Sécurité des protocoles cryptographiques : décidabilité et résultats de réduction

Myrto Arapinis

Octobre 2008

Table des matières

Chapitre 1

Introduction

1.1 Enjeux

Le problème de l'échange sécurisé d'informations n'est pas nouveau. En effet, depuis l'Antiquité de nombreuses techniques ont été développées visant à préserver la confidentialité de certaines informations critiques, en particulier d'ordre militaire. Nous pensons ici aux scytales utilisées par les spartiates au V$^{\text{è}}$ siècle avant J.C. ou encore au célèbre « chiffrement de César » sur lequel nous reviendrons dans un instant. Le problème de la sécurisation des communications est cependant devenu de plus en plus complexe, en particulier avec l'introduction massive des différents outils de télécommunication dans une large partie de nos activités de tous les jours. Par exemple, les échanges sur Internet font désormais partie de notre quotidien, nous permettant d'effectuer des opérations aussi variées que l'achat auprès de boutiques en ligne, l'émission d'ordres bancaires, ou encore le simple échange de messages remplaçant la correspondance papier. La profusion de ces applications rend alors le besoin de communications sécurisées de plus en plus pressant.

S'opérant à distance et transitant sur des réseaux publics de plus en plus larges, toutes ces applications requièrent en effet des garanties de sécurité. Nous nous attendons par exemple à ce que, en donnant un ordre bancaire, nos informations personnelles (numéro de carte de crédit, codes secrets, etc.) ne soient ni interceptées ni réutilisées frauduleusement ; à l'inverse, lors d'un achat en ligne le commerçant veut être sûr que le client est bien en mesure de régler sa facture. La sécurisation de ces applications et de bien d'autres se fait à l'aide de protocoles cryptographiques. Selon les besoins de chacune, ces protocoles visent à garantir des propriétés de sécurité différentes. Nous pouvons ainsi souhaiter garantir la *confidentialité* d'une communication (une tierce personne ne pourra lire un message qui ne lui est pas destiné), l'*intégrité* des messages échangés (ces messages ne pourront être modifiée), l'*anonymat* d'une personne (il ne sera pas possible de remonter à l'émetteur d'un message), ou encore l'*authentification* qui, consiste à s'assurer de l'identité de nos interlocuteurs, pour ne citer que quelques unes des propriétés de sécurité usuelles.

En réalité, cette image est quelque peu simplifiée, puisque la plupart des applications requiert non pas la garantie d'une seule de ces propriétés de sécurité, mais bien la satisfaction simultanée de plusieurs de ces propriétés. Nous n'accepterons par exemple de voter par internet qu'à condition que personne ne puisse faire le rapprochement entre notre identité et notre bulletin de vote, usurper notre identité pour voter à notre place, ni modifier le contenu de notre bulletin de vote. Tout protocole de vote électronique vise précisément à garantir simultanément l'anonymat, l'authentification et l'intégrité. De même, lorsque nous communiquons par mail, nous nous attendons à ce que les messages que nous envoyons ne puissent être interceptés ou modifiés par un « pirate », et que notre adresse ne puisse être empruntée afin d'envoyer des mails en notre nom. Nous attendons donc des protocoles de messagerie qu'ils garantissent la confidentialité, l'intégrité et l'authentification de nos communications par mail.

Au regard des quelques exemples que nous venons de mentionner, il apparaît clairement que le problème de la sécurité des communications est un problème complexe. La conception des protocoles cryptographiques doit en effet prendre en considération de nombreux paramètres, telle que la taille importante des réseaux qui les expose à des risques de piratage, ainsi que la complexité des propriétés de sécurité visées. Il s'agit donc d'un problème difficile dont les enjeux sont importants, toute faille dans les protocoles conçus et mis sur le marché ayant des conséquences dramatiques à de nombreux niveaux (protection de la vie privée, économique, politique etc.). Autant de conséquences qui justifient le déploiement de moyens importants à la conception de protocoles fiables.

1.2 Protocoles cryptographiques

1.2.1 Cryptographie

L'objectif de la cryptographie est de protéger, ou cacher un message de sorte à ce qu'il ne puisse être compris par tout un chacun. Il s'agit alors de transformer un message donné en un autre, de sorte à garantir la *confidentialité* de ce premier. Cette opération est appelée *chiffrement* et fait généralement appel à une *clé* dite de chiffrement. Pour un message initial m, dit *message en clair*, nous noterons $\{m\}_k$, dit chiffré ou encore cryptogramme, le message obtenu après chiffrement de m avec la clé k. L'opération inverse qui consiste à recouvrer le message en clair à partir du chiffré est appelée *déchiffrement*. A toute clé de chiffrement correspond alors une clé de déchiffrement. Notons qu'il existe deux grandes familles d'algorithmes de chiffrement : le chiffrement symétrique et le chiffrement asymétrique. Dans le premier cas (symétrique) la clé de chiffrement et de déchiffrement est la même. Il est donc important qu'elle soit privée, non connue de tous. Dans le second cas (asymétrique) les deux clés sont distinctes, la clé de chiffrement étant généralement publique contrairement à la clé de déchiffrement qui est sensée quant à elle rester privée.

Afin de rendre ces notions générales plus concrètes, illustrons-les par le célèbre algorithme de chiffrement dit 'de Jules César'. Cette opération, inventée par César, consiste à décaler de façon circulaire chaque lettre de quelques crans. Par exemple, décalons les lettres de 3 rangs, comme le faisait Jules César. En appliquant cette procédure au message « Ave Caesar morituri te salutant » nous obtenons le message « Dyh Fdhvdu prulwxul wh vdoxwdqw ». Dans cet exemple, la clé de chiffrement est '3', le message en clair « Ave Caesar morituri

te salutant », et le chiffré « Dyh Fdhvdu prulwxul wh vdoxwdqw ». La clé de déchiffrement correspondante est évidemment '23'.

Cet algorithme de chiffrement manque de toute évidence de robustesse, étant bien trop facile à casser. Etant donné un message de taille suffisante, nous devrions vraisemblablement pouvoir retrouver le message en clair à partir du chiffré en testant simplement toutes les clés possibles, à savoir les 26 possibilités de décalage. Une autre façon de casser ce code consisterait à procéder en comparant la fréquence d'apparition de chaque lettre dans le chiffré à la fréquence moyenne d'apparition des lettres dans la langue du message en clair.

Dans le contexte actuel, et avec l'avènement d'outils calculatoires et de technologies de plus en plus puissantes, ce manque de robustesse pourrait être dramatique. En effet, personne ne ferait de nos jours confiance à ce type de chiffrement s'agissant de communiquer des données personnelles sensibles sur un réseau public. Partant, de nombreux efforts ont été et continuent d'être déployés par les cryptographes afin d'élaborer des algorithmes de chiffrement de plus en plus robustes. A noter que nous disposons à l'heure actuelle d'algorithmes de chiffrement suffisamment sûrs reposant en particulier sur la théorie des nombres.

1.2.2 Les protocoles

Bien qu'importantes, les considérations touchant au chiffrement à proprement parler restent néanmoins orthogonales à nos préoccupations concernant les protocoles. En effet, quand bien même nous tiendrions l'algorithme de chiffrement « parfait », « incassable », cela ne suffirait pas à nous prémunir contre toute attaque. Typiquement, les *attaques* dites *par rejeu* consistent à renvoyer un message déjà émis sans même le décrypter. Imaginons par exemple qu'Alice donne à sa banque un ordre de crédit sur le compte d'un commerçant, en envoyant un message chiffré comportant ses informations bancaires. Une personne malintentionnée pourrait alors intercepter ce message et le rejouer en débitant plusieurs fois le compte d'Alice.

Dans des contextes hostiles, les *protocoles de sécurité* visent alors, comme leur nom l'indique, à établir, entre deux ou plusieurs *participants*, des communications répondant à certaines *propriétés de sécurité* telles que la *confidentialité* ou encore l'*authentification* mentionnées précédemment. Il s'agit de petits programmes qui spécifient une séquence d'*émissions/réceptions* de messages, en précisant la forme et le contenu de ces derniers, l'ordre de cette séquence d'échanges, ainsi que les participants impliqués. Notons $A \rightarrow B : m$ l'émission du message m en provenance du participant A et à destination du participant B.

Considérons le protocole Π_{Toy} défini par la séquence d'*émissions/réceptions* suivante :

$$\begin{aligned} A \rightarrow B : &\quad \{\{s\}_{KB}, A\}_{KB} \\ B \rightarrow A : &\quad \{\{s\}_{KA}, B\}_{KA} \end{aligned}$$

A la première étape, Alice envoie à Bob son identité A ainsi qu'un message secret s chiffré avec la clé publique de Bob KB, le tout chiffré avec cette même clé. Puis, Bob accuse réception de ce message en envoyant à Alice son identité B ainsi que le message secret s chiffré à présent avec la clé publique d'Alice

KA, le tout chiffré avec cette dernière. Bob étant le seul à connaître la clé de déchiffrement correspondant à KB, Alice est convaincue que Bob a reçu son message et que c'est bien lui qui lui a répondu.

Ce protocole décrit implicitement deux programmes appelés *rôles*, l'un décrivant les actions d'Alice et noté R_A, l'autre celles de Bob noté R_B. Une *session* de ce protocole correspond à l'exécution d'un des deux rôles, chaque rôle pouvant être exécuté plusieurs fois, et ce, par différents participants.

Imaginons à présent qu'Alice veuille jouer ce protocole avec Bob, et voyons comment un troisième participant, appelons-le Charlie, pourrait s'interposer dans cet échange, récupérer le secret d'Alice, et se faire passer pour Bob aux yeux de cette dernière. Notons que, Bob ne connaissant pas a priori le secret s, il répondra à tout message de la forme $\{\{X\}_{KB}, A\}_{KB}$ par le message $\{\{X\}_{KA}, B\}_{KA}$. L'attaque se déroule de la manière suivante :

1. Alice initie une session du rôle R_A avec Bob :
$$A \rightarrow B : \{\{s\}_{KB}, A\}_{KB}$$
2. Charlie intercepte ce message et en construit un de la forme attendue par Bob :
$$C \rightarrow B : \{\{\{s\}_{KB}, A\}_{KB}, C\}_{KB}$$
La clé de chiffrement de Bob étant publique, Charlie peut effectivement construire ce message à partir de celui qu'il a intercepté.
3. Bob croit alors jouer une session du rôle R_B avec Charlie. Ayant reçu un message de la forme qu'il attendait il lui répond :
$$C \rightarrow B : \{\{\{s\}_{KB}, A\}_{KC}, B\}_{KC}$$
4. Charlie, connaissant évidemment sa propre clé privée, peut alors déchiffrer les chiffrements faits avec sa clé publique et obtenir le message $\{s\}_{KB}$. Il peut alors envoyer à Bob le message suivant :
$$C \rightarrow B : \{\{s\}_{KB}, C\}_{KB}$$
5. Bob croit alors jouer une nouvelle session du rôle R_B avec Charlie. Le message qu'il a reçu étant cette fois encore de la forme qu'il attendait. Il lui répond :
$$B \rightarrow C : \{\{s\}_{KC}, B\}_{KC}$$
6. Charlie peut maintenant récupérer le secret s et peut se faire passer pour Bob auprès d'Alice puisqu'il sait construire l'accusé de réception attendu par cette dernière :
$$C(B) \rightarrow A : \{\{s\}_{KA}, B\}_{KA}.$$

Aux vues de cet exemple, deux conclusions s'imposent : *(i)* il est possible d'apprendre le contenu d'un message chiffré sans s'attaquer à l'algorithme de chiffrement ; *(ii)* les attaques ne sont pas immédiatement visibles au regard de la spécification du protocole. Il s'en suit qu'il est nécessaire de consacrer de nombreux efforts, non seulement à l'analyse des algorithmes de chiffrement, mais également et de façon indépendante à l'analyse systématique des protocoles de sécurité.

1.3 Vérification des protocoles de sécurité

Le problème de la vérification des protocoles de sécurité est un *problème de décision* qui s'énonce informellement de la façon suivante : étant donné un *protocole*, un *environnement hostile d'exécution*, et une *propriété de sécurité* visée, le protocole vérifie-t-il la propriété visée lorsque celui-ci est exécuté dans l'environnement en question ? Pour tenter de répondre à cette question, la première étape consiste évidemment à formaliser les trois paramètres en jeu.

1.3.1 Modélisation.

Les protocoles. Les notations que nous avons introduites jusqu'ici afin de spécifier le protocole Π_{Toy} sont extrêmement ambiguës. Seul le déroulement *normal* est spécifié dans une telle représentation. Or, l'attaque exhibée nous a montré que d'autres exécutions de ce protocole sont possibles dès lors que nous considérons plusieurs exécutions parallèles de ce dernier, ainsi que la présence de participants malhonnêtes. Il est donc impératif de fournir un modèle précis pour la spécification des protocoles. En particulier, le modèle doit permettre d'expliciter clairement la forme des messages attendus et émis par chaque participant. Des choix de modélisation dépendra alors la fiabilité des résultats de vérification avancés.

L'attaquant. L'attaque sur le protocole Π_{Toy} mentionnée précédemment repose en grande partie sur la capacité de l'*intrus* à *intercepter* des messages avant qu'ils ne parviennent à leur destinataire. Par ailleurs, l'intrus considéré dans cette attaque est capable de générer de nouveaux messages et d'usurper l'identité d'un des participants. Or, selon l'environnement que nous souhaitons considérer, les capacités prêtées à l'intrus varieront grandement. L'intrus ne peut par exemple empêcher un message de parvenir à son destinataire dans le cadre de communications sans fil.

Les propriétés de sécurité. L'éventail des propriétés de sécurité visées par les protocoles de sécurité est large, ce qui est dû à la grande diversité des applications ayant recours à de tels protocoles. A titre d'exemple, citons la propriété du *secret* et celle de l'*authentification*. Une version de la première consiste à déclarer une donnée comme étant critique, et à demander qu'elle ne soit pas révélée à l'intrus. Pour en revenir au protocole Π_{Toy}, si nous spécifions la donnée s comme étant critique alors celui-ci ne vérifie évidemment pas cette version de la propriété du secret. Une version simple de la propriété de l'authentification consiste à s'assurer qu'à chaque fois qu'un participant A finit d'exécuter une session avec pour interlocuteur un participant B, ce dernier a au moins initié l'exécution d'une session. Notons que l'exécution du protocole Π_{Toy} exhibée vérifie bien cette faible version de l'authentification. En effet, nous avons eu l'occasion de remarquer que, bien que vérifiant la susdite propriété, Alice a été dupée quant à l'identité de son interlocuteur.

1.3.2 Méthodes de vérification

Une fois le système entièrement spécifié, nous pouvons nous attaquer à la question même de la vérification, qui se présente comme un cas particulier de

model-checking [16, 10]. Nous nous heurtons alors rapidement à des difficultés dues au caractère non borné de ces systèmes, dont le comportement est à branchement infini (taille des messages, nombre de secrets et de participants non bornés), et de profondeur non bornée (nombre non borné de sessions). Il s'avère d'ailleurs que le problème est indécidable [32, 19]. Face à cette indécidabilité plusieurs approches ont été considérées.

Recherche d'attaques. Quoi qu'une des principales sources d'indécidabilité soit le nombre non borné de sessions, les attaques connues à ce jour impliquent tout au plus trois ou quatre sessions. Ainsi une des premières solutions envisagées a consisté à concevoir des outils de vérification se restreignant à un nombre borné de sessions (tels que FDR [13], Murϕ [44] ou Brutus [17]), grâce auxquels un grand nombre d'attaques ont été découvertes. Il s'est d'ailleurs avéré par la suite que, dans un tel cadre, le problème de la vérification pour la seule propriété du secret devient décidable.

Néanmoins, prouver qu'un protocole préserve son secret dans le cas où seules n sessions sont considérées ne dit rien quant à sa sécurité si une session supplémentaire est autorisée. Notons que pour monter l'attaque sur notre protocole Π_{Toy} trois sessions ont été nécessaires. Ainsi, si nous nous étions contenté de le vérifier sur deux sessions nous aurions pu conclure à tort qu'il préserve son secret. De plus, pour tout entier k, il est possible de construire un protocole non sûr mais dont l'attaque requiert k sessions (nous renvoyons le lecteur au chapitre 7 pour un tel exemple de protocole).

Preuves par abstraction. En toute généralité, les preuves par abstraction consistent à sur-approximer les états accessibles par le système considéré ou encore à sous-approximer les états sûrs du système. Dans le cadre de la vérification des protocoles de sécurité plusieurs options d'abstraction sont possibles.

Il est par exemple possible, dans le but de vérifier la propriété du secret, de sur-approximer la connaissance de l'intrus, *i.e.* l'ensemble des messages accessibles à l'intrus. Telle est l'approche adoptée dans [45, 34, 35].

Une seconde possibilité consiste à faire abstraction des sessions et de l'ordre d'exécution des règles du protocole, comme c'est le cas dans la modélisation par clauses de Horn introduite par [53]. Celle-ci est par ailleurs à la base de nombreux outils efficaces tels que ProVerif ou encore `h1` (voir [11] et [35] respectivement).

Ces méthodes sont correctes contrairement à la précédente. Un protocole vérifiant une propriété sous de telles abstractions la vérifiera nécessairement sans celles-ci. Néanmoins, cette méthode n'est pas complète, *i.e.* elle peut introduire de fausses attaques.

Recherche de classes décidables. Bien que, comme déjà mentionné, le problème de la vérification soit indécidable dans le cas général, il est néanmoins possible de définir des classes de protocoles et de propriétés non triviales pour lesquelles il devient décidable. Une première classe a été proposée en 1981 par D. Dolev et A. Yao [29]. Celle-ci étant néanmoins très restreinte, de nombreux travaux ont été consacrés depuis à la spécification de classes décidables plus larges (pour un panorama de ces travaux voir [23]).

Le problème de la définition de telles classes est toujours d'actualité. En effet, une grande partie des classes définies à ce jour exclue plusieurs mécanismes mis en œuvre par de nombreux protocoles (comme les *secrets temporaires* ou les *copies en aveugle*) et se restreignent souvent à la propriété du secret.

Il est bien entendu possible de combiner les preuves par abstraction à la recherche de classes décidables. Ainsi, B. Blanchet dans [12], et H. Comon-Lundh et V. Cortier dans [19], pour ne citer qu'eux, ont conçu des classes de protocoles et de propriétés pour lesquelles le problème de la vérification est décidable, sous l'abstraction correcte d'un nombre borné de nonces.

1.4 Contenu et plan de la thèse

Comme nous l'avons vu, tout travail sur la vérification des protocoles est soumis à certains choix de modélisation. Avant de présenter la structure de notre travail, il nous semble donc important d'expliciter dès à présent les deux principales caractéristiques du cadre dans lequel s'inscrit celui-ci. Soulignons en effet que nous nous sommes d'emblée placés dans un modèle symbolique, dans lequel les messages, qui ne sont en réalité rien d'autre que des suites de bits, sont abstraits par des termes. Par ailleurs, nous avons toujours travaillé sous l'hypothèse du chiffrement parfait.

Notre démarche a été la suivante : sachant que le problème de la vérification est indécidable dans le cas général, c'est tout naturellement que nous avons commencé par une étude approfondie des preuves d'indécidabilité existantes, afin d'en bien saisir les sources. Il en est ressorti que la question de l'indécidabilité dans le cadre de messages de taille bornée revenait de façon récurrente, les résultats existants ne permettant pas réellement de trancher. La première contribution de ce travail de thèse a donc consisté à fournir une preuve formelle et détaillée de ce résultat négatif.

Nous avons par ailleurs obtenu deux résultats positifs. Le premier est un résultat de décidabilité pour une classe de protocoles assez large et une classe de propriétés incluant un grand nombre de propriétés usuelles de sécurité. Cette classe de protocoles est définie de manière constructive à l'aide d'une transformation de protocoles. Pour des protocoles obtenus par application de cette transformation, il est alors possible de décider s'ils satisfont ou non des propriétés de trace telles que le secret ou l'authentification. Contrairement à la plupart des classes décidables existantes, nous n'avons pas eu besoin d'interdire de la notre des protocoles mettant en œuvre des mécanismes tels que les secrets temporaires ou les copies en aveugles pour obtenir ce résultat de décidabilité.

Le second de nos résultats positifs est un résultat de réduction sur la taille des messages à considérer pour trouver une attaque. Ce résultat est démontré pour la bien connue classe de protocoles satisfaisant le critère de non-unifiabilité des sous-termes énoncé par Abadi et Needham dans [1]. Nous montrons qu'un protocole satisfaisant ce simple critère syntaxique admet une attaque si et seulement si il admet une attaque bien typée suivant un système de typage fort.

Le travail présenté dans cette thèse s'articule en deux parties. La première pose le modèle dans lequel nous avons raisonné afin d'établir les résultats de réduction présentés en seconde partie.

Partie I : Modélisation

Nous commençons, au **Chapitre 2**, par poser clairement notre modèle des protocoles de sécurité. Nous y spécifions l'ensemble des primitives cryptographiques ici considérées, ainsi que les hypothèses faites sur chacune. L'intrus et ses capacités y sont ensuite définis. De plus, nous prenons soin de définir en détail le modèle d'exécution des protocoles de sécurité, dans le cadre d'un nombre non-borné de session.

Au **Chapitre 3** nous présentons la logique qui nous permettra de spécifier les propriétés de sécurités visées par ce travail.

Enfin, au **Chapitre 4** nous définissons formellement le problème de la vérification des protocoles de sécurité qui nous a intéressé. Nous y présentons un premier résultat négatif. En effet, nous prouvons formellement que même en se restreignant à des messages de taille bornée, le problème de la vérification reste indécidable. Nous comparons notre résultat aux preuves déjà existantes, et expliquons les raisons pour lesquelles celles-ci n'avait pas réussit à clore le débat sur ce problème.

Partie II : Vérification

Dans cette seconde partie, consacrée à la vérification, nous commençons, au **Chapitre 5**, par définir les outils requis afin d'établir les résultats de réduction présentés aux chapitres suivants. Nous y présentons en particulier la procédure de décision par résolution de contraintes, dans le cadre d'un nombre borné de sessions, sur laquelle reposent les preuves des chapitres 6 et 7.

Au chapitre **Chapitre 6** nous définissons une classe de protocoles ainsi qu'une classe de propriétés pour lesquelles nous sommes parvenus à un résultat quant au nombre de sessions à considérer, *i.e.* une classe décidable. En d'autres termes, il suffit, pour ces protocoles et propriétés, de considérer des exécutions n'impliquant qu'un nombre borné de sessions pour trouver une attaque.

Le chapitre **Chapitre 7** est quant à lui consacré à la définition d'une classe de protocoles ainsi que d'une classe de propriétés pour lesquelles nous sommes parvenus à un résultat quant à la taille des messages à considérer. En d'autres termes, il suffit, pour ces protocoles et propriétés, de considérer des exécutions n'impliquant que des messages de taille bornée pour trouver une attaque.

Les résultats présentés aux chapitres 6 et 7 ont fait l'objet des publications [5] et [4] respectivement.

Première partie

Modélisation des protocoles de sécurité

Chapitre 2

Modélisation des protocoles de sécurité

L'objectif de ce chapitre est d'introduire le modèle utilisé par la suite pour spécifier, et raisonner sur, des protocoles de sécurité. Il s'agît là d'un modèle symbolique, par opposition aux modèles calculatoires, et plus précisément d'un modèle par rôle avec filtrage. Dans un tel modèle, un certain nombre d'abstractions sont faites, telle que l'« hypothèse du chiffrement parfait ». Aussi, plutôt que de considérer les messages comme des chaînes de bits, ils sont modélisés par les termes de l'algèbre libre construite sur la signature $\mathcal{F} = \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$ des primitives cryptographiques. Ce sont ces hypothèses, relatives à la modélisation, que nous allons expliciter dans ce chapitre.

Nous commencerons par introduire à la section 2.1 certaines notions et notations que nous utiliserons tout au long de cette thèse. Puis, nous dresserons la listes des primitives cryptographiques considérée ici tout en explicitant les hypothèses faites sur ces dernières (section 2.2). Nous présenterons ensuite la modélisation faite des message (section 2.3), et le langage de spécification des protocoles (section 2.4). A la section 2.5 nous définirons les capacités de l'intrus. Enfin, nous décrirons en détail le modèle d'exécution dans le cadre non-borné de session de nos protocoles (section 2.6).

2.1 Préliminaires

La notion d'ensemble et les opérations ensemblistes quoique bien connues et très largement utilisées dans ce travail, se sont parfois avérées insuffisantes s'agissant de rendre compte de tous les aspects des protocoles de sécurité. La notion de *séquence ordonnée d'éléments* s'est dans ces cas avérée plus adéquate. Afin de pouvoir intégrer cette notion, il nous a fallu définir des opérations et notations propres à celle-ci. Dans la présente section, nous nous proposons de résumer l'ensemble des notions et notations relatives aux séquences ordonnées d'éléments.

Soit un ensemble d'éléments $\mathcal{E}$, une séquence ordonnée d'éléments seq sur $\mathcal{E}$ est une expression de la forme $[e_1; \ldots; e_n]$, avec $e_i \in \mathcal{E}$ pour tout i tel que $1 \leq i \leq n$, *i.e.* une liste finie d'éléments de $\mathcal{E}$. Pour tout i tel que $1 \leq i \leq n$, $seq[i]$ dénotera le $i^{ème}$ élément de la séquence seq, *i.e.* $seq[i] = e_i$; seq_i dénotera la restriction de la séquence seq à ses i premiers éléments, *i.e.* $seq_i = [e_1; \ldots; e_i]$; et $|seq|$ dénotera la taille de la séquence seq, *i.e.* $|seq| = n$. Contrairement aux ensembles, les éléments d'une séquence ordonnée d'éléments de $\mathcal{E}$ ne sont pas nécessairement tous distincts.

Afin d'accéder à l'*ensemble d'éléments apparaissant dans une séquence ordonnée*, nous définissons la fonction $\mathsf{Elmts}()$. Soit une séquence ordonnée d'éléments $seq = [e_1; \ldots; e_n]$

$$\mathsf{Elmts}(seq) \stackrel{def}{=} \{e_1, \ldots, e_n\}.$$

La première opération sur les séquences ordonnées d'éléments que nous introduisons est la *concaténation*. Cette opération consiste en la mise bout à bout de deux séquences ordonnées d'éléments. L'opérateur de concaténation est représenté par le symbole @[1]. Soient $seq = [e_1; \ldots; e_n]$ et $seq' = [e'_1; \ldots; e'_m]$ deux séquences d'éléments d'un ensemble $\mathcal{E}$ donné, la concaténation de seq' à la suite de seq est notée seq @ seq' et définie de la manière usuelle

$$seq \text{ @ } seq' \stackrel{def}{=} [e_1; \ldots; e_n; e'_1; \ldots; e'_m].$$

Soient n séquences ordonnées $seq^1, \ldots, seq^n$, nous noterons $\underset{1 \leq i \leq n}{@} seq^i$, la séquence ordonnée définie par :

$$\underset{1 \leq i \leq n}{@} seq^i \stackrel{def}{=} \begin{cases} [] & \text{si } n = 0 \\ (\underset{1 \leq i \leq n-1}{@} seq^i) \text{ @ } seq^n & \text{sinon.} \end{cases}$$

Souvent nous aurons besoin de concaténer à la suite d'une séquence $seq = [e_1; \ldots; e_n]$ uniquement les éléments de la séquence $seq' = [e'_1; \ldots; e'_m]$ n'apparaissant pas déjà dans seq, et ce dans leur ordre d'apparition dans seq'. L'opérateur de *concaténation sans répétition* est représenté par le symbole $\ltimes$. Cette opération est définie par

$$seq \ltimes seq' \stackrel{def}{=} \begin{cases} seq & \text{si } m = 0 \\ [e_1; \ldots; e_n; e'_1] \ltimes seq'' & \text{si } e_i \neq e'_1 \text{ pour tout } i \in \{1, \ldots, n\} \\ seq \ltimes seq'' & \text{sinon} \end{cases}$$

où $seq'' = [e'_2; \ldots; e'_m]$. A noter que si seq admet des répétitions, *i.e.* il existe $i, j \in \{1, \ldots, n\}$, $i \neq j$, tels que $e_i = e_j$, alors $seq \ltimes seq'$ aussi admet des répétitions. Par contre, si seq est sans répétitions, alors $seq \ltimes seq'$ l'est aussi. Soient n séquences ordonnées $seq^1, \ldots, seq^n$, nous noterons $\underset{1 \leq i \leq n}{\ltimes} seq^i$ la séquence

[1] Notation empruntée à Caml.

définie par

$$\underset{1\leq i\leq n}{\ltimes} seq^i \stackrel{def}{=} \begin{cases} [] & \text{si } n = 0 \\ seq^1 & \text{si } n = 1 \\ (\underset{1\leq i\leq n-1}{\ltimes} seq^i) \ltimes seq^n & \text{sinon.} \end{cases}$$

Les éléments d'une séquence ordonnée n'étant pas nécessairement distincts, nous aurons aussi besoin de connaître le nombre d'occurrences d'un élément donné dans la séquence considérée. La fonction $\mathsf{Occ}()$ se charge de retourner cette information étant donnés une séquence d'éléments seq, ainsi qu'un élément e. Formellement, cette fonction est définie comme indiqué ci-dessous.

$$\mathsf{Occ}(seq, e) \stackrel{def}{=} \begin{cases} 0 & \text{si } seq = [] \\ 1 + \mathsf{Occ}(seq', e) & \text{si } seq = [e]@seq' \\ \mathsf{Occ}(seq', e) & \text{sinon.} \end{cases}$$

Il nous faut encore introduire une dernière notation avant de passer à la modélisation à proprement parler des protocoles de sécurité. Celle-ci ne relève pas des séquences ordonnées d'éléments. Soit un entier $n \in \mathbb{N}$, $[\![n]\!]$ dénotera l'ensemble des entiers compris entre 1 et n, *i.e.*

$$[\![n]\!] \stackrel{def}{=} \{1, \ldots, n\}.$$

En particulier, $[\![0]\!] = \emptyset$.

2.2 Primitives cryptographiques considérées

Les primitives cryptographiques seront représentées par les symboles de fonction de la signature $\mathcal{F} = \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$. Nous discutons ici des hypothèses faites sur chacune respectivement. En particulier, nous ne considérerons pas les propriétés algébriques de ces fonctions.

Concaténation. La concaténation est l'opération de mise bout à bout de deux messages. Elle est représentée par le symbole de fonction binaire $\langle\rangle$, aussi appelé *paire*. Ainsi, $\langle m_1, m_2\rangle$ dénote la juxtaposition du message m_2 à la suite du message m_1. Contrairement aux modèles calculatoires, nous ne considérerons pas cette opération comme associative. En particulier, les messages $\langle\langle m_1, m_2\rangle, m_3\rangle$ et $\langle m_1, \langle m_2, m_3\rangle\rangle$ sont deux messages distincts.

Chiffrement. L'opération de chiffrement est la transformation d'un message m en un message m', dans le but de rendre sa compréhension impossible, et de garantir ainsi sa confidentialité. Cette opération se fait à l'aide d'une clé de chiffrement. L'opération inverse est le déchiffrement, et nécessite la clé de déchiffrement, parfois notée k^{-1}, correspondant à la clé de chiffrement k utilisée pour chiffrer m en m'. Nous distinguons deux familles d'algorithmes de chiffrement : symétrique et asymétrique.

Chiffrement symétrique. Le déchiffrement d'un message m chiffré symétriquement avec la clé k se fait avec la même clé k, *i.e.* $k^{-1} = k$. Deux entités a et b

échangeant des messages chiffrés symétriquement doivent donc partager la clé de chiffrement. Dans de nombreux protocoles, la clé symétrique partagée par les participants est préalablement donnée aux participants. Nous parlons alors de *clé symétrique long-terme* et nous la représentons à l'aide du symbole de fonction binaire shk. Ainsi, $\mathsf{shk}(a, b)$ dénote la clé symétrique long-terme partagée entre les participants a et b ; mais celle-ci devrait être la même que la clé symétrique long-terme $\mathsf{shk}(b, a)$ partagée entre b et a. Plutôt que de considérer ce symbole de fonction comme étant commutatif, nous nous donnons un ordre total $\leq$ sur l'ensemble des participants, et chaque couple a, b de participants partage une unique clé symétrique long-terme ($\mathsf{shk}(a, b)$ si $a \leq b$ ou $\mathsf{shk}(b, a)$ si $b \leq a$[2]. L'opération de chiffrement symétrique est représentée par le symbole de fonction binaire senc. Ainsi, $\mathsf{senc}(m, \mathsf{shk}(a, b))$ dénote le chiffrement symétrique du message m à l'aide de la clé long-terme $\mathsf{shk}(a, b)$ partagée par a et b.

Chiffrement asymétrique. Contrairement au chiffrement symétrique, la clé de chiffrement asymétrique et la clé de déchiffrement asymétrique correspondante sont distinctes, *i.e.* $k^{-1} \neq k$. La première est souvent publique (connue de tous) et la seconde secrète. Soit k une clé de chiffrement publique, nous noterons $\mathsf{pvk}(k)$ la clé de déchiffrement correspondante, *i.e.* $k^{-1} = \mathsf{pvk}(k)$. L'opération de chiffrement asymétrique est représentée par le symbole de fonction binaire aenc. Ainsi, $\mathsf{aenc}(m, k)$ dénote le chiffrement asymétrique du message m à l'aide de la clé publique k.

Lorsque nous ne souhaiterons pas distinguer le chiffrement symétrique du chiffrement asymétrique, nous noterons $\mathsf{enc}(m, k)$ pour dénoter le message m chiffré (symétriquement ou asymétriquement indifféremment) avec la clé k.

Dans les modèles symboliques, et en particulier dans celui que nous considérons dans ce travail, les primitives de chiffrement symétrique et asymétrique sont satisfont l'hypothèse dite du chiffrement parfait, qui repose sur le fait suivant : un message chiffré avec une clé de chiffrement k ne peut être déchiffré que par une entité connaissant la clé de déchiffrement correspondante k^{-1}. Aussi, nous supposons que deux messages chiffrés $\mathsf{enc}(m, k)$ et $\mathsf{enc}(m', k')$ sont identiques si et seulement si $m = m'$ et $k = k'$; que l'opération de chiffrement n'est pas commutative, *i.e.* $\mathsf{enc}(\mathsf{enc}(m, k), k')$ et $\mathsf{enc}(\mathsf{enc}(m, k'), k)$ sont deux messages distincts ; ou encore, que le message obtenu en chiffrant symétriquement le message $\mathsf{senc}(m, k)$ avec la clé k, *i.e.* $\mathsf{senc}(\mathsf{senc}(m, k), k)$, est différent du message m lui-même.

Ces hypothèses ne sont pas toujours pas satisfaites (*e.g.* RSA) dans la réalité. Néanmoins, même sous ces hypothèses, le problème de la vérification des protocoles[3] n'est pas trivial. En effet, comme nous le verrons au chapitre 4, le problème reste indécidable même si nous nous permettons les abstractions décrites dans cette section. Ceci est dû au fait qu'un grand nombre d'attaques sur les protocoles de sécurité ne repose pas sur de possibles failles du chiffrement qui seraient abstraites par une telle modélisation des primitives cryptographiques. De plus, un certain nombre de travaux récents initiés par Martìn Abadi et Philip Rogaway avec [2] s'attèle avec succès à transposer les résultats obtenus sous de telles hypothèses au modèle calculatoire, ce qui confère une certaine légitimité à ce type d'hypothèses.

[2] Il nous arrivera, par abus de notation, de noter $\mathsf{shk}(a, b)$ alors que $b < a$.

[3] Le problème de la vérification des protocoles sera formellement défini au chapitre 4.

Signature. Comme l'opération de chiffrement, la signature transforme un message m en un message m', mais dans le but, cette fois, pour le signataire de s'authentifier (par analogie avec la signature manuscrite), *i.e.* assurer son interlocuteur de son identité et donc de l'origine du message m'. Cette opération nécessite également une clé pour signer et une autre pour authentifier (vérifier) la signature. La signature d'un message m avec la clé k^{-1} peut être comprise comme le chiffrement asymétrique de ce message avec la clé privée k^{-1}. Cette opération est représentée pas le symbole de fonction binaire sign. Ainsi, $\mathsf{sign}(m, k^{-1})$ dénote la signature du message m à l'aide de la clé privée k^{-1}. En fonction du schéma de signature, cette opération peut être ou ne pas être une opération inversible. Dans le premier cas, une entité connaissant la clé de vérification k peut recouvrer un message m à partir de sa signature avec la clé privée correspondante k^{-1}, *i.e.* à partir du message $\mathsf{sign}(m, k^{-1})$. L'opération de signature aussi satisfait l'hypothèse selon laquelle les messages $\mathsf{sign}(m, k^{-1})$ et $\mathsf{sign}(m', k'^{-1})$ sont identiques si et seulement si $m = m'$ et $k^{-1} = k'^{-1}$.

Hachage. Le hachage est une opération considérée, dans les modèles symboliques, comme non-inversible et injective. Cette opération est représentée par le symbole de fonction unaire h. Ainsi, $\mathsf{h}(m)$ dénote l'application d'une fonction de hachage au message m. La non-inversibilité de h correspond au fait qu'il n'est pas possible de recouvrer m en ne connaissant que $\mathsf{h}(m)$; l'injectivité (encore appelée résistance aux collisions) de h correspond à l'hypothèse selon laquelle $\mathsf{h}(m) = \mathsf{h}(m')$ si et seulement si $m = m'$.

2.3 Messages

Un protocole de sécurité étant une séquence d'échanges de messages entre participants, il convient en premier lieu de modéliser ces messages. Comme nous l'avons dit en introduction de ce chapitre, ils seront modélisés par les termes de l'algèbre libre construite sur la signature $\mathcal{F}$ des primitives crypotgraphiques, présentée à la section précédente. Les parties atomiques des messages se décomposent en quatre ensembles dénombrables disjoints.

Participants. Nous distinguons deux sortes de participants, les *agents* $\mathcal{A} = \{a, b, \ldots\} \uplus \{\epsilon\}$, et les *serveurs* $\mathcal{S} = \{s_1, s_2, \ldots\}$. Nous avons distingué un agent ϵ pour représenter l'« intrus » (*cf.* section 2.5). Nous considérons ces deux ensembles de participants disjoints, *i.e.* $\mathcal{A} \cap \mathcal{S} = \emptyset$. L'ensemble $\mathcal{P} = \mathcal{A} \uplus \mathcal{S}$ dénotera l'ensemble des participants, *i.e.* les agents et les serveurs indifféremment. Les noms des participants sont des données publiques, *i.e* connues de tous. Comme nous l'expliquions à la section précédente, nous munissons l'ensemble $\mathcal{P}$ d'un ordre total $\leq_{\mathcal{P}}$, et supposons que pour tout $p \in \mathcal{P}$, $\epsilon \leq_{\mathcal{P}} p$. Rappelons qu'il nous arrivera de noter $\mathsf{shk}(p, p')$ avec $p' \leq_{\mathcal{P}} p$ au lieu de $\mathsf{shk}(p', p)$ pour plus de simplicité.

Constantes. Les constantes sont des données publiques, *i.e.* connues de tous, et modélisées par l'ensemble $\mathcal{C}$. Nous les retrouverons en particulier aux chapitres 4 et 7.

Nonces. Les participants d'un protocole peuvent générer des données aléatoires, appelées nonces, et modélisées par l'ensemble $\mathcal{N} = \{n, m, \ldots\}$. Les nonces auront pour objectif de distinguer deux « exécutions » distinctes du même protocole, d'assurer la « fraîcheur » de chaque « session ».

Variables. Intuitivement, l'ensemble de variables $\mathcal{V} = \{x, y, \ldots\}$ nous servira lors de la modélisation de la réception par un participant d'un message dont il ne peut analyser toutes les parties. Ceci peut par exemple survenir lors de la réception par un participant d'un nonce généré par un autre, nonce dont il ne connaît pas la valeur au préalable ; ou encore lors de la réception d'un message chiffré dont il ne connaît pas la clé de déchiffrement correspondante, et ne peut donc pas en vérifier le contenu.

Nous sommes à présent en mesure de définir l'ensemble des termes qui modéliserons les messages.

Définition 2.3.1 (Terme)**.** *L'ensemble des termes $\mathcal{T}$ est engendré par la grammaire suivante :*

$$\begin{array}{rcll} t, u, \ldots & ::= & p & p \in \mathcal{P} \\ & | & c & c \in \mathcal{C} \\ & | & n & n \in \mathcal{N} \\ & | & x & x \in \mathcal{V} \\ & | & \mathsf{pvk}(p) & p \in \mathcal{P} \\ & | & \mathsf{shk}(p_1, p_2) & p_1, p_2 \in \mathcal{P} \text{ et } p_1 \leq_{\mathcal{P}} p_2 \\ & | & f(t_1, t_2) & f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ & | & \mathsf{h}(t) & \end{array}$$

A noter que l'application du symbole pvk est restreinte à l'ensemble des participants $\mathcal{P}$. Ainsi, un participant et sa clé publique sont confondus pour plus de simplicité. Nous dénoterons $\mathcal{K}^{-1}$ l'ensemble de clés long-terme de déchiffrement, *i.e.*

$$\mathcal{K}^{-1} \stackrel{def}{=} \mathcal{P}\{\mathsf{pvk}(p_1), \mathsf{shk}(p_1, p_2) \mid p_1, p_2 \in \mathcal{P}\}.$$

Un terme t sera dit *atomique* si $t \in (\mathcal{C} \cup \mathcal{N} \cup \mathcal{V} \cup \mathcal{K}^{-1})$.

Exemple 2.3.2. *Soient x une variable dans $\mathcal{V}$, et a et b deux agents dans $\mathcal{A}$,*

$$t = \mathsf{aenc}(\langle \mathsf{aenc}(x, b), a \rangle, b)$$

est un terme de $\mathcal{T}$ représentant le chiffrement avec la clé publique de b du message obtenu en concaténant le chiffré de x avec la clé publique de b, et l'identité de a.

Définition 2.3.3 ($|t|$)**.** *Soit t un terme, la taille de t, dénotée $|t|$, est définie inductivement comme indiqué ci-dessous*

$$|t| \stackrel{def}{=} \begin{cases} 1 & si\ t \in (\mathcal{P} \cup \mathcal{C} \cup \mathcal{N} \cup \mathcal{V}) \\ 1 + |t_1| + |t_2| & si\ t = f(t_1, t_2)\ et\ f \in \{\mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ 1 + |t_1| & si\ t = f(t_1)\ et\ f \in \{\mathsf{pvk}, \mathsf{h}\}. \end{cases}$$

Par la suite, nous aurons besoin de manipuler les termes et d'accéder à leur parties. Nous introduisons à cet effet les fonctions $\mathsf{St}()$, $\mathsf{Est}()$, $\mathsf{Plaintext}()$, $\mathsf{Agts}()$, $\mathcal{A}()$, $\mathsf{Partcpts}()$, $\mathcal{P}()$, $\mathsf{Csts}()$, $\mathcal{C}()$, $\mathsf{Nces}()$, $\mathcal{N}()$, $\mathsf{Vars}()$, et $\mathcal{V}()$.

Soit un terme $t \in \mathcal{T}$, $\mathsf{St}(t)$ dénote l'ensemble des sous-termes de t :

$$\mathsf{St}(t) \stackrel{def}{=} \begin{cases} \{t\} \cup \mathsf{St}(t_1) \cup \mathsf{St}(t_2) & \text{si } t = f(t_1, t_2) \text{ et } f \in \{\mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ \{t\} \cup \mathsf{St}(t_1) & \text{si } t = f(t_1) \text{ et } f \in \{\mathsf{pvk}, \mathsf{h}\} \\ \{t\} & \text{sinon,} \end{cases}$$

$\mathsf{Est}(t)$ dénote l'ensemble des sous-termes chiffrés de t :

$$\mathsf{Est}(t) \stackrel{def}{=} \begin{cases} \mathsf{Est}(t_1) \cup \mathsf{Est}(t_2) & \text{si } t = \langle t_1, t_2 \rangle \\ \{t\} \cup \mathsf{Est}(t_1) \cup \mathsf{Est}(t_2) & \text{si } t = f(t_1, t_2) \text{ et } f \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ \{t\} \cup \mathsf{Est}(t_1) & \text{si } t = \mathsf{h}(t_1) \\ \emptyset & \text{sinon,} \end{cases}$$

et $\mathsf{Plaintext}(t)$ dénote l'ensemble défini comme suit :

$$\mathsf{Plaintext}(t) \stackrel{def}{=} \begin{cases} \mathsf{Plaintext}(t_1) & \text{si } t = f(t_1, t_2) \\ & \text{et } f \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ \mathsf{Plaintext}(t_1) & \text{si } t = \mathsf{h}(t_1) \\ \mathsf{Plaintext}(t_1) \cup \mathsf{Plaintext}(t_2) & \text{si } t = \langle t_1, t_2 \rangle \\ \{t\} & \text{sinon.} \end{cases}$$

Les fonctions $\mathsf{St}()$, $\mathsf{Est}()$ et $\mathsf{Plaintext}()$ s'étendent naturellement aux ensembles de termes. Soit T un ensemble de termes :

$$\mathsf{St}(T) \stackrel{def}{=} \bigcup_{t \in T} \mathsf{St}(t), \qquad \mathsf{Est}(T) \stackrel{def}{=} \bigcup_{t \in T} \mathsf{Est}(t), \text{ et} \qquad \mathsf{Plaintext}(T) \stackrel{def}{=} \bigcup_{t \in T} \mathsf{Plaintext}(t).$$

Soit un terme $t \in \mathcal{T}$, $\mathcal{A}(t)$ dénote l'*ensemble d'agents apparaissant dans* t, *i.e.*

$$\mathcal{A}(t) \stackrel{def}{=} \mathsf{St}(t) \cap \mathcal{A},$$

et $\mathsf{Agts}(t)$ dénote la *séquence ordonnée par ordre d'apparition dans* t *et sans répétitions des agents de* t, soit

$$\mathsf{Agts}(t) \stackrel{def}{=} \begin{cases} [t] & \text{si } t \in \mathcal{A} \\ \mathsf{Agts}(t_1) \ltimes \mathsf{Agts}(t_2) & \text{si } t = f(t_1, t_2) \\ & \text{et } f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{shk}\} \\ \mathsf{Agts}(t_1) & \text{si } t = f(t_1), \text{ et } f \in \{\mathsf{h}, \mathsf{pvk}\} \\ [] & \text{sinon.} \end{cases}$$

$\mathcal{P}(t)$ dénote l'*ensemble des participants (serveurs et agents indifféremment) apparaissant dans* t, *i.e.*

$$\mathcal{P}(t) \stackrel{def}{=} \mathsf{St}(t) \cap \mathcal{P},$$

et $\mathsf{Partcpts}(t)$ dénote la *séquence ordonnée par ordre d'apparition dans t et sans répétitions des participants de t*, soit

$$\mathsf{Partcpts}(t) \stackrel{def}{=} \begin{cases} [t] & \text{si } t \in \mathcal{P} \\ \mathsf{Partcpts}(t_1) \ltimes \mathsf{Partcpts}(t_2) & \text{si } t = f(t_1, t_2) \\ & \text{et} f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{shk}\} \\ \mathsf{Partcpts}(t_1) & \text{si } t = f(t_1), \text{ et } f \in \{\mathsf{h}, \mathsf{pvk}\} \\ [] & \text{sinon.} \end{cases}$$

$\mathcal{C}(t)$ dénote l'*ensemble des constantes apparaissant dans t, i.e.*

$$\mathcal{C}(t) \stackrel{def}{=} \mathsf{St}(t) \cap \mathcal{C},$$

et $\mathsf{Csts}(t)$ dénote la *séquence ordonnée par ordre d'apparition dans t et sans répétitions des constantes de t*, soit

$$\mathsf{Csts}(t) \stackrel{def}{=} \begin{cases} [t] & \text{si } t \in \mathcal{C} \\ \mathsf{Csts}(t_1) \ltimes \mathsf{Csts}(t_2) & \text{si } t = f(t_1, t_2) \\ & \text{et } f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{shk}\} \\ \mathsf{Csts}(t_1) & \text{si } t = f(t_1), \text{ et } f \in \{\mathsf{h}, \mathsf{pvk}\} \\ [] & \text{sinon.} \end{cases}$$

De même, $\mathcal{N}(t)$ dénote l'*ensemble de nonces apparaissant dans t, i.e.*

$$\mathcal{N}(t) \stackrel{def}{=} \mathsf{St}(t) \cap \mathcal{N},$$

et $\mathsf{Nces}(t)$ dénote la *séquence ordonnée par ordre d'apparition dans t et sans répétitions des nonces de t*, soit

$$\mathsf{Nces}(t) \stackrel{def}{=} \begin{cases} [t] & \text{si } t \in \mathcal{N} \\ \mathsf{Nces}(t_1) \ltimes \mathsf{Nces}(t_2) & \text{si } t = f(t_1, t_2) \\ & \text{et } f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ \mathsf{Nces}(t_1) & \text{si } t = \mathsf{h}(t_1) \\ [] & \text{sinon} \end{cases}$$

Et, $\mathcal{V}(t)$ dénote l'*ensemble de variables apparaissant dans t, i.e.*

$$\mathcal{V}(t) \stackrel{def}{=} \mathsf{St}(t) \cap \mathcal{V},$$

et $\mathsf{Vars}(t)$ dénote la *séquence ordonnée par ordre d'apparition dans t et sans répétitions des variables de t*, soit

$$\mathsf{Vars}(t) \stackrel{def}{=} \begin{cases} [t] & \text{si } t \in \mathcal{V} \\ \mathsf{Vars}(t_1) \ltimes \mathsf{Vars}(t_2) & \text{si } t = f(t_1, t_2) \\ & \text{et } f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ \mathsf{Vars}(t_1) & \text{si } t = \mathsf{h}(t_1) \\ [] & \text{sinon} \end{cases}$$

Un terme *clos* est un terme sans variables. $\mathcal{M}$ dénote l'ensemble des termes clos, *i.e.* $\mathcal{M} = \{t \in \mathcal{T} \mid \mathcal{V}(t) = \emptyset\}$.

Exemple 2.3.4. *Soit* $t = \mathsf{aenc}(\langle \mathsf{aenc}(x, b), a\rangle, b)$ *le terme de l'exemple 2.3.2,* $|t| = 7$*, et*

$$\mathsf{St}(t) = \{t, \langle \mathsf{aenc}(x, b), a\rangle, b, \mathsf{aenc}(x, b), a, x\},$$

$$\mathsf{Est}(t) = \{t, \mathsf{aenc}(x, b)\}, \qquad \mathsf{Plaintext}(t) = \{x, a\}$$

$$\mathcal{A}(t) = \{b, a\}, \quad \mathsf{Agts}(t) = [b; a], \quad \mathcal{P}(t) = \{b, a\}, \quad \mathsf{Partcpts}(t) = [b; a],$$

$$\mathcal{C}(t) = \emptyset, \quad \mathsf{Csts}(t) = [], \quad \mathcal{N}(t) = \emptyset, \quad \mathsf{Nces}(t) = [], \quad \mathcal{V}(t) = \{x\}, \quad \mathsf{Vars}(t) = [x].$$

Définition 2.3.5 (Substitution)**.** *Une substitution* σ *est une fonction d'un sous-ensemble fini (appelé domaine, noté* $\mathrm{dom}(\sigma)$*) de l'ensemble des variables dans les termes,* i.e. $\sigma : \mathcal{V} \to \mathcal{T}$ *telle que pour toute variable* $x \in \mathrm{dom}(\sigma)$*,* $\sigma(x) \neq x$*.*

La substitution σ *est* close *si pour tout* $x \in \mathrm{dom}(\sigma)$*,* $\sigma(x)$ *est clos,* i.e. $\sigma(x) \in \mathcal{M}$*.*

Une substitution σ de domaine $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$ sera souvent représentée par l'expression $\{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}$. L'application d'une substitution σ à un terme t, dénotée $\sigma(t)$ ou $t\sigma$ indifféremment, est définie de la manière suivante

$$t\sigma \stackrel{def}{=} \begin{cases} x & \text{si } x \in (\mathcal{V} \setminus \mathrm{dom}(\sigma)) \\ \sigma(x) & \text{si } x \in \mathrm{dom}(\sigma) \\ f(\sigma(t_1), \sigma(t_2)) & \text{si } t = f(t_1, t_2) \text{ et } f \in \{\mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ f(\sigma(t_1)) & \text{si } t = f(t_1) \text{ et } f \in \{\mathsf{pvk}, \mathsf{h}\}. \end{cases}$$

Définition 2.3.6 (Unificateur)**.** *Deux termes* t *et* u*. Une substitution* σ *est un unificateur de* t *et* u *si et seulement si* $t\sigma = u\sigma$*.* t *et* u *sont dits être* unifiables *si une substitution existe.*

Théorème 2.3.7 (mgu)**.** *Soient* t *et* u *deux termes unifiables. Il existe un* unificateur le plus général *de* t *et de* u*, dénoté* $\mathrm{mgu}(t, u)$*, c'est à dire une substitution* σ *telle que*

- σ *est un unificateur de* t *et de* u*,* i.e. $t\sigma = u\sigma$*, et*
- *tout unificateur* θ *de* t *et de* u *est une instance de* σ*,* i.e. $\exists\theta'.\ \theta = \sigma\theta$*.*

2.4 Protocoles

Nous pouvons à présent définir les éléments atomiques de nos protocoles : les *événements.*

Définition 2.4.1 (Evénement)**.** *Un événement est*

- *soit, l'*émission *initiée par un participant* $p_1 \in \mathcal{P}$ *d'un message* $t \in \mathcal{T}$ *à destination d'un participant* $p_2 \in \mathcal{P}$*, dénotée* $\mathsf{snd}(p_1, p_2, t)$*,*
- *soit, la* réception *initiée par un participant* $p_1 \in \mathcal{P}$ *d'un message* $t \in \mathcal{T}$ *en provenance d'un participant* $p_2 \in \mathcal{P}$*, dénotée* $\mathsf{rcv}(p_1, p_2, t)$*,*
- *soit, la* déclaration *initiée par un participant* $p \in \mathcal{P}$ de son état relativement à un prédicat Q*, dénotée* $\mathsf{Q}(p, t_1, \ldots, t_n)$*, avec* $t_1, \ldots, t_n \in \mathcal{T}$*.*

Les deux premiers types d'événements sont ceux auxquels nous nous attendions suite à notre introduction : l'émission et la réception de messages. Le troisième type d'événements, aussi appelés *status events*, est introduit dans le but de spécifier les propriétés de sécurité considérées. Cette notion sera précisée et illustrée au chapitre 3.

Les fonctions $\mathsf{St}()$, $\mathsf{Est}()$, $\mathsf{Plaintext}()$, $\mathcal{A}()$, $\mathsf{Agts}()$, $\mathcal{P}()$, $\mathsf{Partcpts}()$, $\mathcal{C}()$, $\mathsf{Csts}()$, $\mathcal{N}()$, $\mathsf{Nces}()$, $\mathcal{V}()$, et $\mathsf{Vars}()$, définies sur les termes, s'étendent naturellement aux évènements et aux séquences d'évènements. Soit e un évènement.

Si $e = \mathsf{snd}(p_1, p_2, t)$ ou $e = \mathsf{rcv}(p_1, p_2, t)$, alors
$\forall \mathcal{G} \in \{\mathsf{St}, \mathsf{Est}, \mathsf{Plaintext}, \mathcal{A}, \mathsf{Agts}, \mathcal{P}, \mathsf{Partcpts}, \mathcal{C}, \mathsf{Csts}, \mathcal{N}, \mathsf{Nces}, \mathcal{V}, \mathsf{Vars}\}.\ \ \mathcal{G}(e) \stackrel{def}{=} \mathcal{G}(t).$

Si $e = \mathsf{Q}(t_1, \ldots, t_n)$, alors
$\forall \mathcal{G} \in \{\mathsf{St}, \mathsf{Est}, \mathsf{Plaintext}, \mathcal{A}, \mathcal{P}, \mathcal{C}, \mathcal{N}, \mathcal{V}\}. \quad \mathcal{G}(e) \stackrel{def}{=} \bigcup_{i \in [\![n]\!]} \mathcal{G}(t_i),$ et
$\forall \mathcal{G} \in \{\mathsf{Agts}, \mathsf{Partcpts}, \mathsf{Csts}, \mathsf{Nces}, \mathsf{Vars}\}. \quad \mathcal{G}(e) \stackrel{def}{=} \ltimes_{i \in [\![n]\!]} \mathcal{G}(t_i).$

De même, soit une séquence d'événements $seq = [e_1; \ldots; e_\ell]$,
$\forall \mathcal{G} \in \{\mathsf{St}, \mathsf{Est}, \mathsf{Plaintext}, \mathcal{A}, \mathcal{P}, \mathcal{C}, \mathcal{N}, \mathcal{V}\}. \quad \mathcal{G}(seq) \stackrel{def}{=} \bigcup_{i \in [\![n]\!]} \mathcal{G}(e_i),$ et
$\forall \mathcal{G} \in \{\mathsf{Agts}, \mathsf{Partcpts}, \mathsf{Csts}, \mathsf{Nces}, \mathsf{Vars}\}. \quad \mathcal{G}(seq) \stackrel{def}{=} \ltimes_{i \in [\![n]\!]} \mathcal{G}(e_i).$

Soit $seq = [e_1; \ldots; e_\ell]$ une séquence d'événements. La *restriction de seq aux communications*, *i.e.* aux émissions et aux réceptions, est notée $\mathsf{Coms}(seq)$ et définie par :

$$\mathsf{Coms}(seq) \stackrel{def}{=} \begin{cases} [] & \text{si } \ell = 0 \\ [e_1]@(\mathsf{Coms}(seq')) & \text{si } \ell > 0 \quad \text{et } e_1 = \mathsf{snd}(p_1, p_2, t) \\ & \qquad\quad \text{ou } e_1 = \mathsf{rcv}(p_1, p_2, t), \\ \mathsf{Coms}(seq') & \text{sinon,} \end{cases}$$

avec $seq' = [e_2; \ldots; e_\ell]$.

La *restriction de seq aux événements initiés par un participant* $p_1 \in \mathcal{P}$ est notée $seq(p_1)$ et définie par :

$$seq(p_1) \stackrel{def}{=} \begin{cases} [] & \text{si } \ell = 0 \\ [e_1]@(seq'(p_1)) & \text{si } \ell > 0 \quad \text{et } e_1 = \mathsf{snd}(p_1, p_2, t) \\ & \qquad\quad \text{ou } e_1 = \mathsf{rcv}(p_1, p_2, t) \\ [\mathsf{Q}(t_1, \ldots, t_n)]@(seq'(p_1)) & \text{si } \ell > 0 \quad \text{et } e_1 = \mathsf{Q}(p_1, t_1, \ldots, t_n) \\ seq' & \text{sinon,} \end{cases}$$

avec $seq' = [e_2; \ldots; e_\ell]$. Remarquons que dans le cas où $e_1 = \mathsf{Q}(p_1, t_1, \ldots, t_n)$, e_1 est remplacé dans la restriction de seq aux événements initiés par p_1, par le status event $\mathsf{Q}(t_1, \ldots, t_n)$. Nous n'expliquons pas ici ce choix de modélisation, mais nous y reviendrons une fois la définition des protocoles posée.

Nous en venons maintenant à la définition de protocole considérée dans ce travail.

Définition 2.4.2 (Protocole)**.** *Un protocole est une séquence d'événements* $\Pi = [e_1; \ldots; e_\ell]$ *vérifiant :*

1. $\forall p, p' \in \mathcal{P}.\quad p \neq p' \quad \Rightarrow \quad (\mathcal{N}(\Pi(p)) \cap \mathcal{N}(\Pi(p')) = \emptyset \wedge \mathcal{V}(\Pi(p)) \cap \mathcal{V}(\Pi(p')) = \emptyset)$
2. $\forall i \in [\![\ell]\!]$, *si* $e_i = \mathsf{Q}(p, t_1, \ldots, t_n)$, *alors* $p \in \mathcal{P}$ *et*

   $$\forall j \in [\![n]\!].\ \forall x \in \mathcal{V}(t_j).\ \exists p' \in \mathcal{P}.\ \exists k \in [\![i]\!].\ e_k = \mathsf{rcv}(p, p', t) \wedge x \in \mathcal{V}(t)$$

3. $\mathsf{Coms}(\Pi)$ *est de la forme* $[e'_1; e''_1; \ldots; e'_m; e''_m]$, *avec pour tout* $i \in [\![m]\!]$
   a) $e'_i = \mathsf{snd}(p_i, p'_i, u_i)$ *et* $e''_i = \mathsf{rcv}(p'_i, p_i, v_i)$,
   b) *si* $x \in \mathcal{V}(u_i)$, *alors il existe* $j \in [\![i-1]\!]$ *tel que* $x \in \mathcal{V}(v_j)$,
   c) $\delta_i \neq \bot$, *avec*

   $$\begin{cases} \delta_1 = \mathrm{mgu}(u_1, v_1), \\ \delta_k = \mathrm{mgu}(u_k\delta_1 \ldots \delta_{k-1}, v_k\delta_1 \ldots \delta_{k-1}) \end{cases}$$

   La substitution $\delta_\Pi = \delta_1 \ldots \delta_m$ *est appelée la* substitution honnête *de* Π.

La première condition stipule que chaque nonce apparaît dans les événements initiés par au plus un des participants du protocole. Cela correspond au fait que chaque nonce est engendré par un unique participant. Il en va de même pour les variables apparaissant dans la spécification du protocole. Les conditions 2 et 3b combinées avec la condition 1 assurent qu'une variable n'est introduite dans un status event ou dans un envoi initié par un participant, qu'à condition que ce dernier ait bien reçu au préalable cette même variable. Intuitivement, cela correspond au fait qu'un participant ne peut envoyer, ou déclarer, que des messages qu'il sait construire. Nous ne considérerons ainsi que des protocoles déterministes. Les conditions 3a et 3c restreignent les protocoles aux séquences d'événements dans lesquelles chaque envoi correspond à une réception (et *vice versa*). Cela permet en particulier d'exclure un certain nombre de « protocoles » qui n'admettent pas d'« exécution normale », notamment dû au fait que certaines réceptions ne correspondent à aucun envoi. Nous verrons au chapitre 4 un exemple (voir figure 4.1) d'un tel « protocole », et tenterons d'expliquer en quoi est-il légitime de l'exclure de notre modèle.

Remarquons que la condition 2 impose que le premier argument d'un status event soit un participant, ceci n'est pas sans rappeler la remarque que nous faisions un peu plus haut sur la restriction d'une séquence d'événements à un participant. En effet, afin que la condition 1 et la notion de « rôle » que nous verrons un peu plus loin dans cette section, soient consistante nous avons besoin que l'initiateur de chaque status event soit unique et déterminé. Afin donc de pouvoir déterminer l'initiateur de chaque status event, nous demandons que celui-ci apparaisse comme le premier argument du status event considéré. C'est pour retrouver toute l'expressivité des status events que nous supprimons

finalement le premier argument de chaque status event dans la restriction de Π à un participant. Une fois l'initiateur identifié, il n'est plus nécessaire que son identité apparaisse parmi les arguments du status event.

Définition 2.4.3 ($\mathsf{B}(\Pi)$)**.** *Soit Π un protocole, la taille de Π, dénotée $\mathsf{B}(\Pi)$, est définie comme indiqué ci-dessous*

$$\mathsf{B}(\Pi) \stackrel{def}{=} \max\{|\delta_\Pi(t)| \mid t \in \mathsf{St}(\Pi)\}.$$

Exemple 2.4.4. *Le protocole présenté informellement en introduction comme la séquence d'envois*

$$\begin{array}{lcll} a & \rightarrow & b: & \{\{n\}_b, a\}_b \\ b & \rightarrow & a: & \{\{n\}_a, b\}_a \end{array}$$

se formalise dans notre modèle par la séquence d'événements suivante :

$$\begin{array}{rl} \Pi_{\mathsf{Toy}} = [& \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n, b), a\rangle, b)); \\ & \mathsf{rcv}(b, a, \mathsf{aenc}(\langle \mathsf{aenc}(x, b), a\rangle, b)); \\ & \mathsf{snd}(b, a, \mathsf{aenc}(\langle \mathsf{aenc}(x, a), b\rangle, a)); \\ & \mathsf{rcv}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n, a), b\rangle, a)) \quad] \end{array}$$

La taille de ce protocole est $\mathsf{B}(\Pi_{\mathsf{Toy}}) = 7$ et la substitution honnête est d'après la définition 2.4.2

$$\delta_{\Pi_{\mathsf{Toy}}} = \{x \mapsto n\}.$$

Définition 2.4.5 (Rôle)**.** *Soit Π un protocole. Les rôles de Π sont les participants $p \in \mathcal{P}$ actifs dans Π,* i.e. *initiant au moins un événement dans Π :*

$$\mathsf{Roles}(\Pi) \stackrel{def}{=} \{r_p \mid p \in \mathcal{P},\ \Pi(p) \neq []\}.$$

Soit Π un protocole, nous appellerons *corps du rôle* $r \in \mathsf{Roles}(\Pi)$ la restriction de Π à r, soit la séquence d'événements $\Pi(r)$.

Exemple 2.4.6. *Les deux rôles du protocoles de l'exemple 2.4.4 sont r_a et r_b avec :*

$$\begin{array}{rl} \Pi_{\mathsf{Toy}}(r_a) = [& \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n, b), a\rangle, b)); \\ & \mathsf{rcv}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n, a), b\rangle, a)) \quad] \end{array}$$

$$\begin{array}{rl} \Pi_{\mathsf{Toy}}(r_b) = [& \mathsf{rcv}(b, a, \mathsf{aenc}(\langle \mathsf{aenc}(x, b), a\rangle, b)); \\ & \mathsf{snd}(b, a, \mathsf{aenc}(\langle \mathsf{aenc}(x, a), b\rangle, a)) \quad] \end{array}$$

2.5 L'intrus

Les communications ayant habituellement lieu sur un support public (*e.g.* Internet), et ayant pour objectif la vérification, il est raisonnable de supposer qu'elles ont lieu en présence d'un attaquant (*intrus*) cherchant à exploiter les éventuelles failles du protocole. Il nous faut donc inclure dans notre modèle une description formelle des capacités de cet attaquant, ce à quoi est dédiée cette section.

L'intrus que nous considérons dans ce travail contrôle entièrement le réseau, d'où le nom d'*intrus actif* consacré. Il peut en effet écouter les messages émis par les participants, mais il peut aussi les intercepter, ainsi qu'en construire et émettre de nouveaux en utilisant sa propre identité ou en usurpant celle d'un autre participant. Comme nous le verrons à la section 2.6, la modélisation d'un tel intrus nous amènera à considérer que *(i)* tout envoi est intercepté par l'intrus, et que *(ii)* tout message m reçu par un participant a été « synthétisé » par l'intrus.

2.5.1 Les capacités de l'intrus

Nous en venons ainsi à la capacité de l'intrus à synthétiser un message u à partir d'un ensemble de messages T qu'il connaît déjà, dénoté par le séquent $T \vdash u$. Celle-ci est modélisée par un ensemble de règles de déduction décrit à la Figure 2.1. Cette modélisation de l'intrus étant due à D. Dolev et A. C. Yao [30], il est souvent question de l'« intrus de Dolev-Yao ».

Les cinq premières règles décrivent les règles de composition. Intuitivement, elles stipulent que l'intrus peut combiner deux messages u et v qu'il connaît en les concaténant, en chiffrant symétriquement ou asymétriquement le premier avec le second, ou encore en signant le premier avec le second. Aussi l'intrus peut appliquer une fonction de hachage à un message u qu'il connaît déjà.

Inversement, l'intrus peut sous certaines conditions décomposer des messages qu'il connaît. Il peut en effet, projeter une paire sur une de ses deux composantes, mais aussi déchiffrer un message chiffré symétriquement ou asymétriquement à condition de connaître la clé de déchiffrement correspondante. Concernant la signature, l'intrus peut vérifier que le message $\mathsf{sign}(u, v)$ correspond bien à la signature du message u à condition de connaître la clé de vérification v^{-1} correspondant à la clé de signature v. Ceci ne génère pas pour autant un message et ne nécessite donc pas d'être représenté par une règle de déduction. La règle dite de vérification stipule que l'intrus peut recouvrer un message u à partir du message $\mathsf{sign}(u, v)$. Ceci dépend du schéma de signature considéré et c'est pourquoi cette règle est optionnelle. Les résultats présentés dans la suite de cette thèse restent vrai indépendemment de la prise en considération ou non de cette règle de déduction. L'intrus n'a par contre aucun moyen, comme nous l'expliquions à la section 2.2, d'inverser le hachage. D'où l'absence de règle de décomposition pour le symbole de fonction h.

L'axiome décrit le fait que l'intrus connaît tous les termes de T.

Les deux dernières règles munissent l'intrus des constantes et clés nécessaires pour incarner, comme tout autre agent, les rôles d'un protocole. L'ensemble $\mathcal{K}_\epsilon$ y dénote l'ensemble de clés long-terme de l'intrus, et est défini par $\mathcal{K}_\epsilon = \{p, \mathsf{shk}(\epsilon, p) \mid p \in \mathcal{P}\} \cup \{\mathsf{pvk}(\epsilon)\}$. En particulier, l'intrus connaît les clés publiques long-terme de tous les participants, ainsi que les constantes (qui sont par définition des données publiques) ; il partage une clé symétrique long-terme avec chaque participant ; et connaît sa propre clé privée long-terme.

L'intrus peut combiner ces règles afin de synthétiser (nous dirons aussi « déduire », ou encore « dériver ») de nouveaux messages. Ceci est formellement décrit par la notion de *déductibilité* définie comme suit :

Définition 2.5.1 (Déductibilité, Dérivabilité)**.** *Soient un ensemble de termes*

Composition

Concaténation *(P)*	$\dfrac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle}$	Chiffrement symétrique *(Cs)*	$\dfrac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{senc}(u, v)}$
Chiffrement asymétrique *(Ca)*	$\dfrac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{aenc}(u, v)}$	Signature *(S)*	$\dfrac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{sign}(u, v)}$
Hachage *(H)*	$\dfrac{T \vdash u}{T \vdash \mathsf{h}(u)}$		

Décomposition

Projection gauche *(Pg)*	$\dfrac{T \vdash \langle u, v \rangle}{T \vdash u}$	Projection droite *(Pd)*	$\dfrac{T \vdash \langle u, v \rangle}{T \vdash v}$
Déchiffrement symétrique *(Ds)*	$\dfrac{T \vdash \mathsf{senc}(u, v) \quad T \vdash v}{T \vdash u}$	Déchiffrement asymétrique *(Da)*	$\dfrac{T \vdash \mathsf{aenc}(u, v) \quad T \vdash \mathsf{pvk}(v)}{T \vdash u}$
Vérification *(V)* (optionnelle)	$\dfrac{T \vdash \mathsf{sign}(u, v)}{T \vdash u}$		

Axiome

Axiome *(A)*	$\dfrac{}{T, u \vdash u}$

Connaissance Initiale Minimale

Constante *(C)*	$\dfrac{}{T \vdash u}\ u \in \mathcal{C}$	Clé long-terme *(Ltk)*	$\dfrac{}{T \vdash u}\ u \in \mathcal{K}_\epsilon$

Fig. 2.1: Règles de déduction de l'intrus

$U \subseteq \mathcal{T}$ et un terme $u \in \mathcal{T}$, u est déductible (ou encore dérivable) par l'intrus à partir de U, noté $U \vdash u$, s'il existe un arbre dont les nœuds sont étiquetés par des séquents de la forme $U \vdash t$ et tels que :

- *la racine est étiquetée par le séquent $U \vdash u$,*
- *pour tout nœud intermédiaire étiqueté $U \vdash v$ avec pour fils des nœuds étiquetés $U \vdash v_1 \ \ldots \ U \vdash v_n$, il existe une substitution σ et une règle de la Figure 2.1 $\dfrac{T \vdash w_1 \ \ldots \ T \vdash w_n}{T \vdash w}$ tels que $v_i = w_i\sigma$ pour tout $i \in [\![n]\!]$, et $v = w\sigma$,*

– *toute feuille est étiquetée par un séquent* $U \vdash v$ *avec* $v \in (U \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$.

Exemple 2.5.2. *Soit* T_0 *un ensemble de termes clos quelconque et* $T_1 \stackrel{def}{=} T_0 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)\}$*, l'arbre de déduction ci-dessous témoigne du fait que :*

$$T_1 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b), \epsilon\rangle, b),$$

$$\dfrac{\dfrac{\dfrac{}{T_1 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)} \quad \dfrac{}{T_1 \vdash \epsilon}}{T_1 \vdash \langle \mathsf{aenc}(\langle \mathsf{aenc}(n,^1 b), a\rangle, b), \epsilon\rangle} \quad \dfrac{}{T_1 \vdash b}}{T_1 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b), \epsilon\rangle, b)}$$

De même, soit $T_2 \stackrel{def}{=} T_1 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, \epsilon), b\rangle, \epsilon)\}$*, l'arbre de déduction ci-dessous témoigne du fait que :*

$$T_2 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), \epsilon\rangle, b)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{T_2 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, \epsilon), b\rangle, \epsilon)} \quad \dfrac{}{T_2 \vdash \mathsf{pvk}(\epsilon)}}{T_2 \vdash \langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, \epsilon), b\rangle}}{T_2 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, \epsilon)} \quad \dfrac{}{T_2 \vdash \mathsf{pvk}(\epsilon)}}{T_2 \vdash \langle \mathsf{aenc}(n^1, b), a\rangle}}{T_2 \vdash \mathsf{aenc}(n^1, b)} \quad \dfrac{}{T_2 \vdash \epsilon}}{T_2 \vdash \langle \mathsf{aenc}(n^1, b), \epsilon\rangle} \quad \dfrac{}{T_2 \vdash b}}{T_2 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), \epsilon\rangle, b)}$$

Et, soit $T_3 \stackrel{def}{=} T_2 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(n^1, \epsilon), b\rangle, \epsilon)\}$*, l'arbre de déduction ci-dessous témoigne du fait que :* $T_3 \vdash n^1$*,*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{T_3 \vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1, \epsilon), b\rangle, \epsilon)} \quad \dfrac{}{T_3 \vdash \mathsf{pvk}(\epsilon)}}{T_3 \vdash \langle \mathsf{aenc}(n^1, \epsilon), b\rangle}}{T_3 \vdash \mathsf{aenc}(n^1, \epsilon)} \quad \dfrac{}{T_3 \vdash \mathsf{pvk}(\epsilon)}}{T_3 \vdash n^1}$$

2.5.2 Quelques lemmes

Dans cette section, nous rappelons quelques lemmes bien connus sur la relation de déductibilité. Le premier de ces lemmes stipule que la relation de déductibilité est stable par application d'une substitution.

Lemme 2.5.3. *Soit* T *un ensemble de termes,* t *un terme et* σ *une substitution.*

$$\textit{Si } T \vdash t, \textit{ alors } T\sigma \vdash t\sigma.$$

Ce deuxième lemme établit la propriété « d'élimination des coupures » pour cette même relation de déductibilité.

Lemme 2.5.4. *Soient un ensemble de termes T et deux termes u et v.*

$$\textit{Si } T \vdash u \textit{ et } T, u \vdash v, \textit{ alors } T \vdash v.$$

En conséquence, si $T \subseteq T'$, $T' \vdash v$ et $T \vdash u$ pour tout $u \in T' \setminus T$, alors $T \vdash v$.

Nous rappelons aussi le lemme de localité, selon lequel s'il existe une preuve de $T \vdash u$, alors il en existe une ne faisant intervenir que des sous-termes de T et u.

Lemme 2.5.5. *Soient un ensemble de termes T et deux termes u. Si $T \vdash u$, alors il existe une preuve dont tout les nœuds sont de la forme $T \vdash v$ avec $v \in (\mathsf{St}(T \cup \{u\}) \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$. De plus, si la dernière règle appliquée est une des règles de décomposition ou l'axiome, alors $v \in (\mathsf{St}(T) \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$.*

Ce dernier lemme que nous énoncerons dans cette section, établit que si un terme t est déductible à partir d'un ensemble de termes T, alors les sous-termes qui apparaissent en position de plaintext dans t, apparaissent aussi en position de plaintext dans T.

Lemme 2.5.6. *Soient un ensemble de termes T et un terme t. Si $T \vdash t$, alors*

$$\mathsf{Plaintext}(t) \subseteq (\mathsf{Plaintext}(T) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Démonstration. Nous procédons par induction sur la profondeur de l'arbre π qui témoigne de $T \vdash t$.

Si la dernière règle appliquée dans π est l'axiome, alors $t \in T$, d'où $\mathsf{Plaintext}(t) \subseteq T$.

Si la dernière règle appliquée dans π est une de celles décrivant la connaissance initiale minimale de l'intrus, alors

$$\mathsf{Plaintext}(t) = t \in \mathcal{C} \cup \mathcal{K}_\epsilon \subseteq (\mathsf{Plaintext}(T) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Si la dernière règle appliquée dans la dérivation π est une règle de composition ou de décomposition, *i.e.* π est de la forme

$$\dfrac{\dfrac{\pi_1}{T \vdash t_1} \quad \ldots \quad \dfrac{\pi_n}{T \vdash t_n}}{T \vdash t}$$

alors au regard ces règles nous concluons que $\mathsf{Plaintext}(t) \subseteq \bigcup_{i \in [\![n]\!]} \mathsf{Plaintext}(t_i)$. Or, par hypothèse d'induction, pour tout $i \in [\![n]\!]$,

$$\mathsf{Plaintext}(t_i) \subseteq (\mathsf{Plaintext}(T) \cup \mathcal{C} \cup \mathcal{K}_\epsilon),$$

et donc

$$\mathsf{Plaintext}(t) \subseteq \bigcup_{i \in [\![n]\!]} \mathsf{Plaintext}(t_i) \subseteq (\mathsf{Plaintext}(T) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Ce qui achève notre induction. □

2.6 Modèle d'exécution des protocoles de sécurité

A la section 2.4 nous formalisions la notion de protocole. C'est à la formalisation de l'exécution d'un protocole que nous nous attelons maintenant. Le passage de la description d'un protocole à la description d'une exécution de ce dernier est un peu délicat. En effet, un protocole tel que défini à la section 2.4 (voir définition 2.4.2) « fixe » le nombre d'exécutions de chaque rôle à un, impose l'ordre d'exécution des événements de ces rôles, et détermine complètement les agents incarnant chaque rôle ainsi que les nonces engendrés par chacun de ces rôles. Néanmoins, un rôle peut être exécuté un nombre arbitraire de fois par une entité quelconque de $\mathcal{P}$, engendrant à chaque fois un ensemble de nonces différents, *frais*. Aussi le réseau étant entièrement sous le contrôle de l'intrus, l'ensemble des rôles d'une exécution peuvent être entrelacés de différentes façons. Ce sont ces aspects dynamiques introduits lors de l'exécution d'un protocole (par opposition à sa description purement statique) que nous allons à présent expliciter afin de définir le modèle d'exécution de nos protocoles.

Nous appellerons *session* une instantiation (partielle) d'un rôle. La première notion introduite est celle de scénario. Cette notion capture *(i)* l'entrelacement d'un nombre arbitraire de sessions, ainsi que *(ii)* la détermination des participants de chaque session du point de vue de l'entité incarnant effectivement le rôle exécuté par cette session.

Définition 2.6.1 (Scénario)**.** *Soit* Π *un protocole. Un scénario de* Π *est un couple* $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ *tel que :*

- $\mathsf{interlvg}$ *est une séquence* $[(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$*, avec pour tout* $i, j \in [\![\ell]\!]$
  1. $r_i \in \mathsf{Roles}(\Pi)$,
  2. $sid_i \in (\mathbb{N} \setminus \{0\})$,
  3. $\mathsf{Occ}(\mathsf{interlvg}, (r_i, sid_i)) \leq |r_i|$,
  4. *si* $sid_i = sid_j$*, alors* $r_i = r_j$ *;*
- $\mathsf{initagts} : (\mathbb{N} \setminus \{0\}) \to \mathcal{A}^k$ *avec* $k = |\mathcal{A}(\Pi)|$.

Les sid_i (avec $i \in [\![\ell]\!]$) sont des identificateurs de sessions. L'entrelacement $\mathsf{interlvg}$ fixe les sessions impliquées dans sc et le rôle joué par chacune. Ainsi, la condition 4 assure que chaque session n'incarne qu'un seul rôle. Chaque rôle ayant une longueur fixée par la spécification du protocole, la condition 3 assure qu'une session n'initie pas plus d'événements que le nombre d'événements dans le corps du rôle qu'elle incarne. La fonction $\mathsf{initagts}$ fixe les participants effectifs de chaque session. En effet, chaque rôle pouvant être incarné par une entité différente, ce sont tous les agents du protocole qui peuvent être différents, du point de vue d'une session, de ceux spécifiés dans la description du protocole. Les serveurs sont par contre fixés dès la spécification et sont les mêmes pour toute session d'un même rôle. Telle que définie, la notion de scénario autorise les participants à se parler à eux-mêmes, en ce sens qu'il est possible pour une session sid_i (avec $i \in [\![\ell]\!]$) d'avoir $\mathsf{initagts}(sid_i) = (a_1^{sid_i}, \ldots, a_k^{sid_i})$ avec les $a_j^{sid_i}$ non tous distincts. Il est possible d'ajouter sur la fonction $\mathsf{initagts}$ la condition supplémentaire suivante

– $\forall i \in [\![\ell]\!].\ \forall j, h \in [\![k]\!].\ \mathsf{initagts}(sid_i) = (a_1^{sid_i}, \ldots, a_k^{sid_i}) \wedge j \neq h \ \Rightarrow\ a_j^{sid_i} \neq a_h^{sid_i}$,

afin d'empêcher tout participant d'un scénario de se parler à lui-même. Les résultats présentés dans la suite de ce travail sont indépendants du choix fait concernant la possibilité pour les participants de se parler à eux-mêmes, et c'est la raison pour laquelle cette condition n'apparaît pas dans la définition de scénario.

Soient Π un protocole, $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ un scénario de Π avec pour entrelacement $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, et $S \subseteq \mathbb{N}^*$ un ensemble fini d'identificateurs de sessions. La *restriction de* sc *aux sessions de* S, dénotée $\mathsf{sc}|_S$ est définie par $\mathsf{sc}|_S \stackrel{def}{=} (\mathsf{interlvg}|_S, \mathsf{initagts})$ où

$$\mathsf{interlvg}|_S = \begin{cases} [] & \text{si } \ell = 0 \\ [(r_1, sid_1)]@(\mathsf{interlvg}'|_S) & \text{si } sid_1 \in S \\ (\mathsf{interlvg}'|_S) & \text{sinon} \end{cases}$$

où $\mathsf{interlvg}' = [(r_2, sid_2); \ldots; r_\ell, sid_\ell)]$.

Soit un entrelacement $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_n, sid_n)]$, $\mathsf{Sess}(\mathsf{interlvg})$ *dénote la séquence ordonnée (dans l'ordre d'apparition dans* $\mathsf{interlvg}$*) et sans répétitions des couples* (r, sid) *apparaissant dans* $\mathsf{interlvg}$, *i.e*

$$\mathsf{Sess}(\mathsf{interlvg}) \stackrel{def}{=} \ltimes_{i \in [\![n]\!]} [(r_i, sid_i)].$$

Exemple 2.6.2. $\mathsf{sc}_{\mathsf{Toy}} = (\mathsf{interlvg}_{\mathsf{Toy}}, \mathsf{initagts}_{\mathsf{Toy}})$ *avec :*

– $\mathsf{interlvg}_{\mathsf{Toy}} = [(r_a, 1); (r_b, 2); (r_b, 2); (r_b, 3); (r_b, 3)]$, *et*
– $\mathsf{initagts}_{\mathsf{Toy}}(1) = (a, b)$,
  $\mathsf{initagts}_{\mathsf{Toy}}(2) = (\epsilon, b)$,
  $\mathsf{initagts}_{\mathsf{Toy}}(3) = (\epsilon, b)$

est un scénario pour le protocole de l'exemple 2.4.4, et

$$\mathsf{Sess}(\mathsf{interlvg}_{\mathsf{Toy}}) = [(r_a, 1); (r_b, 2); (r_b, 3)].$$

Comme nous l'expliquions, chaque session d'un rôle engendre un ensemble frais de nonces, et chaque rôle peut être incarné par une entité différente de celle spécifiée dans la description du protocole. Ceci est formalisé par la fonction d'initialisation d'une session $\mathsf{init}_{r,sid}$ que nous définissons à présent.

Définition 2.6.3 (Fonction $\mathsf{init}_{r,sid}$)**.** *Soient un protocole* Π *et un rôle* $r \in \mathsf{Roles}(\Pi)$ *tels que :*

– $[a_1; \ldots; a_k] = \mathsf{Agts}(\Pi)$ *les agents du protocole* Π,
– $[x_1; \ldots; x_n] = \mathsf{Vars}(\Pi(r))$ *les variables de* $\Pi(r)$, *et*
– $[n_1; \ldots; n_m] = \mathsf{Nces}(\Pi(r))$ *les nonces de* $\Pi(r)$.

Soit un numéro de session $sid \in (\mathbb{N} \setminus \{0\})$, *et une fonction d'initialisation des participants* $\mathsf{initagts} : (\mathbb{N} \setminus \{0\}) \to \mathcal{A}^k$ *avec :*

– $\mathsf{initagts}(sid) = (a_1^{sid}, \ldots, a_k^{sid})$.

Soient aussi un ensemble fini de variables $V \subseteq \mathcal{V}$ *et un ensemble fini de nonces* $N \subseteq \mathcal{N}$. *Puis, soient*
- n *variables fraîches* $(x_1^{sid}, \ldots, x_n^{sid}) \in (\mathcal{V} \setminus V)^n$ *et distinctes, et*
- m *nonces frais* $(n_1^{sid}, \ldots, n_m^{sid}) \in (\mathcal{N} \setminus N)^m$ *et distincts.*

La fonction $\mathsf{init}_{r,sid}$ d'initialisation de *sid* jouant le rôle r associée à $\mathsf{initagts}$, N et V *est définie sur les atomes par*

- $\mathsf{init}_{r,sid}(a_i) = a_i^{sid}$ *pour tout* $i \in [\![k]\!]$,
- $\mathsf{init}_{r,sid}(s) = s$, *pour tout* $s \in \mathcal{S}$,
- $\mathsf{init}_{r,sid}(c) = c$, *pour tout* $c \in \mathcal{C}$,
- $\mathsf{init}_{r,sid}(n_i) = n_i^{sid}$ *pour tout* $i \in [\![m]\!]$,
- $\mathsf{init}_{r,sid}(x_i) = x_i^{sid}$ *pour tout* $i \in [\![n]\!]$.

et s'étend aux termes composés et aux événements de façon standard.

A chaque événement de réception, seules les variables de la session ayant initié cet événement devraient être liées. Afin de s'en assurer la fonction $\mathsf{init}_{r,sid}$ engendre des variables fraîches pour la session *sid*, évitant ainsi toute collision entre variables de sessions distinctes. Intuitivement, les ensembles V et N de cette définition modélisent les variables et les nonces non-frais, engendrés au cours d'autres sessions que *sid*. Ainsi, les n variables et les m nonces engendrés par la fonction $\mathsf{init}_{r,sid}$ pour la session *sid* sont sélectionnés parmi $(\mathcal{V} \setminus V)$ et $(\mathcal{N} \setminus N)$ respectivement, et sont donc frais.

Notons que d'après la définition 2.6.1 de scénario, l'intrus peut lui aussi incarner n'importe lequel des rôles du protocole considéré (à condition que ce ne soit pas un rôle dont l'initiateur est un serveur). Mais il peut aussi engendrer des nonces, sans même avoir à initier de session (et donc sans passer par la fonction d'initialisation $\mathsf{init}_{r,sid}$ où *sid* serait une session dont l'intrus serait l'initiateur). Afin de modéliser cette aptitude de l'intrus nous définissons l'ensemble de nonces $\mathcal{N}_\epsilon(\Pi)$ définit comme suit.

Définition 2.6.4 ($\mathcal{N}_\epsilon(\Pi)$, $\mathsf{Nces}_\epsilon(\Pi)$)**.** *Soit* Π *un protocole avec* $\mathsf{Nces}(\Pi) = [n_1; \ldots; n_m]$. *Soient aussi* m *nonces frais* $(n_1^\epsilon, \ldots, n_m^\epsilon) \in (\mathcal{N} \setminus \{n_1, \ldots, n_m\})^m$ *distincts.*

$$\mathcal{N}_\epsilon = \{n_1^\epsilon, \ldots, n_m^\epsilon\} \qquad et \qquad \mathsf{Nces}_\epsilon = [n_1^\epsilon; \ldots; n_m^\epsilon].$$

A chaque nonce de Π nous faisons correspondre un nonce frais dans $\mathcal{N}_\epsilon(\Pi)$.

Etant donné un scénario, nous définissons à présent la séquence d'événements correspondant à l'entrelacement du scénario, telle que les nonces et les variables de chaque session du scénario soient frais et les participants de chaque session correspondent à ceux spécifiés par le scénario.

Définition 2.6.5 (Trace symbolique)**.** *Soient un protocole* Π *et un scénario* $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ *de* Π, *avec* $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$ *et* $\mathsf{Sess}(\mathsf{interlvg}) = [(r'_1, sid'_1); \ldots; (r'_n, sid'_n)]$. *Soit* $\mathsf{init}_{r'_i,sid'_i}$ *la fonction d'initiali-*

sation de sid'_i jouant le rôle r'_i associée à $\mathsf{initagts}$*, V_i et N_i, avec*

$$\begin{cases} V_1 = \mathcal{V}(\Pi) \\ V_i = V_{i-1} \cup \bigcup\limits_{x \,\in\, \mathcal{V}(\Pi(r'_{i-1}))} \{\mathsf{init}_{r'_{i-1},sid'_{i-1}}(x)\}, \text{ et} \end{cases}$$

$$\begin{cases} N_1 = \mathcal{N}(\Pi) \cup \mathcal{N}_\epsilon(\Pi) \\ N_i = N_{i-1} \cup \bigcup\limits_{n \,\in\, \mathcal{N}(\Pi(r'_{i-1}))} \{\mathsf{init}_{r'_{i-1},sid'_{i-1}}(n)\}, \end{cases}$$

pour tout $i \in [\![n]\!]$. La trace symbolique associée à sc *est la séquence d'événements $tr = [e_1; \ldots; e_\ell]$, où $e_i = \mathsf{init}_{r_i,sid_i}(\Pi(r_i)[q_i])$ avec $q_i = \mathsf{Occ}(tr_i, (r_i, sid_i))$ pour tout $i \in [\![\ell]\!]$.*

Exemple 2.6.6. *La trace symbolique associée au scénario de l'exemple 2.6.2 est la suivante :*

$$\begin{aligned} tr_{\mathsf{Toy}} \quad = [\ & \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)); \\ & \mathsf{rcv}(\epsilon, b, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, b), \epsilon\rangle, b)); \\ & \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, \epsilon), b\rangle, \epsilon)); \\ & \mathsf{rcv}(\epsilon, b, \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b)); \\ & \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^3, \epsilon), b\rangle, \epsilon)) \quad] \end{aligned}$$

Soit Π un protocole, $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ un scénario de Π avec pour entrelacement $[(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, $tr = [e_1; \ldots; e_\ell]$ la trace symbolique associée à sc, et S un ensemble d'identificateurs de sessions. Nous noterons $tr|_S$ la trace symbolique associée à $\mathsf{sc}|_S$. Il est aisé de voir que

$$tr|_S = \begin{cases} [] & \text{si } \ell = 0 \\ [e_1]@(tr'|_S) & \text{si } sid_1 \in S \\ (tr'|_S) & \text{sinon} \end{cases}$$

où $tr' = [e_2; \ldots; e_\ell]$.

Soit Π un protocole. Une *trace* de Π est une instantiation d'une trace symbolique associée à un scénario de Π, *i.e* une séquence d'événements tr est une trace s'il existe une substitution σ, ainsi qu'une trace symbolique tr' associée à un scénario de Π, telles que $tr = tr'\sigma$.

Toutes les traces de Π ne correspondent pas à des exécutions valides du protocole. Soit T_0 la connaissance initiale de l'intrus telle que $n^1 \notin T_0$. Reprenons la trace symbolique tr_{Toy}, et considérons la substitution $\sigma = \{x^2 \mapsto n^1; x^3 \mapsto n^1\}$. La trace

$$\begin{aligned} tr_{\mathsf{Toy}}\sigma \quad = [\ & \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)); \\ & \mathsf{rcv}(\epsilon, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), \epsilon\rangle, b)); \\ & \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, \epsilon), b\rangle, \epsilon)); \\ & \mathsf{rcv}(\epsilon, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), \epsilon\rangle, b)); \\ & \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, \epsilon), b\rangle, \epsilon)) \quad] \end{aligned}$$

n'est pas valide. En effet, aucun agent n'émet le message $\mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), \epsilon\rangle, b)$, et l'intrus n'a aucun moyen de construire ce message par lui-même à partir des messages qu'il connaît déjà (*i.e.* $T_0 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)\}$). Le deuxième événement de $tr_{\mathsf{Toy}}\sigma$ ne peut donc pas se produire. Afin de définir formellement la notion d'exécution valide, nous introduisons un opérateur K qui associe à une trace la connaissance, l'ensemble de termes pas forcément clos, connu de l'intrus.

Définition 2.6.7 (Opérateur K)**.** *Soit* Π *un protocole et* $tr = [e_1; \ldots; e_\ell]$ *une trace de* Π*. La connaissance associée à cette trace est notée* $\mathsf{K}(tr)$ *et définie inductivement par :*

$$\mathsf{K}(tr) = \begin{cases} [] & \text{si } \ell = 0 \\ \{m\} \cup \mathsf{K}(tr') & \text{si } \ell > 2 \quad \text{et } e_1 = \mathsf{snd}(p, p', m), \\ \mathsf{K}(tr') & \text{sinon} \end{cases}$$

avec $tr' = [e_2; \ldots; e_\ell]$*,* $p, p' \in \mathcal{P}$*, et* $m \in \mathcal{T}$*.*

Définition 2.6.8 (Trace valide, Exécution valide)**.** *Soit* Π *un protocole,* $tr = [e_1; \ldots; e_\ell]$ *une trace de* Π*, et* T_0 *un ensemble de termes clos.* tr *est une trace valide de* Π*,* au regard de la connaissance initiale de l'intrus T_0 *si pour tout* $i \in [\![\ell]\!]$*,*

$$\text{si } e_i = \mathsf{rcv}(p, p', m), \text{ alors } T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup K(tr_i) \vdash m.$$

Si de plus tr *est close,* i.e. $\mathcal{V}(tr) = \emptyset$*, alors* tr *est une exécution valide de* Π *au regard de la connaissance initiale de l'intrus* T_0*.*

Cette définition d'exécution valide modélise bien le contrôle du réseau par l'intrus. En effet, chaque réception $\mathsf{rcv}(p, p', m)$ est telle que m est déductible par l'intrus à partir des messages déjà émis, de sa connaissance initiale, et de l'ensemble de nonces $\mathcal{N}_\epsilon(\Pi)$. Nous retrouvons bien ce que nous décrivions à la section 2.5 : *(i)* tout envoi est intercepté par l'intrus, et *(ii)* tout message m reçu par un participant a été synthétisé par l'intrus. En munissant l'intrus de l'ensemble $\mathcal{N}_\epsilon(\Pi)$, et en le mettant à sa disposition dans la construction de m, nous décrivons donc effectivement sa capacité à engendrer ses propres nonces[4].

Exemple 2.6.9. *Considérons la substitution* σ *telle que :*

- $\sigma(x^2) = \langle \mathsf{aenc}(n^1, b), a\rangle$*, et*
- $\sigma(x^3) = n^1$

et la trace symbolique tr_{Toy} *de l'exemple 2.6.6,*

$$\begin{aligned} \mathsf{exec} = tr_{\mathsf{Toy}}\sigma \quad = [\ & \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)); \\ & \mathsf{rcv}(\epsilon, b, \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b), \epsilon\rangle, b)); \\ & \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, \epsilon), b\rangle, \epsilon)); \\ & \mathsf{rcv}(\epsilon, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), \epsilon\rangle, b)); \\ & \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, \epsilon), b\rangle, \epsilon)) \qquad\qquad] \end{aligned}$$

[4]Nous aurions pu munir l'intrus d'un ensemble infini de nonces, mais cela n'augmenterait pas sa capacité à monter des attaques contre Π, et augmenterait qui plus est la complexité de la suite de notre exposé.

est une exécution valide du protocole Π_{Toy} *défini à l'exemple 2.4.4, et ce quel que soit la connaissance initiale de l'intrus* T_0. *En effet, nous avons vu à l'exemple 2.5.2 que*

$$T_0 \cup \mathsf{K}(\mathsf{exec}_2) \vdash \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b), \epsilon\rangle, b),$$

ainsi que

$$T_0 \cup \mathsf{K}(\mathsf{exec}_4) \vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), \epsilon\rangle, b)$$

pour tout T_0.

Lemme 2.6.10. *Soient* $\Pi = [e_1; \ldots; e_k]$ *un protocole et* T_0 *un ensemble de termes clos. Soient aussi* sc *un scénario* Π *de trace symbolique associée* tr, *et* σ *une substitution close telle que* $\mathsf{exec} = tr\sigma$ *soit une exécution valide de* Π, *au regard de la connaissance initiale de l'intrus* T_0. *Si* Π *satisfait la condition suivante :*

> *pour tout* $i \in [\![k]\!]$ *et pour toute variable* $x \in \mathcal{V}$, *si* e_i *est une émission ou un status event initié par un participant* $p \in \mathcal{P}$, *et* $x \in \mathsf{Plaintext}(e_i)$, *alors il existe* $j \in [\![i-1]\!]$ *tel que* e_j *soit une réception initiée par ce même participant* p *et* $x \in \mathsf{Plaintext}(e_j)$,

alors

$$\mathsf{Plaintext}(\mathsf{exec}) \subseteq (\mathsf{Plaintext}(tr) \cup T_0 \cup \mathcal{N}_\Pi(\epsilon) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Démonstration. Posons $\mathsf{exec} = [e'_1; \ldots; e'_\ell]$, et $tr = [e''_1; \ldots; e''_\ell]$. Nous procédons par induction sur la longueur ℓ de l'exécution exec.

Cas de base : $\ell = 0$. Alors $\mathsf{Plaintext}(\mathsf{exec}) = \emptyset$, et donc $\mathsf{Plaintext}(\mathsf{exec}) \subseteq (\mathsf{Plaintext}(tr) \cup T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$.

Cas inductif : $\ell \geq 1$. Par définition que $\mathsf{exec}_{\ell-1}$ est aussi une exécution valide de $\Pi_{\mathcal{P}}$ au regard de la connaissance initiale de l'intrus T_0. Nous pouvons donc appliquer notre hypothèse d'induction et conclure que

$$\mathsf{Plaintext}(\mathsf{exec}_{\ell-1}) \subseteq (\mathsf{Plaintext}(tr) \cup T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Nous procédons par analyse de cas sur le type e'_n.

Cas e'_ℓ *est une réception.*
Il existe donc deux entités $p_1, p_2 \in \mathcal{P}$ et un terme $t \in \mathcal{T}$ tels que $e'_\ell = \mathsf{rcv}(p_1, p_2, t)$. Par définition d'une exécution valide nous savons que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}_{\ell-1}) \vdash t$, et par définition de la fonction $\mathsf{Plaintext}()$, nous savons aussi que $\mathsf{Plaintext}(\mathsf{exec}) = \mathsf{Plaintext}(\mathsf{exec}_{\ell-1}) \cup \mathsf{Plaintext}(t)$. Récapitulons

$$\begin{array}{lll}
\mathsf{Plaintext}(t) & \subseteq & (\mathsf{Plaintext}(\mathsf{K}(\mathsf{exec}_{\ell-1})) \cup T_0 \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon) \qquad \text{lemme 2.5.6} \\
& \subseteq & (\mathsf{Plaintext}(\mathsf{exec}_{\ell-1}) \cup T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon) \qquad \text{définition de K()} \\
& \subseteq & (\mathsf{Plaintext}(tr) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon) \qquad \text{hypothèse d'induction.}
\end{array}$$

Cas e'_ℓ *est une émission ou un status event.*
Les deux cas pouvant être traités de manière analogue, nous ne détaillons ici que le premier. Il existe donc deux entités $p, p' \in \mathcal{P}$ ainsi que deux termes $t, t' \in \mathcal{T}$

tels que $e'_\ell = \mathsf{snd}(p_1, p_2, t)$, $e''_\ell = \mathsf{snd}(p_1, p_2, t')$ et $t = t'\sigma$. Par définition de la fonction $\mathsf{Plaintext}()$, nous savons que

$$\mathsf{Plaintext}(t) \subseteq (\mathsf{Plaintext}(t') \cup \bigcup_{x \in (\mathsf{Plaintext}(t') \cap \mathcal{V}(t'))} \mathsf{Plaintext}(\sigma(x))).$$

Or, par définition de $\mathsf{Plaintext}(tr)$, nous savons aussi que

$$\mathsf{Plaintext}(t') \subseteq \mathsf{Plaintext}(tr) \subseteq (\mathsf{Plaintext}(tr) \cup T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Il nous reste donc à montrer que

$$\bigcup_{x \in (\mathsf{Plaintext}(t') \cap \mathcal{V}(t'))} \mathsf{Plaintext}(\sigma(x)) \subseteq (\mathsf{Plaintext}(tr) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Soit $y \in (\mathsf{Plaintext}(t') \cap \mathcal{V}(t'))$. Par hypothèse sur Π nous savons qu'il existe une réception antérieure à e''_ℓ initiée par p_1 et dans laquelle y apparaît en position de plaintext, *i.e ;* il existe $i \in [\![\ell - 1]\!]$ tel que $e''_i = \mathsf{rcv}(p_1, p_3, u)$ pour une certaine entité $p_3 \in \mathcal{P}$ et un certain terme u, et tel que $y \in \mathsf{Plaintext}(u)$. D'où,

$$\mathsf{Plaintext}(\sigma(y)) \subseteq \mathsf{Plaintext}(u\sigma) \subseteq \mathsf{Plaintext}(\mathsf{K}(\mathsf{exec}_i)) \subseteq \mathsf{Plaintext}(\mathsf{exec}_{\ell-1}).$$

Nous pouvons donc à nouveau faire appel à notre hypothèse d'induction et conclure que

$$\mathsf{Plaintext}(\sigma(y)) \subseteq (\mathsf{Plaintext}(tr) \cup T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon),$$

et donc que

$$\bigcup_{x \in (\mathsf{Plaintext}(t') \cap \mathcal{V}(t'))} \mathsf{Plaintext}(\sigma(x)) \subseteq (\mathsf{Plaintext}(tr) \cup T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

□

Nous en venons maintenant aux deux dernières définitions de ce chapitre. Elle nous seront très utile dès le chapitre suivant. Il s'agît de la définition d'un *participant compromis*, ainsi que de celle d'une *session compromise*.

Définition 2.6.11 (Participant compromis)**.** *Soient un ensemble de termes clos T_0 représentant la connaissance initiale de l'intrus, et un participant $p \in \mathcal{P}$. p est dit compromis au regard de la connaissance initiale de l'intrus T_0, si $T_0 \vdash \mathsf{pvk}(p)$ ou s'il existe $p' \in (\mathcal{P} \setminus \{\epsilon\})$ tels que $T_0 \vdash \mathsf{shk}(p, p')$.*

En d'autres termes, p est dit compromis au regard de la connaissance initiale de l'intrus T_0, si l'intrus connaît une des clés de déchiffrement long-terme. En particulier, l'intrus est compromis.

Définition 2.6.12 (Session compromise)**.** *Soient un protocole Π , un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de Π , et un ensemble de termes clos T_0. Une session sid est dite compromise au regard de la connaissance initiale de l'intrus T_0,*

- *si $\mathsf{initagts}(sid) = (a_1^{sid}, \ldots, a_n^{sid})$ et un des a_i^{sid} (avec $i \in [\![n]\!]$) est compromis,*
- *ou s'il existe un serveur $s \in \mathcal{P}(\Pi) \cap \mathcal{S}$ qui soit compromis.*

En d'autres termes la session *sid* est dite compromise au regard de la connaissance initiale de l'intrus T_0 si l'un de ses participants effectifs de cette session, agent ou serveur indifféremment, est compromis.

Chapitre 3

Modélisation des propriétés de sécurité : $\mathcal{PS}$-LTL

Il convient maintenant de décrire formellement les propriétés de sécurité que nous considérons dans ce travail. Ce chapitre y est consacré. Nous y présentons $\mathcal{PS}$-LTL, une logique du temps linéaire avec modalités du passé, introduite dans [20, 21], qui permet la spécification d'un grand nombre de propriétés de sécurité. Nous allons, dans un premier temps, introduire la logique : sa syntaxe (section 3.1.1) et sa sémantique (section 3.1.2). Nous verrons ensuite comment les propriétés usuelles, telles que le secret (section 3.2.1) et l'authentification (section 3.2.2), peuvent être exprimées dans cette logique.

3.1 Syntaxe et sémantique de $\mathcal{PS}$-LTL

3.1.1 La syntaxe de $\mathcal{PS}$-LTL

Définition 3.1.1 (Formules $\mathcal{PS}$-LTL)**.** *L'ensemble des formules de $\mathcal{PS}$-LTL est engendré par la grammaire suivante :*

$$\begin{array}{rcccccccccc} \phi & ::= & \mathsf{true} & | & \mathsf{Q}(t_1,\ldots,t_n) & | & \mathsf{learn}(t) & | & \mathsf{C}(u) & & \\ & | & \neg\phi & | & \phi \vee \phi & | & \mathcal{Y}\phi & | & \phi\mathcal{S}\phi & | & \exists x.\phi \end{array}$$

avec $t, t_1, \ldots, t_n \in \mathcal{T}$ tels que $\mathcal{N}(t) = \mathcal{N}(t_1) = \cdots = \mathcal{N}(t_n) = \emptyset$, et $u \in (\mathcal{P} \cup \mathcal{V})$.

true renvoie à la valeur de vérité *vrai*. La formule $\mathsf{Q}(t_1, \ldots, t_n)$ est un status event (*cf.* définition 2.4.1). $\mathsf{learn}(t)$ se lit de la façon suivante : « l'intrus peut déduire le terme t ». $\mathsf{C}(t)$ se lit : « le terme u est compromis ». Les connecteurs logiques standards ($\neg$, $\vee$) gardent leur interprétation usuelle. Sans entrer dans les détails de la sémantique, indiquons que $\mathcal{Y}\phi$ se lit « yesterday ϕ » et signifie que ϕ était satisfaite à l'« état précédent ». $\phi_1\mathcal{S}\phi_2$ se lit « ϕ_1 since ϕ_2 », et signifie qu'à un moment dans le passé ϕ_2 a été satisfaite, et que depuis, ϕ_1 est satisfaite. Pour $x \in \mathcal{V}$, $\exists x.\ \phi$ lie la variable x dans ϕ, ce quantificateur étant employé de la façon usuelle.

Exemple 3.1.2. *En guise d'exemple nous introduisons et définissons certaines notations qui nous seront très utiles par la suite, telles que la constante* false *renvoyant à la valeur de vérité* faux*, et* $\mathsf{NC}()$ *pour spécifier qu'un terme est non compromis. Nous définissons aussi les connecteurs usuels* $\wedge$ *et* $\Rightarrow$ *comme suit :*

$$\mathsf{false} \stackrel{def}{=} \neg\mathsf{true}, \qquad \mathsf{NC}(u) \stackrel{def}{=} \neg\mathsf{C}(u)$$
$$\phi_1 \wedge \phi_2 \stackrel{def}{=} \neg(\neg\phi_1 \vee \neg\phi_2), \qquad \phi_1 \Rightarrow \phi_2 \stackrel{def}{=} \neg\phi_1 \vee \phi_2,$$

ainsi que le quantificateur universel :

$$\forall x.\ \phi \stackrel{def}{=} \neg(\exists x.\ \neg\phi).$$

Nous pouvons également définir la modalité $\mathcal{O}$ *comme suit :*

$$\mathcal{O}\phi \stackrel{def}{=} \mathsf{true}\ \mathcal{S}\ \phi.$$

qui se lit « once ϕ *» et signifie à un moment dans le passé* ϕ *a été satisfaite.*

L'ensemble des variables qui apparaissent dans une formule ϕ, dénoté $\mathcal{V}(\phi)$ est défini inductivement :

$$\mathcal{V}(\phi) \stackrel{def}{=} \begin{cases} \emptyset & \text{si } \phi = \mathsf{true} \\ \bigcup\limits_{i\in[\![n]\!]} \mathcal{V}(t_i) & \text{si } \phi = \mathsf{Q}(t_1,\ldots,t_n) \\ \mathcal{V}(t) & \text{si } \phi = \mathsf{learn}(t) \\ \mathcal{V}(t) & \text{si } \phi = \mathsf{C}(t) \\ \mathcal{V}(\phi_1) & \text{si } \phi = \neg\phi_1 \text{ ou } \phi = \mathcal{Y}\phi_1 \\ \mathcal{V}(\phi_1) \cup \mathcal{V}(\phi_2) & \text{si } \phi = \phi_1 \vee \phi_2 \text{ ou } \phi = \phi_1 \mathcal{S} \phi_2 \\ \{x\} \cup \mathcal{V}(\phi_1) & \text{si } \phi = \exists x.\ \phi_1 \end{cases}$$

L'ensemble $\mathsf{Free}(\phi)$ dénote l'ensemble de *variables libres* de ϕ, *i.e.* l'ensemble des variables qui ne sont pas dans la portée d'un quantificateur :

$$\mathsf{Free}(\phi) \stackrel{def}{=} \begin{cases} \emptyset & \text{si } \phi = \mathsf{true} \\ \bigcup\limits_{i\in[\![n]\!]} \mathcal{V}(t_i) & \text{si } \phi = \mathsf{Q}(t_1,\ldots,t_n) \\ \mathcal{V}(t) & \text{si } \phi = \mathsf{learn}(t) \\ \mathcal{V}(t) & \text{si } \phi = \mathsf{C}(t) \\ \mathsf{Free}(\phi_1) & \text{si } \phi = \neg\phi_1 \text{ ou } \phi = \mathcal{Y}\phi_1 \\ \mathsf{Free}(\phi_1) \cup \mathsf{Free}(\phi_2) & \text{si } \phi = \phi_1 \vee \phi_2 \text{ ou } \phi = \phi_1 \mathcal{S} \phi_2 \\ \mathsf{Free}(\phi_1) \setminus \{x\} & \text{si } \phi = \exists\ x.\phi_1 \end{cases}$$

Une formule ϕ est dite *close* si toutes ses variables sont liées, soit $\mathsf{Free}(\phi) = \emptyset$. Une formule ϕ dont les quantificateurs apparaissent tous en tête est dite *prénexe*, *i.e.* $\phi = \mho_1 x_1 \ldots \mho_n x_n.\ \phi'$ avec $\mho_i \in \{\exists, \forall\}$ pour tout $i \in [\![n]\!]$ et aucun quantificateur n'apparaît dans ϕ'. Une formule *existentielle* est une formule prénexe close dont les seuls quantificateurs sont existentiels, *i.e.* $\phi =$

$\exists_1 x_1 \dots \exists_n x_n.\ \phi'$ et aucun quantificateur n'apparaît dans ϕ'. Une formule *universelle* est une formule de la forme $\neg\phi$ avec ϕ une formule existentielle. En d'autres termes une formule universelle est une formule prénexe close dont les seuls quantificateurs sont universels, *i.e.* $\phi = \forall_1 x_1 \dots \forall_n x_n.\ \phi'$ et aucun quantificateur n'apparaît dans ϕ'.

L'ensemble $\mathsf{SubForm}^-(\phi)$ dénote l'ensemble des *sous-formules négatives* de ϕ et $\mathsf{SubForm}^+(\phi)$ l'ensemble des *sous-formules positives* de ϕ. Ils sont définis par

$$\mathsf{SubForm}^-(\phi) \stackrel{def}{=} \begin{cases} \mathsf{SubForm}^+(\phi_1) & \text{si } \phi = \neg\phi_1 \\ \mathsf{SubForm}^-(\phi_1) \cup \mathsf{SubForm}^-(\phi_2) & \text{si } \phi = \phi_1 \vee \phi_2 \\ & \text{ou si } \phi = \phi_1 \mathcal{S} \phi_2 \\ \mathsf{SubForm}^-(\phi_1) & \text{si } \phi = \mathcal{Y}\phi_1 \\ & \text{ou } \phi = \exists \phi_1 \\ \emptyset & \text{sinon,} \end{cases}$$

et

$$\mathsf{SubForm}^+(\phi) \stackrel{def}{=} \{\phi\} \cup \begin{cases} \mathsf{SubForm}^-(\phi_1) & \text{si } \phi = \neg\phi_1 \\ \mathsf{SubForm}^+(\phi_1) \cup \mathsf{SubForm}^+(\phi_2) & \text{si } \phi = \phi_1 \vee \phi_2 \\ & \text{ou } \phi = \phi_1 \vee \phi_2 \\ \mathsf{SubForm}^+(\phi_1) & \text{si } \phi = \mathcal{Y}\phi_1 \\ & \text{ou } \phi = \exists \phi_1 \\ \emptyset & \text{sinon.} \end{cases}$$

Nous étendons la fonction $\mathsf{St}()$ aux formules de $\mathcal{PS}$-LTL. Soit ϕ une formule

$$\mathsf{St}(\phi) \stackrel{def}{=} \begin{cases} \emptyset & \text{si } \phi = \mathsf{true} \\ \bigcup\limits_{i \in [\![n]\!]} \mathsf{St}(t_i) & \text{si } \phi = \mathsf{Q}(t_1, \dots, t_n) \\ \mathsf{St}(t) & \text{si } \phi = \mathsf{learn}(t) \\ \mathsf{St}(t) & \phi = \mathsf{C}(t) \\ \mathsf{St}(\phi_1) & \text{si } \phi = \neg\phi_1 \\ & \text{ou } \phi = \mathcal{Y}\phi_1 \\ & \text{ou } \phi = \exists x.\ \phi_1 \\ \mathsf{St}(\phi_1) \cup \mathsf{St}(\phi_2) & \text{si } \phi = \phi_1 \vee \phi_2 \\ & \text{ou } \phi = \phi_1 \mathcal{S} \phi_2 \end{cases}$$

L'ensemble $\mathsf{Est}^-(\phi)$ dénote l'ensemble des sous-termes chiffrés de ϕ apparaissant négativement dans ϕ, et est défini comme attendu par :

$$\begin{aligned} \mathsf{Est}^-(\phi) \stackrel{def}{=}\ & \{\mathsf{Est}(t) \mid \mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)\} \\ \cup\ & \{\mathsf{Est}(t_1), \dots, \mathsf{Est}(t_n) \mid \mathsf{Q}(t_1, \dots, t_n) \in \mathsf{SubForm}^-(\phi)\} \end{aligned}$$

Dans la seconde partie de cette thèse, nous nous intéresserons plus particulièrement aux formules du fragment $\mathcal{PS}$-LTL$^-$, défini comme suit :

Définition 3.1.3 ($\mathcal{PS}$-LTL$^{\pm}$)**.** *$\mathcal{PS}$-LTL^{-} est le fragment de $\mathcal{PS}$-LTL contenant uniquement les formules universelles dont toute occurrence de* $\mathsf{learn}(t)$ *apparaît négativement,* i.e.

$$\begin{aligned}\mathcal{PS}\text{-}LTL^{-} = \{\phi \in \mathcal{PS}\text{-}LTL \mid\ & \phi = \forall x_1 \ldots \forall x_n.\phi', \\ & \phi \textit{ close} \\ & \textit{aucun quantificateur n'apparaît dans } \phi', \textit{ et} \\ & \forall t \in \mathcal{T}.\ \mathsf{learn}(t) \notin \mathsf{SubForm}^{+}(\phi') \qquad \}\end{aligned}$$

$\mathcal{PS}$-LTL^{+} est le fragment de $\mathcal{PS}$-LTL contenant uniquement les formules existentielles dont toute occurrence de $\mathsf{learn}(t)$ *apparaît positivement,* i.e.

$$\begin{aligned}\mathcal{PS}\text{-}LTL^{+} = \{\phi \in \mathcal{PS}\text{-}LTL \mid\ & \phi = \exists x_1 \ldots \exists x_n.\phi', \\ & \phi \textit{ close} \\ & \textit{aucun quantificateur n'apparaît dans } \phi', \textit{ et} \\ & \forall t \in \mathcal{T}.\ \mathsf{learn}(t) \notin \mathsf{SubForm}^{-}(\phi') \qquad \}\end{aligned}$$

Comme nous le verrons par la suite, nous considérons uniquement des propriétés de sécurité qui peuvent être exprimées à l'aide de formules de $\mathcal{PS}$-LTL^{-}. Ceci signifie que la formule d'attaque est exprimable à l'aide de formules de $\mathcal{PS}$-LTL^{+}.

3.1.2 La sémantique de $\mathcal{PS}$-LTL

Les formules de $\mathcal{PS}$-LTL expriment des propriétés sur une exécution valide d'un protocole. Nous définissons maintenant la sémantique de $\mathcal{PS}$-LTL.

Définition 3.1.4 ($\langle tr, T_0\rangle \models \phi$)**.** *Soient Π un protocole, T_0 un ensemble de termes clos, et tr une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0. Soit aussi ϕ une formule close de $\mathcal{PS}$-LTL. La relation $\models$ est définie inductivement :*

$$\begin{aligned}
&\langle tr, T_0\rangle \models \mathsf{true} \\
&\langle tr, T_0\rangle \models \mathsf{Q}(t_1, \ldots, t_n) && \textit{ssi } tr = tr'@[\mathsf{Q}(u_1, \ldots, u_n)] \\
&&& \qquad \textit{et } t_i = u_i \textit{ pour tout } i \in [\![n]\!] \\
&\langle tr, T_0\rangle \models \mathsf{learn}(t) && \textit{ssi } T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(tr) \vdash t \\
&\langle tr, T_0\rangle \models \mathsf{C}(t) && \textit{ssi } T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash \mathsf{pvk}(t) \\
&&& \textit{ou } \exists p \in (\mathcal{P} \setminus \{\epsilon\}).\quad T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash \mathsf{shk}(p, t) \\
&&& \qquad\qquad \vee \quad T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash \mathsf{shk}(t, p) \\
&\langle tr, T_0\rangle \models \neg\phi && \textit{ssi } \langle tr, T_0\rangle \not\models \phi \\
&\langle tr, T_0\rangle \models \phi_1 \vee \phi_2 && \textit{ssi } \langle tr, T_0\rangle \models \phi_1 \textit{ ou } \langle tr, T_0\rangle \models \phi_2 \\
&\langle tr, T_0\rangle \models \mathcal{Y}\phi && \textit{ssi } tr = tr'@[e] \textit{ et } \langle tr', T_0\rangle \models \phi \\
&\langle tr, T_0\rangle \models \phi_1 \mathcal{S} \phi_2 && \textit{ssi } \exists i,\ 0 \leq i \leq \ell,\ \langle tr_i, T_0\rangle \models \phi_2 \\
&&& \textit{et } \forall j \in [\![i+1, \ell]\!]\ \langle tr_j, T_0\rangle \models \phi_1 \textit{ avec } tr = [e_1, \ldots, e_\ell] \\
&\langle tr, T_0\rangle \models \exists x.\phi && \textit{ssi } \exists t \in \mathcal{M} \textit{ tel que } \langle tr, T_0\rangle \models \phi\{x \mapsto t\}
\end{aligned}$$

Lorsque $\langle tr, T_0\rangle \models \phi$ *est vraie, nous disons que* tr *satisfait* ϕ *au regard de la connaissance initiale de l'intrus* T_0 *donnée.*

Nous verrons au chapitre 5 qu'il est possible de vérifier pour un nombre borné de sessions si un protocole Π satisfait une formule ϕ de $\mathcal{PS}$-LTL$^-$, en s'assurant que l'ensemble fini dans le cadre d'un nombre borné de sessions des traces symboliques de Π vérifie ϕ. Pour cela, la relation $\models$ est d'abord étendue aux traces symboliques.

Définition 3.1.5. *Soit* Π *un protocole,* tr *une trace symbolique de* Π, i.e. *la trace symbolique associée à un scénario de* Π, T_0 *un ensemble de termes clos et* ϕ *une formule close de* $\mathcal{PS}$*-LTL, tels que* $\mathcal{V}(tr) \cap \mathcal{V}(\phi) = \emptyset$ *(ceci peut être obtenu en renommant les variables de* ϕ*). La trace symbolique* tr *satisfait* ϕ*, au regard de la connaissance initiale de l'intrus* T_0*, si pour toute substitution* σ *telle que* $tr\sigma$ *est une exécution valide de* Π *au regard de la connaissance initiale de l'intrus* T_0*,* $\langle tr\sigma, T_0\rangle \models \phi$.

Inversement, la trace symbolique tr viole ϕ s'il existe une substitution σ telle que $tr\sigma$ soit une exécution valide de Π au regard de T_0, et $\langle tr\sigma, T_0\rangle \models \neg\phi$.

Définition 3.1.6 ($\langle \Pi, T_0\rangle \models \phi$)**.** *Soit* Π *un protocole,* ϕ *une formule close de* $\mathcal{PS}$*-LTL et* T_0 *un ensemble de termes clos.* Π *satisfait* ϕ *étant donnée la connaissance initiale de l'intrus* T_0*, dénoté* $\langle \Pi, T_0\rangle \models \phi$*, si et seulement si* $\langle tr, T_0\rangle \models \phi$ *pour toute trace symbolique de* Π *(*i.e. *pour toute trace symbolique associée à un scénario de* Π*).*

Nous dirons que Π *viole la propriété* ϕ étant donnée la connaissance initiale de l'intrus T_0 si et seulement si étant donnée T_0, la connaissance initiale de l'intrus, Π ne satisfait pas ϕ, *i.e.* $\langle \Pi, T_0\rangle \not\models \phi$.

3.2 Propriétés de sécurité usuelles

Nous allons à présent voir comment spécifier des propriétés de sécurité à l'aide de formules de $\mathcal{PS}$-LTL, et plus précisément de $\mathcal{PS}$-LTL$^-$. Nous nous imposons cette restriction, car comme nous l'expliquions un peut plus haut, et comme nous le verrons au chapitre 5 c'est pour ce fragment de $\mathcal{PS}$-LTL que le problème de la vérification dans le cadre d'un nombre borné de sessions est décidable.

3.2.1 Le secret

La propriété du secret est l'inhabilité pour l'intrus d'apprendre un message dont il est spécifié qu'il doit rester « secret ». Souvent, le secret est un nonce du protocole Π considéré, *i.e.* un terme $n \in \mathcal{N}(\Pi)$. Précisons que, lorsque n est spécifié comme étant le secret de Π, ce qui est implicitement spécifié comme devant rester secret c'est l'ensemble des instances de n engendrées par des sessions non-compromises. Nous allons à présent préciser ces idées et voir comment modéliser cette propriété à l'aide de $\mathcal{PS}$-LTL$^-$.

Soient un protocole $\Pi = [e_1; \ldots; e_\ell]$ et un nonce de ce protocole $n \in \mathcal{N}(\Pi)$. Il existe alors un rôle $r_p \in \mathsf{Roles}(\Pi)$ (avec $p \in \mathcal{P}$) ayant engendré n, *i.e.* $n \in \mathcal{N}(\Pi(p))$. Il existe aussi un entier $i \in [\![\ell]\!]$ tel que la première occurrence de n

apparaisse au cours de l'événement e_i, *i.e.* $\exists i \in [\![\ell]\!].\ \forall j \in [\![i-1]\!].\ n \notin \mathsf{St}(e_j) \wedge n \in \mathsf{St}(e_i)$. Afin d'énoncer la propriété du secret (pour n) à l'aide de $\mathcal{PS}$-LTL$^-$, nous construisons à partir du protocole Π un protocole Π' comme suit. Soit Secret un prédicat n'apparaissant pas dans Π,

$$\Pi' = [e_1; \ldots; e_i; \mathsf{Secret}(p, a_1, \ldots, a_k, n); e_{i+1}; \ldots; e_\ell]$$

où $[a_1; \ldots; a_k] = \mathsf{Agts}(\Pi)$. Au cours d'une exécution, le prédicat Secret liera chaque instance de n aux participants effectifs de la session à laquelle il a été engendré. Nous n'avons pas besoin de préciser les serveurs dans le prédicats Secret parce que par définition les serveurs sont les mêmes pour toute exécution d'un même protocole.

La formule suivante exprime que les instances non-compromises de n, doivent rester secrètes.

$$\phi_{\mathsf{S}} = \left\{ \begin{array}{c} \forall y_1 \ldots \forall y_k . \forall z. \\ [\mathcal{O}(\mathsf{Secret}(y_1, \ldots, y_k, z)) \ \wedge \bigwedge_{i \in [\![k]\!]} \mathsf{NC}(y_i) \ \wedge \bigwedge_{s \in (\mathcal{P}(\Pi) \cap \mathcal{S})} \mathsf{NC}(s)] \\ \Rightarrow \\ \neg \mathsf{learn}(z). \end{array} \right.$$

Plus précisément, ϕ_{S} stipule que pour toute instance z de n issue d'une session avec pour agent effectifs $y_1, \ldots, y_k$ ($\mathsf{Secret}(y_1, \ldots, y_k, z)$), si les participants effectifs (serveurs inclus) sont non-compromis ($\mathsf{NC}(y_i)$ pour $i \in [\![k]\!]$, et $\mathsf{NC}(s)$ pour $s \in (\mathcal{P}(\Pi) \cap \mathcal{S})$) alors l'intrus ne devrait pas pouvoir déduire z, l'instance de n issue de cette session ($\neg \mathsf{learn}(z)$).

A noter que telle qu'énoncée, si un des serveurs du protocole est compromis, alors la propriété du secret est trivialement satisfaite par Π'. En effet, dans un tel cas l'antécédent de l'implication ne sera jamais vrai. Il est toutefois possible d'énoncer la propriété du secret sans exiger que les serveurs soient non-compromis. Nous ne rentrerons néanmoins pas dans le débat de savoir quelle serait la propriété du secret à considérer, car ceci dépasse notre simple intention ici d'illustrer le type de propriétés qui peuvent être énoncées à l'aide de formules de $\mathcal{PS}$-LTL$^-$. De plus, les résultats présentés en seconde partie de cette thèse ne sont pas liés à telle ou telle propriété, mais à une classe de propriété plus large qui comprend les deux définitions de la propriété du secret mentionnées à l'instant.

Nous dirons que Π préserve le secret de n au regard de la connaissance initiale de l'intrus T_0 (où T_0 est un ensemble de termes clos) si et seulement si $\langle \Pi', T_0 \rangle \models \phi_{\mathsf{S}}$. Ici, nous avons montré comment spécifier la propriété du secret pour un nonce du protocole considéré. Il est bien entendu possible de spécifier des termes composés (et pas nécessairement frais) comme étant le secret. Néanmoins, présenter en toute généralité la propriété du secret, *i.e.* même pour des termes composés, est quelque peu compliqué et n'est pas nécessaire pour la suite de notre exposé.

Exemple 3.2.1. *Reprenons notre protocole Π_{Toy} de l'exemple 2.4.4. Le secret de Π_{Toy} est le nonce n engendré par a. Afin d'exprimer la propriété du secret pour n, nous construisons le protocole Π'_{Toy}. D'après la construction donnée*

ci-dessus nous obtenons le protocole avec les deux rôles suivants :

$$r'_a = [\ \mathsf{snd}(a,b,\mathsf{aenc}(\langle \mathsf{aenc}(n,b),a\rangle,b));\ \mathsf{Secret}(a,b,n);\ \mathsf{rcv}(a,b,\mathsf{aenc}(\langle \mathsf{aenc}(n,a),b\rangle,a))]$$

$$r'_b = [\ \mathsf{rcv}(b,a,\mathsf{aenc}(\langle \mathsf{aenc}(x,b),a\rangle,b));\ \mathsf{snd}(b,a,\mathsf{aenc}(\langle \mathsf{aenc}(x,a),b\rangle,a))]$$

ainsi que la formule :

$$\phi^{\mathsf{S}}_{\mathsf{Toy}} = \left\{ \begin{array}{c} \forall y_a.\forall y_b.\forall y_n. \\ [\mathcal{O}(\mathsf{Secret}(y_a,y_b,y_n)) \ \wedge\ \mathsf{NC}(y_a) \ \wedge\ \mathsf{NC}(y_b)] \ \Rightarrow\ \neg\mathsf{learn}(y_n). \end{array} \right.$$

Cette formule stipule, comme nous l'expliquions un peu plus haut, que toute instance y_n de n issue d'une session où les participants effectifs y_a et y_b sont non-compromis devrait rester secrète, i.e. *non déductible de l'intrus ($\neg\mathsf{learn}(y_n)$).*

Considérons le scénario $\mathsf{sc}'_{\mathsf{Toy}} = (\mathsf{interlvg}'_{\mathsf{Toy}}, \mathsf{initagts}'_{\mathsf{Toy}})$ *avec :*

$$\mathsf{interlvg}'_{\mathsf{Toy}} = [(r'_a,1);(r'_a,1);(r'_b,2);(r'_b,2);(r'_b,3);(r'_b,3);(r'_a,1)],$$

$$\mathsf{initagts}'_{\mathsf{Toy}}(1) = (a,b), \qquad \mathsf{initagts}'_{\mathsf{Toy}}(2) = (\epsilon,b), \ \textit{et} \qquad \mathsf{initagts}'_{\mathsf{Toy}}(3) = (\epsilon,b).$$

La trace symbolique tr'_{Toy} *associée à ce scénario est la séquence d'événements suivante :*

$$\begin{array}{rl} tr'_{\mathsf{Toy}} = [& \mathsf{snd}(a,b,\mathsf{aenc}(\langle \mathsf{aenc}(n^1,b),a\rangle,b)); \mathit{Secret}(a,b,n^1); \\ & \mathsf{rcv}(b,\epsilon,\mathsf{aenc}(\langle \mathsf{aenc}(x^2,b),\epsilon\rangle,b)); \mathsf{snd}(b,\epsilon,\mathsf{aenc}(\langle \mathsf{aenc}(x^2,\epsilon),b\rangle,\epsilon)); \\ & \mathsf{rcv}(b,\epsilon,\mathsf{aenc}(\langle \mathsf{aenc}(x^3,b),\epsilon\rangle,b)); \mathsf{snd}(b,\epsilon,\mathsf{aenc}(\langle \mathsf{aenc}(x^3,\epsilon),b\rangle,\epsilon)); \\ & \mathsf{rcv}(a,b,\mathsf{aenc}(\langle \mathsf{aenc}(n^1,a),b\rangle,a)) \qquad] \end{array}$$

Si nous reprenons la substitution de l'exemple 2.6.9

$$\sigma = \{x^2 \mapsto \langle \mathsf{aenc}(n^1,b),a\rangle; x^3 \mapsto n^1\},$$

il est aisé de voir que la trace $tr'_{\mathsf{Toy}}\sigma$ *est une exécution valide de* Π'_{Toy} *au regard de* T_0 *(pour tout ensemble de termes clos* T_0*), qui viole la propriété* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$, i.e. $\langle tr'_{\mathsf{Toy}}\sigma, T_0\rangle \models \neg\phi^{\mathsf{S}}_{\mathsf{Toy}}$. *En effet, il suffit de noter que*

- $(tr'_{\mathsf{Toy}}\sigma)[2] = \mathsf{Secret}(a,b,n^1)$, *et*
- $T_0 \cup \mathsf{K}(tr'_{\mathsf{Toy}}\sigma) \vdash n^1$ *(vu à l'exemple 2.5.2)*

pour déduire que

$$\langle tr'_{\mathsf{Toy}}\sigma, T_0\rangle \models \mathcal{O}\ \mathsf{Secret}(a,b,n^1) \ \wedge\ \mathsf{NC}(a) \ \wedge\ \mathsf{NC}(b) \ \wedge\ \mathsf{learn}(n),$$

et donc que

$$\langle tr'_{\mathsf{Toy}}\sigma, T_0\rangle \models \exists y_a.\exists y_b.\exists y_n.(\mathcal{O}\ \mathsf{Secret}(y_a,y_b,y_n) \wedge \mathsf{NC}(y_a) \wedge \mathsf{NC}(y_b) \wedge \mathsf{learn}(y_n)).$$

Donc $\langle tr'_{\mathsf{Toy}}, T_0\rangle \not\models \phi^{\mathsf{S}}_{\mathsf{Toy}}$, *et donc notre protocole* Π_{Toy} *ne garantit pas le secret de n quelle que soit la connaissance initiale de l'intrus* T_0 *considérée.*

3.2.2 L'authentification

Nous allons à présent nous intéresser à l'authentification. Il existe plusieurs formes d'authentification (voir [39] pour une classification des diverses définitions envisagées). Exceptée celle de l'« accord injectif », toutes les autres (recensées dans [39]) peuvent, comme nous allons le voir, être exprimées par des formules de $\mathcal{PS}$-LTL, et plus précisément par des formules de $\mathcal{PS}$-LTL$^-$.

Les propriétés d'authentification énoncées par Lowe dans [39] le sont pour des protocoles à deux participants. En passant à un nombre non fixé de participants nous avons dû faire certains choix quant à la façon d'étendre les définitions de ces propriétés. Néanmoins, notre objectif étant rappelons le, uniquement d'illustrer le type de propriétés exprimables à l'aide de formules de $\mathcal{PS}$-LTL$^-$, nous ne rentrerons pas ici dans des explications concernant ces choix. Nous sommes cependant conscients du fait que nous aurions pu définir ces extensions différemment.

La vivacité (Aliveness)

La vivacité est la plus faible des formes d'authentification dans la hiérarchie de Lowe [39]. Informellement, un protocole Π satisfait la propriété de vivacité si et seulement si à chaque fois qu'un participant p finit d'exécuter une session complète non-compromise de Π incarnant n'importe lequel des rôles de Π, en croyant communiquer avec une autre entité p', alors p' a précédemment exécuté, au moins partiellement, un des rôles de Π. En d'autres termes, à chaque fois qu'un participant termine une session non-compromise, alors chacun des participants effectifs de cette session a auparavant initié une session.

Afin d'exprimer cette propriété nous avons besoin de détecter dans une exécution le début et la fin de chaque session. Ceci peut être accompli en ajoutant des status events en début et en fin de chaque rôle. Plus précisément, soit $\Pi = [e_1; \ldots; e_\ell]$ un protocole avec les rôles $\mathsf{Roles}(\Pi) = \{r_{p_1}, \ldots, r_{p_k}\}$, et avec les agents $[a_1; \ldots; a_h] = \mathsf{Agts}(\Pi)$. Soient aussi les prédicats $\mathsf{Start}_1, \ldots, \mathsf{Start}_k$ et $\mathsf{End}_1, \ldots, \mathsf{End}_k$ n'apparaissant pas dans Π. Nous construisons le protocole Π'' en insérant des status events dans la séquence d'événements de Π comme suit.

$$\begin{aligned}\Pi'' = [\quad & \mathsf{Start}_1(p_1, p_1); \ldots; \mathsf{Start}_k(p_k, p_k);\\ & e_1; \ldots; e_\ell;\\ & \mathsf{End}_1(p_1, a_1, \ldots, a_h); \ldots; \mathsf{End}_k(p_k, a_1, \ldots, a_h) \quad]\end{aligned}$$

Les prédicats Start_i et End_i (pour $i \in [\![k]\!]$) marqueront dans une exécution le début et la fin de chaque session, et lieront entre eux les participants effectifs de chaque session. Ainsi, non seulement nous pourrons détecter dans une exécution le début et la fin de chaque session, mais nous pourrons aussi distinguer les sessions non-compromises (celles où l'intrus n'est pas un des participants effectifs par exemple), des sessions compromises.

La vivacité peut alors être modélisée par la formule :

$$\phi_{\mathsf{A}} = \forall y_1^1 \ldots . \forall y_h^1 \ldots . \forall y_1^k \ldots . \forall y_h^k .$$
$$\bigwedge_{i\in[\![k]\!]} [(\mathsf{End}_i(y_1^i,\ldots,y_h^i) \ \wedge \ \bigwedge_{j\in[\![h]\!]} \mathsf{NC}(y_j^i) \ \wedge \ \bigwedge_{s\in(\mathcal{P}(\Pi)\cap\mathcal{S})} \mathsf{NC}(s))$$
$$\Rightarrow$$
$$\bigwedge_{j\in[\![h]\!]} \mathcal{O}(\bigvee_{m\in[\![k]\!]} \mathsf{Start}_m(y_j^i)) \ \wedge \ \bigwedge_{s\in(\mathcal{P}(\Pi)\cap\mathcal{S})} \mathcal{O}((\bigvee_{m\in[\![k]\!]} \mathsf{Start}_m(s))].$$

qui stipule, qu'à chaque fois qu'une session non-compromise du rôle r_{p_i} est complétée, avec pour participants effectifs les agents $y_1^i, \ \ldots, \ y_h^i$ ($\mathsf{End}_i(y_1^i,\ldots,y_h^i)$), alors chacun d'entre eux doit avoir initié au moins une session.

Nous dirons que Π vérifie la propriété de vivacité au regard de la connaissance initiale de l'intrus T_0 (où T_0 est un ensemble de termes clos) si et seulement si $\langle \Pi'', T_0\rangle \models \phi_{\mathsf{A}}$.

Exemple 3.2.2. *Illustrons la propriété de vivacité à l'aide de notre protocole* Π_{Toy}. *En appliquant la construction générique ci-dessus à notre protocole nous obtenons le protocole* Π''_{Toy} *avec les deux rôles du suivants :*

$$r''_a = [\ \mathsf{Start}_1(a,a);\ \mathsf{snd}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,b),a\rangle,b));\ \mathsf{rcv}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,a),b\rangle,a));\ \mathsf{End}_1(a,a,b)\ ]$$

$$r''_b = [\ \mathsf{Start}_2(b,b);\ \mathsf{rcv}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,b),a\rangle,b));\ \mathsf{snd}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,a),b\rangle,a));\ \mathsf{End}_2(b,a,b)\ ]$$

ainsi que la formule

$$\phi^{\mathsf{A}}_{\mathsf{Toy}} = \begin{cases} \forall y_1.\ \forall y_2.\ \forall z_1.\ \forall z_2. \\ [\mathsf{End}_1(y_1,y_2) \ \wedge \ \mathsf{NC}(y_1) \ \wedge \ \mathsf{NC}(y_2) \\ \Rightarrow \\ \mathcal{O}(\mathsf{Start}_1(y_1)\vee\mathsf{Start}_2(y_1)) \ \wedge \ \mathcal{O}(\mathsf{Start}_1(y_2)\vee\mathsf{Start}_2(y_2))] \\ \wedge \\ [\mathsf{End}_2(z_1,z_2) \ \wedge \ \mathsf{NC}(z_1) \ \wedge \ \mathsf{NC}(z_2) \\ \Rightarrow \\ \mathcal{O}(\mathsf{Start}_1(z_1)\vee\mathsf{Start}_2(z_1)) \ \wedge \ \mathcal{O}(\mathsf{Start}_1(z_2)\vee\mathsf{Start}_2(z_2))]. \end{cases}$$

Considérons le scénario $\mathsf{sc}''_{\mathsf{Toy}} = (\mathsf{interlvg}''_{\mathsf{Toy}}, \mathsf{initagts}''_{\mathsf{Toy}})$ *avec :*

$\mathsf{interlvg}''_{\mathsf{Toy}} = [(r''_b,1);(r''_b,1);(r''_b,1);(r''_b,1)]$ *et* $\mathsf{initagts}''_{\mathsf{Toy}}(1) = (a,b)$.

La trace symbolique tr''_{Toy} *associée à ce scénario est la séquence d'événements suivante :*

$$tr''_{\mathsf{Toy}} = [\ \mathsf{Start}_2(b);\ \mathsf{rcv}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x^1,b),a\rangle,b));\ \mathsf{snd}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x^1,a),b\rangle,a));\ \mathsf{End}_2(a,b)\ ].$$

Si nous considérons la substitution $\sigma = \{x^1 \mapsto \epsilon\}$, il est aisé de voir que $tr''_{\mathsf{Toy}}\sigma$ est une exécution valide de Π''_{Toy}, à l'issue de laquelle l'agent b terminé sa session du rôle r''_b en croyant avoir parlé à l'agent a (événement $\mathsf{End}_2(a,b)$ de $tr''_{\mathsf{Toy}}\sigma$), alors que ce dernier n'a initialisé aucune session de Π''_{Toy} (aucun événement $\mathsf{Start}_1(a)$ ni $\mathsf{Start}_2(a)$ n'apparaît dans $tr''_{\mathsf{Toy}}\sigma$). Donc Π_{Toy} ne satisfait pas la propriété de vivacité, et ce quelle que soit la connaissance initiale de l'intrus considérée.

L'accord faible (Weak agreement)

L'accord faible est une forme d'authentification légèrement plus forte que la vivacité. Informellement, un protocole Π satisfait la propriété dite de l'accord faible si et seulement si à chaque fois qu'une session non-compromise initiée par un participant p est complétée, chacun des interlocuteurs de cette session à préalablement initiée une session qui compte p parmi ses interlocuteurs effectifs.

Comme pour la propriété de la vivacité, nous aurons besoin de détecter dans une exécution le début ainsi que la fin de chaque session ; mais là nous aurons besoin de connaître les participants effectifs de chaque session dès le début de celle-ci. Soit $\Pi = [e_1; \ldots; e_l]$ un protocole avec les rôles $\mathsf{Roles}(\Pi) = \{r_{p_1}; \ldots; r_{p_k}\}$ et avec $[a_1; \ldots; a_h] = \mathcal{A}(\Pi)$. Nous procédons de la manière suivante. Soient les prédicats $\mathsf{Start}_{i,j}$ pour tout $i \in [\![k]\!]$ et $j \in [\![h]\!]$, ainsi que les prédicats $\mathsf{End}_1, \ldots, \mathsf{End}_k$ n'apparaissant pas dans Π. Nous construisons le protocole Π''' en insérant des status events dans la séquence d'événements de Π tels qu'indiqué ci-après

$$
\begin{aligned}
\Pi''' = [\quad & \mathsf{Start}_{1,1}(p_1, p_1, a_1); \ldots; \mathsf{Start}_{1,h}(p_1, p_1, a_h); \\
& \ldots \\
& \mathsf{Start}_{k,1}(p_k, p_k, a_1); \ldots; \mathsf{Start}_{k,h}(p_k, p_k, a_h); \\
& e_1; \ldots; e_\ell; \\
& \mathsf{End}_1(p_1, p_1, a_1, \ldots, a_h); \ldots; \mathsf{End}_k(p_k, p_k, a_1, \ldots, a_k)].
\end{aligned}
$$

Comme pour la propriété précédente les prédicats $\mathsf{Start}_{i,j}$ et End_i (pour $i \in [\![k]\!]$ et $j \in [\![h]\!]$) marqueront dans une exécution le début et la fin de chaque session. De plus ces prédicats lient l'initiateur de chaque session à ses interlocuteurs.

Le lecteur aura peut-être pensé à d'autres formulations, plus directes, de la propriété visée, où en particulier, tous les participants effectifs d'une session serait liés en début de session à l'aide d'un même prédicat. En effet, la première idée serait de reprendre le protocole Π'' présenté pour la propriété de vivacité, et de passer tous les agents de Π en paramètre des prédicats Start_i. Mais il ne serait, en procédant ainsi, pas possible d'énoncer la propriété dite de l'accord faible avec une formule du fragment $\mathcal{PS}$-LTL$^-$, car il aurait fallu quantifier existentiellement certaines variables.

L'accord faible peut alors être modélisée par la formule :

$$\phi_{\mathsf{WA}} = \forall z_1.\forall y_1^1.\ldots.\forall y_h^1.\ldots.\forall z_k.\forall y_1^k.\ldots.\forall y_h^k.$$
$$\bigwedge_{i\in[\![k]\!]}[(\mathsf{End}_i(z_i,y_1^i,\ldots,y_h^i)\wedge\mathsf{NC}(z_i)\wedge\mathsf{NC}(y_1^i)\wedge\cdots\wedge\mathsf{NC}(y_h^i))$$
$$\Rightarrow$$
$$\bigwedge_{j\in[\![h]\!]}\mathcal{O}(\bigvee_{m\in[\![k]\!]}\bigvee_{m'\in[\![h]\!]}\mathsf{Start}_{m,m'}(y_j^i,z_i))$$
$$\wedge$$
$$\bigwedge_{s\in(\mathcal{P}(\Pi)\cap\mathcal{S})}\mathcal{O}(\bigvee_{m\in[\![k]\!]}\bigvee_{m'\in[\![h]\!]}\mathsf{Start}_{m,m'}(s,z_i))].$$

Nous dirons que Π vérifie la propriété dite de l'accord faible au regard de la connaissance initiale de l'intrus T_0 (où T_0 est un ensemble de termes clos) si et seulement si $\langle\Pi''',T_0\rangle\models\phi_{\mathsf{WA}}$.

Exemple 3.2.3. *Nous considérons toujours notre protocole* Π_{Toy}*. En appliquant la construction générique ci-dessus à notre protocole nous obtenons le protocoles* Π'''_{Toy} *avec les deux rôles suivants*

$$r_a''' = [\ \mathsf{Start}_{1,1}(a,a);\ \mathsf{Start}_{1,2}(a,b);\ \mathsf{snd}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,b),a\rangle,b));\ \mathsf{rcv}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,a),b\rangle,a));\ \mathsf{End}_1(a,a,b)\ ]$$

$$r_b''' = [\ \mathsf{Start}_{2,1}(b,a);\ \mathsf{Start}_{2,2}(b,b);\ \mathsf{rcv}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,b),a\rangle,b));\ \mathsf{snd}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,a),b\rangle,a));\ \mathsf{End}_2(b,a,b)\ ]$$

ainsi que la formule

$$\phi^{\mathsf{A}}_{\mathsf{Toy}} = \begin{cases} \forall y.\ \forall y_1.\ \forall y_2.\ \forall z.\ \forall z_1.\ \forall z_2. \\ [\mathsf{End}_1(y,y_1,y_2)\ \wedge\ \mathsf{NC}(y)\ \wedge\ \mathsf{NC}(y_1)\ \wedge\ \mathsf{NC}(y_2) \\ \Rightarrow \\ \mathcal{O}\ (\mathsf{Start}_{1,1}(y_1,y)\vee\mathsf{Start}_{1,2}(y_1,y)\vee\mathsf{Start}_{2,1}(y_1,y)\vee\mathsf{Start}_{2,2}(y_1,y)) \\ \wedge \\ \mathcal{O}\ (\mathsf{Start}_{1,1}(y_2,y)\vee\mathsf{Start}_{1,2}(y_2,y)\vee\mathsf{Start}_{2,1}(y_2,y)\vee\mathsf{Start}_{2,2}(y_2,y)) \\ \wedge \\ [\mathsf{End}_2(z,z_1,z_2)\ \wedge\ \mathsf{NC}(z)\ \wedge\ \mathsf{NC}(z_1)\ \wedge\ \mathsf{NC}(z_2) \\ \Rightarrow \\ \mathcal{O}\ (\mathsf{Start}_{1,1}(z_1,z)\vee\mathsf{Start}_{1,2}(z_1,z)\vee\mathsf{Start}_{2,1}(z_1,z)\vee\mathsf{Start}_{2,2}(z_1,z)) \\ \wedge \\ \mathcal{O}\ (\mathsf{Start}_{1,1}(z_1,z)\vee\mathsf{Start}_{1,2}(z_1,z)\vee\mathsf{Start}_{2,1}(z_1,z)\vee\mathsf{Start}_{2,2}(z_1,z))]. \end{cases}$$

Comme nous l'avons dit cette forme d'authentification est plus forte que la précédente, i.e. *pour tout ensemble de termes clos* T_0, $\langle\Pi,T_0\rangle\models\phi_{\mathsf{WA}}\ \Rightarrow\ \langle\Pi,T_0\rangle\models\phi_{\mathsf{A}}$*. Or nous avons déjà vu que notre protocole* Π *ne vérifie pas la propriété de vivacité quelle que soit la connaissance initiale de l'intrus considérée* T_0*, il*

ne vérifie donc pas non-plus la propriété de l'accord faible quelle que soit la connaissance initiale de l'intrus considérée T_0, i.e. $\langle \Pi_{\mathsf{Toy}}, T_0 \rangle \not\models \phi^{\mathsf{WA}}_{\mathsf{Toy}}$.

Il existe encore d'autres versions de l'authentification, et en particulier la propriété d'authentification dite de l'accord non-injectif. Cette dernière est légèrement plus forte que celle de l'accord faible que nous venons de présenter. Nous arrêtons néanmoins là notre exposé sur les propriétés de sécurité exprimables à l'aide de $\mathcal{PS}$-LTL$^-$ et renvoyons le lecteur à [20, 21]. Les auteurs y illustrent sur un exemple la modélisation de la propriété de l'accord non-injectif, mais aussi la propriété dite de fraîcheur. Pour notre part, nous pensons que les propriétés présentées jusque là suffisent à répondre à notre but initial, à savoir illustrer le type de propriétés exprimables dans $\mathcal{PS}$-LTL et plus précisément dans $\mathcal{PS}$-LTL$^-$.

Chapitre 4

Indécidabilité du problème de la vérification (même dans le cadre de messages de taille bornée)

Le problème auquel nous nous sommes intéressés au cours de ce travail est celui de la vérification des protocoles de sécurité. Il s'agît là d'un problème de décision qui s'énonce comme suit.

> Etant donnés un protocole de sécurité Π, un ensemble de termes clos T_0, et une formule ϕ de $\mathcal{PS}$-LTL, Π satisfait-il la propriété ϕ au regard de la connaissance initiale de l'intrus T_0, *i.e.* $\langle \Pi, T_0 \rangle \models \phi$?

Même en se restreignant à la seule propriété du secret, ce problème est bien connu pour être indécidable. Il existe en effet plusieurs preuves de son indécidabilité (voir [19, 32] pour ne citer qu'elles), au regard desquelles, il semblerait que parmi les principales sources d'indécidabilité, nous retrouvions la taille non-bornée des messages, ainsi que le nombre non-borné de sessions, impliqués dans une attaque. Une question qui se pose alors naturellement est celle de savoir si ce problème serait toujours indécidable si nous nous restreignions à des messages de taille bornée, ou à un nombre borné de sessions. La réponse à cette deuxième question a été apportée par M. Rusinowitch et M. Turuani dans [49], qui ont montré qu'en se restreignant à un nombre borné de sessions et à la seule propriété du secret, le problème devient décidable. R. Corin et *al.* dans [21] ont par la suite étendu ce résultat positif à toute propriété exprimable dans $\mathcal{PS}$-LTL^{-}.

Le premier problème, à savoir celui de la vérification des protocoles de sécurité dans un cadre où la taille des messages est bornée, est un problème pour lequel plusieurs preuves d'indécidabilité ont été proposées (quatre à notre connaissance [48, 3, 32, 31]). Elles montrent qu'il est possible de réduire des problèmes indécidables, tels que l'accessibilité pour les machines à deux compteurs, ou encore le problème de correspondance de Post, au problème de la vérification de la propriété du secret pour les protocoles de sécurité.

Néanmoins, toutes ces preuves exploitent des propriétés liées à la modélisation des protocoles que les « vrais » protocoles ne satisfont pas. Nous rejoignons ainsi S. Fröschle qui dans [33] pose la question de savoir si en se restreignant à des modèles plus réalistes, ce problème reste indécidable. La principale critique émise dans [33] à l'encontre des protocoles mis en œuvre dans ces preuves d'indécidabilité est celle de l'absence d'*exécution honnête*[1] de ces derniers. Ce problème fut initialement pointé par H. Comon-Lundh et V. Cortier dans [19], qui ont eux-mêmes proposé une preuve d'indécidabilité du problème de la vérification des protocoles *honnêtement exécutables*[1], mais pour un nombre fini de nonces, et ce dans le cadre de messages de taille non-bornée.

Nous détaillerons dans ce chapitre une de ces preuves [31], la plus simple et la plus intuitive à notre sens, et verrons en quoi celle-ci peut ne pas être pleinement convaincante. Les trois preuves laissées de coté souffrent cependant des mêmes défauts. Puis, nous montrerons qu'il est possible de pallier aux faiblesses de cette preuve, en exhibant un codage qui résulte en des protocoles plus « réalistes », des protocoles *honnêtement exécutables*. Nous montrerons ainsi que même en se restreignant à des protocoles *honnêtement exécutables*, et des messages de taille bornée, le problème de la vérification des protocoles de sécurité reste indécidable.

Afin de faciliter la lecture de ce chapitre nous introduisons les quelques notations suivantes.

Notation 4.0.4. *Soient $n+1$ termes $u, u_1, \ldots, u_{n-1}, u_n \in \mathcal{T}$ et deux agents $a_1, a_2 \in \mathcal{A}$,*

$$\{u_1, \ldots, u_{n-1}, u_n\}_u \stackrel{def}{=} \mathsf{senc}(\langle u_1, \langle \ldots, \langle u_{n-1}, u_n\rangle \ldots\rangle\rangle, u)$$

$$\mathsf{snd}(a_1, a_2, u_1, \ldots, u_{n-1}, u_n) \stackrel{def}{=} \mathsf{snd}(a_1, a_2, \langle u_1, \langle \ldots, \langle u_{n-1}, u_n\rangle \ldots\rangle\rangle)$$

$$\mathsf{rcv}(a_1, a_2, u_1, \ldots, u_{n-1}, u_n) \stackrel{def}{=} \mathsf{rcv}(a_1, a_2, \langle u_1, \langle \ldots, \langle u_{n-1}, u_n\rangle \ldots\rangle\rangle)$$

4.1 Le problème de correspondance de Post

La preuve d'indécidabilité proposée par N. Durgin et *al.* dans [31] procède par réduction du problème de correspondance de Post, vers la vérification de la propriété du secret pour les protocoles de sécurité. Dans cette section, nous rappelons brièvement les définitions relatives au problème de correspondance de Post.

Définition 4.1.1 (Instance du problème de correspondance de Post)**.** *Soit Σ un alphabet. Une instance du problème de correspondance de Post sur Σ est un ensemble fini de couples de mots de $\Sigma^* \times \Sigma^*$,* i.e.

$$\mathcal{P} = \{(u_i, v_i) \mid u_i, v_i \in \Sigma^*\}_{i \in [\![h]\!]},$$

pour un certain $h \in \mathbb{N}^*$. Nous dirons que $\mathcal{P}$ est de taille h. Une solution de $\mathcal{P}$ *est un k-uplet $(i_1, \ldots, i_k) \in [\![h]\!]^k$ pour un certain $k > 0$ tel que*

$$u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}.$$

[1] Nous verrons un peu plus loin dans ce chapitre ce que nous entendons par « exécution honnête » et par protocole « honnêtement exécutable ».

Un couple de mots $(u, v) \in \Sigma^* \times \Sigma^*$ *est un* mot de $\mathcal{P}$*, s'il existe un* k*-uplet* $(i_1, \ldots, i_k) \in [\![h]\!]^k$ *pour un certain* $k > 0$*, tel que* $u = u_{i_1} \ldots u_{i_k}$*, et* $v = v_{i_1} \ldots v_{i_k}$.
Un couple de mots $(u, v) \in \Sigma^* \times \Sigma^*$ *est une* présolution de $\mathcal{P}$*, s'il existe* (u', v') *un mot de* $\mathcal{P}$*, ainsi qu'un mot* $w \in \Sigma^*$ *tels que* $u' = uw$ *et* $v' = vw$.

A noter que tout mot de $\mathcal{P}$ est une présolution de $\mathcal{P}$, il suffit de prendre le mot vide pour w, *i.e.* $w = \varepsilon$.

Exemple 4.1.2. *Soit l'alphabet* $\Sigma^{\mathsf{exmp}} = \{c, d\}$*. L'ensemble*

$$\mathcal{P}^{\mathsf{exmp}} = \{(c, cdd), (cd, cdd)\}$$

est une instance du problème de correspondance de Post sur l'alphabet Σ^{exmp}.

Il est facile de voir que pour tout entier $k \in \mathbb{N}^+$ *et pour toute séquence* $(i_1, \ldots, i_k) \in [\![2]\!]^k$*,* $u_{i_1} \ldots u_{i_k} \in (c^+d)^+$ *et* $v_{i_1} \ldots v_{i_k} \in (cdd)^+$*, et que donc* $u_{i_1} \ldots u_{i_k} \neq v_{i_1} \ldots v_{i_k}$ *(en effet* $(c^+d)^+ \cap (cdd)^+ = \emptyset$*). Ainsi l'instance du problème de correspondance de Post* $\mathcal{P}^{\mathsf{exmp}}$ *n'admet aucune solution.*

Par contre, $(cc, cddcd)$ *est une présolution de* $\mathcal{P}^{\mathsf{exmp}}$*, car* $(ccd, cddcdd)$ *est un mot de* $\mathcal{P}^{\mathsf{exmp}}$.

Définition 4.1.3 (Problème de correspondance de Post)**.** *Le problème de correspondance de Post consiste à décider, étant donnée une instance* $\mathcal{P}$ *du problème, si* $\mathcal{P}$ *admet une solution.*

Théorème 4.1.4. *Le problème de correspondance de Post est indécidable.*

Une preuve de ce résultat peut être trouvée dans [51].

4.2 Codage proposé par N. Durgin et *al.* dans [31]

D'après le codage proposé dans [31], le « protocole »[2] écrits à la figure 4.1 correspondant à l'instance $\mathcal{P}^{\mathsf{exmp}}$ du problème de correspondance de Post définie à l'exemple 4.1.2 est déterminé par les huit « rôles »[2] décrits à la figure 4.1. Notre but ici étant uniquement de pointer les faiblesses que présentent les codages existants nous ne rentrerons pas dans des explications sur l'intuition du codage proposé par N. Durgin et *al.* Nous nous contenterons de montrer à l'aide de cet exemple que les « protocoles » ainsi construits sont plus des artefacts du modèle, que des « vrais » protocoles tels que nous les envisageons communément.

Les termes K, L et R dénotent trois clés symétriques long-terme partagées entre les 16 entités a_0, b_0, …, a_7, $b_7 \in \mathcal{A}$. Afin de modéliser de telles clés il faudrait ajouter à notre modèle trois constructeurs de clés symétriques long-terme shk_1, shk_2 et shk_3 qui prendraient comme arguments des sous-ensembles finis de $\mathcal{A}$. Nous pourrions alors noter $\mathsf{K} = \mathsf{shk}_1(a_0, b_0 \ldots, a_7, b_7)$, $\mathsf{L} = \mathsf{shk}_2(a_0, b_0 \ldots, a_7, b_7)$, et $\mathsf{R} = \mathsf{shk}_3(a_0, b_0 \ldots, a_7, b_7)$. La première critique émise par S. Fröschle dans [33] concerne précisément ces clés partagées par plus de deux entités de $\mathcal{A}$. En effet, quoiqu'il ne soit pas déraisonnable de vouloir modéliser de telles clés, il n'en reste pas moins que les protocoles existants

[2] Entre guillemets, car comme nous l'expliquons dans ce qui suit, il ne s'agît pas de protocoles ni de rôles tels que formellement définis en 2.4.2.

Initialisation

$r_0^{\mathsf{exmp}} = [\quad \mathsf{snd}(a_0, b_0, \quad \{1,2\}_{\mathsf{K}} \quad]$

Concaténation du Couple $(c, cdd) \in \mathcal{P}^{\mathsf{exmp}}$

$r_1^{\mathsf{exmp}} = [\quad \mathsf{rcv}(a_1, b_1, \quad \{x_1, x_2\}_{\mathsf{K}});$
$\mathsf{snd}(a_1, b_1, \quad \{x_1, c, n_1\}_{\mathsf{L}},$
$\{x_2, c, n_2\}_{\mathsf{R}}, \{n_2, d, n_3\}_{\mathsf{R}}, \{n_3, d, n_4\}_{\mathsf{R}}$
$\{n_1, n_4\}_{\mathsf{K}}) \quad]$

Concaténation du Couple $(cd, cdd) \in \mathcal{P}^{\mathsf{exmp}}$

$r_2^{\mathsf{exmp}} = [\quad \mathsf{rcv}(a_2, b_2, \quad \{x_3, x_4\}_{\mathsf{K}});$
$\mathsf{snd}(a_2, b_2, \quad \{x_3, c, n_5\}_{\mathsf{L}}, \{n_5, d, n_6\}_{\mathsf{L}}$
$\{x_4, c, n_7\}_{\mathsf{R}}, \{n_7, d, n_8\}_{\mathsf{R}}, \{n_8, d, n_9\}_{\mathsf{R}}$
$\{n_6, n_9\}_{\mathsf{K}}) \quad]$

Codage du Passage à la Phase de Vérification

$r_3^{\mathsf{exmp}} = [\quad \mathsf{rcv}(a_3, b_3, \quad \{x_5, x_6\}_{\mathsf{K}});$
$\mathsf{snd}(a_3, b_3, \quad \{\langle \$, x_5\rangle, \langle \$, x_6\rangle\}_{\mathsf{K}}) \quad]$

Vérification Un à Un des Couples de Lettres du Mot de Post Construit

$r_4^{\mathsf{exmp}} = [\quad \mathsf{rcv}(a_4, b_4, \quad \{\langle \$, x_7\rangle, \langle \$, x_8\rangle\}_{\mathsf{K}},$
$\{x_9, c, x_7\}_{\mathsf{L}},$
$\{x_{10}, c, x_8\}_{\mathsf{R}});$
$\mathsf{snd}(a_4, b_4, \quad \{\langle \$, x_9\rangle, \langle \$, x_{10}\rangle\}_{\mathsf{K}} \quad]$

$r_5^{\mathsf{exmp}} = [\quad \mathsf{rcv}(a_5, b_5, \quad \{\langle \$, x_{11}\rangle, \langle \$, x_{12}\rangle\}_{\mathsf{K}},$
$\{x_{13}, d, x_{11}\}_{\mathsf{L}},$
$\{x_{14}, d, x_{12}\}_{\mathsf{R}});$
$\mathsf{snd}(a_5, b_5, \quad \{\langle \$, x_{13}\rangle, \langle \$, x_{14}\rangle\}_{\mathsf{K}} \quad]$

Vérification du Dernier Couple de Lettres du Mot de Post Construit
et Révélation du Secret (n_{10} ou n_{11}) si ces Deux Dernières Lettres Correspondent

$r_6^{\mathsf{exmp}} = [\quad \mathsf{rcv}(a_6, b_6, \quad \{\langle \$, x_{15}\rangle, \langle \$, x_{16}\rangle\}_{\mathsf{K}},$
$\{1, c, x_{15}\}_{\mathsf{L}},$
$\{2, c, x_{16}\}_{\mathsf{R}});$
$\mathsf{snd}(a_6, b_6, \quad n_{10});$
$\mathsf{Secret}(a_6, a_0, b_0, \ldots, a_7, b_7, n_{10}) \quad]$

$r_7^{\mathsf{exmp}} = [\quad \mathsf{rcv}(a_7, b_7, \quad \{\langle \$, x_{17}\rangle, \langle \$, x_{18}\rangle\}_{\mathsf{K}},$
$\{1, d, x_{17}\}_{\mathsf{L}},$
$\{2, d, x_{18}\}_{\mathsf{R}});$
$\mathsf{snd}(a_7, b_7, \quad n_{11});$
$\mathsf{Secret}(a_7, a_0, b_0, \ldots, a_7, b_7, n_{11}) \quad]$

Fig. 4.1: « Protocole » associé à l'instance du problème de correspondance de Post $\mathcal{P}^{\mathsf{exmp}}$ d'après le codage proposé dans [31]

n'en font pas l'usage. Toutes les clés symétriques long-terme mises en œuvre dans de « vrais » protocoles de sécurité sont partagées entre exactement deux participants[3].

La seconde critique émise pas S. Fröschle dans [33] reprend le problème de l'existence d'une exécution honnête pointé initialement par H. Comon-Lundh et V. Cortier dans [19]. Cette critique concerne l'exécutabilité de certains événements spécifiés dans le « protocole » décrit à la figure 4.1. Il est clair qu'aucune exécution de ce « protocole » ne peut impliquer de session non-compromise incarnant le « rôle » r_6^{exmp} ou encore le « rôle » r_7^{exmp}. En effet, soient α_0, β_0, …, α_7, β_7 seize agents non-compromis, aucun message de la forme

$$\langle\{\langle\$, t_1\rangle, \langle\$, t_2\rangle\}_{\mathsf{K}}, \langle\{1, c, t_3\}_{\mathsf{L}}, \{2, c, t_4\}_{\mathsf{R}}\rangle\rangle,$$

ni de la forme

$$\langle\{\langle\$, t_1\rangle, \langle\$, t_2\rangle\}_{\mathsf{K}}, \langle\{1, d, t_3\}_{\mathsf{L}}, \{2, d, t_4\}_{\mathsf{R}}\rangle\rangle,$$

où K, L et R sont trois clés symétriques long-terme partagées entre les seize participants α_0, β_0, …, α_7, β_7, n'est jamais émis et ce quelle que soit l'exécution du « protocole » décrit à la figure 4.1, puisque notre instance du problème de correspondance de Post $\mathcal{P}^{\mathsf{exmp}}$ n'admet pas de solution. De plus, l'intrus ne peut synthétiser par lui-même de tels messages, car α_0, β_0, …, α_7, et β_7 sont non-compromis, et l'intrus n'a, au vu du « protocole » associé à $\mathcal{P}^{\mathsf{exmp}}$, aucun moyen d'apprendre K, L, ou R. Les événements des « rôles » r_6^{exmp} et r_7^{exmp} n'admettent donc aucune instantiation non-compromise ; la description du « protocole » spécifie donc des événements qui ne peuvent être exécutés au cours d'aucune session non-compromise.

En d'autres termes, soit un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de $\Pi_{\mathcal{P}^{\mathsf{exmp}}}$, avec $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$. Si une session sid_j ($j \in [\![\ell]\!]$) est non-compromise, alors le « rôle » incarné par sid_j ne peut être ni le « rôle » r_6^{exmp} ($r_j \neq r_6^{\mathsf{exmp}}$), ni le « rôle » r_7^{exmp} ($r_j \neq r_7^{\mathsf{exmp}}$). Et donc aucune exécution honnête de ces deux « rôles » n'est possible. C'est en ce sens que nous disons d'ailleurs que le « protocole » n'admet pas d'exécution honnête, ou encore qu'il n'est pas honnêtement exécutable ; il spécifie des événements qui ne pourront être exécutés au cours d'aucune session non-compromise.

A noter que ce « protocole » ne satisfait pas les conditions 3a et 3b de la définition 2.4.2, *i.e.* il n'admet pas d'exécution type, alors que celles-ci nous assurent de l'existence d'une exécution honnête. Ainsi le « protocole » décrit à la figure 4.1 n'en est de toutes façons pas un au sens de la définition 2.4.2, et plus généralement les « protocoles » associés à des instances du problème de correspondance de Post n'admettant aucune solution ne sont pas des protocoles au sens de la définition 2.4.2.

4.3 Vers un codage plus réaliste

Nous présentons à présent un codage des instances du problème de correspondance de Post, très proche de celui proposé dans [31], mais mettant en

[3] Il existe bien sûrs les protocoles dits *de groupes* dont le but est précisément d'établir des clés symétriques partagées entre un nombre non-borné de participants ; ce type de protocoles est néanmoins traité de façon indépendante, en faisant appel à des modèles et des techniques différentes, et ne rentre pas dans le cadre que nous nous sommes fixé.

œuvre uniquement des protocoles honnêtement exécutables, dont les clés symétriques long-terme sont partagées entre exactement deux participants. Les protocoles ainsi obtenus satisfont toutes les conditions de la définition 2.4.2, et admettent donc bien une exécution honnête.

4.3.1 Le codage

Soient $\Sigma = \{\mathfrak{f}_1, \ldots, \mathfrak{f}_r\}$ un alphabet, et $\mathcal{P} = \{(u_i, v_i)\}_{i \in [\![h]\!]}$ une instance du problème de correspondance de Post sur l'alphabet Σ, avec pour tout $i \in [\![h]\!]$, $u_i = u_i^1 \ldots u_i^{p_i}$ et $v_i = v_i^1 \ldots v_i^{q_i}$ (où $u_i^j, \in \Sigma$ pour tout $j \in [\![p_i]\!]$, et $v_i^j \in \Sigma$ pour tout $j \in [\![q_i]\!]$). Nous noterons $\Pi_{\mathcal{P}}$ le protocole associé à $\mathcal{P}$, $\phi_{\mathcal{P}}$ la propriété du secret associée à $\mathcal{P}$.

Soient les $r+4$ constantes distinctes $\{0, 1, 2, 3, \mathfrak{f}_1, \ldots, \mathfrak{f}_r\} \subseteq \mathcal{C}$, ainsi que les $2(h+3)$ agents distincts $\{a_0, b_0, \ldots, a_{h+2}, b_{h+2}\} \subseteq \mathcal{A}$. Soient aussi les $9 + 2(\sum_{1 \leq i \leq h} p_i + q_i + 4)$ nonces distincts $N \subseteq \mathcal{N}$, où

$$\begin{aligned} N \;=\; & \{n\} \\ \uplus \; & \biguplus_{1 \leq i \leq h} \; \biguplus_{0 \leq j \leq p_i + 1} \{n_{(i,j)}, n_{(h+i,j)}\} \\ \uplus \; & \biguplus_{1 \leq i \leq h} \; \biguplus_{0 \leq j \leq q_i + 1} \{m_{(i,j)}, m_{(h+i,j)}\} \\ \uplus \; & \{n_{(2h+1,0)}, n_{(2h+2,0)}, n_{(2h+2,1)}, n_{(2h+2,2)}, \\ & m_{(2h+1,0)}, m_{(2h+2,0)}, m_{(2h+2,1)}, m_{(2h+2,2)}\}, \end{aligned}$$

et les $9 + 2(\sum_{1 \leq i \leq h} p_i + q_i + 4)$ variables distinctes $V \subseteq \mathcal{V}$, où

$$\begin{aligned} V \;=\; & \{x\} \\ \uplus \; & \biguplus_{1 \leq i \leq h} \; \biguplus_{0 \leq j \leq p_i + 1} \{x_{(i,j)}, x_{(h+i,j)}\} \\ \uplus \; & \biguplus_{1 \leq i \leq h} \; \biguplus_{0 \leq j \leq q_i + 1} \{y_{(i,j)}, y_{(h+i,j)}\} \\ \uplus \; & \{x_{(2h+1,0)}, x_{(2h+2,0)}, x_{(2h+2,1)}, x_{(2h+2,2)}, \\ & y_{(2h+1,0)}, y_{(2h+2,0)}, y_{(2h+2,1)}, y_{(2h+2,2)}\}. \end{aligned}$$

Les mots de $\mathcal{P}$ seront comme dans [31] représentés par des couples de chaînes dont les maillons seront reliés par des nonces. Par exemple, le mot $(ccd, cddcdd)$ de $\mathcal{P}^{\mathsf{exmp}}$ sera représenté par deux chaînes notées $\overline{ccd}$ et $\overline{cddcdd}$, de la forme

$$\overline{ccd} \;=\; \{2, \nu_1, c, \nu_2\}_{\mathsf{shk}(\alpha,\beta)}, \{2, \nu_2, c, \nu_3\}_{\mathsf{shk}(\alpha,\beta)}, \{2, \nu_3, d, \nu_4\}_{\mathsf{shk}(\alpha,\beta)}$$

et

$$\begin{aligned} \overline{cddcdd} \;=\; & \{3, \mu_1, c, \mu_2\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_2, d, \mu_3\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_3, d, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}, \\ & \{3, \mu_4, c, \mu_5\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_5, d, \mu_6\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_6, d, \mu_7\}_{\mathsf{shk}(\alpha,\beta)} \end{aligned}$$

où les ν_i et les μ_j (pour $i \in [\![4]\!]$ et $j \in [\![7]\!]$) sont des nonces, et $\alpha, \beta \in \mathcal{A}$. Comme le laisse transparaître cet exemple, en fonction de s'il s'agît du membre gauche ou du membre droit d'un mot de $\mathcal{P}$, les maillons seront marqués avec

la constante 2 ou 3 respectivement. Nous dirons que le nonce ν_1 (respectivement le nonce μ_1) est en tête de $\overline{ccd}$ (respectivement de $\overline{cddcdd}$), et que le nonce ν_4 (respectivement le nonce μ_7) est en queue de $\overline{ccd}$ (respectivement de $\overline{cddcdd}$).

Nous construisons le protocole

$$\Pi_{\mathcal{P}} \;=\; \Pi^0_{\mathcal{P}} \;@\; \Pi^1_{\mathcal{P}} \;@\; \ldots \;@\; \Pi^{h+2}_{\mathcal{P}}.$$

Ce protocole comporte trois phases : l'initialisation (sous-protocole $\Pi^0_{\mathcal{P}}$), la concaténation (sous-protocole $\Pi^1_{\mathcal{P}}@\ldots@\Pi^h_{\mathcal{P}}$), et la vérification (sous-protocole $\Pi^{h+1}_{\mathcal{P}}@\Pi^{h+2}_{\mathcal{P}}$). Chacune de ces étapes est présentée séparément.

L'initialisation : le sous-protocole $\Pi^0_{\mathcal{P}}$. Le sous-protocole $\Pi^0_{\mathcal{P}}$ est le protocole d'initialisation. Avant de le définir formellement, nous présentons le sous-protocole d'initialisation associé à notre exemple $\mathcal{P}^{\mathsf{exmp}}$ dans le modèle plus intuitif d'Alice et Bob .

$\underline{\Pi^0_{\mathcal{P}^{\mathsf{exmp}}}}$

$$\begin{aligned}
b_0 \to a_0 :\ & \{2, n_0, n, n_1\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_0, n, m_1\}_{\mathsf{shk}(a_0,b_0)}, \\
& \left.\begin{array}{l} \{2, n_1, c, n_2\}_{\mathsf{shk}(a_0,b_0)}, \\ \{3, m_1, c, m_2\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_2, d, m_3\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_3, d, m_4\}_{\mathsf{shk}(a_0,b_0)}, \end{array}\right\} (\overline{c}, \overline{cdd}) \\
& \{1, n_2, m_4\}_{\mathsf{shk}(a_0,b_0)}, \\
& \\
& \{2, n_3, n, n_4\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_5, n, m_6\}_{\mathsf{shk}(a_0,b_0)}, \\
& \left.\begin{array}{l} \{2, n_4, c, n_5\}_{\mathsf{shk}(a_0,b_0)}, \{2, n_5, d, n_6\}_{\mathsf{shk}(a_0,b_0)}, \\ \{3, m_6, c, m_7\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_7, d, m_8\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_8, d, m_9\}_{\mathsf{shk}(a_0,b_0)}, \end{array}\right\} (\overline{cd}, \overline{cdd}) \\
& \{1, n_6, m_9\}_{\mathsf{shk}(a_0,b_0)},
\end{aligned}$$

où les n_i et les m_j (pour $i \in [\![6]\!]$ et $j \in [\![9]\!]$) sont des nonces.

Imaginons que nous souhaitions construire une représentation d'un mot de $\mathcal{P}^{\mathsf{exmp}}$ commençant par le couple (c, cdd), disons le mot $(ccd, cddcdd)$. Il suffit d'ouvrir une session initiée par un agent β incarnant le rôle r_{b_0} ; nous obtiendrons alors une représentation de (c, cdd) de la forme

$$\begin{aligned}
& \{2, \nu_1, c, \nu_2\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{3, \mu_1, c, \mu_2\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_2, d, \mu_3\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_3, d, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}
\end{aligned}$$

où ν_1, ν_2, μ_1, μ_2, μ_3 et μ_4 sont quatre nonces frais.

De plus, à l'issue de cette session, β aura engendré un secret ν qui sera lié à la représentation du couple (c, cdd) grâce aux deux nonces ν_1, et μ_1 apparaissant aussi bien en tête de la représentation de c et de cdd respectivement, que dans des messages contenant le secret de la forme $\{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha,\beta)}$ et $\{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha,\beta)}$.

L'agent β émet aussi le message $\{1, \nu_2, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}$ contenant les nonces en queue des représentation de c et de cdd. Ce message, nous permettra comme

nous allons le voir un peu plus loin de concaténer à la suite de notre représentation du mot (c, cdd) de $\mathcal{P}^{\mathsf{exmp}}$ le couple $(cd, cdd) \in \mathcal{P}^{\mathsf{exmp}}$, et de construire une représentation du mot $(ccd, cddcdd)$ que nous visons.

Formellement et en toute généralité, $\Pi^0_{\mathcal{P}}$ correspond à la séquence d'envois-réceptions suivante :

$$
\begin{aligned}
\Pi^0_{\mathcal{P}} = [\,&\mathsf{snd}(b_0, a_0, \{2, n_{(1,0)}, n, n_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(1,0)}, n, m_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)}\\
&\{2, n_{(1,1)}, u_1^1, n_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{2, n_{(1,p_1)}, u_1^{p_1}, n_{(1,p_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{3, m_{(1,1)}, v_1^1, m_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{3, m_{(1,q_1)}, v_1^{q_1}, m_{(1,q_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{1, n_{(1,p_1+1)}, m_{(1,q_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\ldots\\
&\{2, n_{(h,0)}, n, n_{(h,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(h,0)}, n, m_{(h,1)}\}_{\mathsf{shk}(a_0,b_0)}\\
&\{2, n_{(h,1)}, u_h^1, n_{(h,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{2, n_{(h,p_h)}, u_h^{p_h}, n_{(h,p_h+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{3, m_{(h,1)}, v_h^1, m_{(h,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{3, m_{(h,q_h)}, v_h^{q_h}, m_{(h,q_h+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{1, n_{(h,p_h+1)}, m_{(h,q_h+1)}\}_{\mathsf{shk}(a_0,b_0)});\\
&\mathsf{Secret}(b_0, a_0, b_0, \ldots, a_{h+2}, b_{h+2}, n);\\
&\mathsf{rcv}(a_0, b_0, \{2, x_{(1,0)}, x, x_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(1,0)}, x, y_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)}\\
&\{2, x_{(1,1)}, u_1^1, x_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{2, x_{(1,p_1)}, u_1^{p_1}, x_{(1,p_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{3, y_{(1,1)}, v_1^1, y_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{3, y_{(1,q_1)}, v_1^{q_1}, y_{(1,q_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{1, x_{(1,p_1+1)}, y_{(1,q_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\ldots\\
&\{2, x_{(h,0)}, x, x_{(h,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(h,0)}, x, y_{(h,1)}\}_{\mathsf{shk}(a_0,b_0)}\\
&\{2, x_{(h,1)}, u_h^1, x_{(h,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{2, x_{(h,p_h)}, u_h^{p_h}, x_{(h,p_h+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{3, y_{(h,1)}, v_h^1, y_{(h,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{3, y_{(h,q_h)}, v_h^{q_h}, y_{(h,q_h+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
&\{1, x_{(h,p_h+1)}, y_{(h,q_h+1)}\}_{\mathsf{shk}(a_0,b_0)})]
\end{aligned}
$$

$\Pi^0_{\mathcal{P}}$ construit une représentation de chacun des couple de $\mathcal{P}$. Le couple $(u_i, v_i) \in \mathcal{P}$ (avec $i \in [\![h]\!]$) est représenté par les deux chaînes $\overline{u_i}$ et $\overline{v_i}$ avec

$$
\begin{aligned}
\overline{u_i} &= \{2, n_{(i,1)}, u_i^1, n_{(i,2)}\}, \ldots, \{2, n_{(i,p_i)}, u_i^{p_i}, n_{(i,p_i+1)}\}_{\mathsf{shk}(a_0,b_0)}\\
\overline{v_i} &= \{3, m_{(i,1)}, v_i^1, m_{(i,2)}\}, \ldots, \{3, m_{(i,q_i)}, v_i^{q_i}, m_{(i,q_i+1)}\}_{\mathsf{shk}(a_0,b_0)}
\end{aligned}
$$

A l'issue de cette première phase, le secret n est lié à chaque couple (u_i, v_i) grâces aux nonces $n_{(i,1)}$ et $m_{(i,1)}$ apparaissant aussi bien en tête de la représentation de u_i et de v_i respectivement, que dans les messages contenant le secret n : $\{2, n_{(i,0)}, n, n_{(i,1)}\}_{\mathsf{shk}(a_0,b_0)}$ et $\{3, m_{(i,0)}, n, m_{(i,1)}\}_{\mathsf{shk}(a_0,b_0)}$. Nous avons eu le besoin de lier le secret du protocole au cours de cette phase à chacun des couples de $\mathcal{P}$, car toute solution de $\mathcal{P}$ est un k-uplet avec $k > 0$.

Le deuxième événement de $\Pi^0_{\mathcal{P}}$ nous permettra de spécifier n comme étant le secret de $\Pi_{\mathcal{P}}$.

Notons que nous avons dû explicitement spécifier la réception par a_0 du message émis par b_0 afin que notre protocole admette une substitution honnête $\delta_{\Pi^0_{\mathcal{P}}}$, et soit par là-même honnêtement exécutable.

D'après la représentation des mots de $\mathcal{P}$ que nous avons choisi, afin de concaténer à la suite de (u_k, v_k) le couple (u_j, v_j) (avec $k, j \in [\![h]\!]$), il faut construire une représentation de (u_j, v_j), soit $(\overline{u_j}, \overline{v_j})$, commençant par les nonces en queue de la représentation de (u_k, v_k), *i.e.* les nonces en têtes de $\overline{u_j}$ et de $\overline{v_j}$ de-

vraient être $n_{(k,p_k+1)}$ et $m_{(k,q_k+1)}$ respectivement. C'est selon cette idée qu'est construit, comme nous allons le voir à présent, le sous-protocole $\Pi^j_{\mathcal{P}}$.

Concaténation : le sous-protocole $\Pi^1_{\mathcal{P}}@\ldots@\Pi^h_{\mathcal{P}}$. $\Pi^i_{\mathcal{P}}$, avec $i \in [\![h]\!]$, est le sous-protocole qui permet de concaténer à la suite d'un mot de $\mathcal{P}$ le couple (u_i, v_i). Avant de le définir formellement, nous présentons les sous-protocoles $\Pi^1_{\mathcal{P}^{\mathsf{exmp}}}$ et $\Pi^2_{\mathcal{P}^{\mathsf{exmp}}}$ associés à notre exemple $\mathcal{P}^{\mathsf{exmp}}$, dans le modèle plus intuitif d'Alice et Bob.

$\underline{\Pi^1_{\mathcal{P}^{\mathsf{exmp}}}}$

$a_1 \to b_1 : \{1, n_7, m_{10}\}_{\mathsf{shk}(a_1,b_1)}$

$b_1 \to a_1 : \left.\begin{array}{l}\{2, n_7, c, n_8\}_{\mathsf{shk}(a_1,b_1)},\\ \{3, m_{10}, c, m_{11}\}_{\mathsf{shk}(a_1,b_1)}, \{3, m_{11}, d, m_{12}\}_{\mathsf{shk}(a_1,b_1)}, \{3, m_{12}, d, m_{13}\}_{\mathsf{shk}(a_1,b_1)},\end{array}\right\} \overline{(c, cdd)}$

$\{1, n_8, m_{13}\}_{\mathsf{shk}(a_1,b_1)}$

$\underline{\Pi^2_{\mathcal{P}^{\mathsf{exmp}}}}$

$a_2 \to b_2 : \{1, n_9, m_{14}\}_{\mathsf{shk}(a_2,b_2)}$

$b_2 \to a_2 : \left.\begin{array}{l}\{2, n_9, c, n_{10}\}_{\mathsf{shk}(a_2,b_2)}, \{2, n_{10}, d, n_{11}\}_{\mathsf{shk}(a_2,b_2)},\\ \{3, m_{14}, c, m_{15}\}_{\mathsf{shk}(a_2,b_2)}, \{3, m_{15}, d, m_{16}\}_{\mathsf{shk}(a_2,b_2)}, \{3, m_{16}, d, m_{17}\}_{\mathsf{shk}(a_2,b_2)}\end{array}\right\} \overline{(cd, cdd)}$

$\{1, n_{11}, m_{17}\}_{\mathsf{shk}(a_2,b_2)},$

Soit le mot de $\mathcal{P}^{\mathsf{exmp}}$, (c, cdd) et sa représentation

$$\{2, \nu_1, c, \nu_2\}_{\mathsf{shk}(\alpha,\beta)}$$
$$\{3, \mu_1, c, \mu_2\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_2, d, \mu_3\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_3, d, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}.$$

à notre disposition. Pour concaténer le couple (cd, cdd) à la suite du mot (c, cdd) de $\mathcal{P}^{\mathsf{exmp}}$, et obtenir une représentation du mot $(ccd, cddcdd)$ de $\mathcal{P}^{\mathsf{exmp}}$, il faudra ouvrir une session du rôle r_{b_1} incarné par l'agent β, et fournir à β le message $\{1, \nu_2, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}$; ce dernier nous a été fourni rappelons-le, en même temps que la représentation de (c, cdd). L'agent β construira alors une représentation $(\overline{cd}, \overline{cdd})$ de (cd, cdd) telle que le nonce en tête de $\overline{cd}$ soit ν_2, et celui en tête de $\overline{cdd}$ soit μ_4. En d'autres termes, il retournera un message de la forme

$$\{2, \nu_2, c, \nu_3\}_{\mathsf{shk}(\alpha,\beta)}, \{2, \nu_3, c, \nu_4\}_{\mathsf{shk}(\alpha,\beta)}$$
$$\{3, \mu_4, c, \mu_5\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_5, d, \mu_6\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_6, d, \mu_7\}_{\mathsf{shk}(\alpha,\beta)}$$

où ν_3, ν_4, μ_5, μ_6 et μ_7 seront des nonces fraîchement engendrés. De la même façon, pour pouvoir concaténer à la suite de $(ccd, cddcdd)$ un autre couple de $\mathcal{P}^{\mathsf{exmp}}$, nous aurons besoin du message $\{1, \nu_4, \mu_7\}_{\mathsf{shk}(\alpha,\beta)}$, ce qui explique le dernier message envoyé par b_1.

Formellement et en toute généralité, $\Pi^i_{\mathcal{P}}$ correspond à la séquence d'envois-réceptions suivante :

$$\begin{aligned}\Pi^i_{\mathcal{P}} = [&\mathsf{snd}(a_i, b_i, \{1, n_{(h+i,1)}, m_{(h+i,1)}\}_{\mathsf{shk}(a_i,b_i)});\\ &\mathsf{rcv}(b_i, a_i, \{1, x_{(h+i,1)}, y_{(h+i,1)}\}_{\mathsf{shk}(a_i,b_i)});\\ &\mathsf{snd}(b_i, a_i, \{2, x_{(h+i,1)}, u_i^1, n_{(h+i,2)}\}_{\mathsf{shk}(a_i,b_i)}, \ldots, \{2, n_{(h+i,p_i)}, u_i^{p_i}, n_{(h+i,p_i+1)}\}_{\mathsf{shk}(a_i,b_i)},\\ &\qquad \{3, y_{(h+i,1)}, v_i^1, m_{(h+i,2)}\}_{\mathsf{shk}(a_i,b_i)}, \ldots, \{3, m_{(h+i,q_i)}, v_i^{q_i}, m_{(h+i,q_i+1)}\}_{\mathsf{shk}(a_i,b_i)},\\ &\qquad \{1, n_{(h+i,p_i+1)}, m_{(h+i,q_i+1)}\}_{\mathsf{shk}(a_i,b_i)});\\ &\mathsf{rcv}(a_i, b_i, \{2, n_{(h+i,1)}, u_i^1, x_{(h+i,2)}\}_{\mathsf{shk}(a_i,b_i)}, \ldots, \{2, x_{(h+i,p_i)}, u_i^{p_i}, x_{(h+i,p_i+1)}\}_{\mathsf{shk}(a_i,b_i)},\\ &\qquad \{3, m_{(h+i,1)}, v_i^1, y_{(h+i,2)}\}_{\mathsf{shk}(a_i,b_i)}, \ldots, \{3, y_{(h+i,q_i)}, v_i^{q_i}, y_{(h+i,q_i+1)}\}_{\mathsf{shk}(a_i,b_i)},\\ &\qquad \{1, x_{(h+i,p_i+1)}, y_{(h+i,q_i+1)}\}_{\mathsf{shk}(a_i,b_i)})]\end{aligned}$$

Soit (u, v) un mot de $\mathcal{P}$ représenté par $(\overline{u}, \overline{v})$ avec n le nonce en queue de $\overline{u}$, et m le nonce en queue de $\overline{v}$. Pour concaténer le couple (u_i, v_i) à la suite du mot (u, v) et obtenir une représentation du mot (uu_i, vv_i) de $\mathcal{P}$, il faudra fournir à b_i le message $\{1, n, m\}_{\mathsf{shk}(a_i,b_i)}$. L'agent b_i construira alors une représentation $(\overline{u_i}, \overline{v_i})$ de (u_i, v_i) telle que le nonce en tête de $\overline{u_i}$ soit n, et celui en tête de $\overline{v_i}$ soit m.

Une fois que l'agent b_i a concaténé à la suite de la représentation du mot (u, v) de $\mathcal{P}$ la représentation du couple (u_i, v_i), il émet le message

$$\{1, n_{(h+i,p_i+1)}, m_{(h+i,q_i+1)}\}_{\mathsf{shk}(a_i,b_i)},$$

où $n_{(h+i,p_i+1)}$ est le nonce en queue de $\overline{u_i}$ et $m_{(h+i,q_i+1)}$ le nonce en queue de $\overline{v_i}$. Ainsi, si pour construire une solution de $\mathcal{P}$ il faut concaténer la représentation $(\overline{u_j}, \overline{v_j})$ d'un couple $(u_j, v_j) \in \mathcal{P}$ à la suite de la représentation du mot (uu_i, vv_i), le couple de nonces en queue de $(\overline{uu_i}, \overline{vv_i})$ nécessaire sera disponible.

Là encore les actions de l'agent a_i ne sont spécifiées que pour que $\Pi^i_{\mathcal{P}}$ admette une exécution honnête, *i.e.* que $\delta_{\Pi^i_{\mathcal{P}}}$ soit défini.

A ce stade, nous sommes capables de construire une représentation de chaque mot de $\mathcal{P}$. Le reste du protocole $\Pi_{\mathcal{P}}$ est consacré à vérifier si un mot de $\mathcal{P}$ donné est une solution de $\mathcal{P}$.

La vérification : le sous-protocole $\Pi^{h+1}_{\mathcal{P}}$@$\Pi^{h+2}_{\mathcal{P}}$. Vérifier si la représentation d'un mot (u, v) de $\mathcal{P}$ est une solution, consiste à vérifier si les lettres de u et de v correspondent deux à deux. La vérification se fait en lisant u et v de la droite vers la gauche. Pour concaténer un couple (u_i, v_i) à la suite d'un mot (u, v), il fallait pouvoir construire une représentation $(\overline{u}, \overline{v})$ du mot (u, v) de $\mathcal{P}$ considéré, ainsi que le message $\{1, n, m\}_{\mathsf{shk}(a_i,b_i)}$ où n est le nonce en queue de la représentation de u construite, et m celui en queue de la représentation de v construite. Maintenant, pour vérifier qu'un mot (u, v) est une présolution de $\mathcal{P}$ il faudra pouvoir construire une représentation $(\overline{u}, \overline{v})$ de (u, v) considéré, ainsi que le message $\{0, n, m\}_{\mathsf{shk}(\alpha,\beta)}$, pour certains agents $\alpha, \beta \in \mathcal{A}$, où n est le nonce en queue de la représentation $\overline{u}$, et m celui en queue de la représentation $\overline{v}$.

$\Pi^{h+1}_{\mathcal{P}}$ code le passage à la phase de vérification. D'après la discussion menée à l'instant, une fois la représentation du mot de $\mathcal{P}$ visé construite, il faut vérifier que les deux mots le constituant sont identiques. Formellement, dans notre modèle, $\Pi^{h+1}_{\mathcal{P}}$ correspond à la séquence d'envois-réceptions suivante :

$$\begin{aligned}\Pi_{\mathcal{P}}^{h+1} = [\quad & \mathsf{snd}(a_{h+1}, b_{h+1}, \quad \{1, n_{(2h+1,0)}, m_{(2h+1,0)}\}_{\mathsf{shk}(a_{h+1},b_{h+1})});\\ & \mathsf{rcv}(b_{h+1}, a_{h+1}, \quad \{1, x_{(2h+1,0)}, y_{(2h+1,0)}\}_{\mathsf{shk}(a_{h+1},b_{h+1})});\\ & \mathsf{snd}(b_{h+1}, a_{h+1}, \quad \{0, x_{(2h+1,0)}, y_{(2h+1,0)}\}_{\mathsf{shk}(a_{h+1},b_{h+1})});\\ & \mathsf{rcv}(a_{h+1}, b_{h+1}, \quad \{0, n_{(2h+1,0)}, m_{(2h+1,0)}\}_{\mathsf{shk}(a_{h+1},b_{h+1})})]\end{aligned}$$

Ainsi défini le sous-protocole $\Pi_{\mathcal{P}}^{h+1}$, nous avons bien que si le message $\{1, n, m\}_{\mathsf{shk}(a_{h+1},b_{h+1})}$ est fourni à l'agent b_{h+1}, avec n le nonce en queue de la représentation de u construite, et m celui en queue de la représentation de v construite, alors celui-ci lance le processus de vérification en retournant le message nécessaire $\{0, n, m\}_{\mathsf{shk}(a_{h+1},b_{h+1})}$.

Reprenons notre mot $(ccd, cddcdd)$ de $\mathcal{P}^{\mathsf{exmp}}$. Nous avons déjà vu que le mot $(ccd, cddcd)$ est une présolution de $\mathcal{P}^{\mathsf{exmp}}$. D'après la description informelle du processus de vérification faite ci-dessus, il faut pouvoir construire le message $\{0, \nu_4, \mu_7\}_{\mathsf{shk}(\alpha,\beta)}$. Pour cela, il faudra ouvrir une session du rôle $r_{b_{h+1}} = r_{b_3}$ incarné par β et fournir à ce dernier le message $\{1, \nu_4, \mu_7\}_{\mathsf{shk}(\alpha,\beta)}$ que nous avons obtenu lors de la concaténation de (cd, cdd) à la suite de (c, cdd). L'agent β construira alors le message $\{0, \nu_4, \mu_7\}_{\mathsf{shk}(\alpha,\beta)}$ voulu.

A présent il faut vérifier une à une les lettres de la présolution de $\mathcal{P}$ construite. C'est le sous-protocole $\Pi_{\mathcal{P}}^{h+2}$ qui s'en charge. Formellement, dans notre modèle, $\Pi_{\mathcal{P}}^{h+2}$ correspond à la séquence d'envois-réceptions suivante :

$$\begin{aligned}\Pi_{\mathcal{P}}^{h+2} = [\quad & \mathsf{snd}(a_{h+2}, b_{h+2}, \quad \{0, n_{(2h+2,1)}, m_{(2h+2,1)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})},\\ & \qquad\qquad \{2, n_{(2h+2,2)}, n_{(2h+2,0)}, n_{(2h+2,1)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})},\\ & \qquad\qquad \{3, m_{(2h+2,2)}, n_{(2h+2,0)}, m_{(2h+2,1)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})});\\ & \mathsf{rcv}(b_{h+2}, a_{h+2}, \quad \{0, x_{(2h+2,1)}, y_{(2h+2,1)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})},\\ & \qquad\qquad \{2, x_{(2h+2,2)}, x_{(2h+2,0)}, x_{(2h+2,1)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})},\\ & \qquad\qquad \{3, y_{(2h+2,2)}, x_{(2h+2,0)}, y_{(2h+2,1)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})});\\ & \mathsf{snd}(b_{h+2}, a_{h+2}, \quad \{0, x_{(2h+2,2)}, y_{(2h+2,2)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})},\\ & \qquad\qquad x_{(2h+2,0)});\\ & \mathsf{rcv}(a_{h+2}, b_{h+2}, \quad \{0, n_{(2h+2,2)}, m_{(2h+2,2)}\}_{\mathsf{shk}(a_{h+2},b_{h+2})},\\ & \qquad\qquad n_{(2h+2,0)})]\end{aligned}$$

Soit (u, v) une présolution de $\mathcal{P}$ de représentation $(\overline{u}, \overline{v})$. Imaginons que $\{2, n, \mathfrak{f}, m\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$ soit le dernier maillon de la représentation $\overline{u}$, et que $\{2, n', \mathfrak{f}, m'\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$ soit le dernier maillon de la représentation $\overline{v}$, avec $\mathfrak{f} \in \Sigma$, *i.e.* $u = u'\mathfrak{f}$ et $v = v'\mathfrak{f}$, pour certains $u', v' \in \Sigma^*$. Donc (u', v') aussi est une présolution de $\mathcal{P}$. A partir de la représentation $(\overline{u}, \overline{v})$ il est trivial de construire une représentation de (u', v'). Néanmoins il manque le message $\{0, n, n'\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$. En fournissant à b_{h+2} les maillons $\{2, n, \mathfrak{f}, m\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$, et $\{3, n', \mathfrak{f}, m'\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$ en queue de la représentation $(\overline{u}, \overline{v})$, ainsi que le message $\{0, m, m'\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$, b_{h+2} retourne, après avoir vérifié que les deux lettres correspondent $(\mathfrak{f} = \mathfrak{f})$, le message $\{0, n, n'\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$ manquant pour continuer la vérification du mot (u, v). Ainsi, il est possible à partir d'une représentation d'un mot de $\mathcal{P}$ de vérifier deux à deux les lettres de sa représentation de la droite vers la gauche.

En effet, si nous ouvrons une session du rôle $r_{b_{h+2}} = r_{b_4}$ et fournissons à l'agent β les messages

$$\{0, \nu_4, \mu_7\}_{\mathsf{shk}(\alpha,\beta)},$$
$$\{2, \nu_3, d, \nu_4\}_{\mathsf{shk}(\alpha,\beta)}, \text{ et}$$
$$\{3, \mu_6, d, \mu_7\}_{\mathsf{shk}(\alpha,\beta)}$$

à notre disposition, celui-ci vérifiera que les deux lettres correspondent, et nous répondra par le message $\{0, \nu_3, \mu_6\}_{\mathsf{shk}(\alpha,\beta)}$ qui nous manquait et nous permettra de continuer notre processus de vérification.

Supposons maintenant que $\mathcal{P}$ admette une solution (u, v). D'après notre codage, il devrait être possible (c'est le cas comme nous le prouverons formellement à la section 4.4) de construire une représentation de (u, v) puis de vérifier de la droite vers la gauche les lettres de (u, v) deux à deux. De plus, si cette représentation à été construite en commençant avec le protocole $\Pi^0_{\mathcal{P}}$ d'initialisation, alors elle est liée à une instance ν du secret n en ce sens qu'il existe quatre nonces ν_0, ν_1, μ_0 et μ_1 tels que ν_1 soit le nonce en tête de la représentation de u, μ_1 soit le nonce en tête de la représentation de v, et que le secret apparaisse dans le message $\{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$ ainsi que dans le message $\{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$. Il sera alors possible de construire le message $\{0, \nu_1, \mu_1\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$, à l'aide d'une session du rôle $r_{b_{h+2}}$. Finalement en fournissant à b_{h+2} les messages $\{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$, $\{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$, et $\{0, \nu_1, \mu_1\}_{\mathsf{shk}(a_{h+2},b_{h+2})}$, ce dernier révélera le secret.

Maintenant que nous avons présenté le protocole associé à $\mathcal{P}$, il nous reste à énoncer la propriété associée à cette même instance. La propriété $\phi_{\mathcal{P}}$ que nous considérons est la suivante :

$$\phi_{\mathcal{P}} = \begin{cases} \forall z_{a_0}.\forall z_{b_0}.\ldots.\forall z_{a_{h+2}}.\forall z_{b_{h+2}}.\forall z_n. \\ \mathcal{O}(\mathsf{Secret}(z_{a_0}, z_{b_0}, \ldots, z_{a_{h+2}}, z_{b_{h+2}}, z_n)) \wedge \bigwedge_{0\le i\le h+2} (\mathsf{NC}(z_{a_i} \wedge \mathsf{NC}(z_{b_i}))) \\ \Rightarrow \\ \neg\mathsf{learn}(z_n). \end{cases}$$

Comme nous l'avons vu au chapitre 3, cette formule spécifie que toute instance honnête de n doit restée secrète de l'intrus, *i.e.* n est spécifié comme étant le secret du protocole $\Pi_{\mathcal{P}}$.

Exemple 4.3.1. *En appliquant le codage ci-dessus à l'instance $\mathcal{P}^{\mathsf{exmp}}$ du problème de correspondance de Post définie à l'exemple 4.1.2, nous obtenons le protocole $\Pi_{\mathcal{P}^{\mathsf{exmp}}}$ décrit aux figures 4.2 et 4.3. Cette séquence d'événements est bien un protocole au sens de la définition 2.4.2, et admet bien une exécution honnête ($\delta_{\Pi_{\mathcal{P}^{\mathsf{exmp}}}} = \{x_{(i,j)} \mapsto n_{(i,j)}, y_{(i,j)} \mapsto m_{(i,j)} \mid i, j \in \mathbb{N}\}$ est bien définie).*

4.3.2 Equivalence entre $\mathcal{P}$ et $(\Pi_{\mathcal{P}}, \emptyset, \phi_{\mathcal{P}})$

L'indécidabilité du problème de la vérification, dans le cadre de messages de taille bornée, repose sur l'équivalence suivante.

INITIALISATION

$\Pi_{\mathcal{P}^{\mathsf{exmp}}} = [$ $\mathsf{snd}(b_0, a_0,$ $\{2, n_{(1,0)}, n, n_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(1,0)}, n, m_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{2, n_{(1,1)}, c, n_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{3, m_{(1,1)}, c, m_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(1,2)}, d, m_{(1,3)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(1,3)}, d, m_{(1,4)}\}_{\mathsf{shk}(a_0,b_0)},$ $\Big\}$ $(\overline{c}, \overline{cdd})$
$\{1, n_{(1,2)}, m_{(1,4)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{2, n_{(2,0)}, n, n_{(2,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(2,0)}, n, m_{(2,1)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{2, n_{(2,1)}, c, n_{(2,2)}\}_{\mathsf{shk}(a_0,b_0)}, \{2, n_{(2,2)}, d, n_{(2,3)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{3, m_{(2,1)}, c, m_{(2,2)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(2,2)}, d, m_{(2,3)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m_{(2,3)}, d, m_{(2,4)}\}_{\mathsf{shk}(a_0,b_0)},$ $\Big\}$ $(\overline{cd}, \overline{cdd})$
$\{1, n_{(2,3)}, m_{(2,4)}\}_{\mathsf{shk}(a_0,b_0)})$;

$\mathsf{Secret}(b_0, a_0, b_0, a_1, b_1, a_2, b_2, a_3, b_3, a_4, b_4, n)$;

$\mathsf{rcv}(a_0, b_0,$ $\{2, x_{(1,0)}, x, x_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(1,0)}, x, y_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{2, x_{(1,1)}, c, x_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{3, y_{(1,1)}, c, y_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(1,2)}, d, y_{(1,3)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(1,3)}, d, y_{(1,4)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{1, x_{(1,2)}, y_{(1,4)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{2, x_{(2,0)}, x, x_{(2,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(2,0)}, x, y_{(2,1)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{2, x_{(2,1)}, c, x_{(2,2)}\}_{\mathsf{shk}(a_0,b_0)}, \{2, x_{(2,2)}, d, x_{(2,3)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{3, y_{(2,1)}, c, y_{(2,2)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(2,2)}, d, y_{(2,3)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, y_{(2,3)}, d, y_{(2,4)}\}_{\mathsf{shk}(a_0,b_0)},$
$\{1, x_{(2,3)}, y_{(2,4)}\}_{\mathsf{shk}(a_0,b_0)})$;

CONCATÉNATION DU COUPLE $(c, cdd) \in \mathcal{P}^{\mathsf{exmp}}$

$\mathsf{snd}(a_1, b_1,$ $\{1, n_{(3,1)}, m_{(3,1)}\}_{\mathsf{shk}(a_1,b_1)})$;
$\mathsf{rcv}(b_1, a_1,$ $\{1, x_{(3,1)}, y_{(3,1)}\}_{\mathsf{shk}(a_1,b_1)})$;
$\mathsf{snd}(b_1, a_1,$ $\{2, x_{(3,1)}, c, n_{(3,2)}\}_{\mathsf{shk}(a_1,b_1)},$
$\{3, y_{(3,1)}, c, m_{(3,2)}\}_{\mathsf{shk}(a_1,b_1)}, \{3, m_{(3,2)}, d, m_{(3,3)}\}_{\mathsf{shk}(a_1,b_1)}, \{3, m_{(3,3)}, d, m_{(3,4)}\}_{\mathsf{shk}(a_1,b_1)},$ $\Big\}$ $(\overline{c}, \overline{cdd})$
$\{1, n_{(3,2)}, m_{(3,4)}\}_{\mathsf{shk}(a_1,b_1)})$;
$\mathsf{rcv}(b_1, a_1,$ $\{2, n_{(3,1)}, c, x_{(3,2)}\}_{\mathsf{shk}(a_1,b_1)},$
$\{3, m_{(3,1)}, c, y_{(3,2)}\}_{\mathsf{shk}(a_1,b_1)}, \{3, y_{(3,2)}, d, y_{(3,3)}\}_{\mathsf{shk}(a_1,b_1)}, \{3, y_{(3,3)}, d, y_{(3,4)}\}_{\mathsf{shk}(a_1,b_1)},$
$\{1, x_{(3,2)}, y_{(3,4)}\}_{\mathsf{shk}(a_1,b_1)})$;

FIG. 4.2: Protocole $\Pi_{\mathcal{P}^{\mathsf{exmp}}}$ associé à l'instance $\mathcal{P}^{\mathsf{exmp}}$ d'après le codage proposé à la section 4.3 (1)

Concaténation du Couple $(cd, cdd) \in \mathcal{P}^{\mathsf{exmp}}$

$\mathsf{snd}(a_2, b_2,$ $\{1, n_{(4,1)}, m_{(4,1)}\}_{\mathsf{shk}(a_2,b_2)})$;
$\mathsf{rcv}(b_2, a_2,$ $\{1, x_{(4,1)}, y_{(4,1)}\}_{\mathsf{shk}(a_2,b_2)})$;
$\mathsf{snd}(b_2, a_2,$ $\{2, x_{(4,1)}, c, n_{(4,2)}\}_{\mathsf{shk}(a_2,b_2)}, \{2, n_{(4,2)}, d, n_{(4,3)}\}_{\mathsf{shk}(a_2,b_2)},$
$\{3, y_{(4,1)}, c, m_{(4,2)}\}_{\mathsf{shk}(a_2,b_2)}, \{3, m_{(4,2)}, d, m_{(4,3)}\}_{\mathsf{shk}(a_2,b_2)}, \{3, m_{(4,3)}, d, m_{(4,4)}\}_{\mathsf{shk}(a_2,b_2)},$ $\Big\}(\overline{cd}, \overline{cdd})$
$\{1, n_{(4,3)}, m_{(4,4)}\}_{\mathsf{shk}(a_2,b_2)})$;
$\mathsf{rcv}(b_2, a_2,$ $\{2, n_{(4,1)}, c, x_{(4,2)}\}_{\mathsf{shk}(a_2,b_2)}, \{2, x_{(4,2)}, d, n_{(4,3)}\}_{\mathsf{shk}(a_2,b_2)},$
$\{3, m_{(4,1)}, c, y_{(4,2)}\}_{\mathsf{shk}(a_2,b_2)}, \{3, y_{(4,2)}, d, y_{(4,3)}\}_{\mathsf{shk}(a_1,b_1)}, \{3, y_{(4,3)}, d, y_{(4,4)}\}_{\mathsf{shk}(a_2,b_2)},$
$\{1, x_{(4,3)}, y_{(4,4)}\}_{\mathsf{shk}(a_2,b_2)})$;

Passage à la Phase de Vérification

$\mathsf{snd}(a_3, b_3,$ $\{1, n_{(5,0)}, m_{(5,0)}\}_{\mathsf{shk}(a_3,b_3)})$;
$\mathsf{rcv}(b_3, a_3,$ $\{1, x_{(5,0)}, y_{(5,0)}\}_{\mathsf{shk}(a_3,b_3)})$;
$\mathsf{snd}(b_3, a_3,$ $\{0, x_{(5,0)}, y_{(5,0)}\}_{\mathsf{shk}(a_3,b_3)})$;
$\mathsf{rcv}(a_3, b_3,$ $\{0, n_{(5,0)}, m_{(5,0)}\}_{\mathsf{shk}(a_3,b_3)})$;

Vérification

$\mathsf{snd}(a_4, b_4,$ $\{0, n_{(6,1)}, m_{(6,1)}\}_{\mathsf{shk}(a_4,b_4)}, \{2, n_{(6,2)}, n_{(6,0)}, n_{(6,1)}\}_{\mathsf{shk}(a_4,b_4)}, \{3, m_{(6,2)}, n_{(6,0)}, m_{(6,1)}\}_{\mathsf{shk}(a_4,b_4)})$;
$\mathsf{rcv}(b_4, a_4,$ $\{0, x_{(6,2)}, y_{(6,1)}\}_{\mathsf{shk}(a_4,b_4)}, \{2, x_{(6,2)}, x_{(6,0)}, x_{(6,1)}\}_{\mathsf{shk}(a_4,b_4)}, \{3, y_{(6,2)}, x_{(6,0)}, y_{(6,1)}\}_{\mathsf{shk}(a_4,b_4)})$;
$\mathsf{snd}(b_4, a_4,$ $\{0, x_{(6,2)}, y_{(6,2)}\},$
$x_{(6,0)})$;
$\mathsf{rcv}(a_4, b_4,$ $\{0, n_{(6,2)}, m_{(6,2)}\},$
$n_{(6,0)})$]

Fig. 4.3: Protocole $\Pi_{\mathcal{P}^{\mathsf{exmp}}}$ associé à l'instance $\mathcal{P}^{\mathsf{exmp}}$ d'après le codage proposé à la section 4.3 (2)

Théorème 4.3.2. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, ainsi que le protocole $\Pi_{\mathcal{P}}$ et la formule $\phi_{\mathcal{P}}$ associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1.*

> *$\mathcal{P}$ admet une solution si et seulement si il existe une exécution valide exec de $\Pi_{\mathcal{P}}$ au regard d'une connaissance initiale de l'intrus vide telle que $\langle \mathsf{exec}, \emptyset \rangle \models \neg\phi_{\mathcal{P}}$, et telle que pour tout $m \in \mathsf{St}(\mathsf{exec})$, $|m| \leq \mathsf{B}(\Pi_{\mathcal{P}})$.*

Rappelons que $\mathsf{B}()$ avait été définie au chapitre 2 (voir définition 2.4.3) comme étant la fonction qui retourne la taille du protocole passé en argument.

La preuve de ce théorème repose sur trois résultats intermédiaires. Nous les énonçons ici afin de pouvoir procéder à la démonstration du théorème au plus vite. Nous remettons leur preuve à la section 4.4.

Formalisons avant tout la notion de représentation de mot de $\mathcal{P}$ et de présolution de $\mathcal{P}$ qui s'est dessinée au cours de notre description du protocole $\Pi_{\mathcal{P}}$.

Définition 4.3.3. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soient aussi $\mathsf{exec} = [e_1; \ldots; e_\ell]$ une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide, et (u, v) un couple de mot sur Σ, tels que $u = \mathsf{f}_{i_1} \ldots \mathsf{f}_{i_p}$ et $v = \mathsf{f}_{j_1} \ldots \mathsf{f}_{j_q}$. Nous dirons que (u, v)* admet dans exec une représentation de type mot de $\mathcal{P}$ *(respectivement* de type présolution de $\mathcal{P}$*) si*

- *il existe $2(h+3)$ agents honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$,*
- *il existe un entier $k \in [\![\ell]\!]$,*
- *il existe $p+q+5$ nonces distincts $\nu, \nu_0, \nu_1, \ldots, \nu_{p+1}, \mu_0, \mu_1, \ldots, \mu_{q+1} \in \mathcal{N}$*

tels que

- $e_k = \mathsf{Secret}(\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}, \nu)$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{T, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{shk(\alpha_0,\beta_0)}$, *et*
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{shk(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, \mathsf{f}_{i_j}, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ *pour tout* $j \in [\![p]\!]$ *et*
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_i, \mathsf{f}_{j_i}, \mu_{i+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ *pour tout* $i \in [\![q]\!]$.

où $T = 1$ (respectivement $T = 0$).

Le premier résultat dont nous avons besoin stipule qu'aucune exécution valide de $\Pi_{\mathcal{P}}$ ne révèle à l'intrus de clés long-terme qu'il ne connaîtrait pas initialement.

Lemme 4.3.4. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soient aussi* exec *une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide, et deux entités honnêtes $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$.*

$$\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta).$$

Nous aurons aussi besoin du résultat suivant selon lequel si $\Pi_{\mathcal{P}}$ viole $\phi_{\mathcal{P}}$, alors $\Pi_{\mathcal{P}}$ admet une exécution valide qui viole $\phi_{\mathcal{P}}$, au regard d'une connaissance

initiale de l'intrus vide, en n'impliquant que des messages de tailles bornée, et qui plus est en n'impliquant que des sessions honnêtes.

Proposition 4.3.5. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Si $\Pi_{\mathcal{P}}$ viole la propriété $\phi_{\mathcal{P}}$ au regard d'une connaissance initiale de l'intrus vide,* i.e. $\langle \Pi_{\mathcal{P}}, \emptyset \rangle \models \neg\phi_{\mathcal{P}}$, *alors il existe deux agents honnêtes $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$ et une exécution* exec *de $\Pi_{\mathcal{P}}$, valide au regard d'une connaissance initiale de l'intrus vide, qui viole $\phi_{\mathcal{P}}$ et telle que*

- *pour tout $i \in [\![\ell]\!]$ et pour tout η, $0 \leq \eta \leq h+2$*

$$r_i \in \{r_{b_\eta}, r_{a_\eta}\} \wedge \mathsf{initagts}(sid_i) = (\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}) \Rightarrow \{\alpha_\eta, \beta_\eta\} = \{\alpha, \beta\}$$

- *pour tout message $m \in \mathsf{Est}(\mathsf{exec})$, m est de la forme*

$$\{T, \nu, \mu\}_{\mathsf{shk}(\alpha,\beta)} \text{ ou } \{T', \nu, \mathfrak{f}, \mu\}_{\mathsf{shk}(\alpha,\beta)} \text{ ou encore } \{T', \nu, \lambda, \mu\}_{\mathsf{shk}(\alpha,\beta)}$$

où $T \in \{0,1\}$, $T' \in \{2,3\}$, $\mathfrak{f} \in \Sigma$, et $\lambda, \mu, \nu \in \mathcal{N}$.

D'après cette proposition et pour une connaissance initiale de l'intrus vide, nous pouvons donc nous restreindre à l'ensemble des exécutions dénoté $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, et défini comme suit.

Définition 4.3.6 ($\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$)**.** *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soient aussi* $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ *un scénario de $\Pi_{\mathcal{P}}$ avec* $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, *et une substitution close σ tels que* $\mathsf{exec} = tr\sigma$ *soit une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide, où tr est la trace symbolique associée à $\Pi_{\mathcal{P}}$.* $\mathsf{exec} \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ *si et seulement si*

- *il existe deux entités honnêtes $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$ telles que pour tout $i \in [\![\ell]\!]$ et pour tout η, $0 \leq \eta \leq h+2$*

$$r_i \in \{r_{b_\eta}, r_{a_\eta}\} \wedge \mathsf{initagts}(sid_i) = (\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}) \Rightarrow \{\alpha_\eta, \beta_\eta\} = \{\alpha, \beta\}$$

- *pour tout message $m \in \mathsf{Est}(\mathsf{exec})$, m est de la forme*

$$\{T, \nu, \mu\}_{\mathsf{shk}(\alpha,\beta)} \text{ ou } \{T', \nu, \mathfrak{f}, \mu\}_{\mathsf{shk}(\alpha,\beta)} \text{ ou encore } \{T', \nu, \lambda, \mu\}_{\mathsf{shk}(\alpha,\beta)}$$

où $T \in \{0,1\}$, $T' \in \{2,3\}$, $\mathfrak{f} \in \Sigma$, et $\lambda, \mu, \nu \in \mathcal{N}$.

Le dernier résultat dont nous aurons besoin afin de procéder à la preuve du théorème 4.3.2 est énoncé ci-après, et établit que pour toute présolution de $\mathcal{P}$ il existe une exécution valide de $\Pi_{\mathcal{P}}$ dans laquelle elle admet une représentation de type présolution ; et réciproquement.

Proposition 4.3.7. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soit aussi (u, v) un couple de mots sur Σ,* i.e. *$u, v \in \Sigma^*$.*

> *(u, v) est une présolution de $\mathcal{P}$ si et seulement si il existe une exécution $\mathsf{exec} \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ telle que (u, v) admette dans exec une représentation de type présolution de $\mathcal{P}$.*

Preuve du théorème 4.3.2 Nous avons désormais tous les ingrédients nécessaires pour mener à bien notre preuve du théorème 4.3.2.

Démonstration. Soient $\mathcal{P} = \{(u_i, v_i)\}_{i \in [\![h]\!]}$ pour un certain $h \in \mathbb{N}^*$ une instance du problème de correspondance de Post, et $\Pi_{\mathcal{P}}$ le protocole et $\phi_{\mathcal{P}}$ la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1

($\Rightarrow$) Supposons que $\mathcal{P}$ admette une solution. Alors par définition $(\varepsilon, \varepsilon)$ est une présolution de $\mathcal{P}$. D'après la proposition 4.3.7 nous savons qu'il existe un scénario sc de $\Pi_{\mathcal{P}}$ ainsi qu'une substitution σ, tels que $\mathsf{exec} = tr\sigma = [e_1; \ldots; e_\ell] \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, où tr est la trace symbolique associée à sc, et tels que $(\varepsilon, \varepsilon)$ admette une représentation de type présolution de $\mathcal{P}$ dans exec, *i.e.*

- il existe $k \in [\![\ell]\!]$,
- il existe $2(h+3)$ entités honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$, et
- il existe 5 nonces distincts $\nu, \nu_0, \nu_1, \mu_0, \mu_1 \in \mathcal{N}$

tels que

- $e_k = \mathsf{Secret}(\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}, \nu)$
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \nu_1, \mu_1\}_{\mathsf{shk}(\alpha_0, \beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0, \beta_0)}$, et
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0, \beta_0)}$.

Soit $\mathsf{sc}' = (\mathsf{interlvg}, \mathsf{initagts})$, avec $\mathsf{interlvg} = [(r_{b_{h+2}}, sid); (r_{b_{h+2}}, sid)]$ où sid est un identificateur de session n'apparaissant pas déjà dans sc, et $\mathsf{initagts}(sid) = (\alpha'_0, \beta'_0, \ldots, \alpha'_{h+2}, \beta'_{h+2})$ quelconque mais avec $\alpha'_{h+2} = \alpha_0$ et $\beta'_{h+2} = \beta_0$. La trace symbolique associée à $\mathsf{sc} \mathbin{@} \mathsf{sc}'$ est

$$
\begin{aligned}
tr' = tr \mathbin{@} [\quad & \mathsf{rcv}(\beta_0, \alpha_0, \quad \{0, x^{sid}_{(2h+2,1)}, y^{sid}_{(2h+2,1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\
& \qquad\qquad\quad \{2, x^{sid}_{(2h+2,2)}, x^{sid}_{(2h+2,0)}, x^{sid}_{(2h+2,1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\
& \qquad\qquad\quad \{3, y^{sid}_{(2h+2,2)}, x^{sid}_{(2h+2,0)}, y^{sid}_{(2h+2,1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}); \\
& \mathsf{snd}(\beta_0, \alpha_0, \quad \{0, x^{sid}_{(2h+2,2)}, y^{sid}_{(2h+2,2)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\
& \qquad\qquad\quad x^{sid}_{(2h+2,0)})]
\end{aligned}
$$

où $x^{sid}_{(2h+2,0)}$, $x^{sid}_{(2h+2,1)}$, $x^{sid}_{(2h+2,2)}$, $y^{sid}_{(2h+2,1)}$, et $y^{sid}_{(2h+2,2)}$ sont des variables fraîches, *i.e.* n'apparaissant pas dans tr. Il est aisé de voir que pour

$$
\sigma' = \{x^{sid}_{(2h+2,0)} \mapsto \nu, x^{sid}_{(2h+2,1)} \mapsto \nu_1, x^{sid}_{(2h+2,2)} \mapsto \nu_0, y^{sid}_{(2h+2,1)} \mapsto \mu_1, y^{sid}_{(2h+2,2)} \mapsto \mu_0\},
$$

$\mathsf{exec}' = tr'\sigma\sigma'$ est une exécution dans $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ et que $\langle \mathsf{exec}', \emptyset \rangle \models \neg\phi_{\mathcal{P}}$. En effet, le secret ν passe en clair sur le réseau.

($\Leftarrow$) Supposons que $\Pi_{\mathcal{P}}$ viole la propriété $\phi_{\mathcal{P}}$. Nous savons alors d'après la proposition 4.3.5 qu'il existe $\mathsf{sc} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$ un scénario de $\Pi_{\mathcal{P}}$ ainsi qu'une substitution σ tels que $\mathsf{exec} = tr\sigma = [e_1; \ldots; e_\ell] \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ où tr est la trace symbolique associée à sc, et tels que $\langle \mathsf{exec}, \emptyset \rangle \models \neg\phi_{\mathcal{P}}$, *i.e.*

- il existe $k \in [\![\ell]\!]$
- il existe $2(h+3)$ entités honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$, et
- il existe un nonce ν,

tels que

- $e_k = \mathsf{Secret}(\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}, \nu)$, et
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\Pi(\epsilon) \vdash \nu$.

Or par définition de $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, toutes les sessions de sc sont honnêtes, et qui plus est pour tout $i \in [\![\ell]\!]$, si $r_i \in \{r_{a_\eta, r_{b_\eta}}\}$ pour un certain η, $0 \leq \eta \leq h+2$ et $\mathsf{initagts}(sid_i) = (\alpha'_0, \beta'_0, \ldots, \alpha'_{h+2}, \beta'_{h+2})$, alors $\{\alpha'_\eta, \beta'_\eta\} = \{\alpha_0, \beta_0\}$. De plus, nous savons d'après le lemme 4.3.4 que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha_0, \beta_0)$. Donc, le seul moyen que le secret ν soit dévoilé, *i.e.* $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \nu$, c'est qu'il passe en clair sur le réseau. Par construction de $\Pi_{\mathcal{P}}$ nous savons alors que

- il existe un entier $k_1 \in [\![\ell]\!]$,
- il existe 2 nonces ν_0, μ_0

tels que $r_{k_1} = r_{b_{h+2}}$ et donc que $e_{k_1} = \mathsf{snd}(\beta_0, \alpha_0, t_1)$ avec

$$t_1 = \begin{array}{l} \{0, \nu_0, \mu_0\}_{\mathsf{shk}(\alpha_0,\beta_0)} \\ \nu \end{array}$$

où ν_0 et μ_0 sont deux nonces (ceci tient du fait que $\mathsf{exec} \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$). Mais toujours par construction de $r_{b_{h+2}}$ nous savons qu'il existe $k_2 \in [\![k_1 - 1]\!]$, et deux nonces ν_1, μ_1 tels que $r_{k_2} = r_{k_1}$, $sid_{k_2} = sid_{k_1}$, et $e_{k_2} = \mathsf{rcv}(\beta_0, \alpha_0, t_2)$ avec

$$t_2 = \begin{array}{l} \{0, \nu_1, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)} \end{array}$$

De plus comme exec est une exécution valide de $\Pi_{\mathcal{P}}$, il est nécessaire que $\mathsf{K}(\mathsf{exec}_{k-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash t_2$. En récapitulant donc tout ce que nous venons de dire nous avons donc que

- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \nu_1, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$.

Mais par définition (voir définition 4.3.3), cela signifie que $(\varepsilon, \varepsilon)$ admet une représentation de type présolution dans exec. Or, d'après la proposition 4.3.7 cela implique que $(\varepsilon, \varepsilon)$ est une présolution de $\mathcal{P}$, et donc par définition que $\mathcal{P}$ admet une solution.

Pour conclure quant à la tailles des messages impliqués dans cette exécution exec, il suffit de noter que exec étant dans $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, pour toute variable $x \in \mathcal{V}(tr)$ $\sigma(x) \in \mathcal{N} \cup \Sigma$). Nous avons donc bien que pour tout message $m \in \mathsf{St}(\mathsf{exec})$, $|m| \leq \mathsf{B}(\Pi_{\mathcal{P}})$. □

Ayant à présent prouvé l'équivalence énoncée au théorème 4.3.2, nous pouvons faire appel à l'indécidabilité du problème de correspondance de Post (voir théorème 4.1.4) et conclure que même dans le cadre de messages de taille bornée le problème de la vérification des protocoles de sécurité est indécidable.

4.4 Preuves des résultats intermédiaires

Comme nous l'annoncions, cette section est dédiée à la preuve formelle des résultats intermédiaires sur lesquels repose notre preuve d'indécidabilité présentée ci-dessus.

Le premier résultat dont nous avons besoin stipule qu'aucune exécution valide de $\Pi_{\mathcal{P}}$ ne révèle à l'intrus de clés long-terme qu'il ne connaîtrait pas initialement.

Lemme 4.3.4. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soient aussi* exec *une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide, et deux entités honnêtes $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$.*

$$\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta).$$

Démonstration. Par construction, $\Pi_{\mathcal{P}}$ vérifie bien l'hypothèse sur l'introduction des variables en position de plaintext du lemme 2.6.10. D'où,

$$\mathsf{Plaintext}(\mathsf{exec}) \subseteq (\mathsf{Plaintext}(tr) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon).$$

Considérons deux entités $\alpha, \beta \in \mathcal{P}$, et supposons que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \mathsf{shk}(\alpha, \beta)$. En faisant appel au lemme 2.5.6 nous déduisons alors que $\mathsf{shk}(\alpha, \beta) = \mathsf{Plaintext}(\mathsf{shk}(\alpha, \beta)) \subseteq (\mathsf{Plaintext}(\mathsf{exec}) \cup \mathcal{C} \cup \mathcal{N}_\epsilon(\Pi))$, et donc que $\mathsf{shk}(\alpha, \beta) \subseteq (\mathsf{Plaintext}(tr) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$. Or, par construction de $\Pi_{\mathcal{P}}$, nous savons que $\mathsf{shk}(\alpha, \beta) \notin \mathsf{Plaintext}(\Pi_{\mathcal{P}})$ et donc $\mathsf{shk}(\alpha, \beta) \notin \mathsf{Plaintext}(tr)$. De plus, par définition de l'ensemble des constantes $\mathsf{shk}(\alpha, \beta) \notin \mathcal{C}$. De même, par définition de l'ensemble de nonces engendré par l'intrus, $\mathsf{shk}(\alpha, \beta) \notin \mathcal{N}_\epsilon(\Pi)$. Ce qui implique nécessairement que $\mathsf{shk}(\alpha, \beta) \in \mathcal{K}_\epsilon$. Finalement, nous pouvons conclure par définition de $\mathcal{K}_\epsilon$ que soit $\alpha = \epsilon$, ou bien $\beta = \epsilon$, *i.e.* $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \mathsf{shk}(\alpha, \beta) \Rightarrow (\alpha = \epsilon \vee \beta = \epsilon)$. Ce qui est équivalent par contraposée à ce que nous cherchions initialement à montrer, à savoir que si α et β sont honnêtes (*i.e.* $\neq \epsilon$), alors $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$. □

4.4.1 Preuve de la proposition 4.3.5

Afin d'établir qu'il est en effet possible de se restreindre, dans le but de trouver une attaque, à l'ensemble des exécutions $\mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$, il nous faut d'abord montrer un résultat intermédiaire. Intuitivement, le lemme suivant stipule que si au cours d'un exécution l'intrus peut construire un messages chiffré avec la clé symétrique de deux agents honnêtes, alors un de ces agents a antérieurement envoyé ce message.

Lemme 4.4.1. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soit aussi,* $\mathsf{exec} = [e_1; \ldots; e_\ell]$ *une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide. Pour tous agents honnêtes $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$, et pour tout message $m \in \mathcal{M}$, si $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)}$, alors*

– $\{m\}_{\mathsf{shk}(\alpha,\beta)}$ *est de la forme*

$$\{T,\nu,\mu\}_{\mathsf{shk}(\alpha,\beta)} \quad ou \quad \{T',\nu,\mathfrak{f},\mu\}_{\mathsf{shk}(\alpha,\beta)} \quad ou\ encore \quad \{T',\nu,\lambda,\mu\}_{\mathsf{shk}(\alpha,\beta)}$$

où $T \in \{0,1\}$, $T' \in \{2,3\}$, $\mathfrak{f} \in \Sigma\}$, *et* $\lambda,\mu,\nu \in \mathcal{N}$, *et*

– *pour tout message* $m' \in (\mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)} \setminus \mathcal{C}))$ *tel que* $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$, *il existe un entier* $i \in [\![\ell]\!]$, *un entier* $n \in \mathbb{N}^*$, *un entier* $j \in [\![n]\!]$, *ainsi que* n *messages* $m_1, \ldots, m_n \in \mathcal{M}$ *tels que* $(e_i = \mathsf{snd}(\alpha,\beta,m_1,\ldots,m_n)$ *ou* $e_i = \mathsf{snd}(\beta,\alpha,m_1,\ldots,m_n))$ *et* $m_j = m'$.

Démonstration. Soient $\Sigma = \{\mathfrak{f}_1,\ldots,\mathfrak{f}_r\}$ un alphabet, $\mathcal{P} = \{(u_i,v_i)\}_{i\in[\![h]\!]}$ une instance du problème de correspondance de Post sur Σ pour un certain h, et $\Pi_\mathcal{P}$ et $\phi_\mathcal{P}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. D'après ce codage, nous avons $\mathcal{C}(\Pi_\mathcal{P}) = \{0,1,2,3,\mathfrak{f}_1,\ldots,\mathfrak{f}_r\}$, $\mathcal{A}(\Pi_\mathcal{P}) = \{a_i,b_i \mid 0 \leq i \leq (h+2)\}$, et $\mathsf{Roles}(\Pi_\mathcal{P}) = \{r_{a_i},r_{b_i} \mid 0 \leq i \leq (h+2)\}$.

Soit $\mathsf{exec} = [e_1;\ldots;e_\ell]$ une exécution valide de $\Pi_\mathcal{P}$, au regard d'une connaissance initiale de l'intrus vide. Soit le scénario $\mathsf{sc} = (\mathsf{interlvg},\mathsf{initagts})$ de $\Pi_\mathcal{P}$, avec $\mathsf{interlvg} = [(r_1,sid_1);\ldots;(r_\ell,sid_\ell)]$, et σ une substitution close, tels que $\mathcal{V}(tr) \subseteq \mathrm{dom}(\sigma)$ et $\mathsf{exec} = tr\sigma$, où tr est la trace symbolique associée à sc.

Nous procédons par induction sur la longueur ℓ de exec.

Cas de base : $\ell = 0$. Dans ce cas, $\mathsf{K}(\mathsf{exec}) = \emptyset$ et donc pour tout message $m \in \mathcal{M}$ et pour touts agents honnêtes $\alpha,\beta \in (\mathcal{A} \setminus \{\epsilon\})$, $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \nvdash \{m\}_{\mathsf{shk}(\alpha,\beta)}$. Nous pouvons donc conclure immédiatement.

Cas inductif : $\ell \geq 1$. Si e_ℓ est une réception ou un status event, alors $\mathsf{K}(\mathsf{exec}_\ell) = \mathsf{K}(\mathsf{exec}_{\ell-1})$. Nous pouvons donc conclure par hypothèse d'induction. Supposons maintenant que e_ℓ est une émission, *i.e.* $e_\ell = \mathsf{snd}(\gamma,\delta,t)$. Soient deux agents honnêtes $\alpha,\beta \in (\mathcal{A} \setminus \{\epsilon\})$ et un message $m \in \mathcal{M}$ tels que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)}$. Nous distinguerons plusieurs cas en fonction du rôle r_ℓ incarné par la session sid_ℓ, mais avant d'aller plus loin, notons que de $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \nvdash \mathsf{shk}(\alpha,\beta)$ (lemme 4.3.4) nous déduisons que la dernière règle appliquée dans la dérivation de $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)}$ est soit l'axiome soit une règle de décomposition. Nous concluons donc en faisant appel au lemme 2.5.5 que $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(\mathsf{K}(\mathsf{exec}))$.

Cas $r_\ell = r_{a_0}$. Ce cas ne peut pas survenir car aucune émission n'est spécifiée dans le corps de r_{a_0}.

Cas $r_\ell = r_{b_0}$. t est alors de la forme :

$$\begin{aligned} t = \quad & \{2, \kappa_{(1,0)}, \kappa, \kappa_{(1,1)}\}_{\mathsf{shk}(\gamma,\delta)}, \{3, \lambda_{(1,0)}, \kappa, \lambda_{(1,1)}\}_{\mathsf{shk}(\gamma,\delta)} \\ & \{2, \kappa_{(1,1)}, u_1^1, \kappa_{(1,2)}\}_{\mathsf{shk}(\gamma,\delta)}, \ldots, \{2, \kappa_{(1,p_1)}, u_1^{p_1}, \kappa_{(1,p_1+1)}\}_{\mathsf{shk}(\gamma,\delta)}, \\ & \{3, \lambda_{(1,1)}, v_1^1, \lambda_{(1,2)}\}_{\mathsf{shk}(\gamma,\delta)}, \ldots, \{3, \lambda_{(1,q_1)}, v_1^{q_1}, \lambda_{(1,q_1+1)}\}_{\mathsf{shk}(\gamma,\delta)}, \\ & \{1, \kappa_{(1,p_1+1)}, \lambda_{(1,q_1+1)}\}_{\mathsf{shk}(\gamma,\delta)}, \\ & \ldots \\ & \{2, \kappa_{(h,0)}, \kappa, \kappa_{(h,1)}\}_{\mathsf{shk}(\gamma,\delta)}, \{3, \lambda_{(h,0)}, \kappa, \lambda_{(h,1)}\}_{\mathsf{shk}(\gamma,\delta)} \\ & \{2, \kappa_{(h,1)}, u_h^1, \kappa_{(h,2)}\}_{\mathsf{shk}(\gamma,\delta)}, \ldots, \{2, \kappa_{(h,p_h)}, u_h^{p_h}, \kappa_{(h,p_h+1)}\}_{\mathsf{shk}(\gamma,\delta)}, \\ & \{3, \lambda_{(h,1)}, v_h^1, \lambda_{(h,2)}\}_{\mathsf{shk}(\gamma,\delta)}, \ldots, \{3, \lambda_{(h,q_h)}, v_h^{q_h}, \lambda_{(h,q_h+1)}\}_{\mathsf{shk}(\gamma,\delta)}, \\ & \{1, \kappa_{(h,p_h+1)}, \lambda_{(h,q_h+1)}\}_{\mathsf{shk}(\gamma,\delta)} \end{aligned}$$

où κ, $\kappa_{(j_1,j_2)}$ et $\lambda_{(j_1,j_2)}$ sont des nonces frais pour tout $j_1, j_2 \in \mathbb{N}$.
Si $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(t)$, alors

$$\{m\}_{\mathsf{shk}(\alpha,\beta)} = \begin{cases} \{1, \kappa_{(j_1,p_{j_1}+1)}, \lambda_{(j,q_{j_1}+1)}\}_{\mathsf{shk}(\gamma,\delta)} & j_1 \in [\![h]\!] \\ \qquad\qquad \text{ou} & \\ \{2, \kappa_{(j_1,j_2)}, u_{j_1}^{j_2}, \kappa_{(j_1,j_2+1)}\}_{\mathsf{shk}(\gamma,\delta)} & j_1 \in [\![h]\!],\ 0 \le j_2 \le p_{j_1} \\ \qquad\qquad \text{ou encore} & \\ \{3, \lambda_{(j_1,j_2)}, v_{j_1}^{j_2}, \lambda_{(j_1,j_2+1)}\}_{\mathsf{shk}(\gamma,\delta)} & j_1 \in [\![h]\!],\ 0 \le j_2 \le q_{j_1} \end{cases}$$

Donc $\{m\}_{\mathsf{shk}(\alpha,\beta)}$ est bien d'une des formes spécifiées. De plus, et toujours dû au fait que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha,\beta)$, le seul sous- terme m' de $\{m\}_{\mathsf{shk}(\alpha,\beta)}$ non dans $\mathcal{C}$ que l'intrus puisse dérivé est $m' = \{m\}_{\mathsf{shk}(\alpha,\beta)}$ lui-même. Le second point est donc vérifié aussi.
Si $\{m\}_{\mathsf{shk}(\alpha,\beta)} \notin \mathsf{St}(t)$, alors les nonces apparaissant dans t étant tous frais, et sachant que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha,\beta)$ nous avons nécessairement que

$$\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)}$$

et que

$$\forall m' \in (\mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)}) \setminus \mathcal{C}).\ \mathsf{K}(\mathsf{exec}_\ell) \cup \mathcal{N}_\epsilon(\Pi) \vdash m' \ \Rightarrow\ \mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'.$$

Nous pouvons donc dans ce cas conclure par hypothèse d'induction.

Cas $r_\ell = r_{a_\eta}$ *pour un* $\eta \in [\![h]\!]$. Ce cas est analogue au cas $r_\ell = r_{b_0}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_\ell = r_{b_\eta}$ *pour un* $\eta \in [\![h]\!]$. t est alors de la forme :

$$\begin{aligned} t = \quad & \{2, t_1, u_\eta^1, \kappa_{(h+\eta,2)}\}_{\mathsf{shk}(\gamma,\delta)}, \ldots, \{2, \kappa_{(h+\eta,p_\eta)}, u_\eta^{p_\eta}, \kappa_{(h+\eta,p_\eta+1)}\}_{\mathsf{shk}(\gamma,\delta)}, \\ & \{3, t_2, v_\eta^1, \lambda_{(h+\eta,2)}\}_{\mathsf{shk}(\gamma,\delta)}, \ldots, \{3, \lambda_{(h+\eta,q_\eta)}, v_\eta^{q_\eta}, \lambda_{(h+\eta,q_\eta+1)}\}_{\mathsf{shk}(\gamma,\delta)}, \\ & \{1, \kappa_{(h+\eta,p_\eta+1)}, \lambda_{(h+\eta,q_\eta+1)}\}_{\mathsf{shk}(\gamma,\delta)} \end{aligned}$$

où $\kappa_{h+\eta,j}$ et $\lambda_{h+\eta,j}$ sont des nonces frais pour tout $j \in \mathbb{N}$, et $t_1, t_2 \in \mathcal{T}$. exec étant une exécution valide de $\Pi_\mathcal{P}$ au regard d'une connaissance initiale de l'intrus vide, nous savons qu'il existe $k \in [\![\ell-1]\!]$ tel que $e_k = \mathsf{rcv}(\delta, \gamma, \{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)})$

et que $\mathsf{K}(\mathsf{exec}_{k-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$.
Supposons dans un premier temps que les agents γ et δ sont honnêtes, *i.e.* $\epsilon \notin \{\gamma, \delta\}$. Nous pouvons alors conclure par hypothèse d'induction sur $\{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$ que t_1 et t_2 sont des nonces. Si $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(t)$, alors

$$\{m\}_{\mathsf{shk}(\alpha,\beta)} = \begin{cases} \{2, t_1, u^1_\eta, \kappa_{(h+\eta,2)}\}_{\mathsf{shk}(\gamma,\delta)} \\ \qquad\qquad \text{ou} \\ \{3, t_2, v^1_\eta, \lambda_{(h+\eta,2)}\}_{\mathsf{shk}(\gamma,\delta)} \\ \qquad\qquad \text{ou} \\ \{1, \kappa_{(h+\eta,p_\eta+1)}, \lambda_{(h+\eta,q_\eta+1)}\}_{\mathsf{shk}(\gamma,\delta)} \\ \qquad\qquad \text{ou} \\ \{2, \kappa_{(h+\eta,j)}, u^j_\eta, \kappa_{(h+\eta,j+1)}\}_{\mathsf{shk}(\gamma,\delta)} \qquad j \in [\![p_\eta]\!] \\ \qquad\qquad \text{ou encore} \\ \{3, \lambda_{(h+\eta,j)}, v^j_\eta, \lambda_{(h+\eta,j+1)}\}_{\mathsf{shk}(\gamma,\delta)} \qquad j \in [\![q_\eta]\!]. \end{cases}$$

Dans tous ces cas $\{m\}_{\mathsf{shk}(\alpha,\beta)}$ est d'une des formes spécifiées. Soit un message $m' \in (\mathsf{St}(\{\{m\}_{\mathsf{shk}(\alpha,\beta)} \setminus \mathcal{C}) \setminus \mathcal{C})$. Les $\kappa_{(h+\eta,j)}$ et $\lambda_{h+\eta,j}$ (avec $2 \leq j$) étant frais et $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$ les seuls cas pouvant survenir sont $m' = \{m\}_{\mathsf{shk}(\alpha,\beta)}$, $m' = t_1$ ou $m' = t_2$. Si $m' = \{m\}_{\mathsf{shk}(\alpha,\beta)}$ nous pouvons conclure immédiatement quant au second point. Si $m' = t_1$, sachant que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$ et que $\kappa_{(h+\eta,2)}$ est frais, nous concluons que $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash t_1$, mais alors en faisant appel au fait déjà établi que $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$ nous concluons quant au second point en faisant appel à notre hypothèse d'induction. Le cas $m' = t_2$ étant analogue au cas $m' = t_1$ nous ne le détaillons pas ici.
Si $\{m\}_{shk(\alpha,\beta)} \notin \mathsf{St}(t)$, ayant supposé que γ et δ sont honnêtes et sachant que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$ et que les κ_j et λ_j (pour $2 \leq j$) sont frais, alors il est nécessaire que

$$\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)}$$

et que

$$\forall m' \in (\mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)}) \setminus \mathcal{C}).\ \mathsf{K}(\mathsf{exec}_\ell) \cup \mathcal{N}_\epsilon(\Pi) \vdash m' \ \Rightarrow\ \mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'.$$

Nous pouvons donc dans ce cas conclure par hypothèse d'induction.
Supposons à présent que γ ou δ est malhonnête, *i.e.* $\epsilon \in \{\gamma, \delta\}$. Alors nous déduisons que $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \{t_1, t_2\})$. De plus, sachant que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$ et dû au fait que les κ_j et les λ_j (pour $2 \leq j$) sont frais, nous déduisons que pour tout $m' \in (\mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)}))$, si $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$, alors $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \{t_1, t_2\} \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$. Or, le fait $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$ déjà établi nous autorise à conclure que $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash t_i$ (pour $i \in \{1, 2\}$). En faisant appel au lemme 2.5.4 d'élimination des coupures nous en arrivons donc aux conclusions suivantes :

$$\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)},$$

et

$$\forall m' \in \mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)} \setminus \mathcal{C}).\ \mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m' \ \Rightarrow\ \mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'.$$

Nous pouvons donc pour conclure faire appel à notre lemme d'induction.

Cas $r_\ell = r_{a_{h+1}}$. Ce cas est analogue au cas $r_\ell = r_{b_0}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_\ell = r_{b_{h+1}}$. t est alors de la forme

$$t = \{0, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}.$$

exec étant une exécution valide de $\Pi_{\mathcal{P}}$ au regard d'une connaissance initiale de l'intrus vide, nous savons qu'il existe un entier $k \in [\![\ell - 1]\!]$ tel que $e_k = \mathsf{rcv}(\delta, \gamma, \{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)})$ et que $\mathsf{K}(\mathsf{exec}_{k-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$.
Supposons dans un premier temps que les agents γ et δ sont honnêtes, *i.e.* $\epsilon \notin \{\gamma, \delta\}$. Nous pouvons alors conclure par hypothèse d'induction que t_1 et t_2 sont des nonces.
Si $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(t)$, alors $\{m\}_{\mathsf{shk}(\alpha,\beta)} = t$ et est donc d'une des formes spécifiées. Soit $m' \in (\mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)} \setminus \mathcal{C}))$, alors $m' \in \{\{m\}_{\mathsf{shk}(\alpha,\beta)}\} \cup \mathsf{St}(\langle t_1, t_2 \rangle)$.
Si $m' = \{m\}_{\mathsf{shk}(\alpha,\beta)}$, alors nous concluons immédiatement quant au second point. Si par contre $m' \in \mathsf{St}(\langle t_1, t_2 \rangle)$, alors étant donné que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$ il est nécessaire que $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$. Mais alors, du fait que $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$, nous pouvons faire appelle à notre hypothèse d'induction et conclure.
Si $\{m\}_{\mathsf{shk}(\alpha,\beta)} \notin \mathsf{St}(t)$, alors $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(\mathsf{K}(\mathsf{exec}_{\ell-1}))$ et nous pouvons donc conclure par hypothèse d'induction quant à la forme de $\{m\}_{\mathsf{shk}(\alpha,\beta)}$. De plus, de $\{m\}_{\mathsf{shk}(\alpha,\beta)} \notin \mathsf{St}(t)$ découle que

$$\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)}$$

et

$$\forall m' \in \mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)} \setminus \mathcal{C}).\ \mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m' \ \Rightarrow\ \mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'.$$

Nous pouvons donc pour conclure faire appel à notre lemme d'induction.
Supposons à présent que γ ou δ est malhonnête, *i.e.* $\epsilon \in \{\gamma, \delta\}$. Alors nous déduisons que $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \{t_1, t_2\})$, et que de plus, pour tout $m' \in (\mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)}) \setminus \mathcal{C})$, si $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$, alors $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \{t_1, t_2\} \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$. Or, le fait $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$ déjà établi nous autorise à conclure que $\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash t_i$ (pour $i \in \{1, 2\}$). En faisant appel au lemme 2.5.4 d'élimination des coupures nous en arrivons donc aux conclusions suivantes :

$$\mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha,\beta)},$$

et

$$\forall m' \in \mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)} \setminus \mathcal{C}).\ \mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m' \ \Rightarrow\ \mathsf{K}(\mathsf{exec}_{\ell-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'.$$

Nous pouvons donc pour conclure faire appel à notre lemme d'induction.

Cas $r_\ell = r_{a_{h+2}}$. Ce cas est analogue au cas $r_\ell = r_{b_0}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_\ell = r_{b_{h+2}}$. t est alors de la forme :

$$t = \begin{array}{l} \{0, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)} \\ t_3 \end{array}$$

exec étant une exécution valide de $\Pi_\mathcal{P}$ au regard d'une connaissance initiale de l'intrus vide, nous savons qu'il existe un entier $k \in [\![\ell - 1]\!]$ tel que $e_k = \mathsf{rcv}(\delta, \gamma, t')$ avec

$$t' = \begin{array}{l} \{0, t_4, t_5\}_{\mathsf{shk}(gamma,\delta)}, \\ \{2, t_1, t_3, t_4\} \\ \{3, t_2, t_3, t_5\} \end{array}$$

et que $\mathsf{K}(\mathsf{exec}_{k-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash t'$.
Supposons dans un premier temps que les agents γ et δ sont honnêtes, *i.e.* $\epsilon \notin \{\gamma, \delta\}$. Nous pouvons alors conclure par hypothèse d'induction que t_1 et t_2 sont des nonces, et t_3 est soit un nonce soit une constante.
Si $\{m\}_{\mathsf{shk}(\alpha,\beta)} \in \mathsf{St}(t)$, alors $\{m\}_{\mathsf{shk}(\alpha,\beta)} = \{0, t_1, t_2\}_{\mathsf{shk}(\gamma,\delta)}$ et est donc d'une des formes spécifiées. Soit $m' \in (\mathsf{St}(\{m\}_{\mathsf{shk}(\alpha,\beta)}) \setminus \mathcal{C})$, alors $m' \in \{\{m\}_{\mathsf{shk}(\alpha,\beta)}, t_3\} \cup \mathsf{St}(\langle t_1, t_2\rangle)$. Si $m' = \{m\}_{\mathsf{shk}(\alpha,\beta)}$ ou $m' = t_3$, alors nous concluons immédiatement quant au second point. Si par contre $m' \in \mathsf{St}(\langle t_1, t_2\rangle)$, alors pour conclure il suffit de mimer les arguments exhibés au cas $r_\ell = r_{b_{h+1}}$. □

La proposition suivante stipule comme nous l'expliquions un peu plus haut que si $\Pi_\mathcal{P}$ viole $\phi_\mathcal{P}$, alors il existe une exécution $\mathsf{exec} \in \mathcal{E}^{\mathsf{bnd}}_\mathcal{P}$ qui viole $\phi_\mathcal{P}$.

Proposition 4.3.5. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_\mathcal{P}$ et $\phi_\mathcal{P}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Si $\Pi_\mathcal{P}$ viole la propriété $\phi_\mathcal{P}$ au regard d'une connaissance initiale de l'intrus vide,* i.e. *$\langle \Pi_\mathcal{P}, \emptyset\rangle \models \neg\phi_\mathcal{P}$, alors il existe deux agents honnêtes $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$ et une exécution* exec *de $\Pi_\mathcal{P}$, valide au regard d'une connaissance initiale de l'intrus vide, qui viole $\phi_\mathcal{P}$ et telle que*

- *pour tout $i \in [\![\ell]\!]$ et pour tout η, $0 \leq \eta \leq h + 2$*

$$\begin{array}{c} r_i \in \{r_{b_\eta}, r_{a_\eta}\} \wedge \mathsf{initagts}(sid_i) = (\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}) \\ \Rightarrow \\ \{\alpha_\eta, \beta_\eta\} = \{\alpha, \beta\} \end{array}$$

- *pour tout message $m \in \mathsf{Est}(\mathsf{exec})$, m est de la forme*

$$\{T, \nu, \mu\}_{\mathsf{shk}(\alpha,\beta)} \quad \textit{ou} \quad \{T', \nu, \mathfrak{f}, \mu\}_{\mathsf{shk}(\alpha,\beta)} \quad \textit{ou encore} \quad \{T', \nu, \lambda, \mu\}_{\mathsf{shk}(\alpha,\beta)}$$

où $T \in \{0, 1\}$, $T' \in \{2, 3\}$, $\mathfrak{f} \in \Sigma$, et $\lambda, \mu, \nu \in \mathcal{N}$.

Démonstration. Soient un alphabet $\Sigma = \{\mathfrak{f}_1, \ldots, \mathfrak{f}_r\}$, $\mathcal{P} = \{(u_i, v_i)\}_{i \in [\![h]\!]}$ une instance du problème de correspondance de Post sur Σ pour un certain h, et $\Pi_\mathcal{P}$ et $\phi_\mathcal{P}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. D'après ce codage $\Pi_\mathcal{P}$ est tel que $\mathcal{C}(\Pi_\mathcal{P}) = \{0, 1, 2, 3, \mathfrak{f}_1, \ldots, \mathfrak{f}_r\}$, $\mathcal{A}(\Pi_\mathcal{P}) = \{a_i, b_i \mid 0 \leq i \leq (h + 2)\}$, et $\mathsf{Roles}(\Pi_\mathcal{P}) = \{r_{a_i}, r_{b_i} \mid 0 \leq i \leq (h + 2)\}$.

Supposons que $\Pi_{\mathcal{P}}$ viole la propriété $\phi_{\mathcal{P}}$, et soit $\mathsf{exec} = [e_1, \ldots, e_\ell]$ une attaque sur $\phi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide, *i.e.* $\langle \mathsf{exec}, \emptyset \rangle \models \neg\phi_{\mathcal{P}}$. Il existe alors, d'après la sémantique de $\mathcal{PS}$-LTL, un entier $k \in [\![\ell]\!]$, $2(h+3)$ agents honnêtes α_0, β_0, …, α_{h+2}, $\beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$, ainsi qu'un nonce $\nu \in \mathcal{N}$ tels que

- $e_k = \mathsf{Secret}(\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}, \nu)$, et
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \nu$.

Etant donné que exec est une exécution valide de $\Pi_{\mathcal{P}}$, il existe par définition un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de $\Pi_{\mathcal{P}}$ avec pour entrelacement $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, et une substitution close σ tels que $\mathcal{V}(tr) \subseteq \mathrm{dom}(\sigma)$ et $\mathsf{exec} = tr\sigma$ où tr est la trace symbolique associée à sc. Soit l'ensemble de sessions $S^{(\alpha_0,\beta_0)}$ défini comme suit :

$$
\begin{aligned}
S^{(\alpha_0,\beta_0)} = \{ sid_i \mid\ & i \in [\![\ell]\!], \\
& \mathsf{initagts}(sid_i) = (\alpha'_0, \beta'_0, \ldots, \alpha'_{h+2}, \beta'_{h+2}) \in \mathcal{A}^{2(h+3)}, \\
& r_i \in \{r_{a_\eta}, r_{b_\eta}\} \text{ pour un certain } \eta \text{ } t.q. \text{ } 0 \leq \eta \leq (h+2), \\
& \{\alpha'_\eta, \beta'_\eta\} = \{\alpha_0, \beta_0\} \qquad \}.
\end{aligned}
$$

Informellement, $S^{(\alpha_0,\beta_0)}$ contient les sessions de exec dans lesquelles toute émission et toute réception a lieu entre α_0 et β_0.

Soit aussi $\mathsf{exec}^{(\alpha_0,\beta_0)} = (tr|_{S^{(\alpha_0,\beta_0)}})\sigma = [e_{i_1}; \ldots; e_{i_n}]$ $(1 \leq i_1 < \cdots < i_n \leq \ell)$, la restriction de exec à l'ensemble de sessions $S^{(\alpha_0,\beta_0)}$. Dans un premier temps, nous allons montrer que

- $\mathsf{exec}^{(\alpha_0,\beta_0)}$ est une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide, et que
- pour tout message $m \in \mathsf{Est}(\mathsf{exec}^{(\alpha_0,\beta_0)})$, m est de la forme

  $\{T, \nu, \mu\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ ou $\{T', \nu, \mathfrak{f}, \mu\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ ou encore $\{T', \nu, \lambda, \mu\}_{\mathsf{shk}(\alpha_0,\beta_0)}$

  où $T \in \{0, 1\}$, $T' \in \{2, 3\}$, $\mathfrak{f} \in \Sigma\}$, et $\lambda, \mu, \nu \in \mathcal{N}$, et

par induction sur n.

Cas de base : $n = 0$. Alors $\mathsf{exec}^{(\alpha_0,\beta_0)} = [\,]$ est par définition une exécution valide au regard de toute connaissance initiale de l'intrus. Et comme $\mathsf{Est}(\mathsf{exec}) = \emptyset$ nous pouvons conclure trivialement quant à la forme des sous-termes chiffrés de $\mathsf{exec}^{(\alpha_0,\beta_0)}$.

Cas inductif : $n \geq 1$. Nous procédons par analyse de cas sur le type d'évènement qu'est e_{i_n}.

Cas e_{i_n} est un status event. Il existe alors $\alpha'_1, \beta'_1, \ldots, \alpha'_{h+2}, \beta'_{h+2} \in \mathcal{A}$ ainsi que $m \in \mathcal{M}$ tels que

$$
e_{i_n} = \begin{cases} \mathsf{Secret}(\alpha_0, \beta_0, \alpha'_1, \beta'_1, \ldots, \alpha'_{h+2}, \beta'_{h+2}, m) \\ \text{ou} \\ \mathsf{Secret}(\beta_0, \alpha_0, \alpha'_1, \beta'_1, \ldots, \alpha'_{h+2}, \beta'_{h+2}, m) \end{cases}
$$

Par hypothèse d'induction, $\mathsf{exec}^{(\alpha_0,\beta_0)}_{n-1} = [e_{i_1}; \ldots; e_{i_{(n-1)}}]$ est une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide. Donc par

définition, $\mathsf{exec}^{(\alpha_0,\beta_0)} = [e_{i_1}; \ldots; e_{i_n}]$ en est aussi une. De plus, par construction de $\Pi_{\mathcal{P}}$ nous savons que m est nécessairement un nonce, donc $\mathsf{Est}(\mathsf{exec}^{(\alpha_0,\beta_0)}) = \mathsf{Est}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)})$. Nous pouvons donc conclure par hypothèse d'induction quant à la forme des sous-termes chiffrés de $\mathsf{exec}^{(\alpha_0,\beta_0)}$.

Cas e_{i_n} est une émission. Il existe alors un message $m \in \mathcal{M}$ tel que $e_{i_n} = \mathsf{snd}(\alpha_0, \beta_0, m)$ ou $e_{i_n} = \mathsf{snd}(\beta_0, \alpha_0, m)$. Par hypothèse d'induction, $\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)} = [e_{i_1}; \ldots; e_{i_{(n-1)}}]$ est une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide. Donc par définition, $\mathsf{exec}^{(\alpha_0,\beta_0)} = [e_{i_1}; \ldots; e_{i_n}]$ en est aussi une. Distinguons à présent deux cas.

Cas $r_{i_n} \neq r_{b_{h+2}}$. m est dans ce cas de la forme

$$m = \{m_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \ldots, \{m_q\}_{\mathsf{shk}(\alpha_0,\beta_0)}.$$

Donc, pour tout $j \in [\![q]\!]$, $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m_j\}_{\mathsf{shk}(\alpha_0,\beta_0)}$. Or, d'après le lemme 4.4.1, cela implique que $\{m_j\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ est d'une des formes voulues. De là, découle aussi que

$$\mathsf{Est}(\mathsf{exec}^{(\alpha_0,\beta_0)}) = \mathsf{Est}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)}) \cup \{\{m_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \ldots, \{m_k\}_{\mathsf{shk}(\alpha_0,\beta_0)}\}.$$

Nous pouvons donc conclure quant à la forme des sous-termes chiffrés de $\mathsf{exec}^{(\alpha_0,\beta_0)}$.

Cas $r_{i_n} = r_{b_{h+2}}$. m est dans ce cas de la forme

$$\begin{aligned} m \quad = \quad & \{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ & m_3 \end{aligned}$$

Donc, $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)}$. Or, d'après le lemme 4.4.1, cela implique que $\{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ est d'une des formes voulues, et plus précisément $m_1, m_2 \in \mathcal{N}$. De plus, exec étant une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide, il existe $k \in [\![i_n - 1]\!]$ tel que $e_k = \mathsf{rcv}(\beta_0, \alpha_0, m')$ ou $e_k = \mathsf{rcv}(\alpha_0, \beta_0, m')$ avec m' de la forme

$$\begin{aligned} m' \quad = \quad & \{0, m_4, m_5\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ & \{2, m_1, m_3, m_4\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ & \{3, m_2, m_3, m_5\}_{\mathsf{shk}(\alpha_0,\beta_0)}. \end{aligned}$$

et $\mathsf{K}(\mathsf{exec}_{k-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$. D'où, en appelant une seconde fois le lemme 4.4.1 nous savons que $m_3 \in (\mathcal{N} \cup \Sigma)$. Si nous récapitulons, alors

$$\mathsf{Est}(\mathsf{exec}^{(\alpha_0,\beta_0)}) = \mathsf{Est}(\mathsf{exec}_{i_{(n-1)}}^{(\alpha_0,\beta_0)}) \cup \{\{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)}\}.$$

Nous pouvons donc conclure quant à la forme des sous-termes chiffrés de $\Pi_{\mathcal{P}}$.

Cas e_{i_n} est une réception. Il existe alors un message $m \in \mathcal{M}$ tel que $e_{i_n} = \mathsf{rcv}(\alpha_0, \beta_0, m)$ ou $e_{i_n} = \mathsf{rcv}(\beta_0, \alpha_0, m)$. Distinguons à présent deux cas.

Cas $r_i \neq r_{a_{h+2}}$. m est dans ce cas de la forme :

$$m = \{m_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \ldots, \{m_q\}_{\mathsf{shk}(\alpha_0,\beta_0)}.$$

exec étant une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide $\mathsf{K}(\mathsf{exec}_{(i_n)-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m$, et par là même pour tout $j \in [\![q]\!]$, $\mathsf{K}(\mathsf{exec}_{(i_n)-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m_j\}_{\mathsf{shk}(\alpha_0,\beta_0)}$. Or, d'après le lemme 4.4.1, cela implique que

- pour tout $j \in [\![q]\!]$, il existe $k' \in [\![i_n - 1]\!]$ tel que $e_{k'}$ soit de la forme

  $$e_{k'} = \mathsf{snd}(\alpha_0, \beta_0, m'_1, \ldots, m'_{q'}) \quad \text{ou} \quad e_{k'} = \mathsf{snd}(\beta_0, \alpha_0, m'_1, \ldots, m'_{q'})$$

  et il existe $j' \in [\![q']\!]$ tel que $m'_{j'} = m_j$. Mais alors, il existe $k'' \in [\![n-1]\!]$ tel que $e_{i_{k''}} = e_{k'}$, et donc $\mathsf{K}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m_j$. Nous déduisons donc que $\mathsf{K}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m$, et donc que $\mathsf{exec}^{(\alpha_0,\beta_0)}$ est une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide.
- pour tout $j \in [\![q]\!]$, $\{m_j\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ est d'une des formes voulues. De là, découle aussi que

  $$\mathsf{Est}(\mathsf{exec}^{(\alpha_0,\beta_0)}) = \mathsf{Est}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)}) \cup \{\{m_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \ldots, \{m_k\}_{\mathsf{shk}(\alpha_0,\beta_0)}\}.$$

  Nous pouvons donc conclure quant à la forme des sous-termes chiffrés de $\mathsf{exec}^{(\alpha_0,\beta_0)}$.

Cas $r_i = r_{a_{h+2}}$. Nous avons nécessairement que m est de la forme

$$m = \begin{array}{l} \{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)} \\ m_3 \end{array}$$

exec étant une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide $\mathsf{K}(\mathsf{exec}_{(i_n)-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m$. Aussi, nous savons qu'il existe $n' \in [\![i_n - 1]\!]$ tel que $e_{n'}$ soit de la forme $e_{n'} = \mathsf{snd}(\alpha_0, \beta_0, m')$ ou $e_{n'} = \mathsf{snd}(\beta_0, \alpha_0, m')$ avec m' de la forme

$$m' = \begin{array}{l} \{0, m_4, m_5\}_{\mathsf{shk}(\alpha_0,\beta_0)} \\ \{2, m_1, m_3, m_4\}_{\mathsf{shk}(\alpha_0,\beta_0)} \\ \{3, m_2, m_3, m_5\}_{\mathsf{shk}(\alpha_0,\beta_0)}. \end{array}$$

et que donc $\mathsf{K}(\mathsf{exec}_{(i_n)-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m'$. Or, d'après le lemme 4.4.1, tout cela implique que

- il existe $k'_1 \in [\![i_n - 1]\!]$ tel que $e_{k'_1}$ soit de la forme

  $$e_{k'_1} = \mathsf{snd}(\alpha_0, \beta_0, m'_1, \ldots, m'_{q'}) \quad \text{ou} \quad e_{k'_1} = \mathsf{snd}(\beta_0, \alpha_0, m'_1, \ldots, m'_{q'})$$

  et il existe $j' \in [\![q']\!]$ tel que $m'_{j'} = \{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)}$; et il existe aussi $k'_2 \in [\![i_n - 1]\!]$ tel que $e_{k'_2}$ soit de la forme

  $$e_{k'_2} = \mathsf{snd}(\alpha_0, \beta_0, m'_1, \ldots, m'_{q'}) \quad \text{ou} \quad e_{k'_2} = \mathsf{snd}(\beta_0, \alpha_0, m'_1, \ldots, m'_{q'})$$

  et il existe $j' \in [\![q']\!]$ tel que $m'_{j'} = m_3$. Mais alors, il existe $k''_1, k''_2 \in [\![n-1]\!]$ tels que $e_{i_{k''_1}} = e_{k'_1}$ et $e_{i_{k''_2}} = e_{k'_2}$, et donc $\mathsf{K}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m$. Nous déduisons donc que $\mathsf{K}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)}) \cup \mathcal{N}_\epsilon(\Pi) \vdash m$, et donc que $\mathsf{exec}^{(\alpha_0,\beta_0)}$ est une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide.

– $\{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ est d'une des formes voulues. De là, découle aussi que $m_3 \in (\mathcal{N} \cup \Sigma)$ et donc que

$$\mathsf{Est}(\mathsf{exec}^{(\alpha_0,\beta_0)}) = \mathsf{Est}(\mathsf{exec}_{n-1}^{(\alpha_0,\beta_0)}) \cup \{\{0, m_1, m_2\}_{\mathsf{shk}(\alpha_0,\beta_0)}\}.$$

Nous pouvons donc conclure quant à la forme des sous-termes chiffrés de $\mathsf{exec}^{(\alpha_0,\beta_0)}$.

A ce stade, nous avons donc établit que $\mathsf{exec}^{(\alpha_0,\beta_0)}$ est une exécution valide de $\Pi_{\mathcal{P}}$, au regard d'une connaissance de l'intrus vide. Encore faut-il, à présent, établir que $\mathsf{exec}^{(\alpha_0,\beta_0)}$ révèle le secret ν. Mais nous savons par construction que au moment où ν est engendré, il est émis sous la clé de chiffrement $\mathsf{shk}(\alpha_0, \beta_0)$, *i.e.* il existe un message $m \in \mathcal{M}$ tel que $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{m\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ et $\nu \in \mathcal{N}(m)$. Nous pouvons donc faire appel au lemme 4.4.1 pour conclure qu'il existe $n \in [\![\ell]\!]$ tel que e_n soit de la forme $e_n = \mathsf{snd}(\alpha_0, \beta_0, m_1, \ldots, m_q)$ ou $e_n = \mathsf{snd}(\beta_0, \alpha_0, m_1, \ldots, m_q)$, et qu'il existe $j \in [\![q]\!]$ tel que $m_j = \nu$. Mais alors par construction de $S^{(\alpha_0,\beta_0)}$, il existe $n'' \in [\![n-1]\!]$ tel que $e_{i_{n''}} = e_{n'}$, et donc $\mathsf{K}(\mathsf{exec}^{(\alpha_0,\beta_0)}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \nu$. Nous concluons alors que $\mathsf{exec}^{(\alpha_0,\beta_0)}$ est une attaque sur $\phi_{\mathcal{P}}$, au regard d'une connaissance initiale de l'intrus vide. □

4.4.2 Preuve de la proposition 4.3.7

A ce stade, il ne nous reste plus qu'à démontrer la proposition 4.3.7. Pour y parvenir nous établissons deux résultats préliminaires.

Le premier établit que pour tout mot de $\mathcal{P}$ il existe une exécution valide de $\Pi_{\mathcal{P}}$ dans laquelle il admette une représentation de type mot de $\mathcal{P}$; et réciproquement.

Proposition 4.4.2. *Soient Σ un alphabet, $\mathcal{P}$ une instance du problème de correspondance de Post sur Σ, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soit aussi (u, v) un couple de mots sur Σ,* i.e. *$u, v \in \Sigma^*$.*

> *(u, v) est un mot de $\mathcal{P}$ si et seulement si il existe une exécution $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$ telle que (u, v) admette dans exec une représentation de type mot de $\mathcal{P}$.*

Démonstration. Soient un alphabet $\Sigma = \{\mathsf{f}_1, \ldots, \mathsf{f}_r\}$, $\mathcal{P} = \{(u_i, v_i)\}_{i \in [\![h]\!]}$ une instance du problème de correspondance de Post sur Σ pour un certain h, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associés à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soit aussi (u, v) un couple de mots sur Σ, *i.e.* $u, v \in \Sigma^*$. D'après ce codage, $\Pi_{\mathcal{P}}$ est tel que nous avons $\mathcal{C}(\Pi_{\mathcal{P}}) = \{0, 1, 2, 3, \mathsf{f}_1, \ldots, \mathsf{f}_r\}$, $\mathcal{A}(\Pi_{\mathcal{P}}) = \{a_i, b_i \mid 0 \leq i \leq (h+2)\}$, et $\mathsf{Roles}(\Pi_{\mathcal{P}}) = \{r_{a_i}, r_{b_i} \mid 0 \leq i \leq (h+2)\}$.

($\Rightarrow$) Supposons que (u, v) est un mot de $\mathcal{P}$. Par définition il existe alors un entier $n \in \mathbb{N}^*$ tel qu'il existe un n-uplet $(i_1, \ldots, i_n) \in [\![h]\!]^n$ tel que $u = u_{i_1} \ldots u_{i_n}$ et $v = v_{i_1} \ldots v_{i_n}$. Nous allons montrer par induction sur n qu'il existe une exécution $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$ dans laquelle (u, v) admette une représentation de type mot de $\mathcal{P}$.

Cas de base : $n = 1$. Il existe alors $\eta \in [\![h]\!]$ tel que $u = u_\eta$ et $v = v_\eta$. Considérons le scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$, avec pour entrelacement $\mathsf{interlvg} =$

$[(r_{a_0}, sid); (r_{a_0}; sid)]$ et $\mathsf{initagts}(sid) = (a_0, b_0, \ldots, a_{h+2}, b_{h+2})$. La trace symbolique associée à sc est la suivante

$$
\begin{array}{lll}
tr = [& \mathsf{snd}(a_0, b_0, & \{2, n^{sid}_{(1,0)}, n^{sid}, n^{sid}_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m^{sid}_{(1,0)}, n^{sid}, m^{sid}_{(1,1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{2, n^{sid}_{(1,1)}, u^1_1, n^{sid}_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{2, n^{sid}_{(1,p_1)}, u^{p_1}_1, n^{sid}_{(1,p_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{3, m^{sid}_{(1,1)}, v^1_1, m^{sid}_{(1,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{3, m^{sid}_{(1,q_1)}, v^{q_1}_1, m^{sid}_{(1,q_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{1, n^{sid}_{(1,p_1+1)}, m^{sid}_{(1,q_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \ldots\\
& & \{2, n^{sid}_{(\eta,0)}, n^{sid}, n^{sid_1}_{(\eta,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m^{sid}_{(j,0)}, n^{sid}, m^{sid}_{(\eta,1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{2, n^{sid}_{(\eta,1)}, u^1_j, n^{sid}_{(\eta,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{2, n^{sid}_{(\eta,p_\eta)}, u^{p_\eta}_\eta, n^{sid}_{(\eta,p_\eta+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{3, m^{sid}_{(\eta,1)}, v^1_\eta, m^{sid}_{(\eta,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{3, m^{sid}_{(\eta,q_\eta)}, v^{q_\eta}_\eta, m^{sid}_{(\eta,q_1+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{1, n^{sid}_{(\eta,p_\eta+1)}, m^{sid}_{(\eta,q_\eta+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \ldots\\
& & \{2, n^{sid}_{(h,0)}, n^{sid}, n^{sid}_{(h,1)}\}_{\mathsf{shk}(a_0,b_0)}, \{3, m^{sid}_{(h,0)}, n^{sid}, m^{sid}_{(h,1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{2, n^{sid}_{(h,1)}, u^1_h, n^{sid}_{(h,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{2, n^{sid}_{(h,p_h)}, u^{p_h}_h, n^{sid}_{(h,p_h+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{3, m^{sid}_{(h,1)}, v^1_h, m^{sid}_{(h,2)}\}_{\mathsf{shk}(a_0,b_0)}, \ldots, \{3, m^{sid}_{(h,q_h)}, v^{q_h}_h, m^{sid}_{(h,q_h+1)}\}_{\mathsf{shk}(a_0,b_0)},\\
& & \{1, n^{sid}_{(h,p_h+1)}, m^{sid}_{(h,q_h+1)}\}_{\mathsf{shk}(a_0,b_0)});\\
& \mathsf{Secret}(a_0, b_0, \ldots, a_{h+2}, b_{h+2}, n^{sid}) &]
\end{array}
$$

Il est alors évident que

- $e_2 = \mathsf{Secret}(a_0, b_0, \ldots, a_{h+2}, b_{h+2}, n)$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, n^{sid}_{(\eta,p+1)}, m^{sid}_{(\eta,q+1)}\}_{\mathsf{shk}(a_0,b_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, n^{sid}_{(\eta,0)}, n^{sid}, n^{sid}_{(\eta,1)}\}_{shk(a_0,b_0)}$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, m^{sid}_{(\eta,0)}, n^{sid}, m^{sid}_{(\eta,1)}\}_{shk(a_0,b_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, n^{sid}_{(\eta,i)}, u^i_\eta, n^{sid}_{(\eta,i+1)}\}_{\mathsf{shk}(a_0,b_0)}$ pour tout $i \in [\![p_\eta]\!]$ et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, m^{sid}_{(\eta,i)}, v^i_\eta, m^{sid}_{(\eta,i+1)}\}_{\mathsf{shk}(a_0,b_0)}$ pour tout $i \in [\![q_\eta]\!]$.

Ce qui correspond précisément à la définition d'une 'représentation de type mot de $\mathcal{P}$ dans exec' pour (u_η, v_η) ; et donc nous avons bien que $(u, v) = (u_\eta, v_\eta)$ admet une représentation de type mot de $\mathcal{P}$ dans exec. Il est clair que tous les sous-termes chiffrés de exec sont d'une des formes admises dans les exécutions de $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, et que donc $\mathsf{exec} \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$.

Cas inductif : $n \geq 2$. Posons $u' = u_{i_1} \ldots u_{i_{n-1}} = \mathsf{f}_{\eta_1} \ldots \mathsf{f}_{\eta_p}$, et $v' = v_{i_1} \ldots v_{i_{n-1}} = \mathsf{f}_{\vartheta_1} \ldots \mathsf{f}_{\vartheta_q}$, *i.e.* $(u, v) = (u' u_{i_n}, v' v_{i_n})$. Par définition d'un mot de $\mathcal{P}$, (u', v') aussi est un mot de $\mathcal{P}$; auquel nous pouvons appliquer notre hypothèse d'induction et conclure qu'il existe un scénario $\mathsf{sc} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$ de $\Pi_{\mathcal{P}}$ et une substitution σ tels que $tr\sigma \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, où tr est la trace symbolique associée au scénario sc, et tels que (u', v') admette une représentation de type mot de $\mathcal{P}$ dans exec, *i.e.*

- il existe $2(h+3)$ entités honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$,
- il existe un entier $k \in [\![\ell]\!]$,
- il existe $p + q + 5$ nonces distincts $\nu, \nu_0, \ldots \nu_{p+1}, \ldots, \mu_0, \ldots, \mu_{q+1} \in \mathcal{N}$,

tels que

- $e_k = \mathsf{Secret}(\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}, \nu)$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{shk(\alpha_0,\beta_0)}$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{shk(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, \mathsf{f}_{\eta_j}, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$ et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_j, \mathsf{f}_{\vartheta_j}, \mu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![q]\!]$.

Considérons le scénario $\mathsf{sc}' = (\mathsf{interlvg}, \mathsf{initagts})$ avec pour entrelacement $\mathsf{interlvg} = [(r_{b_{i_n}}, sid); (r_{b_{i_n}}, sid)]$ où sid est un identificateur de session n'apparaissant pas déjà sc, *i.e.* $sid \neq sid_j$ pour tout $j \in [\![\ell]\!]$, et $\mathsf{initagts}(sid) = (\alpha'_0, \beta'_0, \ldots, \alpha'_{h+1}, \beta'_{h+2})$ quelconque mais avec $\alpha'_{i_n} = \alpha_0$ et $\beta'_{i_n} = \beta_0$. La trace symbolique associée à $\mathsf{sc} \ @ \ \mathsf{sc}'$ est

$$
\begin{array}{rcll}
tr' & = & tr \ @ \ [& \mathsf{rcv}(\beta_0, \alpha_0, \quad \{1, x^{sid}_{(h+i_n,1)}, y^{sid}_{(h+i_n,1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}); \\
& & & \mathsf{snd}(\beta_0, \alpha_0, \quad \{2, x^{sid}_{(h+i_n,1)}, u^1_{i_n}, n^{sid}_{(h+i_n,2)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \ldots, \\
& & & \qquad \{2, n^{sid}_{(h+i_n,p_{i_n})}, c^{p_{i_n}}_{u^{i_n}_{i_n}}, n^{sid}_{(h+i_n,p_{i_n}+1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\
& & & \{3, y^{sid}_{(h+i_n,1)}, v^1_{i_n}, m^{sid}_{(h+i_n,2)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \ldots, \\
& & & \qquad \{3, m^{sid}_{(h+i_n,q_{i_n})}, v^{q_{i_n}}_{i_k}, m^{sid}_{(h+i_n,q_{i_n}+1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\
& & & \{1, n^{sid}_{(h+i_k,p_{i_n}+1)}, m^{sid}_{(h+i_n,q_{i_n}+1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)})
\end{array}
$$

où $x^{sid}_{(h+i_n,1)}$ et $y^{sid}_{(h+i_n,1)}$ sont des variables fraîches, *i.e.* n'apparaissant pas dans tr. Il est évident que pour la substitution $\sigma' = \{x^{sid}_{(h+i_n,1)} \mapsto \nu_{p+1}, y^{sid}_{(h+i_n,1)} \mapsto \mu_{q+1}\}$, $\mathsf{exec} = tr'\sigma\sigma$ est une exécution dans $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ dans laquelle (u, v) admet une représentation de type mot de $\mathcal{P}$. Ce qui achève notre induction et établit la première implication.

($\Leftarrow$) Supposons, à présent, qu'il existe une exécution $\mathsf{exec} \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ telle que (u, v) admette une représentation de type mot de $\mathcal{P}$ dans exec. Etant donné que exec est une exécution valide de $\Pi_{\mathcal{P}}$, il existe par définition un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de $\Pi_{\mathcal{P}}$ avec $\mathsf{interlvg} = [(sid_1, r_1); \ldots; (sid_\ell, r_\ell)]$, et une substitution close σ, tels que $\mathcal{V}(tr) \subseteq \mathrm{dom}(\sigma)$, et $\mathsf{exec} = tr\sigma$ où tr est la trace symbolique associée à sc.

Posons $(u, v) = (\mathfrak{f}_{i_1} \ldots \mathfrak{f}_{i_p}, \mathfrak{f}_{j_1} \ldots \mathfrak{f}_{j_q})$ (avec $i_1, \ldots, i_p, j_1, \ldots, j_q \in [\![r]\!]$). Par définition de '$(u, v)$ admet une représentation de type mot de $\mathcal{P}$ dans exec',

- il existe $k \in [\![\ell]\!]$,
- il existe $2(h+3)$ entités honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$, et
- il existe $p + q + 5$ nonces $\nu, \nu_0, \ldots, \nu_{p+1}, \mu_0, \ldots, \mu_{q+1} \in \mathcal{N}$

tels que

- $e_k = \mathsf{Secret}(\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2})$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{i_j}, \mathfrak{f}_{i_j}, \nu_{i_{j+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{j_i}, \mathfrak{f}_{j_i}, \nu_{j_{i+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$.

Nous allons montrer par induction sur n, (pour $n \in [\![\ell]\!]$), que si

- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{i_j}, \mathfrak{f}_{i_j}, \nu_{i_{j+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{j_i}, \mathfrak{f}_{j_i}, \nu_{j_{i+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$.

alors (u, v) est un mot de $\mathcal{P}$.

Cas de base : $n = 0$. L'antécédent de notre implication n'étant pas vérifié nous pouvons conclure trivialement.

Cas inductif : $n \geq 1$. Si e_n est une réception ou un status event, alors $\mathsf{K}(\mathsf{exec}_n) = \mathsf{K}(\mathsf{exec}_{n-1})$, et nous pouvons donc conclure par hypothèse d'induction. Supposons que e_n est une émission, et que

- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{i_j}, \mathsf{f}_{i_j}, \nu_{i_{j+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{j_i}, \mathsf{f}_{j_i}, \nu_{j_{i+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$.

Il existe alors deux agents $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$ et un message t tel que $e_n = \mathsf{snd}(\alpha, \beta, t)$. Nous procédons par analyse de cas sur le rôle r_n incarné par sid_n.

Cas $r_n = r_{a_0}$. Ce cas ne peut pas survenir étant donné qu'aucune émission n'est spécifiée dans le corps de r_{a_o}.

Cas $r_n = r_{b_0}$. La seule émission du rôle r_n étant le premier événement du corps de r_{b_0}, t est nécessairement de la forme

$$
\begin{aligned}
t \quad = \quad & \{2, \kappa_{(1,0)}, \kappa_0, \kappa_{(1,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \lambda_{(1,0)}, \kappa_0, \lambda_{(1,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{2, \kappa_{(1,1)}, u_1^1, \kappa_{(1,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{2, \kappa_{(1,p_1)}, u_1^{p_1}, \kappa_{(1,p_1+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{3, \lambda_{(1,1)}, v_1^1, \lambda_{(1,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{3, \lambda_{(1,q_1)}, v_1^{q_1}, \lambda_{(1,q_1+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{1, \kappa_{(1,p_1+1)}, \lambda_{(1,q_1+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \ldots \\
& \{2, \kappa_{(h,0)}, \kappa_0, \kappa_{(h,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \lambda_{(h,0)}, \kappa_0, \lambda_{(h,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{2, \kappa_{(h,1)}, u_h^1, \kappa_{(h,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{2, \kappa_{(h,p_h)}, u_h^{p_h}, \kappa_{(h,p_h+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{3, \lambda_{(h,1)}, v_h^1, \lambda_{(h,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{3, \lambda_{(h,q_h)}, v_h^{q_h}, \lambda_{(h,q_h+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{1, \kappa_{(h,p_h+1)}, \lambda_{(h,q_h+1)}\}_{\mathsf{shk}(\alpha,\beta)}
\end{aligned}
$$

où pour tout $i_1, i_2 \in \mathbb{N}$, κ_0, $\kappa_{(i_1,i_2)}$ et $\lambda_{(i_1,i_2)}$ sont des nonces frais, *i.e.* κ_0, $\kappa_{(i_1,i_2)}$, $\lambda_{(i_1,i_2)} \in \mathcal{N}$, et κ_0, $\kappa_{(i_1,i_2)}$, $\lambda_{(i_1,i_2)} \notin \mathsf{St}(\mathsf{exec}_{n-1})$. De plus, comme $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$, α et β sont honnêtes.

Supposons que les messages impliqués dans la représentation de type mot de $\mathcal{P}$ de (u, v) ne soient pas tous déductibles dès exec_{n-1}, auquel cas nous pourrions conclure par hypothèse d'induction. Alors comme d'après le lemme 4.3.4 $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$, et comme κ_0, les $\kappa_{(i_1,i_2)}$ et les $\lambda_{(i_1,i_2)}$ sont frais, nous avons nécessairement qu'il existe $\eta \in [\![h]\!]$ tel que $(u, v) = (u_\eta, v_\eta)$. Or, par définition des mots de $\mathcal{P}$, (u_η, v_η) est un mot de $\mathcal{P}$. Nous concluons donc que (u, v) est bien un mot de $\mathcal{P}$.

Cas $r_n = r_{a_\eta}$ *avec* $\eta \in [\![h]\!]$. La seule émission du rôle r_n étant le premier événement du corps de r_{a_η}, t est nécessairement de la forme

$$t = \{1, \kappa_{(h+\eta,1)}, \lambda_{(h+\eta,1)}\}_{\mathsf{shk}(\alpha,\beta)},$$

où $\kappa_{(h+\eta,1)}$ et $\lambda_{(h+\eta,1)}$ sont des nonces frais, et α et β sont des agent honnêtes. Or $\kappa_{(h+\eta,1)}$ et $\lambda_{(h+\eta,1)}$ étant frais et comme $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$, il est impossible que t soit impliqué dans la représentation de type mot de $\mathcal{P}$ de (u, v) considérée ; il est donc nécessaire que tous les messages de la représentation de

(u, v) considérée soient déductibles dès exec_{n-1}. Nous pouvons conclure donc par hypothèse d'induction que (u, v) est un mot de $\mathcal{P}$.

Cas $r_n = r_{b_\eta}$ *avec* $\eta \in [\![h]\!]$. La seule émission du rôle r_n étant le second événement du corps de r_{b_η}, et comme $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$, t est nécessairement de la forme

$$\begin{aligned} t = \quad & \{2, \kappa, u_\eta^1, \kappa_{(h+\eta,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{\kappa_{(h+\eta,p_\eta)}, u_\eta^{p_\eta}, \kappa_{(h+\eta,p_\eta+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\ & \{3, \lambda, v_\eta^1, \lambda_{(h+\eta,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{\lambda_{(h+\eta,q_\eta)}, v_i^{q_\eta}, \lambda_{(h+\eta,q_\eta+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\ & \{1, \kappa_{(h+\eta,p_\eta+1)}, \lambda_{(h+\eta,q_\eta+1)}\}_{\mathsf{shk}(\alpha,\beta)} \end{aligned}$$

où $\kappa_{(h+\eta,2)}, \ldots, \kappa_{(h+\eta,p_\eta+1)}, \lambda_{(h+\eta,2)}, \ldots,$ et $\lambda_{(h+\eta,q_\eta+1)}$ sont des nonces frais, $\kappa, \lambda \in \mathcal{N}$, et α et β sont des agent honnêtes. De plus, nous savons par construction de r_{b_η} qu'il existe $n' \in [\![n-1]\!]$, tel que $e_{n'} = \mathsf{rcv}(\alpha, \beta, \{1, \kappa, \lambda\}_{\mathsf{shk}(\alpha,\beta)})$, et comme exec est une exécution valide de $\Pi_{\mathcal{P}}$, il est nécessaire que $\mathsf{K}(\mathsf{exec}_{n'}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, \kappa, \lambda\}_{\mathsf{shk}(\alpha,\beta)}$, et donc $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, \kappa, \lambda\}_{\mathsf{shk}(\alpha,\beta)}$.

Supposons que les messages impliqués dans la représentation de type mot de $\mathcal{P}$ de (u, v) ne soient pas tous déductibles dès exec_{n-1}, auquel cas nous pourrions conclure par hypothèse d'induction. Sachant que $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$, et les nonces $\kappa_{(h+\eta,2)}, \ldots, \kappa_{(h+\eta,p_\eta+1)}, \lambda_{(h+\eta,2)}, \ldots,$ et $\lambda_{(h+\eta,q_\eta+1)}$ étant frais, il est nécessaire que

- $\kappa = \nu_{p-p_\eta+1}$ et $\lambda = \mu_{q-q_\eta+1}$,
- $\kappa_{(h+\eta,j+1)} = \nu_{p-p_\eta+1+j}$ et $\mathsf{f}_{i_{p-p_\eta+j}} = u_\eta^j$ pour tout $j \in [\![p_\eta]\!]$, et
- $\lambda_{(h+\eta,i+1)} = \mu_{q-q_\eta+1+i}$ et $\mathsf{f}_{j_{q-q_\eta+i}} = v_\eta^i$ pour tout $i \in [\![q_\eta]\!]$.

et que

- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, \mathsf{f}_{i_j}, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p - p_\eta]\!]$, et $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_i, \mathsf{f}_{j_i}, \mu_{i+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q - q_\eta]\!]$.

Alors pour $u' = \mathsf{f}_{i_1} \ldots \mathsf{f}_{i_{p-p_\eta}}$, et $v' = \mathsf{f}_{j_1} \ldots \mathsf{f}_{j_{q-q_\eta}}$, nous savons que $u = u'u_\eta$ et $v = v'v_\eta$. De plus, (u', v') admet une représentation de type mot de $\mathcal{P}$ dans exec, et les messages impliqués dans cette représentation sont dérivables dès exec_{n-1}. Par hypothèse d'induction nous sommes donc en droit de conclure que (u', v') est un mot de $\mathcal{P}$. Mais sachant que $(u, v) = (u'u_\eta, v'v_\eta)$, nous concluons par définition d'un mot de $\mathcal{P}$ que (u, v) est un mot de $\mathcal{P}$.

Cas $r_n = r_{a_{h+1}}$. Ce cas est analogue au cas $r_n = r_{a_\eta}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_n = r_{b_{h+1}}$. La seule émission du rôle r_n étant le second événement du corps de $r_{b_{h+1}}$, t est nécessairement de la forme $t = \{0, \kappa, \lambda\}_{\mathsf{shk}(\alpha,\beta)}$, où $\kappa, \lambda \in \mathcal{N}$, et α et β sont des agent honnêtes (ceci tient du fait que $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$). Comme d'après le lemme 4.3.4 $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$, tous les messages impliqués dans la représentation de (u, v) sont dérivables dès exec_{n-1}. Nous pouvons donc conclure par hypothèse d'induction.

Cas $r_n = r_{a_{h+2}}$. Ce cas est analogue au cas $r_n = r_{a_\eta}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_n = r_{b_{h+2}}$. Ce cas est analogue au cas $r_n = r_{a_{b_{h+1}}}$. Pour cette raison nous ne le détaillons pas ici.

Ainsi s'achève notre induction. Nous avons donc établit la seconde implication. □

Lemme 4.4.3. *Soient* Σ *un alphabet,* $\mathcal{P}$ *une instance du problème de correspondance de Post sur* Σ*, et* $\Pi_{\mathcal{P}}$ *et* $\phi_{\mathcal{P}}$ *le protocole et la formule associés à* $\mathcal{P}$ *d'après le codage présenté à la section 4.3.1. Soient une exécution* $\mathsf{exec} = [e_1; \dots; e_\ell] \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$*. Soient aussi*

- $2(h+3)$ *entités honnêtes* $\alpha_0, \beta_0, \dots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$,
- *un entier* $k \in [\![\ell]\!]$,
- $p+q$ *termes* $t_1, \dots, t_q, u_1, \dots, u_q \in \mathcal{T}$*, et*
- $p+q+5$ *nonce* $\nu, \nu_0, \dots, \nu_{p+1}, \mu_0, \dots, \mu_{q+1} \in \mathcal{N}$.

pour deux entiers p *et* q *quelconques. Si*

- $e_k = \mathsf{Secret}(\beta_0, \alpha_0, \beta_0, \dots, \alpha_{h+2}, \beta_{h+2}, \nu)$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$*, et*
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, t_j, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ *pour tout* $i \in [\![p]\!]$*, et*
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_j, u_j, \mu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ *pour tout* $j \in [\![q]\!]$,

alors les t_i *et les* u_j *(pour* $i \in [\![p]\!]$ *et* $j \in [\![q]\!]$*) sont des constantes dans* Σ.

Ce lemme pouvant être prouvé par induction, exactement comme l'implication ($\Rightarrow$) de la proposition précédente, nous avons fait le choix de ne pas la présenté ici.

Nous en venons à présent à la proposition que nous visions initialement.

Proposition 4.3.7. *Soient* Σ *un alphabet,* $\mathcal{P}$ *une instance du problème de correspondance de Post sur* Σ*, et* $\Pi_{\mathcal{P}}$ *et* $\phi_{\mathcal{P}}$ *le protocole et la formule associés à* $\mathcal{P}$ *d'après le codage présenté à la section 4.3.1. Soit aussi* (u, v) *un couple de mots sur* Σ, i.e. $u, v \in \Sigma^*$.

> (u, v) *est une présolution de* $\mathcal{P}$ *si et seulement si il existe une exécution* $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$ *telle que* (u, v) *admette dans* exec *une représentation de type présolution de* $\mathcal{P}$.

Démonstration. Soit $\Sigma = \{\mathfrak{f}_1, \dots, \mathfrak{f}_r\}$, $\mathcal{P} = \{(u_i, v_i)\}_{i \in [\![h]\!]}$ une instance du problème de correspondance de Post sur Σ pour un certain h, et $\Pi_{\mathcal{P}}$ et $\phi_{\mathcal{P}}$ le protocole et la formule associée à $\mathcal{P}$ d'après le codage présenté à la section 4.3.1. Soit (u, v) un couple de mots sur Σ, *i.e.* $u, v \in \Sigma^*$. D'après ce codage, $\Pi_{\mathcal{P}}$ est tel que $\mathcal{C}(\Pi_{\mathcal{P}}) = \{0, 1, 2, 3, \mathfrak{f}_1, \dots, \mathfrak{f}_r\}$, $\mathcal{A}(\Pi) = \{a_i, b_i \mid 0 \leq i \leq (h+2)\}$, et $\mathsf{Roles}(\Pi) = \{r_{a_i}, r_{b_i} \mid 0 \leq i \leq (h+2)\}$.

($\Rightarrow$) Supposons que (u, v) est une présolution de $\mathcal{P}$ avec $u = \mathfrak{f}_{i_1} \dots \mathfrak{f}_{i_p}$ et $v = \mathfrak{f}_{j_1} \dots \mathfrak{f}_{j_q}$. Il existe par définition un mot $w = \mathfrak{f}_{\eta_1} \dots \mathfrak{f}_{\eta_s} \in \Sigma^*$ tel que (uw, vw) soit un mot de $\mathcal{P}$. D'après la proposition 4.4.2 nous savons qu'il existe alors un scénario sc de $\Pi_{\mathcal{P}}$, ainsi qu'une substitution close σ tels que $\mathsf{exec} = tr\sigma =$

$[e'_1; \ldots; e'_\ell]$ où tr est la trace symbolique associée à sc est une exécution dans $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ dans laquelle (uw, vw) admet une représentation de type mot de $\mathcal{P}$, *i.e.*

- il existe $2(h+3)$ entités honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{P} \setminus \{\epsilon\})$,
- il existe un entier $k \in [\![\ell]\!]$,
- il existe $p+q+2s+5$ nonces distincts ν, ν_0, ν_1, …, ν_{p+s+1}, μ_0, μ_1, …, $\mu_{q+s+1} \in \mathcal{N}$

tels que

- $e_i = \mathsf{Secret}(\beta_0, \alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}, \nu)$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \cup\mathcal{N}_\epsilon(\Pi)\{1, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{shk(\alpha_0,\beta_0)}$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{shk(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, \mathsf{f}_{i_j}, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_i, \mathsf{f}_{j_i}, \mu_{i+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{p+j}, \mathsf{f}_{\eta_j}, \nu_{p+j+1}$, pour tout $j \in [\![r]\!]$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \mu_{q+j}, \mathsf{f}_{\eta_j}, \nu_{q+j+1}$, pour tout $j \in [\![r]\!]$.

Nous procédons par induction sur la longueur $s = |w|$.

Cas de base : $s = 0$. Ce cas peut bien survenir car tout mot de $\mathcal{P}$ est une présolution de $\mathcal{P}$. Nous construisons le scénario $\mathsf{sc}' = (\mathsf{interlvg}, \mathsf{initagts})$ avec $[(r_{b_{h+1}}, sid); (r_{b_{h+1}}, sid)]$ où sid est un identificateur de session n'apparaissant pas déjà dans sc et $\mathsf{initagts} = (\alpha'_0, \beta'_0, \ldots, \alpha'_{h+2}, \beta'_{h+2})$ quelconque mais avec $\alpha_{h+1} = \alpha_0$ et $\beta_{h+1} = \beta_0$. La trace symbolique associée à $\mathsf{sc} \mathbin{@} \mathsf{sc}'$ est

$$tr' \;=\; tr \mathbin{@} [\;\; \begin{array}{ll} \mathsf{rcv}(\beta_0, \alpha_0, & \{1, x^{sid}_{(2h+1,0)}, y^{sid}_{(2h+1,0)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}); \\ \mathsf{snd}(\beta_0, \alpha_0, & \{0, x^{sid}_{(2h+1,0)}, y^{sid}_{(2h+1,0)}\}_{\mathsf{shk}(\alpha_0,\beta_0)})] \end{array}$$

où $x^{sid}_{(2h+1,0)}$ et $y^{sid}_{(2h+1,0)}$ sont deux variables fraîches, *i.e.* n'apparaissant pas dans tr. Il est alors évident que pour la substitution

$$\sigma' = \{x^{sid}_{(2h+1,0)} \mapsto \nu_{p+s+1}, y^{sid}_{(2h+1,0)} \mapsto \mu_{q+x+1}\},$$

$\mathsf{exec}' = tr'\sigma\sigma'$ est une exécution dans $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ dans laquelle $(u, v) = (uw, vw)$ admet une représentation de type présolution de $\mathcal{P}$.

Cas inductif : $s \geq 1$. Par définition d'une présolution de $\mathcal{P}$, nous avons que $(u\mathsf{f}_{\eta_1}, v\mathsf{f}_{\eta_1})$ est aussi une présolution de $\mathcal{P}$ à laquelle nous pouvons appliquer notre hypothèse d'induction et conclure qu'il existe un scénario sc de $\Pi_{\mathcal{P}}$ et une substitution close σ tels que $\mathsf{exec} = tr\sigma = [e_1; \ldots; e_\ell]$, où tr est la trace symbolique associée à sc, soit dans $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, et tels que $(u\mathsf{f}_{\eta_1}, v\mathsf{f}_{\eta_1})$ admette une représentation de type présolution dans exec, *i.e.* tels que

- il existe $2(h+3)$ entités honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{P} \setminus \{\epsilon\})$,
- il existe un entier $k \in [\![\ell]\!]$,
- il existe $p+q+7$ nonces distincts $\nu, \nu_0, \nu_1, \ldots, \nu_{p+2}, \mu_0, \mu_1, \ldots, \mu_{q+2} \in \mathcal{N}$

tels que

- $e_k = \mathsf{Secret}(\beta_0, \alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2}, \nu)$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \nu_{p+2}, \mu_{q+2}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{shk(\alpha_0,\beta_0)}$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{shk(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, \mathsf{f}_{i_j}, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$ et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_i, \mathsf{f}_{j_i}, \mu_{i+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$,

– $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{p+1}, \mathfrak{f}_{\eta_1}, \nu_{p+2}\}$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{q+1}, \mathfrak{f}_{\eta_1}, \mu_{q+2}\}$.

Considérons à présent le scénario $\mathsf{sc}' = (\mathsf{interlvg}, \mathsf{initagts})$ avec pour entrelacement $\mathsf{interlvg} = [(r_{b_{h+2}}, sid); (r_{b_{h+2}}, sid)]$ où sid est un identificateur de session n'apparaissant pas déjà dans sc, et $\mathsf{initagts}(sid) = (\alpha'_0, \beta'_0, \ldots, \alpha'_{h+2}, \beta'_{h+2})$ quelconque mais avec $\alpha'_{h+2} = \alpha_0$ et $\beta'_{h+2} = \beta_0$. La trace symbolique associée à $\mathsf{sc} \mathbin{@} \mathsf{sc}'$ est

$$\begin{array}{rcll} tr' & = & tr \mathbin{@} [& \mathsf{rcv}(\beta_0, \alpha_0, & \{0, x^{sid}_{(2h+2,1)}, y^{sid}_{(2h+2,1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ & & & & \{2, x^{sid}_{(2h+2,2)}, x^{sid}_{(2h+2,0)}, x^{sid}_{(2h+2,1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ & & & & \{3, y^{sid}_{(2h+2,2)}, x^{sid}_{(2h+2,0)}, y^{sid}_{(2h+2,1)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}); \\ & & & \mathsf{snd}(\beta_0, \alpha_0, & \{0, x^{sid}_{(2h+2,2)}, y^{sid}_{(2h+2,2)}\}_{\mathsf{shk}(\alpha_0,\beta_0)}, \\ & & & & x^{sid}_{(2h+2,0)})] \end{array}$$

où $x^{sid}_{(2h+2,0)}, x^{sid}_{(2h+2,1)}, y^{sid}_{(2h+2,1)}, x^{sid}_{(2h+2,2)}$, et $y^{sid}_{(2h+2,2)}$ sont des variables fraîches, *i.e.* n'apparaissant pas dans tr. Il est alors facile de voir que pour la substitution $\sigma' = \{x^{sid}_{(2h+2,0)} \mapsto c_{a_{\eta_1}}, x^{sid}_{(2h+2,1)} \mapsto \nu_{p+2}, y^{sid}_{(2h+2,1)} \mapsto \mu_{q+2}, x^{sid}_{(2h+2,2)} \mapsto \nu_{p+1}, y^{sid}_{(2h+2,2)} \mapsto \mu_{q+1}\}$, $\mathsf{exec} = tr\sigma\sigma'$ est une exécution de $\mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$ dans laquelle (u, v) admet une représentation de type présolution de $\mathcal{P}$. Ce qui achève notre induction, et établit la première implication.

($\Leftarrow$) Supposons, à présent, qu'il existe une exécution $\mathsf{exec} \in \mathcal{E}^{\mathsf{bnd}}_{\mathcal{P}}$, telle que (u, v) admette une représentation de type présolution dans exec. Etant donné que exec est une exécution valide de $\Pi_{\mathcal{P}}$, il existe par définition un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de $\Pi_{\mathcal{P}}$ avec $\mathsf{interlvg} = [(sid_1, r_1); \ldots; (sid_\ell, r_\ell)]$, et une substitution close σ, tels que $\mathcal{V}(tr) \subseteq \mathrm{dom}(\sigma)$, et $\mathsf{exec} = tr\sigma$ où tr est la trace symbolique associée à sc.

Posons $(u, v) = (\mathfrak{f}_{i_1} \ldots \mathfrak{f}_{i_p}, \mathfrak{f}_{j_1} \ldots \mathfrak{f}_{j_q})$ (avec $i_1, \ldots, i_p, j_1, \ldots, j_q \in [\![r]\!]$). Par définition de '$(u, v)$ admet une représentation de type présolution de $\mathcal{P}$ dans exec',

– il existe $k \in [\![\ell]\!]$,
– il existe $2(h+3)$ entités honnêtes $\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2} \in (\mathcal{A} \setminus \{\epsilon\})$, et
– il existe $p + q + 5$ nonces $\nu, \nu_0, \ldots, \nu_{p+1}, \mu_0, \ldots, \mu_{q+1} \in \mathcal{N}$

tels que

– $e_k = \mathsf{Secret}(\alpha_0, \beta_0, \ldots, \alpha_{h+2}, \beta_{h+2})$,
– $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
– $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
– $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{i_j}, \mathfrak{f}_{i_j}, \nu_{i_{j+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{j_i}, \mathfrak{f}_{j_i}, \nu_{j_{i+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$.

Nous allons montrer par induction sur n, (pour $n \in [\![\ell]\!]$), que si

– $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
– $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
– $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{i_j}, \mathfrak{f}_{i_j}, \nu_{i_{j+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{j_i}, \mathfrak{f}_{j_i}, \nu_{j_{i+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$.

alors (u, v) est une présolution de $\mathcal{P}$.

Cas de base : $n = 0$. L'antécédent de notre implication n'étant pas vérifié nous pouvons conclure trivialement.

Cas inductif : $n \geq 1$. Si e_n est une réception ou un status event, alors $\mathsf{K}(\mathsf{exec}_n) = \mathsf{K}(\mathsf{exec}_{n-1})$, et nous pouvons donc conclure par hypothèse d'induction. Supposons que e_n est une émission, et que

- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \nu_{p+1}, \mu_{q+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
- $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{i_j}, \mathsf{f}_{i_j}, \nu_{i_{j+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{j_i}, \mathsf{f}_{j_i}, \nu_{j_{i+1}}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$.

Il existe alors deux agents $\alpha, \beta \in (\mathcal{A} \setminus \{\epsilon\})$ et un message t tel que $e_n = \mathsf{snd}(\alpha, \beta, t)$. Nous procédons par analyse de cas sur le rôle r_n incarné par sid_n.

Cas $r_n = r_{a_0}$. Ce cas ne peut pas survenir car aucune émission n'est spécifiée dans le corps de r_{a_0}.

Cas $r_n = r_{b_0}$. La seule émission du rôle r_n étant le premier événement du corps de r_{b_0}, t est nécessairement de la forme

$$
\begin{aligned}
t \quad = \quad & \{2, \kappa_{(1,0)}, \kappa_0, \kappa_{(1,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \lambda_{(1,0)}, \kappa_0, \lambda_{(1,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{2, \kappa_{(1,1)}, u_1^1, \kappa_{(1,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{2, \kappa_{(1,p_1)}, u_1^{p_1}, \kappa_{(1,p_1+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{3, \lambda_{(1,1)}, v_1^1, \lambda_{(1,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{3, \lambda_{(1,q_1)}, v_1^{q_1}, \lambda_{(1,q_1+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{1, \kappa_{(1,p_1+1)}, \lambda_{(1,q_1+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \ldots \\
& \{2, \kappa_{(h,0)}, \kappa_0, \kappa_{(h,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \lambda_{(h,0)}, \kappa_0, \lambda_{(h,1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{2, \kappa_{(h,1)}, u_h^1, \kappa_{(h,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{2, \kappa_{(h,p_h)}, u_h^{p_h}, \kappa_{(h,p_h+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{3, \lambda_{(h,1)}, v_h^1, \lambda_{(h,2)}\}_{\mathsf{shk}(\alpha,\beta)}, \ldots, \{3, \lambda_{(h,q_h)}, v_h^{q_h}, \lambda_{(h,q_h+1)}\}_{\mathsf{shk}(\alpha,\beta)}, \\
& \{1, \kappa_{(h,p_h+1)}, \lambda_{(h,q_h+1)}\}_{\mathsf{shk}(\alpha,\beta)}
\end{aligned}
$$

où pour tout $i_1, i_2 \in \mathbb{N}$, κ_0, $\kappa_{(i_1,i_2)}$ et $\lambda_{(i_1,i_2)}$ sont des nonces frais, *i.e.* κ_0, $\kappa_{(i_1,i_2)}$, $\lambda_{(i_1,i_2)} \in \mathcal{N}$, et κ_0, $\kappa_{(i_1,i_2)}$, $\lambda_{(i_1,i_2)} \notin \mathsf{St}(\mathsf{exec}_{n-1})$. Or d'après le lemme 4.3.4 $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \not\vdash \mathsf{shk}(\alpha, \beta)$, et comme κ_0, les $\kappa_{(i_1,i_2)}$ et les $\lambda_{(i_1,i_2)}$ sont tous des nonces frais, il est nécessaire que les messages impliqués dans la représentation de type présolution de $\mathcal{P}$ de (u, v) considérée soient déductibles dès exec_{n-1}. Nous concluons donc par hypothèse d'induction que (u, v) est une présolution de $\mathcal{P}$.

Cas $r_n = r_{a_\eta}$ *avec* $\eta \in [\![h]\!]$. Ce cas est analogue au cas $r_n = r_{a_0}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_n = r_{b_\eta}$ *avec* $\eta \in [\![h]\!]$. Ce cas est analogue au cas $r_n = r_{a_0}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_n = r_{a_{h+1}}$. Ce cas est analogue au cas $r_n = r_{a_0}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_n = r_{b_{h+1}}$. La seule émission du rôle r_n étant le second événement du corps de $r_{b_{h+1}}$, t est nécessairement de la forme $t = \{0, \kappa, \lambda\}_{\mathsf{shk}(\alpha,\beta)}$, où $\kappa, \lambda \in \mathcal{N}$ (ceci tient du fait que $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$). Supposons que tous les messages impliqués dans la représentation de type présolution de $\mathcal{P}$ de (u, v) considérée ne soient pas tous dérivables dès exec_{n-1}, auquel cas nous pourrions conclure par hypothèse d'induction. Par construction de $r_{b_{h+1}}$, il doit exister $n' \in [\![k-1]\!]$ tel que $e_{n'} = \mathsf{rcv}(\alpha, \beta, \{1, \kappa, \lambda\}_{\mathsf{shk}(\alpha,\beta)})$, et comme exec est une exécution valide de $\Pi_{\mathcal{P}}$, $\mathsf{K}(\mathsf{exec}_{n'}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{1, \kappa, \lambda\}_{\mathsf{shk}(\alpha_0,\beta_0)}$. De plus, comme d'après le lemme 4.3.4 $\mathsf{K}(\mathsf{exec}_n) \cup \mathcal{N}_\epsilon(\Pi) \nvdash \mathsf{shk}(\alpha, \beta)$, il est nécessaire que

- $\kappa = \nu_{p+1}$ et $\lambda = \mu_{q+1}$,
- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,et
  $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, \mathsf{f}_{i_j}, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_i, \mathsf{f}_{j_i}, \mu_{i+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$.

Nous pouvons donc faire appel à la proposition 4.4.2 et conclure que (u, v) est un mot de $\mathcal{P}$. Or, s'il est un mot de $\mathcal{P}$, nous avons déjà vu que (u, v) est aussi une présolution de $\mathcal{P}$.

Cas $r_n = r_{a_{h+2}}$. Ce cas est analogue au cas $r_n = r_{a_0}$. Pour cette raison nous ne le détaillons pas ici.

Cas $r_n = r_{b_{h+2}}$. La seule émission du rôle r_n étant le second événement du corps de $r_{b_{h+2}}$, et exec étant une exécution dans $\mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$, t est nécessairement de la forme

$$t = \{0, \kappa, \lambda\}_{\mathsf{shk}(\alpha,\beta)}, \; t'$$

où $\kappa, \lambda, \nu \in \mathcal{N}$.

Supposons que les messages impliqués dans la représentation de type présolution de $\mathcal{P}$ de (u, v) ne soient pas tous dérivables dès exec_n, auquel cas nous pourrions conclure par hypothèse d'induction. Nous savons par construction de $r_{b_{h+2}}$ qu'il existe un entier $n' \in [\![n-1]\!]$ ainsi qu'un terme t'' tels que $e_{n'} = \mathsf{rcv}(\alpha, \beta, t'')$, et comme $\mathsf{exec} \in \mathcal{E}_{\mathcal{P}}^{\mathsf{bnd}}$, il est nécessaire que t'' soit de la forme

$$t'' = \{0, \kappa_1, \lambda_1\}_{\mathsf{shk}(\alpha,\beta)}, \; \{2, \kappa, t', \kappa_1\}_{\mathsf{shk}(\alpha,\beta)}, \; \{3, \lambda, t', \lambda_1\}_{\mathsf{shk}(\alpha,\beta)}.$$

Mais d'après le lemme 4.4.3 nous savons que $t' \in \Sigma$. De plus, comme exec est une exécution valide de $\Pi_{\mathcal{P}}$ nous avons nécessairement que

$$\mathsf{K}(\mathsf{exec}_{n'}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \kappa_1, \lambda_1\}_{\mathsf{shk}(\alpha,\beta)}, \; \{2, \kappa, t', \kappa_1\}_{\mathsf{shk}(\alpha,\beta)}, \; \{3, \lambda, t', \lambda_1\}_{\mathsf{shk}(\alpha,\beta)}.$$

et donc

$$\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \kappa_1, \lambda_1\}_{\mathsf{shk}(\alpha,\beta)}, \; \{2, \kappa, t', \kappa_1\}_{\mathsf{shk}(\alpha,\beta)}, \; \{3, \lambda, t', \lambda_1\}_{\mathsf{shk}(\alpha,\beta)}.$$

Si nous compilons tout ce que nous venons de voir nous obtenons que
- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{0, \kappa_1, \lambda_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,et
  $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$,
- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_j, \mathsf{f}_{i_j}, \nu_{j+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $j \in [\![p]\!]$, et
  $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_i, \mathsf{f}_{j_i}, \mu_{i+1}\}_{\mathsf{shk}(\alpha_0,\beta_0)}$ pour tout $i \in [\![q]\!]$,
- $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{2, \nu_{p+1}, t', \kappa_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$, et
  $\mathsf{K}(\mathsf{exec}_{n-1}) \cup \mathcal{N}_\epsilon(\Pi) \vdash \{3, \mu_{q+1}, t', \lambda_1\}_{\mathsf{shk}(\alpha_0,\beta_0)}$.

Nous pouvons donc conclure par hypothèse d'induction que le couple de mots $(\mathsf{f}_{i_1} \dots \mathsf{f}_{i_p} \mathsf{f}_\eta, \mathsf{f}_{j_1} \dots \mathsf{f}_{j_p} \mathsf{f}_\eta)$ avec $\mathsf{f}_\eta = t'$, est une présolution de $\mathcal{P}$. Et donc par définition (u, v) aussi est une présolution de $\mathcal{P}$. Ce qui achève notre induction et établit notre seconde implication. □

4.5 Une autre possibilité de codage

Etant donnée une instance du problème de correspondance de Post de taille h, le codage présenté à la section 4.3.1 construit un protocole à $2(h+3)$ rôles. Dans cette section, nous présentons un codage basé sur les mêmes intuitions, mais qui construit un protocole à 3 participants, quelle que soit l'instance du problème de correspondance de Post considérée. Si nous avons décidé de consacrer une section à la présentation de ce codage, c'est dans le but de ne laisser aucun doute quant à l'indécidabilité du problème de la vérification des protocoles de sécurité dans le cadre de messages de taille bornée. En effet, les « vrais » protocoles[4] impliquent tout au plus quatre ou cinq rôles. Il serait alors possible d'affirmer que la preuve d'indécidabilité proposée plus haut n'est plus valable pour la classe des « vrais » protocoles, *i.e.* la classe des protocoles dont le nombre de participants est borné.

Plutôt que de présenter ce codage en toute généralité, comme nous l'avons fait à la section 4.3.1 pour le précédent codage, nous nous contenterons de l'expliquer sur un exemple. Plus précisément, nous présenterons le protocole $\Pi'_{\mathcal{P}^{\mathsf{exmp}}}$ associé à l'instance $\mathcal{P}^{\mathsf{exmp}}$ du problème de correspondance de Post, et ce dans le modèle informel d'Alice et Bob. Il nous semble que les sections précédentes suffiront à convaincre le lecteur.

Le protocole $\Pi'_{\mathcal{P}^{\mathsf{exmp}}}$ associé à $\mathcal{P}^{\mathsf{exmp}}$ est constitué de trois sous-protocoles : le sous-protocole $\Pi^{\mathsf{init}}_{\mathcal{P}^{\mathsf{exmp}}}$ d'initialisation, le sous-protocole $\Pi^{\mathsf{concat}}_{\mathcal{P}^{\mathsf{exmp}}}$ de concaténation, et le sous-protocole de vérification $\Pi^{\mathsf{verif}}_{\mathcal{P}^{\mathsf{exmp}}}$.

$$\Pi'_{\mathcal{P}^{\mathsf{exmp}}} = \Pi^{\mathsf{init}}_{\mathcal{P}^{\mathsf{exmp}}}\ \Pi^{\mathsf{concat}}_{\mathcal{P}^{\mathsf{exmp}}}\ \Pi^{\mathsf{verif}}_{\mathcal{P}^{\mathsf{exmp}}}.$$

L'idée générale est de réunir dans le corps d'un seul et même rôle les actions des rôles r_{a_1}, r_{b_1}, r_{a_2} et r_{b_2} dédiés, dans le précédent codage, à la concaténation de chacun des couples de $\mathcal{P}^{\mathsf{exmp}}$. Nous verrons aussi qu'il est possible de se passer des rôles r_{a_3} et r_{b_3} qui simulaient le passage à la phase de vérification, dans le codage de la section 4.3.1.

[4] Rappelons que nous avons d'entrée exclu de notre cadre de travail les protocoles de groupes, qui eux impliquent un nombre arbitraire de participants

L'initialisation : le sous-protocole $\Pi^{\mathsf{init}}_{\mathcal{P}^{\mathsf{exmp}}}$.
$\Pi^{\mathsf{init}}_{\mathcal{P}^{\mathsf{exmp}}}$ est très similaire au sous-protocole $\Pi^{0}_{\mathcal{P}^{\mathsf{exmp}}}$ d'initialisation du précédent codage, présenté à la section 4.3.1. Il correspond à la séquence d'émission suivante :

$\underline{\Pi^{\mathsf{init}}_{\mathcal{P}^{\mathsf{exmp}}}}$

$$
\begin{aligned}
a \rightarrow b : &\left.\begin{array}{l}\{2, n_0, n, n_1\}_{\mathsf{shk}(a,b)}, \{3, m_0, n, m_1\}_{\mathsf{shk}(a,b)},\\ \{2, n_1, c, n_2\}_{\mathsf{shk}(a,b)},\\ \{3, m_1, c, m_2\}_{\mathsf{shk}(a,b)}, \{3, m_2, d, m_3\}_{\mathsf{shk}(a,b)}, \{3, m_3, d, m_4\}_{\mathsf{shk}(a,b)},\\ \{1, n_2, m_4\}_{\mathsf{shk}(a,b)}, \{0, n_2, m_4\}_{\mathsf{shk}(a,b)},\end{array}\right\} (\overline{c}, \overline{cdd})\\
&\left.\begin{array}{l}\{2, n_3, n, n_4\}_{\mathsf{shk}(a,b)}, \{3, m_5, n, m_6\}_{\mathsf{shk}(a,b)},\\ \{2, n_4, c, n_5\}_{\mathsf{shk}(a,b)}, \{2, n_5, d, n_6\}_{\mathsf{shk}(a,b)},\\ \{3, m_6, c, m_7\}_{\mathsf{shk}(a,b)}, \{3, m_7, d, m_8\}_{\mathsf{shk}(a,b)}, \{3, m_8, d, m_9\}_{\mathsf{shk}(a,b)},\\ \{1, n_6, m_9\}_{\mathsf{shk}(a,b)}, \{0, n_6, m_9\}_{\mathsf{shk}(a,b)},\end{array}\right\} (\overline{cd}, \overline{cdd})\\
&\{1, n_7, m_{10}\}_{\mathsf{shk}(a,b)}
\end{aligned}
$$

La seule différence avec $\Pi^{0}_{\mathcal{P}^{\mathsf{exmp}}}$ réside dans le fait que pour chacun des couples de $\mathcal{P}^{\mathsf{exmp}}$, a construit non seulement une représentation de type 'mot de $\mathcal{P}^{\mathsf{exmp}}$' du couple, mais aussi une représentation de type 'présolution de $\mathcal{P}^{\mathsf{exmp}}$'. Ainsi, nous n'avons plus besoin des rôles r_{a_3} et r_{b_3} afin de construire une représentation de type présolution de $\mathcal{P}^{\mathsf{exmp}}$ d'un couple de $\mathcal{P}^{\mathsf{exmp}}$.

Reprenons l'exemple sur lequel nous avons travaillé à la section 4.3.1. Pour construire une représentation de $(ccd, cddcdd)$ de type mot de $\mathcal{P}^{\mathsf{exmp}}$, nous commençons par construire une représentation de type mot de $\mathcal{P}^{\mathsf{exmp}}$ de (c, cdd). Pour ce faire, il nous faut ouvrir une session du rôle r_a incarnée par un agent α. Celui-ci construira alors pour nous la représentation que nous visons, en émettant un message m de la forme :

$$
\begin{array}{l}
\{2, \nu_0, \nu, \nu_1\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_0, \nu, \mu_1\}_{\mathsf{shk}(\alpha,\beta)},\\
\{2, \nu_1, c, \nu_2\}_{\mathsf{shk}(\alpha,\beta)},\\
\{3, \mu_1, c, \mu_2\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_2, d, \mu_3\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_3, d, \mu_4\}_{\mathsf{shk}(\alpha,\beta)},\\
\{1, \nu_2, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}, \{0, \nu_2, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}\\
\\
\{2, \nu_3, \nu, \nu_4\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_5, \nu, \mu_6\}_{\mathsf{shk}(\alpha,\beta)},\\
\{2, \nu_4, c, \nu_5\}_{\mathsf{shk}(\alpha,\beta)}, \{2, \nu_5, d, \nu_6\}_{\mathsf{shk}(\alpha,\beta)},\\
\{3, \mu_6, c, \mu_7\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_7, d, \mu_8\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_8, d, \mu_9\}_{\mathsf{shk}(\alpha,\beta)},\\
\{1, \nu_6, \mu_9\}_{\mathsf{shk}(\alpha,\beta)}, \{0, \nu_6, \mu_9\}_{\mathsf{shk}(\alpha,\beta)},\\
\\
\{1, \nu_7, \mu_{10}\}_{\mathsf{shk}(\alpha,\beta)}
\end{array}
$$

Notons qu'à l'issue de cette session nous avons aussi bien une représentation de (c, cdd) de type mot de $\mathcal{P}^{\mathsf{exmp}}$, que de type présolution de $\mathcal{P}^{\mathsf{exmp}}$. Ainsi le

passage à la phase de vérification ne nécessite plus de faire appel à un rôle tel que r_{b_3} du le précédent codage.

La concaténation : le sous-protocole $\Pi^{\mathsf{concat}}_{\mathcal{P}^{\mathsf{exmp}}}$.
$\Pi^{\mathsf{concat}}_{\mathcal{P}^{\mathsf{exmp}}}$ réunit dans le corps de b les actions des rôles r_{b_1} et r_{b_2} consacrés, à la section 4.3.1, à la concaténation des couples (c, cdd) et $(cd, cddd)$ respectivement. Il correspond à la séquence d'émissions suivante :

$\underline{\Pi^{\mathsf{concat}}_{\mathcal{P}^{\mathsf{exmp}}}}$

$$
\left.\begin{array}{l} b \rightarrow c : \{2, n_7, c, n_8\}_{\mathsf{shk}(a,b)}, \\ \{3, m_{10}, c, m_{11}\}_{\mathsf{shk}(a,b)}, \{3, m_{11}, d, m_{12}\}_{\mathsf{shk}(a,b)}, \{3, m_{12}, d, m_{13}\}_{\mathsf{shk}(a,b)}, \\ \{1, n_8, m_{13}\}_{\mathsf{shk}(a,b)}, \{0, n_8, m_{13}\}_{\mathsf{shk}(a,b)}, \end{array}\right\} (\overline{c}, \overline{cdd})
$$

$$
\left.\begin{array}{l} \{2, n_7, c, n_9\}_{\mathsf{shk}(a,b)}, \{2, n_9, d, n_{10}\}_{\mathsf{shk}(a,b)}, \\ \{3, m_{10}, c, m_{14}\}_{\mathsf{shk}(a,b)}, \{3, m_{14}, d, m_{15}\}_{\mathsf{shk}(a,b)}, \{3, m_{15}, d, m_{16}\}_{\mathsf{shk}(a,b)}, \\ \{1, n_{10}, m_{16}\}_{\mathsf{shk}(a,b)}, \{0, n_{10}, m_{16}\}_{\mathsf{shk}(a,b)}, \end{array}\right\} (\overline{cd}, \overline{cdd})
$$

Soit la représentation de type mot de $\mathcal{P}^{\mathsf{exmp}}$

$$
\begin{array}{l} \{2, \nu_1, c, \nu_2\}_{\mathsf{shk}(\alpha,\beta)}, \\ \{3, \mu_1, c, \mu_2\}_{\mathsf{shk}(a,b)}, \{3, \mu_2, d, \mu_3\}_{\mathsf{shk}(a,b)}, \{3, \mu_3, d, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}, \\ \{1, \nu_2, \mu_4\}_{\mathsf{shk}(\alpha,\beta)} \end{array}
$$

de (c, cdd), à notre disposition. Pour concaténer le couple (cd, cdd) à la suite du mot (c, cdd) de $\mathcal{P}^{\mathsf{exmp}}$, il faudra ouvrir une session du rôle r_b incarné par l'agent β et fournir à ce dernier le message $\{1, \nu_2, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}$. La différence avec le précédent codage, réside en ce que β ne se bornera pas à la construction d'une représentation du mot de $\mathcal{P}^{\mathsf{exmp}}$ qui nous intéresse. Il construira qui plus est une représentation de tout mot issu de la concaténation d'un couple de $\mathcal{P}^{\mathsf{exmp}}$ à la suite de (c, cdd). Ainsi, à l'issue de cette session, nous aurons à notre disposition, non-seulement une représentation de type mot de $\mathcal{P}^{\mathsf{exmp}}$ de $(ccd, cddcdd)$, mais aussi de $(cc, cddcdd)$. De plus, β émet les deux types de représentations pour chaque mot de $\mathcal{P}^{\mathsf{exmp}}$ issu de la concaténation d'un couple de $\mathcal{P}^{\mathsf{exmp}}$ à la suite de (c, cdd). A nous ensuite de récupérer la représentation du mot et du type qui nous intéresse, à savoir dans le cas du mot $(ccd, cddcdd)$:

$$
\begin{array}{l} \{2, \nu_2, c, \nu_9\}_{\mathsf{shk}(\alpha,\beta)}, \{2, \nu_9, d, \nu_{10}\}_{\mathsf{shk}(\alpha,\beta)}, \\ \{3, \mu_4, c, \mu_{14}\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_{14}, d, \mu_{15}\}_{\mathsf{shk}(\alpha,\beta)}, \{3, \mu_{15}, d, \mu_{16}\}_{\mathsf{shk}(\alpha,\beta)}, \\ \{1, \nu_{10}, \mu_{16}\}_{\mathsf{shk}(\alpha,\beta)} \end{array}
$$

L'agent β a aussi mis à notre disposition le message $\{0, \nu_{10}, \mu_{16}\}_{\mathsf{shk}(\alpha,\beta)}$, qui nous permettra comme nous allons le voir maintenant de procéder à la vérification du mot considéré.

Remarquons que b émet ses messages à destination de c et non de a, comme nous pourrions nous y attendre. Nous n'expliquons pas ici les raisons de ce choix. Nous y reviendrons dans un instant, après avoir présenté le sous-protocole de vérification.

La vérification

Le sous-protocole de vérification $\Pi^{\mathsf{verif}}_{\mathcal{P}^{\mathsf{exmp}}}$ prescrit exactement les mêmes actions que le précédent sous-protocole $\Pi^{h+2}_{\mathcal{P}^{\mathsf{exmp}}}$, excepté que nous n'avons pas consacré de rôles distincts à cette phase. En effet, La vérification est prise en charge par les mêmes rôles r_a et r_b.
$\underline{\Pi^{\mathsf{verif}}_{\mathcal{P}^{\mathsf{exmp}}}}$

$$
\begin{aligned}
b \rightarrow a :& \{0, n_{13}, m_{18}\}_{\mathsf{shk}(a,b)} \\
& \{2, n_{11}, n_{12}, n_{13}\}_{\mathsf{shk}(a,b)}, \\
& \{3, m_{17}, n_{12}, m_{18}\}_{\mathsf{shk}(a,b)}, \\
\\
a \rightarrow b :& \{0, n_{11}, m_{17}\}_{\mathsf{shk}(a,b)}, \\
& n_{12}
\end{aligned}
$$

Soit une nouvelle session du rôle r_a incarné par l'agent α, une fois que ce dernier aura émis les messages d'initialisation, il sera possible de lui demander de vérifier les deux dernières lettre de la représentation de $(ccd, cddcdd)$ construite, en lui fournissant les messages

$$
\begin{aligned}
& \{0, \nu_{10}, \mu_{16}\}_{\mathsf{shk}(\alpha,\beta)} \\
& \{2, \nu_9, d, \nu_{10}\}_{\mathsf{shk}(\alpha,\beta)} \\
& \{3, \mu_{15}, d, \mu_{10}\}_{\mathsf{shk}(\alpha,\beta)}
\end{aligned}
$$

à notre disposition. Il répondra alors par le message $\{0, \nu_9, \mu_{15}\}_{\mathsf{shk}(\alpha,\beta)}$ nécessaire à la poursuite de la vérification.

Nous pouvons à présent justifier l'introduction du rôle r_c dans le sous-protocole $\Pi^{\mathsf{concat}}_{\mathcal{P}^{\mathsf{exmp}}}$. En effet, tel que spécifié α s'attend à ce que β construise les représentations des couples de $\mathcal{P}^{\mathsf{exmp}}$ en partant du message $\{1, \nu_7, \mu_{10}\}_{\mathsf{shk}(\alpha,\beta)}$. Or, afin de concaténer $(cd, cddcdd)$ à la suite de $(c, cddcdd)$ nous avons demandé à β incarnant le rôle r_b de construire les représentations des couples de $\mathcal{P}^{\mathsf{exmp}}$ en partant du message $\{1, \nu_2, \mu_4\}_{\mathsf{shk}(\alpha,\beta)}$. Si β avait envoyé sa réponse à α et non à un agent γ incarnant le rôle r_c, α se serait retrouvé bloqué, et ne procéderait pas à la vérification.

4.6 Conclusion

Dans ce chapitre, nous avons établi que, même en nous restreignant à un modèle plus réaliste des protocoles de sécurité, à savoir qui n'inclut que des protocoles honnêtement exécutables, le problème de la vérification dans le cadre de messages de taille bornée reste indécidable. Nous sommes en effet parvenu à exhiber un codage du problème de correspondance de Post vers les protocoles de sécurité ne mettant en œuvre que de tels protocoles. Comme nous l'indiquions en ouvrant ce chapitre, la question de l'indécidabilité de la vérification dans le cadre de messages de taille bornée est une question qui revient de façon récurrente (*c.f.* [48, 3, 32, 31]).

Notons par ailleurs que [3], [32], et [31] se contentent respectivement de proposer un codage sans donner la preuve formelle de la correction de celui-ci. Dans [48] les auteurs esquissent bien quant à eux une preuve de la correction de leur propre codage. Néanmoins, comme en témoigne le petit exemple qui suit, celle-ci n'est pas correcte.

Dans [48] les auteurs proposent un codage des machines à deux compteurs vers les protocoles de sécurité. Nous supposons connu du lecteur ce qui relève des machines à deux compteurs.

Considérons la machine à deux compteurs $\mathcal{M} = (\mathcal{Q}, q_0, \delta, \mathcal{F})$ à deux états $\mathcal{Q} = \{q_0, q_1\}$, une unique transition $\delta = \{(q_0, 1, 1, q_1, 1, 1)\}$, et un seul état final $\mathcal{F} = \{q_1\}$. L'état final q_1 n'est clairement pas accessible puisque l'unique transition de $\mathcal{M}$ ne peut être enclenchée à partir de la configuration initiale $(q_0, 0, 0)$.

Le « protocole »[5] $\Pi_{\mathcal{M}}$ associé à $\mathcal{M}$ d'après le codage présenté dans [48] correspond aux « rôles » suivants :

$$r_a = [\quad \mathsf{snd}(a, b,\quad \{0,0\}_{\mathsf{shk}(a,b)}, \{n, q_0, 0, 0\}_{\mathsf{shk}(a;b)}, \{0,0\}_{\mathsf{shk}(a,b)});$$
$$\mathsf{Secret}(a, a, b, n) \qquad\qquad]$$

$$r_b = [\quad \mathsf{rcv}(b, a,\quad \{0,0\}_{\mathsf{shk}(a,b)}, \{x, q_0, y_1, y_2\}_{\mathsf{shk}(a;b)}, \{0,0\}_{\mathsf{shk}(a,b)});$$
$$\mathsf{snd}(b, a,\quad \{y_1, n_1\}_{\mathsf{shk}(a,b)}, \{x, q_1, n_1, n_2\}_{\mathsf{shk}(a;b)}, \{y_2, n_2\}_{\mathsf{shk}(a,b)}),$$
$$x \qquad\qquad]$$

Les auteurs reposent alors leur « preuve » d'indécidabilité sur l'affirmation avancée selon laquelle $\Pi_{\mathcal{M}}$ admettrait une attaque si et seulement si q_1 était accessible. Nous avons déjà dit que q_1 n'est pas accessible dans $\mathcal{M}$. Il est néanmoins évident que $\Pi_{\mathcal{M}}$ révèle son secret.

En effet, imaginons qu'un agent honnête α ouvre une session du rôle r_a avec pour interlocuteur l'agent honnête β. L'intrus obtient alors un message m de la forme

$$m \quad = \quad \{0,0\}_{\mathsf{shk}(\alpha,\beta)}, \{\nu, q_0, 0, 0\}_{\mathsf{shk}(\alpha,\beta)}, \{0,0\}_{\mathsf{shk}(\alpha,\beta)}$$

où ν est une instance honnête du secret n. En ouvrant alors une session du rôle r_b incarné par l'agent β avec pour interlocuteur α, et fournissant à β le message m, l'intrus récupère un message de la forme

$$m \quad = \quad \{0, \nu_1\}_{\mathsf{shk}(\alpha,\beta)}, \{\nu, q_0, \nu_1, \nu_2\}_{\mathsf{shk}(\alpha,\beta)}, \{0, \nu_2\}_{\mathsf{shk}(\alpha,\beta)},$$
$$\nu$$

et par là même le secret ν. Leur codage n'est donc pas correct.

En exhibant un codage « réaliste » et en fournissant une preuve formelle et détaillée de la correction de ce dernier, nous espérons avoir levé définitivement le doute quant à l'indécidabilité du problème de la vérification des protocoles de sécurité dans le cadre de messages de taille bornée.

[5] $\Pi_{\mathcal{M}}$ n'est clairement pas un protocole au sens de la définition 2.4.2, mais nous ne réouvrons pas ici la discussion quant à ces considérations.

Deuxième partie

Résultats de réduction

Chapitre 5

Vérification par résolution de systèmes de contraintes

Les preuves des résultats présentés en deuxième partie de cette thèse sont étroitement liées à une procédure particulière de vérification des protocoles dans le cadre d'un nombre borné de sessions. Il s'agît de la procédure proposée par R. Corin et *al.* dans [20, 21]. Nous consacrons ce chapitre à la présentation de cette procédure. Celle-ci repose à son tour largement sur la résolution de contraintes symboliques. En effet, les systèmes de contraintes symboliques sont très répandus en matière de modélisation et de vérification des protocoles de sécurité, dans le cadre d'un nombre borné de sessions (voir [18, 43, 50, 27, 24]). Il nous faudra donc dédier une partie de notre exposé aux définitions relatives aux systèmes de contraintes (section 5.2) ainsi qu'à leur résolution (section 5.3). Nous présenterons plus précisément la procédure de résolution de contraintes proposées par H. Comon-Lundh [18] et reprise par d'autres [27, 24, 28] par la suite. Nous en viendrons alors (sections 5.4 et 5.5) à l'algorithme de vérification de [20, 21].

Nous profitons de ce chapitre ayant pour but de poser les bases dont nous aurons besoin par la suite, pour présenter l'unification (section 5.1) et établir un résultat de conservativité auquel nous aurons souvent recours.

5.1 L'unification

L'unification du premier ordre de deux termes est un problème dont la solution est bien connue. Si nous la rappelons ici c'est dans le seul but d'établir le résultat de conservativité énoncé au lemme 5.1.4. Nous serons donc brefs. Nous nous contenterons d'ailleurs de la présentée sur la signature $\mathcal{F}$ des primitives cryptographiques. Nous rappelons rapidement la définition de *problème d'unification* ainsi que l'algorithme d'unification proposé par A. Martelli et U. Montanari dans [41].

Définition 5.1.1 (Problème d'unification)**.** *Un problème d'unification P est*

un ensemble d'expressions de la forme $t \stackrel{?}{=} u$ *où* t *et* u *sont des termes. Un problème d'unification* $P = \{t_i \stackrel{?}{=} u_i\}_{i \in [\![h]\!]}$ *est dit en* en forme résolue *si les deux conditions suivantes sont satisfaites :*

1. *pour tout* $i \in [\![h]\!]$, $t_i \in [\![h]\!]$,
2. *pour tout* $i \in [\![h]\!]$ *et tout* $j \in [\![h]\!]$, *si* $j \neq i$ *alors* $t_i \notin \mathcal{V}(\{t_j, u_j\})$ *et* $t_i \notin \mathcal{V}(u_i)$.

L'algorithme d'unification proposé dans [41] repose sur l'ensemble de règles décrites à la figure 5.1.

$$(a): \quad P \cup \{t \stackrel{?}{=} u\} \rightsquigarrow_{\mathrm{id}} P \cup \{u \stackrel{?}{=} t\} \quad \text{si } u \in \mathcal{V} \text{ et } t \notin \mathcal{V}$$

$$(b): \quad P \cup \{t \stackrel{?}{=} t\} \rightsquigarrow_{\mathrm{id}} P \quad \text{si } t \in (\mathcal{V} \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{N})$$

$$(c): \quad P \cup \{f(t_1, \ldots, t_n) \stackrel{?}{=} f(u_1, \ldots, u_n)\} \rightsquigarrow_{\mathrm{id}} P \cup \{t_1 \stackrel{?}{=} u_1, \ldots, t_n \stackrel{?}{=} u_n\} \quad \text{pour } f \in \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$$

$$(d): \quad P \cup \{t \stackrel{?}{=} u\} \rightsquigarrow_{\{t \mapsto u\}} P \cup \{t \stackrel{?}{=} u\} \quad \text{si } t \in \mathcal{V},\ t \neq u,\ t \notin \mathcal{V}(u) \text{ et } t \text{ apparait dans } P$$

Fig. 5.1: Règles de simplification d'un problème d'unification

Pour tout $n \geq 1$ nous notons $P_0 \rightsquigarrow_\sigma^n P_n$ pour la dérivation $P_0 \rightsquigarrow_{\sigma_1} P_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} P_n$ telle que $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$. Nous introduisons aussi la notation $P \rightsquigarrow_\sigma^* P'$ pour dénoter que $P \rightsquigarrow_\sigma^n P'$ pour un certain $n \geq 1$, ou que $P = P'$. Un problème d'unification P est dit *admettre une solution* s'il existe un problème d'unification P' en forme résolue tel que $P \rightsquigarrow_\sigma^* P'$.

Théorème 5.1.2 ([41])**.** *Soient* t *et* u *deux termes,* t *et* u *sont unifiables si et seulement si le problème d'unification* $\{t \stackrel{?}{=} u\}$ *admet une solution* P*. De plus, si* t *et* u *sont unifiables, et* P *est une solution du problème d'unification* $\{t \stackrel{?}{=} u\}$*, alors* $\{t \stackrel{?}{=} u\} \rightsquigarrow_{\mathrm{mgu}(t,u)}^* P$.

Soient t et u deux termes, nous noterons $\mathrm{mgu}(t, u) = \bot$ si t et u sont non-unifiables.

Nous avons à présent établi le peu de notions et de résultats relatifs à l'unification dont nous aurons besoin. Nous nous attelons donc la preuve de la propriété de conservativité énoncée au lemme 5.1.4. Ceci requiert néanmoins de caractériser l'ensemble des sous-termes chiffrés d'un terme auquel une substitution a été appliquée, en distinguant les applications de primitives de chiffrement issues de la substitution, de celles déjà présentes dans le terme avant application de la substitution.

Lemme 5.1.3. *Soient deux termes* $u, v \in \mathcal{T}$*, et une substitution* σ*. Si* $u \in \mathsf{Est}(v\sigma)$*, alors*

$$u \in \mathsf{Est}(v)\sigma \ \vee \ (\exists x \in \mathcal{V}(v).\ u \in \mathsf{Est}(\sigma(x))).$$

Démonstration. Soient deux termes u et v ainsi qu'une substitution σ tels que $u \in \mathsf{Est}(v\sigma)$. Nous montrons que

$$u \in \mathsf{Est}(v)\sigma \ \vee \ (\exists x \in \mathcal{V}(v).\ u \in \mathsf{Est}(\sigma(x))).$$

par induction sur la taille du terme v.

Cas $v \in \mathcal{P} \cup \mathcal{C} \cup \mathcal{N}$. Alors $v\sigma = v$ et $\mathsf{Est}(v\sigma) = \emptyset$. Ce cas ne peut donc pas survenir car nous avons supposé que $\mathsf{Est}(v\sigma)$ contient au moins u.

Cas $v \in \mathcal{V}$. Alors la deuxième branche de la disjonction est trivialement vraie. En effet, comme $u \in \mathsf{Est}(v\sigma) = \mathsf{Est}(\sigma(v))$, alors pour existe $x = v \in \mathcal{V}(v)$ $u \in \mathsf{Est}(\sigma(x))$.

Cas $v = f(v_1, \ldots, v_n)$ *pour certains* $v_1, \ \ldots, \ v_n$ *avec* $f \in \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle\}$. Comme par définition $\mathsf{Est}(v\sigma) = \bigcup\limits_{i \in [\![n]\!]} \mathsf{Est}(v_i\sigma)$, il existe nécessairement $i \in [\![n]\!]$ tel que $u \in \mathsf{Est}(v_i\sigma)$. Or, nous savons par induction que

$$(u \in \mathsf{Est}(v_i)\sigma \ \vee \ (\exists x \in \mathcal{V}(v_i).\ u \in \mathsf{Est}(\sigma(x)))).$$

Supposons que $u \in \mathsf{Est}(v_i)\sigma$, alors comme par définition $\mathsf{Est}(v_i) \subseteq \mathsf{Est}(v)$, nous concluons que $u \in \mathsf{Est}(v)\sigma$. Supposons maintenant que $\exists x \in \mathcal{V}(v_i),\ u \in \mathsf{Est}(\sigma(x))$, puisque $\mathcal{V}(v_i) \subseteq \mathcal{V}(v)$ nous concluons que $\exists x \in \mathcal{V}(v),\ u \in \mathsf{Est}(\sigma(x))$.

Cas $v = f(v_1, \ldots, v_n)$ *pour certains* $v_1, \ \ldots, \ v_n$, *avec* $f \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$. Comme par définition $\mathsf{Est}(v\sigma) = \{v\sigma\} \cup \bigcup_{i \in [\![n]\!]} \mathsf{Est}(v_i\sigma)$, il nous faut distinguer deux cas :

1. $u = v\sigma$. Par définition $v \in \mathsf{Est}(v)$ et donc $u = v\sigma \in \mathsf{Est}(v)\sigma$.
2. $u \neq v\sigma$, implique qu'il existe $i \in [\![n]\!]$ tel que $u \in \mathsf{Est}(v_i\sigma)$. Nous savons alors par induction

   $$(u \in \mathsf{Est}(v_i)\sigma \ \vee \ (\exists x \in \mathcal{V}(v_i).\ u \in \mathsf{Est}(\sigma(x)))).$$

   Supposons que $u \in \mathsf{Est}(v_i)\sigma$, alors comme par définition $\mathsf{Est}(v_i) \subseteq \mathsf{Est}(v)$, nous concluons que $u \in \mathsf{Est}(v)\sigma$. Supposons maintenant que $\exists x \in \mathcal{V}(v_i),\ u \in \mathsf{Est}(\sigma(x))$, puisque $\mathcal{V}(v_i) \subseteq \mathcal{V}(v)$ nous concluons que $\exists x \in \mathcal{V}(v),\ u \in \mathsf{Est}(\sigma(x))$.

□

Le lemme suivant établit que la substitution résultant de l'unification de deux termes est conservative.

Lemme 5.1.4. *Soient deux termes* $u, v \in \mathcal{T}$. *Soit* $\sigma = \mathrm{mgu}(u, v)$. *Si* $\sigma \neq \bot$, *alors*

$$\forall x \in \mathrm{dom}(\sigma).\ \forall t \in \mathsf{Est}(\sigma(x)).\ \exists w \in \mathsf{Est}(\{u, v\}).\ t = w\sigma.$$

Démonstration. Soient u et v deux termes tels que $\mathrm{mgu}(u, v) \neq \bot$. La substitution $\sigma = \mathrm{mgu}(u, v)$ peut être calculée à l'aide de l'algorithme d'unification présenté ci-dessus, l'ensemble initial d'équations étant $E_0 = \{u \stackrel{?}{=} v\}$. Soit $E_0 \rightsquigarrow_{\sigma_1} E_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_k} E_k$ une séquence de simplifications aboutissant à la substitution $\mathrm{mgu}(u, v)$, *i.e.* $\mathrm{mgu}(u, v) = \sigma_1\sigma_2 \ldots \sigma_k$.

Nous étendons la fonction $\mathsf{Est}()$ aux ensembles d'équations comme suit :

$$\mathsf{Est}(E) = \bigcup_{u' \stackrel{?}{=} v' \in E} (\mathsf{Est}(u') \cup \mathsf{Est}(v')).$$

Nous allons montrer par induction sur i que pour tout terme $t \in \mathsf{Est}(E_i)$, il existe un terme $w \in \mathsf{Est}(\{u, v\})$ tel que $t = w\sigma_1 \dots \sigma_i$.

Cas de base : $i = 0$. Par définition, $\mathsf{Est}(E_0) = \mathsf{Est}(u, v)$ et donc l'implication est trivialement vraie.

Cas inductif : $i \geq 1$. Nous procédons par analyse de cas sur la règle R impliquée dans la simplification $E_{i-1} \overset{\mathsf{R}}{\rightsquigarrow}_{\sigma_i} E_i$ et distinguons deux cas.

Cas $\mathsf{R} = (a)$, $\mathsf{R} = (b)$, *ou* $\mathsf{R} = (c)$. Dans tous ces cas $\mathsf{Est}(E_i) \subseteq \mathsf{Est}(E_{i-1})$ et $\sigma_i = \mathrm{id}$, or nous savons par induction que pour tout $t \in \mathsf{Est}(E_{i-1})$, il existe $w \in \mathsf{Est}(\{u, v\})$ tel que $t = w\sigma_1 \dots \sigma_{i-1} = w\sigma_1 \dots \sigma_{i-1}\sigma_i$. Ce qui nous permet de conclure.

Cas $\mathsf{R} = (d)$. Il existe alors nécessairement un ensemble d'équations E et une équation $x \stackrel{?}{=} t$ avec $x \in \mathcal{V}$, $t \neq x$, $x \notin \mathcal{V}(t)$, et x apparaissant dans E, tels que $E_{i-1} = E \cup \{x \stackrel{?}{=} t\}$, et $E_i = E\sigma_i \cup \{x \stackrel{?}{=} t\}$ où $\sigma_i = \{x \mapsto t\}$. Soit $t' \in \mathsf{Est}(E_i)$, alors ou bien $t' \in \mathsf{Est}(E\sigma_i)$, ou $t \in \mathsf{Est}(\{x \stackrel{?}{=} t\})$. Dans le second cas, comme E_{i-1} satisfait l'hypothèse d'induction, et que $x \stackrel{?}{=} t \in E_{i-1}$, nous savons qu'il existe $w \in \mathsf{Est}(\{u, v\})$ tel que $t' = w\sigma_1 \dots \sigma_{i-1}$. De plus, comme $x \notin \mathcal{V}(t)$ nous concluons que $w\sigma_1 \dots \sigma_{i-1}\sigma_i = w\sigma_1 \dots \sigma_{i-1}$ et que donc $t' = w\sigma_1 \dots \sigma_{i-1}\sigma_i$. Plaçons nous à présent dans le premier cas (*i.e.* $t' \in \mathsf{Est}(E\sigma_i)$). Il existe alors $t'' \in E$ tel que $t' \in \mathsf{Est}(t''\sigma_i)$. D'après le lemme 5.1.3 nous savons alors qu'ou bien $t' \in \mathsf{Est}(t'')\sigma_i$, ou $\exists y \in \mathcal{V}(t'').\ t' \in \mathsf{Est}(\sigma_i(y))$.

1. Si $t' \in \mathsf{Est}(t'')\sigma_i$, alors $t' \in \mathsf{Est}(E)\sigma_i \subseteq \mathsf{Est}(E_{i-1})\sigma_i$, il existe donc $t''' \in \mathsf{Est}(E_{i-1})$ tel que $t' = t'''\sigma_i$. Par hypothèse d'induction nous déduisons qu'il existe $w \in \mathsf{Est}(\{u, v\})$ tel que $t''' = w\sigma_1 \dots \sigma_{i-1}$. Nous concluons donc que $t' = t'''\sigma_i = w\sigma_1 \dots \sigma_{i-1}\sigma_i$ pour un certain $w \in \mathsf{Est}(\{u, v\})$.

2. Si $t' \in \mathsf{Est}(\sigma_i(y))$ pour une certaine variable $y \in \mathcal{V}(t'')$, et comme $\sigma_i = \{x \mapsto t\}$, nécessairement $y = x$ et $t' \in \mathsf{Est}(t)$. Or $x \stackrel{?}{=} t \in E_{i-1}$ implique que $t' \in \mathsf{Est}(E_{i-1})$. Par induction nous pouvons donc dire qu'il existe $w \in \mathsf{Est}(\{u, v\})$ tel que $t' = w\sigma_1 \dots \sigma_{i-1}$. De plus, nous savons que $x \notin \mathcal{V}(t)$ et donc que $t\sigma_i = t$. Par là même, sachant que $t' \in \mathsf{Est}(t)$ nous savons que $t'\sigma_i = w\sigma_1 \dots \sigma_{i-1}\sigma_i$. Ce qui nous permet de conclure, et achève notre induction.

A présent, considérons $x \in \mathrm{dom}(\sigma)$. D'après l'algorithme d'unification, cela implique que $x \stackrel{?}{=} \sigma(x) \in E_k$. Soit $t \in \mathsf{Est}(\sigma(x))$, alors $t \in \mathsf{Est}(E_k)$. D'après ce que nous venons de montrer, il existe $w \in \mathsf{Est}(\{u, v\})$ tel que $t = w\sigma_1 \dots \sigma_k = w\sigma$. □

5.2 Systèmes de contraintes symboliques

Définition 5.2.1 (Contraintes symboliques)**.** *Une contrainte est une expression de la forme* $T \Vdash u$*, avec* T *un ensemble fini de termes et* u *un terme.* T *est appelé le* membre gauche *de la contrainte, et* u *le* membre droit*. Un* système de contraintes *est soit* $\bot$*, soit un ensemble fini* $C = \{T_i \Vdash u_i\}_{i \in [\![n]\!]}$ *de contraintes tel que :*

- $\forall i \in [\![n-1]\!].\ T_i \subseteq T_{i+1}$*, et*
- $\forall i \in [\![n]\!].\ \mathcal{V}(T_i) \subseteq \{x \mid x \in \mathcal{V}(T_j),\ T_j \subsetneq T_i\}$.

Une substitution close σ *est une solution de* C *si et seulement si* $T_i\sigma \vdash u_i\sigma$ *pour tout* $i \in [\![n]\!]$.

La première condition stipule que les membres gauches (les T_i) sont totalement ordonnés. La seconde condition assure que toute variable apparaît d'abord dans un membre droit (relativement à l'ordre d'inclusion des membres gauches). C sera souvent dénoté comme la conjonction de ses contraintes, *i.e.*

$$C = \bigwedge_{i \in [\![n]\!]} T_i \Vdash u_i.$$

Le membre gauche de C, dénoté $\mathrm{lhs}(C)$, est le membre gauche maximal (pour l'inclusion) des contraintes de C, soit $\mathrm{lhs} = \max_{i \in [\![n]\!]}(T_i)$. Le membre droit de C, dénoté $\mathrm{rhs}(C)$, est l'union des membres droits de ses contraintes, soit $\mathrm{rhs} = \bigcup_{i \in [\![n]\!]} u_i$. L'ensemble des variables apparaissant dans C est dénoté $\mathcal{V}(C)$ et défini de manière usuelle par :

$$\mathcal{V}(C) = \bigcup_{i \in [\![n]\!]} (\mathcal{V}(T_i) \cup \mathcal{V}(u_i)).$$

Nous avons dit en introduction de ce chapitre que les systèmes de contraintes sont souvent utilisés afin de définir le modèle d'exécution des protocoles. Nous allons voir comment, à partir d'un scénario sc et de la connaissance initiale de l'intrus T_0, construire un système de contraintes symboliques C, tel que l'ensemble des solutions de C corresponde à l'ensemble des exécutions valides pour sc et T_0.

Définition 5.2.2 ($\mathcal{C}(tr, T_0)$)**.** *Soit* Π *un protocole,* T_0 *un ensemble de termes clos,* sc *un scénario de* Π*, et* $tr = [e_1; \ldots; e_\ell]$ *la trace symbolique associée à* sc*. Le* système de contraintes symboliques associé à tr et T_0*, dénoté* $\mathcal{C}(tr, T_0)$*, est défini par :*

$$\mathcal{C}(tr, T_0) = \{T_0 \cup \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(tr_i) \Vdash u \mid e_i = \mathsf{rcv}(p_1, p_2, u),\ i \in [\![\ell]\!],\ p_1, p_2 \in \mathcal{P}\}$$

Les membres gauches du système de contraintes ainsi construit représentent les messages envoyés sur le réseau, alors que les membres droits correspondent aux réceptions de la trace tr et donc aux messages que l'intrus devra construire. Il est facile de voir que le système de contraintes ainsi obtenu vérifie bien les deux conditions de la définition 5.2.1. En effet, comme $\mathsf{K}(tr_i) \subseteq \mathsf{K}(tr_{i+1})$ pour tout $i \in [\![\ell - 1]\!]$, les membres gauches de C sont totalement ordonnés. Aussi,

les conditions 2 et 3b sur la définition d'un protocole (voir définition 2.4.2) combinées à la définition de trace symbolique associée à un scénario (voir définition 2.6.5) assurent que toute variable apparaissant dans le membre gauche d'une des contraintes de C apparaît préalablement (relativement à l'ordre d'inclusion des membres gauches des contraintes de C) dans le membre droit d'une des contraintes de C, comme le veut la deuxième condition de la définition de système de contraintes.

Exemple 5.2.3. *Soit T_0 un ensemble de termes clos. Considérons le scénario $\mathsf{sc}_{\mathsf{Toy}}$ de l'exemple 2.6.2. La trace symbolique tr_{Toy} associée à $\mathsf{sc}_{\mathsf{Toy}}$ est celle décrite à l'exemple 2.6.6, et selon la définition 5.2.1 ci-dessus le système de contraintes $C = \mathcal{C}(tr_{\mathsf{Toy}}, T_0)$ associé à tr_{Toy} et T_0 est :*

$$C = \begin{cases} T_1 \stackrel{def}{=} T_0 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)\} & \Vdash \quad \mathsf{aenc}(\langle \mathsf{aenc}(x^2, b), \epsilon\rangle, b) \\ T_2 \stackrel{def}{=} T_1 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(x^2, \epsilon), b\rangle, \epsilon)\} & \Vdash \quad \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b) \end{cases}$$

La substitution $\sigma = \{x^2 \mapsto \langle \mathsf{aenc}(n^1, b), a\rangle; x^3 \mapsto n^1\}$ définie à l'exemple 2.6.9 est une solution de C.

Le lemme suivant stipule que les solutions du système de contraintes associé à un scénario d'un protocole et une connaissance initiale de l'intrus correspondent aux exécutions valides possibles pour ce scénario et cette connaissance initiale.

Lemme 5.2.4. *Soit Π un protocole, T_0 un ensemble de termes clos, sc un scénario de Π, et tr la trace symbolique associée à sc. Soit σ une substitution close telle que $\mathrm{dom}(\sigma) \subseteq \mathcal{V}(tr)$*

> *$tr\sigma$ est une exécution valide de Π pour T_0* ssi *σ est une solution de $\mathcal{C}(tr, T_0)$.*

Ce lemme ne nécessite clairement pas d'être prouvé. En effet, la définition d'une solution d'un système de contraintes associé à une trace symbolique tr d'un protocole Π et une connaissance un connaissance initiale de l'intrus T_0, calque parfaitement avec celle d'exécution de trace symbolique sous-jacente tr valide, au regard de la connaissance initiale de l'intrus T_0.

5.3 Simplification de systèmes de contraintes

Il s'agit à présent de voir que résoudre un système de contraintes peut se ramener à la résolution d'un système de contraintes plus « simple », dit *en forme résolue*. De tels systèmes de contraintes seront utilisés par la suite pour décider les propriétés de sécurité introduites au chapitre 3 précédent.

Définition 5.3.1 (Système de contraintes en forme résolue)**.** *Nous dirons qu'un système de contraintes C est en forme résolue si $C = \bot$, ou si $C = \{T_i \Vdash u_i\}_{i \in [\![n]\!]}$ avec $u_i \in \mathcal{V}$ pour tout $i \in [\![n]\!]$.*

A noter que le système de contraintes vide est en forme résolue. De tels systèmes de contraintes sont particulièrement simples puisqu'ils admettent toujours une solution. En effet, la substitution $\sigma = \{u_i \mapsto \epsilon\}_{i \in [\![n]\!]}$ est une solution de C.

La procédure de simplification d'un système de contraintes en un système de contraintes en forme résolue repose sur l'ensemble de règles de simplification de la Figure 5.2. Toutes ces règles sont en fait indicées par une substitution : quand celle-ci n'apparaît pas, il s'agît alors implicitement de la substitution identité. Pour tout $n \geq 1$ nous notons $C_0 \rightsquigarrow_\sigma^n C_n$ pour la dérivation $C_0 \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} C_n$ avec $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$. Nous introduisons aussi la notation $C \rightsquigarrow_\sigma^* D$ pour dénoter que $C \rightsquigarrow_\sigma^n D$ pour un certain $n \geq 1$, soit que $D = C$ et que σ est la substitution vide.

$$\mathsf{R}_1 : \quad C \wedge T \Vdash u \quad \rightsquigarrow \quad C \qquad \text{si } T \cup \{x \in \mathcal{V} \mid T' \Vdash u' \in C, T' \subsetneq T\} \vdash u$$

$$\mathsf{R}_2 : \quad C \wedge T \Vdash u \quad \rightsquigarrow_\sigma \quad C\sigma \wedge T\sigma \Vdash u\sigma$$
si $\sigma = \mathrm{mgu}(t, u)$ avec $t \in \mathsf{St}(T)$, $t \neq u$, t, u ni des variables ni des paires

$$\mathsf{R}_3 : \quad C \wedge T \Vdash u \quad \rightsquigarrow_\sigma \quad C\sigma \wedge T\sigma \Vdash u\sigma$$
si $\sigma = \mathrm{mgu}(t_1, t_2)$, $t_1, t_2 \in \mathsf{St}(T)$, $t_1 \neq t_2$, t_1, t_2 sont ni des variables ni des paires

$$\mathsf{R}_4 : \quad C \wedge T \Vdash u \quad \rightsquigarrow_\sigma \quad C\sigma \wedge T\sigma \Vdash u\sigma$$
si $\sigma = \mathrm{mgu}(t_2, t_3)$, $\mathsf{aenc}(t_1, t_2) \in \mathsf{St}(T)$, $\mathsf{pvk}(t_3) \in (\mathsf{Plaintext}(T) \cup \{\mathsf{pvk}(\epsilon)\})$, $t_2 \neq t_3$

$$\mathsf{R}_5 : \quad C \wedge T \Vdash u \quad \rightsquigarrow \quad \bot \qquad \text{si } \mathcal{V}(T \cup \{u\}) = \emptyset \text{ et } T \not\vdash u$$

$$\mathsf{R}_f : \quad C \wedge T \Vdash f(u_1, \ldots, u_n) \rightsquigarrow C \wedge \bigwedge_{i \in [\![n]\!]} T \Vdash u_i \quad \text{pour } f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$$

FIG. 5.2: Règles de simplification

Exemple 5.3.2. *Sois le système de contraintes*

$$C = \begin{cases} T_1 \stackrel{def}{=} T_0 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)\} & \Vdash & \mathsf{aenc}(\langle \mathsf{aenc}(x^2, b), \epsilon\rangle, b) \\ T_2 \stackrel{def}{=} T_1 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(x^2, \epsilon), b\rangle, \epsilon)\} & \Vdash & \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b) \\ T_3 \stackrel{def}{=} T_2 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(x^3, \epsilon), b\rangle, \epsilon)\} & \Vdash & n^1 \end{cases}$$

La dérivation suivante réduit C en un système de contraintes en forme résolue :

$$C \overset{\mathsf{R}_{\mathsf{aenc}}}{\rightsquigarrow} \begin{cases} T_1 \Vdash \langle \mathsf{aenc}(x^2, b), \epsilon\rangle \\ T_1 \Vdash b \\ T_2 \Vdash \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b) \\ T_3 \Vdash n^1 \end{cases} \overset{\mathsf{R}_1}{\rightsquigarrow} \begin{cases} T_1 \Vdash \langle \mathsf{aenc}(x^2, b), \epsilon\rangle \\ T_2 \Vdash \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b) \\ T_3 \Vdash n^1 \end{cases}$$

$$\overset{\mathsf{R}_{\langle\rangle}}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathsf{aenc}(x^2, b) \\ T_1 \Vdash \epsilon \\ T_2 \Vdash \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b) \\ T_3 \Vdash n^1 \end{cases} \overset{\mathsf{R}_1}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathsf{aenc}(x^2, b) \\ T_2 \Vdash \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b) \\ T_3 \Vdash n^1 \end{cases}$$

$$\overset{\mathsf{R}_{\mathsf{aenc}}}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathsf{aenc}(x^2, b) \\ T_2 \Vdash \langle \mathsf{aenc}(x^3, b), \epsilon\rangle \\ T_2 \Vdash b \\ T_3 \Vdash n^1 \end{cases} \overset{\mathsf{R}_1}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathsf{aenc}(x^2, b) \\ T_2 \Vdash \langle \mathsf{aenc}(x^3, b), \epsilon\rangle \\ T_3 \Vdash n^1 \end{cases} \overset{\mathsf{R}_{\langle\rangle}}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathsf{aenc}(x^2, b) \\ T_2 \Vdash \mathsf{aenc}(x^3, b) \\ T_2 \Vdash \epsilon \\ T_3 \Vdash n^1 \end{cases}$$

$$\overset{\mathsf{R}_1}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathsf{aenc}(x^2, b) \\ T_2 \Vdash \mathsf{aenc}(x^3, b) \\ T_3 \Vdash n^1 \end{cases} \overset{\mathsf{R}_2}{\rightsquigarrow}_{\sigma_1} \begin{cases} T_1 \Vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b) \\ T_2' \Vdash \mathsf{aenc}(x^3, b) \\ T_3' \Vdash n^1 \end{cases}$$

avec $\sigma_1 = \mathrm{mgu}(\mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b), \mathsf{aenc}(x^2, b)) = \{x^2 \mapsto \langle \mathsf{aenc}(n^1, b), a\rangle\}$, $T_2' = T_1 \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, \epsilon), b\rangle, \epsilon)\}$, *et* $T_3' = T_2' \cup \{\mathsf{aenc}(\langle \mathsf{aenc}(x^3, \epsilon), b\rangle, \epsilon)\}$.

$$\overset{\mathsf{R}_1}{\rightsquigarrow} \begin{cases} T_2' \Vdash \mathsf{aenc}(x^3, b) \\ T_3' \Vdash n^1 \end{cases} \overset{\mathsf{R}_2}{\rightsquigarrow}_{\sigma_2} \begin{cases} T_2' \Vdash \mathsf{aenc}(n^1, b) \\ T_3'' \Vdash n^1 \end{cases} \overset{\mathsf{R}_1}{\rightsquigarrow} \begin{cases} T_3'' \Vdash n^1 \end{cases} \overset{\mathsf{R}_1}{\rightsquigarrow} \emptyset$$

avec $\sigma_2 = \mathrm{mgu}(\mathsf{aenc}(n^1, b), \mathsf{aenc}(x^3, b)) = \{x^3 \mapsto n^1\}$.

L'ensemble de règles de simplification de la Figure 5.2 est correct, complet et termine : un système de contraintes C admet une solution θ si et seulement si il existe un système de contraintes en forme résolue D et deux substitutions σ et θ' tels que $C \rightsquigarrow_\sigma^* D$, θ' est une solution de D, et $\theta = \sigma\theta'$.

Théorème 5.3.3. *Soit C un système de contraintes non en forme résolue :*

1. *(Terminaison) Il n'existe pas de dérivation infinie à partir de C.*
2. *(Correction) Si il existe un système de contraintes D et une substitution σ tels que $C \rightsquigarrow_\sigma^* D$ et si θ' est une solution de D, alors $\theta = \sigma\theta'$ est une solution de C.*
3. *(Complétude) Si θ est une solution de C, alors il existe un système de contrainte en forme résolue D ainsi que deux substitutions σ et θ' tels que, θ' est une solution de D, $\theta = \sigma\theta'$ et $C \rightsquigarrow_\sigma^* D$.*

La paternité de cette procédure revient à H. Comon-Lundh [18]. Il existe un certain nombre de variantes de cette procédure [18, 27, 24, 28] et pour chacune le théorème 5.3.3 doit être re-prouvé. Notre travail s'étant basé sur la version de cette procédure proposée par S. Delaune et *al.* dans [24, 28], pour la preuve de ce théorème, nous renvoyons le lecteur à [24, 28].

5.4 Simplification de formules de $\mathcal{PS}$-LTL$^+$

L'approche proposée par R. Corin [20, 21] se décompose en deux étapes. Soit ϕ la formule de $\mathcal{PS}$-LTL$^-$ représentant la propriété de sécurité considérée. La première étape consiste à traduire $\neg\phi$, *i.e.* la formule d'attaque, en une formule *élémentaire* π équivalente. Cette traduction se fait à l'aide de la fonction de simplification **T** que nous présentons dans cette section.

Définition 5.4.1 (Formules élémentaires, EF)**.** *L'ensemble des formules élémentaires est défini par la grammaire suivante :*

$$\pi \quad ::= \quad \mathsf{true} \quad | \quad t_1 = t_2 \quad | \quad T \Vdash t \quad | \quad \neg\pi \quad | \quad \pi \vee \pi \quad | \quad \exists x.\pi$$

Une formule élémentaire est dite *positive* si toute occurrence de la forme $T \Vdash t$ apparaît dans le champ d'un nombre pair de négations, et *négative* si elle apparaît dans le champ d'un nombre impair de négations.

Soit π une formule élémentaire, nous distinguons l'ensemble des variables libres de π dénoté $\mathsf{Free}_r(\pi)$ apparaissant à droite du symbole $=$ ou $\Vdash$, de l'ensemble des variables libre de π dénoté $\mathsf{Free}_l(\pi)$ apparaissant à gauche du symbole $=$ ou $\Vdash$.

$$\begin{array}{ll}
\mathsf{Free}_l(\mathsf{true}) = \emptyset & \mathsf{Free}_r(\mathsf{true}) = \emptyset \\
\mathsf{Free}_l(t_1 = t_2) = \mathsf{Vars}(t_1) & \mathsf{Free}_r(t_1 = t_2) = \mathsf{Vars}(t_2) \\
\mathsf{Free}_l(T \Vdash t) = \mathsf{Vars}(T) & \mathsf{Free}_r(T \Vdash t) = \mathsf{Vars}(t) \\
\mathsf{Free}_l(\neg\pi) = \mathsf{Free}_l(\pi) & \mathsf{Free}_r(\neg\pi) = \mathsf{Free}_r(\pi) \\
\mathsf{Free}_l(\pi_1 \vee \pi_2) = \mathsf{Free}_l(\pi_1) \cup \mathsf{Free}_l(\pi_2) & \mathsf{Free}_r(\pi_1 \vee \pi_2) = \mathsf{Free}_r(\pi_1) \cup \mathsf{Free}_r(\pi_2) \\
\mathsf{Free}_l(\exists x.\ \pi) = \mathsf{Free}_l(\pi) \setminus \{x\} & \mathsf{Free}_r(\exists x.\ \pi) = \mathsf{Free}_r(\pi) \setminus \{x\}
\end{array}$$

Définition 5.4.2 ($\sigma \models' \pi$)**.** *Soient π une formule élémentaire et σ une substitution close telle que $\mathsf{Free}_r(\pi) = \emptyset$ et $\mathsf{Free}_l(\pi) = \mathrm{dom}(\sigma)$. $\sigma \models' \pi$ est défini inductivement :*

$$\begin{array}{lll}
\sigma \models' \mathsf{true} & & \\
\sigma \models' t_1 = t_2 & \text{ssi} & t_1\sigma = t_2 \\
\sigma \models' T \Vdash t & \text{ssi} & T\sigma \vdash t \\
\sigma \models' \neg\pi & \text{ssi} & \sigma \not\models' \pi \\
\sigma \models' \pi_1 \vee \pi_2 & \text{ssi} & \sigma \models' \pi_1 \text{ ou } \sigma \models' \pi_2 \\
\sigma \models' \exists x.\ \pi & \text{ssi} & \exists t \in \mathcal{T} \text{ tel que } \sigma \models' \pi[x \mapsto t]
\end{array}$$

Définition 5.4.3 (Traduction $\mathbf{T}$.)**.** *Soit $\phi \in \mathcal{PS}$-LTL. Soient aussi Π un protocole, tr une trace symbolique de Π,* i.e. *associée à un scénario de Π, et T_0 un ensemble de termes clos. $\mathbf{T}(\phi, tr, T_0)$ dénote la traduction de ϕ en une formule élémentaire « équivalente » et est définie comme suit :*

$$\begin{array}{rcl}
\mathbf{T}(\mathsf{true}, tr, T_0) & \rightarrow & \mathsf{true} \\
\mathbf{T}(\mathsf{learn}(t), tr, T_0) & \rightarrow & T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(tr) \Vdash t \\
\mathbf{T}(\mathsf{Q}(t_1, \ldots, t_n), tr, T_0) & \rightarrow & \begin{cases} t'_1 = t_1 \wedge \cdots \wedge t'_n = t_n & si\ tr = tr'@[\mathsf{Q}(t'_1, \ldots, t'_n)] \\ \neg\mathsf{true} & sinon \end{cases} \\
\mathbf{T}(\mathsf{C}(t), tr, T_0) & \rightarrow & \epsilon = t \vee \bigvee\limits_{\mathsf{pvk}(u) \in T_0} u = t \vee \bigvee\limits_{\mathsf{shk}(u,t) \in T_0} \epsilon \neq u \\
\mathbf{T}(\neg\phi, tr, T_0) & \rightarrow & \neg\mathbf{T}(\phi, tr, T_0) \\
\mathbf{T}(\phi_1 \vee \phi_2, tr, T_0) & \rightarrow & \mathbf{T}(\phi_1, tr, T_0) \vee \mathbf{T}(\phi_2, tr, T_0) \\
\mathbf{T}(\mathcal{Y}\phi, tr, T_0) & \rightarrow & \begin{cases} \mathbf{T}(\phi, tr, T_0) & si\ tr = tr'@[e] \\ \neg\mathsf{true} & sinon \end{cases} \\
\mathbf{T}(\phi_1 \mathcal{S} \phi_2, tr, T_0) & \rightarrow & \begin{cases} \mathbf{T}(\phi_2, tr, T_0) \vee (\mathbf{T}(\phi_1 \mathcal{S} \phi_2, tr', T_0) \wedge \mathbf{T}(\phi_1, tr, T_0)) & \\ \qquad\qquad si\ tr = tr'@[e] & \\ \neg\mathsf{true} \qquad\qquad sinon & \end{cases} \\
\mathbf{T}(\exists x.\ \phi, tr, T_0) & \rightarrow & \exists x.\ \mathbf{T}(\phi, tr, T_0)
\end{array}$$

Le lemme suivant stipule que la traduction $\mathbf{T}$ est correcte.

Lemme 5.4.4. *Soient $\phi \in \mathcal{PS}$-LTL une formule close, tr une trace, T_0 un ensemble de termes clos, et σ une substitution close telle que $\mathcal{V}(tr) = \mathrm{dom}(\sigma)$.*

$$\langle tr\sigma, T_0\rangle \models \phi \text{ si et seulement si } \sigma \models' \mathbf{T}(\phi, tr, T_0).$$

Aussi, toute formule atomique de la forme $T \Vdash t$ apparaît positivement dans $\mathbf{T}(\phi, tr, T_0)$, i.e. *toute occurrence de $T \Vdash t$ apparaît dans $\mathbf{T}(\phi, tr, T_0)$ sous un nombre pair de négations.*

Exemple 5.4.5. *Reprenons notre protocole Π'_{Toy} et rappelons la trace symbolique tr'_{Toy} et la formule $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ introduits à l'exemple 3.2.1.*

$$\begin{aligned} tr'_{\mathsf{Toy}} = [\ & \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)); \mathit{Secret}(a, b, n^1); \\ & \mathsf{rcv}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, b), \epsilon\rangle, b)); \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, \epsilon), b\rangle, \epsilon)); \\ & \mathsf{rcv}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b)); \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^3, \epsilon), b\rangle, \epsilon)); \\ & \mathsf{rcv}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, a), b\rangle, a)) \qquad] \end{aligned}$$

$$\phi^{\mathsf{S}}_{\mathsf{Toy}} = \begin{cases} \forall y_a.\forall y_b.\forall y_n. \\ [\mathcal{O}(\mathsf{Secret}(y_a, y_b, y_n)) \ \wedge\ \mathsf{NC}(y_a) \ \wedge\ \mathsf{NC}(y_b)] \ \Rightarrow\ \neg\mathsf{learn}(y_n). \end{cases}$$

La formule élémentaire associée par $\mathbf{T}$ à $\neg\phi^{\mathsf{S}}_{\mathsf{Toy}}$, tr'_{Toy} et la connaissance initiale de l'intrus T_0 vide est :

$$\mathbf{T}(\neg\phi^{\mathsf{S}}_{\mathsf{Toy}}, tr'_{\mathsf{Toy}}, \emptyset) = (a = y_a \wedge b = y_b \wedge n = y_n \wedge \mathsf{K}(tr'_{\mathsf{Toy}}) \cup \mathcal{N}_\epsilon(\Pi'_{\mathsf{Toy}}) \Vdash y_n).$$

Dans la suite de ce travail nous aurons besoin de relier toute égalité (ou diségalité) de $\mathbf{T}(\phi, tr, T_0)$ à la sous-formule de ϕ dont elle est la traduction. Nous énonçons à cet effet un lemme qui caractérise l'origine dans ϕ d'une égalité de $\mathbf{T}(\phi, tr, T_0)$.

Lemme 5.4.6. *Soient ϕ une formule de $\mathcal{PS}$-LTL$^+$, Π un protocole, T_0 un ensemble de termes clos tel que*

$$\forall k \in (\mathcal{K} \cap \mathsf{St}(T_0)).\ T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash k \ \Rightarrow k \in T_0.$$

Soient aussi tr une trace valide de Π au regard de la connaissance initiale de l'intrus T_0, et $\psi = \mathbf{T}(\phi, tr, T_0)$.

1. *Pour tout $t = u \in \mathsf{SubForm}^+(\psi)$, ou bien*

$$\exists \mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{SubForm}^+(\phi).\ \exists \mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{Elmts}(tr).\ \exists i \in [\![n]\!].\ t_i = t \ \wedge\ u_i = u,$$

ou

$$\mathsf{C}(u) \in \mathsf{SubForm}^+(\phi) \ \wedge\ t \in \mathcal{P},$$

ou encore

$$t \in \mathcal{P} \ \wedge\ u \in \mathcal{P},$$

2. *Pour tout* $t = u \in \mathsf{SubForm}^-(\psi)$,
ou bien

$$\exists \mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{SubForm}^-(\phi).\ \exists \mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{Elmts}(tr).\ \exists i \in [\![n]\!].$$
$$t_i = t \ \wedge\ u_i = u,$$

ou

$$\mathsf{C}(u) \in \mathsf{SubForm}^-(\phi) \ \wedge\ t \in \mathcal{P},$$

ou encore

$$t \in \mathcal{P} \ \wedge\ u \in \mathcal{P}.$$

Démonstration. Nous procédons par induction sur la mesure $(|tr|, |\phi|)$ relativement à l'ordre lexicographique.

Cas $\phi = \mathsf{true}$.
Par définition $\psi = \mathsf{true}$, et donc $\mathsf{SubForm}^+(\psi) = \{\mathsf{true}\}$, et $\mathsf{SubForm}^-(\psi) = \emptyset$. Aucune égalité n'apparaissant ni négativement ni positivement dans ψ nous concluons immédiatement.

Cas $\phi = \mathsf{learn}(v)$.
Par définition $\psi = T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(tr) \Vdash v$, et donc $\mathsf{SubForm}^+(\psi) = \{\psi\}$, et $\mathsf{SubForm}^-(\psi) = \emptyset$. Aucune égalité n'apparaissant ni négativement ni positivement dans ψ nous concluons immédiatement.

Cas $\phi = \mathsf{Q}(u_1, \ldots, u_n)$.
Si $tr = tr'@[\mathsf{Q}(t_1, \ldots, t_n)]$, alors $\psi = (t_1 = u_1 \wedge \cdots \wedge t_n = u_n)$. Ainsi pour tout $t = u \in \mathsf{SubForm}^+(\phi)$ il existe $i \in [\![n]\!]$ tel que $t = t_i$ et $u = u_i$ avec $\mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{SubForm}^+(\phi)$ et $\mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{Elmts}(tr)$. De plus, $\mathsf{SubForm}^-(\psi) = \emptyset$ nous permet de conclure quant à ce cas.
Si par contre $tr \neq tr'@[\mathsf{Q}(t_1, \ldots, t_n)]$, alors $\psi = \mathsf{false}$, et donc $\mathsf{SubForm}^+(\psi) = \emptyset$ et $\mathsf{SubForm}^-(\psi) = \{\mathsf{false}\}$. Aucune égalité n'apparaissant ni négativement ni positivement dans ψ nous concluons immédiatement.

Cas $\phi = \mathsf{C}(v)$.
Par définition

$$\psi = (\epsilon = v \vee \bigvee_{\mathsf{pvk}(v') \in T_0} v' = v \vee \bigvee_{\mathsf{shk}(v', v) \in T_0} \epsilon \neq v'),$$

et donc pour tout $t = u \in \mathsf{SubForm}^+(\psi)$, $t \in \mathcal{P}$ et $u = v$. De même, pour tout $t = u \in \mathsf{SubForm}^-(\psi)$, $t = \epsilon$ et $u \in \mathcal{P}$.

Cas $\phi = \neg\phi_1$. Posons $\psi_1 = \mathbf{T}(\phi_1, tr, T_0)$. Par définition $\mathsf{SubForm}^+(\psi) = \{\psi\} \cup \mathsf{SubForm}^-(\psi_1)$ et $\mathsf{SubForm}^-(\psi) = \mathsf{SubForm}^+(\psi_1)$.Soit $t = u \in \mathsf{SubForm}^+(\psi)$. Nous savons alors que $t = u \in \mathsf{SubForm}^-(\psi_1)$. Or par hypothèse d'induction nous savons qu'ou bien

$$\exists \mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{SubForm}^-(\phi_1).\ \exists \mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{Elmts}(tr).\ \exists i \in [\![n]\!].$$
$$t_i = t \ \wedge\ u_i = u,$$

ou

$$\mathsf{C}(u) \in \mathsf{SubForm}^-(\phi_1) \ \wedge \ t \in \mathcal{P},$$

ou encore

$$t \in \mathcal{P} \ \wedge \ u \in \mathcal{P}.$$

Il suffit pour conclure de noter que

$$\mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{SubForm}^-(\phi_1) \Rightarrow \mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{SubForm}^+(\phi),$$

et de même que $\mathsf{C}(u) \in \mathsf{SubForm}^-(\phi_1) \Rightarrow \mathsf{C}(u) \in \mathsf{SubForm}^+(\phi)$. Le raisonnement concernant les égalités apparaissant négativement dans ψ étant analogue nous ne le détaillerons pas ici.

Cas $\phi = \phi_1 \vee \phi_2$.
Il suffit pour conclure par induction de faire appel aux définitions de $\mathsf{SubForm}^+()$ et $\mathsf{SubForm}^-()$ et de noter que $\psi = \psi_1 \vee \psi_2$ pour $\psi_1 = \mathbf{T}(\phi_1, tr, T_0)$ et $\psi_2 = \mathbf{T}(\phi_2, tr, T_0)$.

Cas $\phi = \mathcal{Y}\phi_1$.
Si $tr = tr'@[e]$, alors nous concluons par induction en faisant appel aux définitions de $\mathsf{SubForm}^+()$ et $\mathsf{SubForm}^-()$ et en notant que $\psi = \mathcal{Y}\psi_1$ pour $\psi_1 = \mathbf{T}(\phi_1, tr', T_0)$. Dans le cas contraire, *i.e.* $tr = [\,]$, nous savons que $\psi = \neg\mathsf{true}$, et donc que $\mathsf{SubForm}^+(\psi) = \{\neg\mathsf{true}\}$ et $\mathsf{SubForm}^-(\psi) = \{\mathsf{true}\}$. Aucune égalité n'apparaissant ni négativement ni positivement dans ψ nous concluons immédiatement.

Cas $\phi = \phi_1 \mathcal{S} \phi_2$.
Si $tr = tr'@[e]$, alors par définition $\psi = \psi_1 \vee (psi_2 \wedge \psi_3)$ pour $\psi_1 = \mathbf{T}(\phi_2, tr, T_0)$, $\psi_2 = \mathbf{T}(\phi_1 \mathcal{S} \phi_2, tr', T_0)$ et $\psi_3 = \mathbf{T}(\phi_1, tr, T_0)$. Il suffit pour conclure par induction de faire appel aux définitions de $\mathsf{SubForm}^+()$ et $\mathsf{SubForm}^-()$. Dans le cas contraire, *i.e.* $tr = [\,]$, nous savons que $\psi = \neg\mathsf{true}$, et donc que $\mathsf{SubForm}^+(\psi) = \{\neg\mathsf{true}\}$ et $\mathsf{SubForm}^-(\psi) = \{\mathsf{true}\}$. Aucune égalité n'apparaissant ni négativement ni positivement dans ψ nous concluons immédiatement.

Cas $\phi = \exists x.\ \phi_1$.
Il suffit pour conclure par induction de faire appel aux définitions de $\mathsf{SubForm}^+()$ et $\mathsf{SubForm}^-()$ et de noter que $\psi = \exists x.\ \psi_1$ pour $\psi_1 = \mathbf{T}(\phi_1, tr, T_0)$. □

5.5 Procédure de décision

Nous en venons maintenant à la procédure qui étant donnés une trace symbolique tr, un ensemble de termes clos T_0, et une formule ϕ de $\mathcal{PS}$-LTL$^+$ décide si $\langle tr, T_0 \rangle \models \phi$. L'approche proposée dans [20, 21], et que nous reprenons ici, consiste comme nous l'annoncions en introduction de la section 5.4 en deux étapes : *(i)* ϕ est d'abord simplifiée en la formule élémentaire équivalente (dans le sens précisé par le lemme 5.4.4) $\pi = \mathbf{T}(\phi, tr, T_0)$, puis *(ii)* π est passée en argument à la procédure de décision $\mathbf{D}$ que nous décrivons à présent.

Notons que ϕ étant une formule de $\mathcal{PS}$-LTL$^+$, π est nécessairement une formule élémentaire positive de la forme $\pi = \exists x_1 \ldots \exists x_n.\ \varpi$.

Procédure 5.5.1. *Soient C un système de contraintes symboliques en forme résolue, et $\pi = \exists x_1 \dots \exists x_n.\ \varpi$ une formule élémentaire positive telle que ϖ soit en forme normale disjonctive,* i.e. *de la forme $\varpi = \bigvee_{i \in [\![k]\!]} \varpi_i$, et pour tout $i \in [\![k]\!]$, ϖ_i de la forme $\varpi_i = C_i \wedge Eq_i \wedge Deq_i$ avec :*

- $C_i = \{T_j \Vdash t_j\}_{j \in [\![\ell_i]\!]}$,
- $Eq_i = \{s_j = s'_j\}_{j \in [\![m_i]\!]}$,
- $Deq_i = \{u_j \neq u'_j\}_{j \in [\![n_i]\!]}$.

fonction $\mathbf{D}(C, \pi)$
  pour i de 1 à k faire
    $\theta = \mathrm{mgu}(Eq_i)$
    si $\theta \neq \bot$ *alors*
      $E = C_i\theta \wedge C\theta$
      $\Sigma = \{\sigma \mid E \rightsquigarrow^*_\sigma F$ *et* F *est en forme résolue*$\}$
      pour tout $\sigma \in \Sigma$ *faire*
        si $(Deq_i)\theta\sigma \equiv \mathsf{true}$ *alors*
          retourner $\theta\sigma$
  retourner $\bot$
fin

Le lemme suivant établi que la procédure $\mathbf{D}$ est correcte. Nous ne faisons ici que l'énoncer et renvoyons le lecteur à [20, 21] pour une preuve détaillée.

Lemme 5.5.2. *Soient Π un protocole, sc un scénario de Π, T_0 un ensemble de termes clos, tr la trace symbolique associée à sc, ϕ une formule de $\mathcal{PS}$-LTL$^+$. Soient C un système de contraintes symboliques en forme résolue et σ une substitution tels que $\mathcal{C}(tr, T_0) \rightsquigarrow^*_\sigma C$. Soit π la formule élémentaire telle que $\pi = \mathbf{T}(\phi, tr\sigma, T_0)$, et π' la formule élémentaire positive en forme normale disjonctive correspondante.*

1. *si $\mathbf{D}(\pi, C) = \theta$ $(\neq \bot)$, alors $\theta \models' \pi$ et $tr\sigma\theta$ est une trace valide de Π, au regard de la connaissance initiale de T_0 ;*
2. *si $\theta \models' \pi$ et $tr\sigma\theta$ est une trace valide de Π, au regard de la connaissance initiale de T_0 pour une certaine substitution θ, alors il existe une substitution γ telle que $\mathbf{D}(\pi, C) = \gamma$ $(\neq \bot)$.*

Le théorème suivant stipule que la procédure de vérification en deux temps qui consiste à d'abord appliquer la transformation $\mathbf{T}$, puis à appeler la procédure $\mathbf{D}$ résultante est correcte et complète. Le lecteur pourra consulter la preuve de ce résultat dans [20, 21].

Théorème 5.5.3. *Soient Π un protocole, sc un scénario de Π, T_0 un ensemble de termes clos, tr la trace symbolique associée à sc, ϕ une formule de $\mathcal{PS}$-LTL$^-$. Soit $A_\phi = \neg\phi$ la formule d'attaque correspondante. Soit $C = \mathcal{C}(tr, T_0)$. Soit l'ensemble $\mathsf{Sol}(C) = \{(C', \sigma) \mid C \rightsquigarrow^*_\sigma C',\ C'$ en forme résolue$\}$.*

$$\langle tr, T_0 \rangle \models \phi \qquad \text{ssi} \qquad \forall (C', \sigma) \in \mathsf{Sol}(C).\ \mathbf{D}(\pi, C') = \bot$$

avec π la forme normale disjonctive de $\mathbf{T}(A_\phi, tr\sigma, T_0)$.

Chapitre 6

Réduction de l'espace de recherche à des exécutions de taille bornée

Au chapitre 4 nous avons vu que si nous nous donnons un nombre non-borné de sessions, le problème de la vérification des protocoles de sécurité est indécidable, quand bien même nous nous restreindrions à des messages de taille bornée. D'un autre coté, nous avons discuté les raisons pour lesquelles il est important de pouvoir vérifier automatiquement les protocoles de sécurité avant de les implanter dans des applications, et ce dans le cadre d'un nombre non-borné de sessions. Une des options pour faire face à l'indécidabilité du problème dans le cas général est de restreindre l'ensemble de protocoles ainsi que l'ensemble de propriétés considérés ; en d'autre termes, concevoir des classes de protocoles, ainsi que des classes de propriétés pour lesquelles le problème de la vérification devient décidable.

Une telle classe de protocoles (la classe $\mathcal{C}_1$), ainsi que la classe de propriétés associée (la classe Φ_1), sont présentées dans ce chapitre aux sections 6.1 et 6.2 respectivement. Nous montrons plus précisément (sections 6.3 et 6.4) qu'il suffit de considérer un nombre borné de sessions pour décider si un protocole de la classe $\mathcal{C}_1$ vérifie une propriété donnée de la classe Φ_1. La spécification de $\mathcal{C}_1$ repose sur une transformation qui à un protocole quelconque en associe un « équivalent » dans $\mathcal{C}_1$. L'avantage d'un tel résultat de réduction est qu'il existe de nombreux outils efficaces de vérification pour un nombre borné de sessions (l'outil AVISPA [7] pour n'en citer qu'un).

6.1 La classe de protocoles $\mathcal{C}_1$

Habituellement, une classe décidable d'instances d'un problème dont on sait qu'il est indécidable, est définie à l'aide d'un critère « structurel » (pour les protocoles il s'agît de critères syntaxiques sur les termes apparaissant dans leur spécification) vérifié par toutes les instances de la classe. Ici nous procédons légèrement différemment. Nous définissons notre classe de protocoles $\mathcal{C}_1$ de manière constructive, en ce sens qu'un protocole Π appartiendra à la classe $\mathcal{C}_1$

s'il résulte de la « compilation » d'un autre protocole, sur lequel par contre aucune contrainte n'a besoin d'être posée.

Plus précisément, nous définissons une transformation (un compilateur) sur les protocoles, qui à tout protocole Π fait correspondre un unique protocole $\overline{\Pi}$, et définissons la classe $\mathcal{C}_1$ comme l'ensemble des protocoles obtenus par application de cette transformation.

Afin de faciliter la lecture de ce chapitre, nous introduisons la notation suivante.

Notation 6.1.1. *Soient n termes $u_1, \ldots, u_n \in \mathcal{T}$,*

$$\langle u_1, \ldots, u_n \rangle \stackrel{def}{=} \langle u_1, \langle \ldots, \langle u_{n-1}, u_n \rangle \ldots \rangle \rangle.$$

Notre transformation repose sur l'opération de *marquage* formellement définie ci-dessous.

Définition 6.1.2 ($\lfloor t \rfloor_\tau$)**.** *Soient t et τ deux termes, marquer (ou encore* tagger*) t avec τ résulte en le terme dénoté $\lfloor t \rfloor_\tau$, et défini comme suit :*

$$\lfloor t \rfloor_\tau \stackrel{def}{=} \begin{cases} \langle \lfloor t_1 \rfloor_\tau, \lfloor t_2 \rfloor_\tau \rangle & \text{si } t = \langle t_1, t_2 \rangle \\ f(\langle \tau, \lfloor t_1 \rfloor_\tau \rangle, \lfloor t_2 \rfloor_\tau) & \text{si } t = f(t_1, t_2) \text{ avec } f \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\ \mathsf{h}(\langle \tau, \lfloor t_1 \rfloor_\tau \rangle) & \text{si } t = \mathsf{h}(t_1) \\ t & \text{sinon} \end{cases}$$

En d'autres termes tagger t avec le terme τ consiste à marquer dans t toute application d'une primitive cryptographique (à l'exceptions de la concaténation) avec τ.

Définition 6.1.3 (k-tag)**.** *Soit un entier $k \in \mathbb{N}$. Un k-tag est un terme de la forme $\langle t_1, \ldots, t_k \rangle$ où $t_1, \ldots, t_k \in \mathcal{T}$. Nous dirons qu'un terme t est k-taggé s'il existe un terme u et un k-tag τ tel que $t = \lfloor u \rfloor_\tau$. Nous noterons dans ce cas $\mathsf{Tags}(t) = \{\tau\}$.*

Exemple 6.1.4. *Soient les deux termes $t = \mathsf{aenc}(\langle \mathsf{aenc}(n, b), a \rangle, b)$ et $\tau = \langle n_1, n_2 \rangle$. τ est un 2-tag. En taggant t avec τ, nous obtenons le terme*

$$\lfloor t \rfloor_\tau = \mathsf{aenc}(\langle \langle n_1, n_2 \rangle, \langle \mathsf{aenc}(\langle \langle n_1, n_2 \rangle, n \rangle, b), a \rangle \rangle, b)$$

6.1.1 Description informelle de la transformation

Soit un protocole Π impliquant k rôles, posons $\{r_{p_1}, \ldots, r_{p_k}\} = \mathsf{Roles}(\Pi)$. Intuitivement notre compilateur construit un nouveau protocole $\overline{\Pi}$ impliquant lui aussi k rôles $\{\overline{r_{p_1}}, \ldots, \overline{r_{p_k}}\} = \mathsf{Roles}(\overline{\Pi})$ en distinguant deux phases. Au cours de la première phase, les k participants $\{p_1, \ldots, p_k\}$ du protocole résultant cherchent à établir un numéro de session[1]. A cet effet, chaque participant engendre un nonce n_i qu'il diffuse à tous les autres participants. A l'issue de cette première étape, chaque participant calcule le numéro de session τ en

[1] A ne pas confondre avec l'identificateur de session qui apparaît dans l'entrelacement d'un scénario.

concaténant (dans un ordre qui sera précisé dans un instant) les $k-1$ messages qu'il vient de recevoir avec celui qu'il a lui-même émis.

Ensuite, les k participants passent à la seconde phase du protocole $\overline{\Pi}$. Ils exécutent le corps du protocole Π d'origine dans lequel chaque application d'une primitive cryptographique a été marquée avec le numéro de session τ calculé à l'étape précédente.

Nous résumons ces idées en présentant dans un premier temps la transformation décrite ci-dessus dans le modèle plus intuitif d'Alice et Bob. Supposons que Π corresponde dans le modèle Alice et Bob à la séquence d'émissions suivante :

$$\Pi = \begin{cases} p_{i_1} \rightarrow p'_{j_1} : m_1 \\ \vdots \\ p_{i_\ell} \rightarrow p'_{j_\ell} : m_\ell \end{cases}$$

Alors, le protocole $\overline{\Pi}$ construit par notre compilateur correspond à la séquence d'émissions suivante :

$$\overline{\Pi} = \begin{cases} \left.\begin{array}{l} p_1 \rightarrow All : n_1 \\ \vdots \\ p_k \rightarrow All : n_k \end{array}\right\} \text{Phase 1} \\ \\ \left.\begin{array}{l} p_{i_1} \rightarrow p'_{j_1} : \lfloor m_1 \rfloor_\tau \\ \vdots \\ p_{i_\ell} \rightarrow p'_{j_\ell} : \lfloor m_\ell \rfloor_\tau \end{array}\right\} \text{Phase 2 où } \tau = \langle n_1, \ldots, n_k \rangle \end{cases}$$

Si le mécanisme de *diffusion* (*broadcast*) mis en œuvre dans la phase d'initialisation n'est pas possible (*e.g.* si les participants ne se connaissent pas tous), où s'il n'est pas efficace, il est toutesfois possible de procéder par *propagation*, *i.e* le message à diffuser est envoyé à un des participants, qui se charge de le transmettre à un autre, et ainsi de suite, jusqu'à ce que tous les participants l'aient reçu.

Exemple 6.1.5. *Illustrons notre propos sur notre protocole* Π_{Toy}. *Celui-ci était défini, dans le modèle Alice et Bob, comme la séquence d'émissions suivante :*

$$\Pi_{\mathsf{Toy}} = \begin{cases} a \rightarrow b : \mathsf{aenc}(\langle \mathsf{aenc}(n, b), a \rangle, b) \\ b \rightarrow a : \mathsf{aenc}(\langle \mathsf{aenc}(n, a), b \rangle, a). \end{cases}$$

Le protocole $\overline{\Pi_{\mathsf{Toy}}}$ *associé à* Π_{Toy} *d'après la transformation décrite ci-dessus est donc le suivant :*

$$\overline{\Pi_{\mathsf{Toy}}} = \begin{cases} \left.\begin{array}{l} a \rightarrow b : n_1 \\ b \rightarrow a : n_2 \end{array}\right\} \textit{Phase 1} \\ \left.\begin{array}{l} a \rightarrow b : \mathsf{aenc}(\langle\langle n_1, n_2 \rangle, \langle \mathsf{aenc}(\langle\langle n_1, n_2 \rangle, n \rangle, b), a \rangle\rangle, b) \\ b \rightarrow a : \mathsf{aenc}(\langle\langle n_1, n_2 \rangle, \langle \mathsf{aenc}(\langle\langle n_1, n_2 \rangle, n \rangle, a), b \rangle\rangle, a) \end{array}\right\} \textit{Phase 2} \end{cases}$$

A la première étape l'agent a engendre le nonce n_1 *qu'il envoie à l'agent b, et b engendre le nonce* n_2 *qu'il envoie à l'agent a. Les agents calculent alors*

le numéro de sessions $\langle n_1, n_2 \rangle$ qu'ils utilisent pour marquer toute application d'une primitive cryptographique dans les messages émis par la suite.

Le protocole $\overline{\Pi}$ ainsi présenté, les participants semblent tomber d'accord sur le numéro de session τ. En effet, dans le modèle Alice et Bob ce qui est représenté est une exécution honnête, sans interférence de la part de l'intrus, du protocole considéré. A noter que l'intrus peut s'interposer dans cet échange de messages menant chaque participant à l'issue de la première phase à un numéro de session différent. Afin d'analyser le protocole $\overline{\Pi}$ il nous faut donc avant tout le formaliser dans notre modèle présenté au chapitre 2.

6.1.2 Définition formelle de la transformation

Définition 6.1.6 ($\overline{\Pi}$). *Soit $\Pi = [e_1; \ldots; e_\ell]$ un protocole avec $\mathsf{Roles}(\Pi) = \{r_{p_1}, \ldots, r_{p_k}\}$. Soient k nonces distincts n'apparaissant pas dans Π, $n_1, \ldots, n_k \in (\mathcal{N} \setminus \mathcal{N}(\Pi))$. Soient aussi k^2 variables[2] distinctes n'apparaissant pas dans Π, $x_j^i \in (\mathcal{V} \setminus \mathcal{V}(\Pi))$ pour $i, j \in [\![k]\!]$. La compilation du protocole Π résulte en le protocole dénoté $\overline{\Pi}$, et défini comme suit :*

$$\overline{\Pi} = \Pi_1^0 \ @ \ \ldots \ @ \ \Pi_k^0 \ @ \ \Pi'$$

où

$$\Pi_i^0 = \mathop{@}_{\substack{j \in [\![k]\!] \\ j \neq i}} [\mathsf{snd}(p_i, p_j, n_i); \mathsf{rcv}(p_j, p_i, x_j^i)]$$

et $\Pi' = [e'_1; \ldots; e'_\ell]$ avec pour tout $h \in [\![\ell]\!]$

$$e'_h = \begin{cases} \mathsf{snd}(p_i, p_j, \lfloor t \rfloor_{\tau_i}) & \text{si } e_h = \mathsf{snd}(p_i, p_j, t) \quad i, j \in [\![k]\!] \\ \mathsf{rcv}(p_i, p_j, \lfloor t \rfloor_{\tau_i}) & \text{si } e_h = \mathsf{rcv}(p_i, p_j, t) \quad i, j \in [\![k]\!] \\ \mathsf{Q}(p_i, \lfloor t_1 \rfloor_{\tau_i}, \ldots, \lfloor t_n \rfloor_{\tau_i}) & \text{si } e_h = \mathsf{Q}(p_i, t_1, \ldots, t_n) \quad i \in [\![k]\!] \end{cases}$$

où pour tout $i \in [\![k]\!]$, $\tau_i = \langle u_i^1, \ldots, u_i^k \rangle$ avec pour tout $j \in [\![k]\!]$,

$$u_i^j = \begin{cases} n_i & \text{si } j = i \\ x_i^j & \text{sinon.} \end{cases}$$

Le sous-protocole $\Pi_1^0 \ldots \Pi^0$ correspond à la phase d'initialisation décrite à la section 6.1.1, au cours de laquelle les participants échangent des nonces frais afin d'établir un numéro de session. Le sous-protocole Π' correspond quant à lui à la seconde phase décrite à la section 6.1.1. Il est construit en marquant dans le protocole d'origine chaque application d'une primitive cryptographique avec le numéro de session calculé à l'étape précédente.

Exemple 6.1.7. *Illustrons notre transformation sur notre protocole Π'_{Toy}. Rappelons que celui-ci était défini dans notre modèle comme la séquence d'événe-*

[2] Il suffirait de considérer $k \times (k-1)$ variables n'apparaissant pas dans Π. Néanmoins, en considérer k^2 est gratuit et nous évite de rentrer dans un certain nombre de technicalités.

ments suivante :

$$\begin{aligned}\Pi'_{\mathsf{Toy}} = [\quad & \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n, b), a\rangle, b));\\ & \mathsf{Secret}(a, a, b, n);\\ & \mathsf{rcv}(b, a, \mathsf{aenc}(\langle \mathsf{aenc}(x, b), a\rangle, b));\\ & \mathsf{snd}(b, a, \mathsf{aenc}(\langle \mathsf{aenc}(x, a), b\rangle, a));\\ & \mathsf{rcv}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n, a), b\rangle, a))].\end{aligned}$$

La compilation du protocole Π'_{Toy} selon le schéma de compilation décrit à la définition 6.1.6 résulte en le protocole :

$$\begin{aligned}\overline{\Pi'_{\mathsf{Toy}}} = [\quad & \mathsf{snd}(a, b, n_1);\\ & \mathsf{rcv}(b, a, x_1);\\ & \mathsf{snd}(b, a, n_2);\\ & \mathsf{rcv}(a, b, x_2);\\ & \mathsf{snd}(a, b, \mathsf{aenc}(\langle\langle n_1, x_2\rangle, \langle \mathsf{aenc}(\langle\langle n_1, x_2\rangle, n\rangle, b), a\rangle\rangle, b));\\ & \mathsf{Secret}(a, a, b, n);\\ & \mathsf{rcv}(b, a, \mathsf{aenc}(\langle\langle x_1, n_2\rangle, \langle \mathsf{aenc}(\langle\langle x_1, n_2\rangle, n\rangle, b), a\rangle\rangle, b));\\ & \mathsf{snd}(b, a, \mathsf{aenc}(\langle\langle x_1, n_2\rangle, \langle \mathsf{aenc}(\langle\langle x_1, n_2\rangle, n\rangle, a), b\rangle\rangle, a));\\ & \mathsf{rcv}(a, b, \mathsf{aenc}(\langle\langle n_1, x_2\rangle, \langle \mathsf{aenc}(\langle\langle n_1, x_2\rangle, n\rangle, a), b\rangle\rangle, a))]\end{aligned}$$

Phase 1 (4 premières lignes) ; *Phase 2* (4 dernières lignes)

Au regard de cet exemples, nous comprenons qu'avant de déchiffrer le message reçu, b vérifiera le tag utilisé par a, et ne procédera au déchiffrement que si celui-ci correspond au tag qu'il a lui même calculé à l'issue de la première phase. Ainsi, si l'intrus s'interpose au cours de la phase d'initialisation, b s'en apercevra dès le début de la seconde.

Le schéma de compilation à présent défini, nous sommes en mesure de déterminer la classe de protocoles reposant sur ce dernier. La classe $\mathcal{C}_1$ est l'ensemble des protocoles résultants de la compilation d'un protocole quelconque.

Définition 6.1.8 (La classe $\mathcal{C}_1$)**.** *La classe de protocoles $\mathcal{C}_1$ est l'ensemble des protocoles obtenus par compilation selon le schéma décrit à la définition 6.1.6 d'un protocole quelconque,* i.e.

$$\mathcal{C}_1 \stackrel{def}{=} \{\Pi \mid \exists \Pi'.\ \Pi = \overline{\Pi'}\}.$$

Exemple 6.1.9. *D'après la définition ci-dessus, $\overline{\Pi'_{\mathsf{Toy}}} \in \mathcal{C}_1$.*

6.2 La classe de propriétés Φ_1

Nous nous sommes jusqu'à présent concentrés sur la définition de la classe de protocoles $\mathcal{C}_1$, reste encore à définir la classe de formules pour laquelle notre résultat de décidabilité tient. Cette section y est consacrée.

Définition 6.2.1 (La classe Φ_1)**.** *La classe de formules $\Phi_1 \subseteq \mathcal{PS}\text{-}LTL^-$ est l'ensemble des formules ϕ vérifiant les 4 conditions suivantes :*

1. *pour tout* $\mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)$, $t \in (\mathcal{P} \cup \mathcal{C} \cup \mathcal{V} \cup \mathcal{K}^{-1})$, *et pour tout* $\mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{SubForm}^-(\phi)$, $t_1, \ldots, t_n \in (\mathcal{P} \cup \mathcal{C} \cup \mathcal{V} \cup \mathcal{K}^{-1})$,
2. *tout* $\mathsf{learn}(t)$ *apparaît au plus une fois dans* ϕ, *et tout status event* $\mathsf{Q}(t_1, \ldots, t_n)$ *apparaît au plus une fois en tant que sous-formule négative de* ϕ,
3. *pour tous* $\mathsf{Q}(t_1, \ldots, t_n), \mathsf{Q}'(u_1, \ldots, u_m) \in \mathsf{SubForm}^-(\phi)$, $\mathsf{Q}(t_1, \ldots, t_n) \neq \mathsf{Q}'(u_1, \ldots, u_m) \Rightarrow \mathcal{V}(\{t_1, \ldots, t_n\}) \cap \mathcal{V}(\{u_1, \ldots, u_m\}) = \emptyset$.
4. *aucune modalité* ($\mathcal{Y}$ *ou* $\mathcal{S}$ *indifféremment*), *ni aucun* $\mathsf{learn}(t)$ *n'apparaît sous un* $\mathcal{S}$.

La condition 1 contraint les sous-termes négatifs de ϕ à être atomiques. Les conditions 2 et 3 nous assurent que chaque variable apparaît au plus une fois négativement. Sans entrer dans les détails du pourquoi de ces conditions, remarquons juste qu'elles portent toutes (exceptée la 4) sur les sous-formules négatives de ϕ. Or, nous avons vu au chapitre 5 que la procédure **D** de décision dans le cadre d'un nombre borné de sessions opère en unifiant des sous-termes de la trace et des sous-termes négatifs de la formule considérée. En posant donc des conditions sur les sous-termes négatifs d'une formule il est possible, comme nous le verrons par la suite, de contrôler en quelques sortes les unifications possibles au cours de toute application de la procédure **D** à cette formule. C'est précisément dans ce but que nous avons sélectionné ces conditions.

Exemple 6.2.2. *Pour tout protocole* Π, *les formules* ϕ_{Secret}, ϕ_{A}, ϕ_{WA}, *et* ϕ_{NIA} *définies au chapitre 3 sont toutes des formules de* Φ_1.

Pour la classe de protocoles $\mathcal{C}_1$ et la classe de formules Φ_1 nous verrons à la section suivante qu'il est possible de borner *à priori* le nombre de sessions à considérer pour trouver une attaque. Cette borne est fonction de la formule ϕ considérée. Plus précisément, pour décider si un protocole $\Pi \in \mathcal{C}_1$ satisfait une formule $\phi \in \Phi_1$ il suffit de considérer $\max\{1, \mathsf{M}(\phi)\}$ sessions de chaque rôle de Π, avec $\mathsf{M}()$ telle que définie ci-dessous.

Définition 6.2.3 ($\mathsf{M}(\phi)$, $\mathsf{M}^+(\phi)$, et $\mathsf{M}^-(\phi)$)**.** *Nous définissons la mesure* $\mathsf{M}()$ *sur les formules de* $\mathcal{PS}$*-LTL par*

$$\mathsf{M}(\phi) \stackrel{def}{=} \mathsf{M}^+(\phi) + \mathsf{NbLearn}(\phi)$$

où $\mathsf{NbLearn}(\phi) = |\{\mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)\}|$, *et*

$\mathsf{M}^+(\mathsf{true}) \stackrel{def}{=} 0$
$\mathsf{M}^+(\mathsf{Q}(t_1, \ldots, t_n)) \stackrel{def}{=} 1$
$\mathsf{M}^+(\mathsf{learn}(t)) \stackrel{def}{=} 0$
$\mathsf{M}^+(\mathsf{C}(t)) \stackrel{def}{=} 0$
$\mathsf{M}^+(\neg\phi) \stackrel{def}{=} \mathsf{M}^-(\phi)$
$\mathsf{M}^+(\phi_1 \vee \phi_2) \stackrel{def}{=} \max\{\mathsf{M}^+(\phi_1), \mathsf{M}^+(\phi_2)\}$
$\mathsf{M}^+(\mathcal{Y}\phi) \stackrel{def}{=} 1 + \mathsf{M}^+(\phi)$
$\mathsf{M}^+(\phi_1 \mathcal{S} \phi_2) \stackrel{def}{=} \mathsf{M}^+(\phi_2)$
$\mathsf{M}^+(\exists x.\ \phi) \stackrel{def}{=} \mathsf{M}^+(\phi)$

$\mathsf{M}^-(\mathsf{true}) \stackrel{def}{=} 0$
$\mathsf{M}^-(\mathsf{Q}(t_1, \ldots, t_n)) \stackrel{def}{=} 1$
$\mathsf{M}^-(\mathsf{learn}(t)) \stackrel{def}{=} 0$
$\mathsf{M}^-(\mathsf{C}(t)) \stackrel{def}{=} 0$
$\mathsf{M}^-(\neg\phi) \stackrel{def}{=} \mathsf{M}^+(\phi)$
$\mathsf{M}^-(\phi_1 \vee \phi_2) \stackrel{def}{=} \mathsf{M}^-(\phi_1) + \mathsf{M}^-(\phi_2)$
$\mathsf{M}^-(\mathcal{Y}\phi) \stackrel{def}{=} 1 + \mathsf{M}^-(\phi)$
$\mathsf{M}^-(\phi_1 \mathcal{S} \phi_2) \stackrel{def}{=} \mathsf{M}^-(\phi_1)$
$\mathsf{M}^-(\exists x.\ \phi) \stackrel{def}{=} \mathsf{M}^-(\phi)$

Essayons de donner l'intuition derrière cette définition de la fonction $\mathsf{M}()$. Pour ce faire, rappelons que notre but est de réduire le nombre de sessions à considérer pour trouver une attaque. En d'autres termes, nous cherchons à ne garder d'une attaque que les sessions nécessaires à celle-ci, pour montrer par la suite que l'ensemble de sessions nécessaires est de taille bornée. Supposons à présent que la formule d'attaque que nous considérons soit une disjonction, *i.e.* une formule $\phi = \phi_1 \vee \phi_2$ et que l'exécution exec satisfasse cette formule d'attaque ϕ, au regard d'une connaissance initiale de l'intrus T_0, *i.e.* $\langle \mathsf{exec}, T_0 \rangle \models \phi$. Nous savons alors que exec satisfait une des deux sous-formules directes de la disjonction, mais aussi que satisfaire une des deux sous-formules directes de la disjonction suffit à satisfaire ϕ tout entière. Ainsi, il suffit de considérer l'ensemble de sessions nécessaires à la satisfaction de ϕ_1 si $\langle \mathsf{exec}, T_0 \rangle \models \phi_1$, ou bien l'ensemble de sessions nécessaires à la satisfaction de ϕ_2 si $\langle \mathsf{exec}, T_0 \rangle \models \phi_2$. Une telle définition est correcte uniquement par ce que, comme nous le verrons par la suite, la restriction à $\max\{1, \mathsf{M}(\phi)\}$ sessions par rôle résulte en une exécution valide du protocole considéré, au regard de la connaissance initiale de l'intrus.

Nous aurions pu nous contenter du nombre de status events, de learn et d'occurrences de la modalité $\mathcal{Y}$ apparaissant dans la formule ϕ considérée pour borner le nombre de sessions. Néanmoins, comme le montre l'exemple qui suit, il arrivera souvent que $\mathsf{M}(\phi)$ soit bien plus petit.

Exemple 6.2.4. *La propriété de vivacité pour le protocole* Π'_{Toy} *peut être spécifiée, comme nous l'avons vu à l'exemple 3.2.2, à l'aide de la formule*

$$\phi^{\mathsf{A}}_{\mathsf{Toy}} = \left\{ \begin{array}{c} \forall y_1.\ \forall y_2.\ \forall z_1.\ \forall z_2. \\ \left.\begin{array}{c} [\underbrace{\mathsf{End}_1(y_1, y_2) \ \wedge \ \mathsf{NC}(y_1) \ \wedge \ \mathsf{NC}(y_2)}_{\phi_3} \\ \Rightarrow \\ \underbrace{\mathcal{O}(\mathsf{Start}_1(y_1) \vee \mathsf{Start}_2(y_1))}_{\phi_4} \ \wedge \ \underbrace{\mathcal{O}(\mathsf{Start}_1(y_2) \vee \mathsf{Start}_2(y_2))}_{\phi_5}] \end{array}\right\} \phi_1 \\ \wedge \\ \left.\begin{array}{c} [\mathsf{End}_2(z_1, z_2) \ \wedge \ \mathsf{NC}(z_1) \ \wedge \ \mathsf{NC}(z_2) \\ \Rightarrow \\ \mathcal{O}(\mathsf{Start}_1(z_1) \vee \mathsf{Start}_2(z_1)) \ \wedge \ \mathcal{O}(\mathsf{Start}_1(z_2) \vee \mathsf{Start}_2(z_2))]. \end{array}\right\} \phi_2 \end{array} \right.$$

Or, le nombre de status events apparaissant dans $\neg\phi^{\mathsf{A}}_{\mathsf{Toy}}$ *est de 10, tandis que*

$$
\begin{aligned}
\mathsf{M}(\neg\phi^{\mathsf{A}}_{\mathsf{Toy}}) &= \mathsf{M}^{+}(\exists y_1.\ \exists y_2.\ \exists z_1.\ \exists z_2.\ (\neg\phi_1 \vee \neg\phi_2)) \\
&= \mathsf{M}^{+}(\neg\phi_1 \ \vee\ \neg\phi_2) \\
&= \max\{\mathsf{M}^{+}(\neg\phi_1), \mathsf{M}^{+}(\neg\phi_2)\} \\
&= \mathsf{M}^{+}(\neg\phi_1) && (\mathsf{M}(\neg\phi_1) = \mathsf{M}(\neg\phi_2)) \\
&= \mathsf{M}^{+}(\neg[\phi_3 \ \Rightarrow\ (\phi_4 \wedge \phi_5)]) \\
&= \mathsf{M}^{+}(\phi_3 \ \wedge\ \neg(\phi_4 \wedge \phi_5)) \\
&= \mathsf{M}^{+}(\phi_3 \ \wedge\ (\neg\phi_4 \vee \neg\phi_5)) \\
&= \mathsf{M}^{+}(\phi_3) \ +\ \mathsf{M}^{+}(\neg\phi_4 \vee \neg\phi_5)) \\
&= \mathsf{M}^{+}(\phi_3) \ +\ \max\{\mathsf{M}^{+}(\neg\phi_4), \mathsf{M}^{+}(\neg\phi_5)\} \\
&= \mathsf{M}^{+}(\phi_3) \ +\ \mathsf{M}^{+}(\neg\phi_4) && (\mathsf{M}^{+}(\neg\phi_4) = \mathsf{M}^{+}(\neg\phi_5)) \\
&= \mathsf{M}^{+}(\mathsf{End}_1(y_1, y_2) \wedge \mathsf{NC}(y_1) \wedge \mathsf{NC}(y_2)) \ +\ \mathsf{M}^{+}(\neg\phi_4) \\
&= \mathsf{M}^{+}(\mathsf{End}_1(y_1, y_2)) \ +\ \mathsf{M}^{+}(\mathsf{NC}(y_1)) \ +\ \mathsf{M}^{+}(\mathsf{NC}(y_2)) \ +\ \mathsf{M}^{+}(\neg\phi_4) \\
&= \mathsf{M}^{+}(\mathsf{End}_1(y_1, y_2)) \ +\ \mathsf{M}^{+}(\neg\phi_4) \\
&= 1 \ +\ \mathsf{M}^{+}(\neg\phi_4) \\
&= 1 \ +\ \mathsf{M}^{-}(\phi_4) \\
&= 1 \ +\ \mathsf{M}^{-}(\mathsf{true}\ \mathcal{S}\ (\mathsf{Start}_1(y_1) \vee \mathsf{Start}_2(y_1))) \\
&= 1 \ +\ \mathsf{M}^{-}(\mathsf{true}) \\
&= 1
\end{aligned}
$$

Ainsi, nous montrerons que pour trouver une attaque sur $\phi^{\mathsf{A}}_{\mathsf{Toy}}$*, il suffit de considérer une session par rôle, et non pas 10.*

6.3 Décidabilité

Cette section présente le résultat de décidabilité annoncé pour les classes de protocoles et de propriétés $\mathcal{C}_1$ et Φ_1 respectivement.

Définition 6.3.1. *Soient* Π *un protocole* T_0 *un ensemble de terme clos. Une exécution* exec *de* Π *révèle des clés long-terme de déchiffrement si et seulement si il existe une clé long-terme de déchiffrement* $k \in \mathcal{K}^1$ *telle que*

$$T_0 \cup \mathcal{N}_\epsilon(\Pi) \not\vdash k \qquad \textit{et} \qquad T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash k.$$

Nous énonçons à présent le résultat principal de ce chapitre.

Théorème 6.3.2. *Soient un protocole* $\Pi \in \mathcal{C}_1$*, une formule* $\phi \in \Phi_1$*, et* T_0 *un ensemble de termes atomiques clos. Si* Π *viole* ϕ *au regard de la connaissance initiale de l'intrus* T_0*, alors il existe une exécution valide de* Π *au regard de la connaissance initiale de l'intrus* T_0

- *qui viole* ϕ *en impliquant au plus* $\mathsf{M}(\phi)$ *sessions de chaque rôle de* Π*,*
- *ou bien, qui révèle une clé long-terme de déchiffrement en impliquant au plus une session de chaque rôle de* Π*.*

Il suffit donc de considérer $\max\{1, \mathsf{M}(\phi)\}$ sessions de chaque rôle pour décider si Π viole ϕ ou révèle une clé long-terme de déchiffrement.

La preuve de ce théorème fait appel à une notion supplémentaire, à savoir celle d'*exécution bien formée*, ainsi qu'à deux résultats intermédiaires que nous ne ferons qu'énoncer dans un premier temps, afin de procéder à la preuve du théorème 6.3.2 au plus vite. La preuve de ces deux propositions est remise à la section 6.4.

La définition d'une exécution bien formée nécessite de pouvoir accéder aux sous-termes, aux variables, aux nonces, ainsi qu'aux tags apparaissant dans une exécution et issus d'une session donnée. Nous étendons à cet effet les fonctions $\mathsf{St}()$, $\mathcal{V}()$, $\mathcal{N}()$ et $\mathsf{Tags}()$ de la façon suivante. Soient un protocole $\Pi \in \mathcal{C}_1$ à k rôles, *i.e.* $|\mathsf{Roles}(\Pi)| = k$, et $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ un scénario de Π, avec $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, et de trace symbolique associée tr. Soient aussi σ une substitution telle que tout sous-terme de $tr\sigma$ soit k-taggé, *i.e.* pour tout terme $t \in \mathsf{St}(tr\sigma)$, il existe un terme u et un k-tag τ tel que $t = \lfloor u \rfloor_\tau$.

L'ensemble de sous-termes issu de la session sid et apparaissant dans $tr' = tr\sigma$ est

$$\mathsf{St}(tr', sid) \stackrel{def}{=} \bigcup_{\substack{i \in [\![\ell]\!] \\ sid_i = sid}} \mathsf{St}(e_i\sigma).$$

L'ensemble de variables issues de la session sid et apparaissant dans $tr' = tr\sigma$ est

$$\mathcal{V}(tr', sid) \stackrel{def}{=} \bigcup_{\substack{i \in [\![\ell]\!] \\ sid_i = sid}} \mathcal{V}(e_i\sigma).$$

De même, l'ensemble de nonces issues de la session sid et apparaissant dans $tr' = tr\sigma$ est

$$\mathcal{N}(tr', sid) \stackrel{def}{=} \bigcup_{\substack{i \in [\![\ell]\!] \\ sid_i = sid}} \mathcal{N}(e_i\sigma),$$

et l'ensemble de tags issues de la session sid et apparaissant dans $tr' = tr\sigma$ est

$$\mathsf{Tags}(tr', sid) \stackrel{def}{=} \bigcup_{\substack{i \in [\![\ell]\!] \\ sid_i = sid}} \mathsf{Tags}(e_i\sigma).$$

Définition 6.3.3 (Exécution bien formée)**.** *Soient un protocole $\Pi \in \mathcal{C}_1$ à k rôles,* i.e. $|\mathsf{Roles}(\Pi)| = k$*, T_0 un ensemble de termes atomiques clos*[3]*, et* sc *un scénario de Π de trace symbolique associée tr. Soit aussi σ une substitution. La trace $tr\sigma$ est bien formée si elle vérifie :*

1. *pour tout $t \in \mathsf{St}(tr\sigma)$, il existe un terme $u \in \mathcal{T}$ et un k-tag τ tels que $t = \lfloor u \rfloor_\tau$,*
2. *pour toute session sid, $|\mathsf{Tags}(tr\sigma, sid)| \leq 1$,*
3. *pour toute session sid, $\mathsf{Tags}(tr\sigma, sid) = (\mathsf{Tags}(tr, sid))\sigma$,*

[3]Rappelons qu'un ensemble T de termes atomiques clos est tel que $T \subseteq \mathcal{P} \cup \mathcal{C} \cup \mathcal{N} \cup \mathcal{K}^{-1}$

4. *pour toutes sessions* sid_1 *et* sid_2*, si* $\mathsf{Tags}(tr\sigma, sid_1) \neq \mathsf{Tags}(tr\sigma, sid_2) = \emptyset$*, alors*

$$\mathcal{V}(tr\sigma, sid_1) \cap \mathcal{V}(tr\sigma, sid_2) = \emptyset \quad et \quad \mathcal{N}(tr\sigma, sid_1) \cap \mathcal{N}(tr\sigma, sid_2) = \emptyset,$$

5. *pour toutes sessions* sid_1 *et* sid_2*, si* $\mathsf{Tags}(tr\sigma, sid_1) = \emptyset$ *et* $sid_1 \not= sid_2$*, alors*

$$\mathcal{V}(tr\sigma, sid_1) \cap \mathcal{V}(tr\sigma, sid_2) = \emptyset \quad et \quad \mathcal{N}(tr\sigma, sid_1) \cap \mathcal{N}(tr\sigma, sid_2) = \emptyset.$$

Si de plus $tr\sigma$ *est une trace (respectivement une exécution) valide de* Π*, au regard de la connaissance initiale de l'intrus* T_0*, alors nous dirons que* $tr\sigma$ *est une trace (respectivement une exécution) valide de* Π *bien formée, au regard de la connaissance initiale de l'intrus* T_0*.*

En d'autres termes, $tr\sigma$ est bien formée si toute session sid est k-taggée dans $tr\sigma$ et ce avec le seul tag $(\mathsf{Tags}(tr, sid))\sigma$, et si toutes sessions sid_1 et sid_2 taggées différemment ne partagent ni nonces ni variables.

La première proposition dont nous aurons besoin établi que si un protocole Π de $\mathcal{C}_1$ viole une propriété ϕ de Φ_1, alors Π admet une attaque sur ϕ bien formée.

Proposition 6.3.4. *Soient un protocole* $\Pi \in \mathcal{C}_1$*, une formule* $\phi \in \Phi_1$*, et* T_0 *un ensemble de termes atomiques clos. Il existe un scénario* sc *de* Π*, et une substitution close* σ *tels que*

- $tr\sigma$ *soit une exécution valide de* Π*, au regard de la connaissance initiale de l'intrus* T_0*, et*
- $\langle \mathsf{exec}, T_0 \rangle \models \neg\phi$*,*

où tr *est la trace symbolique associée à* sc*, si et seulement si il existe une substitution* σ_{wf} *telle que*

- $tr\sigma_{\mathsf{wf}}$ *soit une exécution valide de* Π*, au regard de la connaissance initiale de l'intrus* T_0*,*
- $tr\sigma_{\mathsf{wf}}$ *soit bien formée,*
- *pour toute session* sid*,* $\mathsf{Tags}(tr\sigma_{\mathsf{wf}}, sid) = (\mathsf{Tags}(tr, sid))\sigma_{\mathsf{wf}}$*, et*
- $\langle tr\sigma_{\mathsf{wf}}, T_0 \rangle \models \neg\phi$*.*

La seconde proposition dont nous avons besoin établi qu'à partir d'une attaque bien formée il est possible d'en construire une autre qui ou bien révèle une clé long-terme de déchiffrement en impliquant au plus une session de chaque rôle, soit viole la propriété ϕ considérée mais implique tout au plus $\mathsf{M}(\phi)$ sessions de chaque rôle.

Proposition 6.3.5. *Soient* Π *un protocole dans* $\mathcal{C}_1$*, une formule* $\phi \in \Phi_1$*, et* T_0 *un ensemble de termes atomiques clos. Si* Π *admet une exécution valide au regard de la connaissance initiale de l'intrus* T_0*, bien formée et violant* ϕ*, alors* Π *admet une exécution valide au regard de la connaissance initiale de l'intrus* T_0

- *qui viole* ϕ *en impliquant au plus* $\mathsf{M}(\phi)$ *sessions de chaque rôle de* Π*,*
- *ou bien, qui révèle une clé long-terme de déchiffrement en impliquant au plus une session de chaque rôle de* Π*.*

Preuve du théorème 6.3.2 La preuve du théorème 6.3.2 consiste uniquement à mettre bout à bout les deux propositions énoncées ci-dessus.

Démonstration. Soient Π un protocole de $\mathcal{C}_1$, ϕ une propriété de Φ_1, et T_0 un ensemble de termes atomiques clos. Soit aussi exec une exécution valide de Π qui viole ϕ, au regard de la connaissance initiale de l'intrus T_0, *i.e.* $\langle \mathsf{exec}, T_0 \rangle \models \neg\phi$. D'après la proposition 6.3.4, il existe une exécution $\mathsf{exec}^{\mathsf{wf}}$ valide et bien formée de Π qui viole ϕ, au regard de la connaissance initiale de l'intrus T_0, *i.e.* $\langle \mathsf{exec}^{\mathsf{wf}}, T_0 \rangle \models \neg\phi$. De plus, Π, ϕ, T_0 et $\mathsf{exec}^{\mathsf{wf}}$ vérifient les hypothèses de la proposition 6.3.5 qui nous permet de conclure qu'il existe une exécution exec_{min} qui

- viole ϕ en impliquant au plus $\mathsf{M}(\phi)$ sessions de chaque rôle de Π,
- ou bien, révèle une clé long-terme de déchiffrement en impliquant au plus une session de chaque rôle de Π.

□

Il est possible de préciser le résultat énoncé au théorème 6.3.2 au prix de restrictions supplémentaires sur la classe des protocole considérés ; restrictions néanmoins satisfaites par la majorité des protocoles. La première condition consiste à interdire dans la spécification du protocole toute occurrence de clé long-terme de déchiffrement en position de plaintext. En d'autres termes, nous ne considérons que les protocoles dans lesquels les clés de déchiffrement long-terme n'apparaissent qu'en position de clé. La seconde restriction permet d'éliminer des protocoles qui sont de toutes façons non implantables et non exécutables. Cette condition exige que toute variable apparaissant dans une émission du protocole en position de plaintext, apparaisse préalablement en position de plaintext dans une réception.

Corollaire 6.3.6. *Soient ϕ une formule de Φ_1, et T_0 un ensemble de termes clos. Soit aussi $\Pi = [e_1; \ldots; e_\ell]$ un protocole de $\mathcal{C}_1$ vérifiant les trois conditions suivantes :*

1. *pour toute clé long terme de déchiffrement $k \in \mathcal{K}^{-1}$, $k \notin \mathsf{Plaintext}(\Pi)$,*
2. *pour tout $i \in [\![\ell]\!]$ et pour toute variable $x \in \mathcal{V}$, si $e_i = \mathsf{snd}(p_1, p_2, t)$ et $x \in \mathsf{Plaintext}(t)$ pour certains participants $p_1, p_2 \in \mathcal{P}$ et un certain terme $t \in \mathcal{T}$, alors il existe $j \in [\![i-1]\!]$ tel que $e_j = \mathsf{rcv}(p_1, p_3, u)$ et $x \in \mathsf{Plaintext}(u)$ pour un certain participant p_3 et certain terme u, et*
3. *pour tout $i \in [\![\ell]\!]$ et pour toute variable $x \in \mathcal{V}$, si $e_i = \mathsf{Q}(p_1, t_1, \ldots, t_n)$ et $x \in \mathsf{Plaintext}(t_i)$ pour un certain participants $p_1 \in \mathcal{P}$, certains termes $t_1, \ldots, t_n \in \mathcal{T}$, et un certain terme $i \in [\![n]\!]$, alors il existe $j \in [\![i-1]\!]$ tel que $e_j = \mathsf{rcv}(p_1, p_2, u)$ et $x \in \mathsf{Plaintext}(u)$ pour un certain participant p_2 et certain terme u.*

Π viole ϕ, au regard de la connaissance initiale de l'intrus T_0 si et seulement, Π admet une attaque sur ϕ impliquant au plus $\mathsf{M}(\phi)$ sessions de chaque rôle.

Démonstration. Nous allons montrer que Π ne révèle aucune clé long-terme de déchiffrement. Soit exec une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0. Il existe alors un scénario de Π, de trace symbolique associée tr, ainsi qu'une substitution σ tels que $\mathsf{exec} = tr\sigma$. Soit aussi une clé long-terme de déchiffrement $k \in \mathcal{K}^{-}1$ telle que

$$T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash k.$$

D'après le lemme 2.5.6, $k \in (T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon(\Pi) \cup \mathsf{Plaintext}(\mathsf{K}(\mathsf{exec})))$. Or, d'après le lemme 2.6.10 nous savons que

$$\mathsf{Plaintext}(\mathsf{K}(\mathsf{exec})) \subseteq \mathsf{Plaintext}(\mathsf{exec}) \subseteq (T_0 \cup \mathcal{C} \cup \mathcal{K}_\epsilon(\Pi) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{Plaintext}(tr)).$$

D'où

$$k \in (T_0 \cup \mathcal{C} \cup \mathcal{K}_\epsilon(\Pi) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{Plaintext}(tr)).$$

Mais par construction de Π nous savons que $k \notin \mathsf{Plaintext}(tr)$. Nous concluons donc que $k \in (T_0 \cup \mathcal{K}_\epsilon(\Pi))$, ce n'est donc pas au cours de l'exécution exec que k à été révélée.

En combinant ce qui vient d'être établi avec le théorème 6.3.2 nous déduisons donc que : Π viole ϕ, au regard de la connaissance initiale de l'intrus T_0 si et seulement, Π admet une attaque sur ϕ impliquant au plus $\mathsf{M}(\phi)$ sessions de chaque rôle. □

6.4 Preuves des résultats intermédiaires

Cette section est entièrement consacrée à la démonstration des deux propositions 6.3.4 et 6.3.5 sur lesquelles repose notre théorème 6.3.2.

6.4.1 Preuve de la proposition 6.3.4

Définitions et notations

Au cours de cette section, les termes seront annotés avec l'identificateur de session[4] de laquelle ils sont issus. Nous dirons qu'ils sont *traçables*. Ainsi, t^{sid}, avec $t \in \mathcal{T}$ et $sid \in \mathbb{N}$, dénote le fait que le terme t est issu de la session sid. Aussi, nous aurons besoin d'accéder aux variables, aux nonces, ainsi qu'aux tags d'un ensemble de termes traçables, issus d'une session donnée. Nous étendons à cet effet les fonctions $\mathcal{V}()$, $\mathcal{N}()$ et $\mathsf{Tags}()$ de la façons attendue :

$$\begin{aligned}
\mathcal{V}(T, sid) &\stackrel{def}{=} \bigcup_{t^{sid} \in T} \mathcal{V}(t) \\
\mathcal{N}(T, sid) &\stackrel{def}{=} \bigcup_{t^{sid} \in T} \mathcal{N}(t) \\
\mathsf{Tags}(T, sid) &\stackrel{def}{=} \bigcup_{t^{sid} \in T} \mathsf{Tags}(t)
\end{aligned}$$

où T est un ensemble de termes k-taggés et traçables pour un certain entier $k \in \mathbb{N}$, et $sid \in \mathbb{N}$ est un identificateur de session.

Définition 6.4.1 (Ensemble de termes bien formé)**.** *Soient k un entier et T un ensemble de termes k-taggés et traçables. Nous dirons que T est bien formé si :*

1. $\mathcal{V}(T, 0) = \emptyset$ *et* $\mathsf{Tags}(T, 0) = \emptyset$,
2. *pour toute session sid,* $|\mathsf{Tags}(T, sid)| \leq 1$, *et*

[4] A ne pas confondre identificateur de session, *i.e.* l'identificateur associé à une session dans un scénario, avec le tag de session établi à l'issue de la phase d'initialisation des protocoles de $\mathcal{C}_1$

3. *pour toutes sessions* $sid_1, sid_2 \in (\mathbb{N} \setminus \{0\})$, *si* $\mathsf{Tags}(T, sid_1) \neq \mathsf{Tags}(T, sid_2)$ *alors*

$$\mathcal{V}(T, sid_1) \cap \mathcal{V}(T, sid_2) = \emptyset \qquad et \qquad \mathcal{N}(T, sid_1) \cap \mathcal{N}(T, sid_2) = \emptyset.$$

4. *pour toutes sessions* $sid_1, sid_2 \in (\mathbb{N} \setminus \{0\})$, *si* $\mathsf{Tags}(T, sid_1) = \emptyset$ *et* $sid_1 \neq sid_2$, *alors*

$$\mathcal{V}(T, sid_1) \cap \mathcal{V}(T, sid_2) = \emptyset \qquad et \qquad \mathcal{N}(T, sid_1) \cap \mathcal{N}(T, sid_2) = \emptyset.$$

En d'autres termes, un ensemble de termes k-taggés et traçables est bien formé si chaque session est taggée de façon unique dans T, et si deux sessions k-taggées différemment ne partagent ni variables ni nonces dans T. L'identificateur de session 0 sera attribué, comme nous allons le voir dans un instant, aux termes de la connaissance initiale de l'intrus considérée. C'est d'ailleurs la raison pour laquelle nous avions interdit, à la définition 2.6.1 de numéroter par 0 une session dans un scénario.

Nous aurons aussi besoin de connaître la session d'origine des termes apparaissant dans un système de contraintes symboliques associé à une trace symbolique d'un protocole et un ensemble de termes clos. A cet effet, nous reformulons les définitions 2.6.7 et 5.2.2 de sorte à ce que cette information soit intégrée.

Définition 6.4.2 (Opérateur K')**.** *Soit* Π *un protocole,* sc *un scénario de* Π *d'entrelacement* $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, *et* $tr = [e_1; \ldots; e_\ell]$ *la trace symbolique associée à* sc. *La connaissance associée à cette trace est notée* $\mathsf{K}'(tr, \mathsf{interlvg})$ *et définie inductivement par :*

$$\mathsf{K}'(tr, \mathsf{interlvg}) = \begin{cases} [] & si\ \ell = 0 \\ \{m^{sid_1}\} \cup \mathsf{K}(tr', \mathsf{interlvg}') & si\ \ell > 2 \quad et\ e_1 = \mathsf{snd}(p, p', m), \\ \mathsf{K}(tr', \mathsf{interlvg}') & sinon \end{cases}$$

avec $tr' = [e_2; \ldots; e_\ell]$, $\mathsf{interlvg}' = [(r_2, sid_2); \ldots; (r_\ell, sid_\ell)]$, $p, p' \in \mathcal{P}$, *et* $m \in \mathcal{T}$.

Définition 6.4.3 ($\mathcal{C}'(tr, T_0)$)**.** *Soit* Π *un protocole,* T_0 *un ensemble de termes clos,* sc *un scénario de* Π *d'entrelacement* $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, *et* $tr = [e_1; \ldots; e_\ell]$ *la trace symbolique associée à* sc. *Le* système de contraintes symboliques associé à tr et T_0, *dénoté* $\mathcal{C}'(tr, T_0)$, *est défini par :*

$$\mathcal{C}'(tr, T_0) = \{T_0' \cup \mathcal{N}_\epsilon'(\Pi) \cup \mathsf{K}'(tr_i, \mathsf{interlvg}_i) \Vdash u^{sid_i} \mid e_i = \mathsf{rcv}(p_1, p_2, u), \\ i \in [\![\ell]\!],\ p_1, p_2 \in \mathcal{P}\}$$

où $T_0' = \bigcup_{t \in T_0} t^0$, *et* $\mathcal{N}_\epsilon'(\Pi) = \bigcup_{n \in \mathcal{N}_\epsilon(\Pi)} n^0$.

Remarquons que, comme nous l'avons annoncé à l'instant, nous avons annoté les termes de T_0 et les nonces de l'intrus avec 0.

De la même façon, nous aurons besoin de connaître la session et le prédicat d'origine des termes apparaissant dans une formule élémentaire associée à une

trace tr et une formule ϕ. A cet effet, nous associons à chaque sous-terme t de ϕ, un entier sid^t_ϕ tel que les identificateurs de sessions de tr et sid^t_ϕ soient distincts, et reformulons la définition 5.4.3 de manière à intégrer cette information.

Définition 6.4.4 (Traduction $\mathbf{T}'$.)**.** *Soit $\phi \in \mathcal{PS}$-LTL. Soient aussi Π un protocole,* sc *un scénario de Π d'entrelacement* $\mathsf{interlvg} = [(t_1, sid_1); \ldots; (r_\ell, sid_\ell)]$ *et de trace symbolique associée $tr = [e_1; \ldots; e_\ell]$, et T_0 un ensemble de termes clos. $\mathbf{T}'(\phi, tr, \mathsf{interlvg}, T_0)$ dénote la traduction de ϕ en une formule élémentaire « équivalente » et est définie comme suit :*

$$
\begin{aligned}
\mathbf{T}'(\mathsf{true}, tr, \mathsf{interlvg}, T_0) &\rightarrow \mathsf{true}^{sid^{\mathsf{true}}_\phi} \\
\mathbf{T}'(\mathsf{learn}(t), tr, \mathsf{interlvg}, T_0) &\rightarrow T'_0 \cup \mathcal{N}'_\epsilon(\Pi) \cup \mathsf{K}'(tr) \Vdash t^{sid^t_\phi} \\
\mathbf{T}'(\mathsf{Q}(t_1, \ldots, t_n), tr, \mathsf{interlvg}, T_0) &\rightarrow \begin{cases} t_1'^{sid_\ell} = t_1^{sid^{t_1}_\phi} \wedge \cdots \wedge t_n'^{sid_\ell} = t_n^{sid^{t_n}_\phi} & si\ \ell > 0 \\ \qquad\qquad et\ e_\ell = \mathsf{Q}(t'_1, \ldots, t'_n) & \\ \neg\mathsf{true}^{sid^{\mathsf{true}}_\phi} & sinon \end{cases} \\
\mathbf{T}'(\mathsf{C}(t), tr, \mathsf{interlvg}, T_0) &\rightarrow \epsilon^0 = t^{sid^t_\phi} \vee \bigvee_{\mathsf{pvk}(u) \in T_0} u^0 = t^{sid^t_\phi} \vee \bigvee_{\mathsf{shk}(u,t) \in T_0} \epsilon^0 \neq u^0 \\
\mathbf{T}'(\neg\phi, tr, \mathsf{interlvg}, T_0) &\rightarrow \neg\mathbf{T}'(\phi, tr, \mathsf{interlvg}, T_0) \\
\mathbf{T}'(\phi_1 \vee \phi_2, tr, \mathsf{interlvg}, T_0) &\rightarrow \mathbf{T}'(\phi_1, tr, \mathsf{interlvg}, T_0) \vee \mathbf{T}'(\phi_2, tr, \mathsf{interlvg}, T_0) \\
\mathbf{T}((\mathcal{Y}\phi, tr, \mathsf{interlvg}, T_0) &\rightarrow \begin{cases} \mathbf{T}'(\phi, tr, \mathsf{interlvg}, T_0) & si\ \ell > 0 \\ \neg\mathsf{true}^{sid^{\mathsf{true}}_\phi} & sinon \end{cases} \\
\mathbf{T}'(\phi_1 \mathcal{S} \phi_2, tr, \mathsf{interlvg}, T_0) &\rightarrow \begin{cases} \mathbf{T}'(\phi_2, tr, \mathsf{interlvg}, T_0) \vee & \\ \qquad (\mathbf{T}'(\phi_1 \mathcal{S} \phi_2, tr_{\ell-1}, \mathsf{interlvg}_{\ell-1}, T_0) & \\ \qquad \wedge \mathbf{T}'(\phi_1, tr, \mathsf{interlvg}, T_0)) & \\ \qquad\qquad si\ \ell > 0 & \\ \neg\mathsf{true}^{sid^{\mathsf{true}}_\phi} & sinon \end{cases} \\
\mathbf{T}'(\exists x.\ \phi, tr, \mathsf{interlvg}, T_0) &\rightarrow \exists x.\ \mathbf{T}'(\phi, tr, \mathsf{interlvg}, T_0)
\end{aligned}
$$

où $T'_0 = \bigcup_{t \in T_0} t^0$, *et* $\mathcal{N}'_\epsilon(\Pi) = \bigcup_{n \in \mathcal{N}_\epsilon(\Pi)} n^0$.

Ces annotations, n'influencent en aucun cas la procédure de résolution de contraintes ni la procédure de vérification $\mathbf{D}$. Nous aurions pu les introduire dès le chapitre 5. Il nous a néanmoins semblé qu'il serait mieux de retarder le plus possible l'introduction de ces annotations pour des raisons de lisibilité, d'autant plus que nous n'en ferons l'usage que dans cette section.

Lemmes intermédiaires

La preuve de la proposition 6.3.4 repose sur les 4 lemmes supplémentaires que nous énonçons et prouvons avant de procéder à la démonstration de la susdite proposition.

Ce premier lemme caractérise les tags apparaissant dans l'ensemble de termes obtenus en appliquant à un ensemble de termes bien formé la substitution résultant de l'unification de deux de ses sous-termes chiffrés.

Lemme 6.4.5. *Soit k un entier et T un ensemble de termes k-taggés et traçables bien formé. Soient aussi deux termes $u_1^{sid_1}, u_2^{sid_2} \in T$ et deux termes*

$v_1 \in \mathsf{Est}(u_1)$ et $v_2 \in \mathsf{Est}(u_2)$. Si v_1 et v_2 sont unifiables, posons $\sigma = \mathrm{mgu}(v_1, v_2)$, alors

1. $\mathsf{Tags}(T\sigma, sid_1) = \mathsf{Tags}(T\sigma, sid_2) = (\mathsf{Tags}(T, sid_1))\sigma = (\mathsf{Tags}(T, sid_2))\sigma$,
2. $|\mathsf{Tags}(T\sigma, sid_1)| = |\mathsf{Tags}(T\sigma, sid_2)| = 1$,
3. *pour toute variable* $x \in \mathrm{dom}(\sigma)$,

$$\mathsf{Tags}(\sigma(x)) \subseteq \mathsf{Tags}(T\sigma, sid_1) \ (= \mathsf{Tags}(T\sigma, sid_2)),$$

4. *pour toutes sessions* sid, sid',

$$\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid') \ \Rightarrow \ \mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid'),$$

5. *pour toutes sessions* sid,

$$\mathsf{Tags}(T\sigma, sid) = \emptyset \Rightarrow \ \mathsf{Tags}(T, sid) = \emptyset.$$

Démonstration. Remarquons que T étant bien formé, $v_1 \in \mathsf{Est}(u_1)$, et $v_2 \in \mathsf{Est}(u_2)$ impliquent $sid_1 \neq 0$ et $sid_2 \neq 0$; et procédons à la preuve de chacun des cinq points séparément.

1. Notons en premiers lieux que $v_1 \in \mathsf{Est}(u_1)$ et $v_2 \in \mathsf{Est}(u_2)$ impliquent que pour $f \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$, v_1 est de la forme $v_1 = f(\langle \tau_1, v_1' \rangle, w_2, \ldots, w_n)$ pour un certain k-tag τ_1 et certains termes $v_1', w_2, \ldots, w_n \in \mathcal{T}$, et v_2 de la forme $v_2 = f(\langle \tau_2, v_2' \rangle, w_2', \ldots, w_n')$ pour un certain k-tag τ_2 et certains termes $v_2', w_2', \ldots, w_n' \in \mathcal{T}$. Mais sachant que T est bien formé nous pouvons faire appel à la condition 2 de la définition 6.4.1 et conclure que $\mathsf{Tags}(T, sid_1) = \{\tau_1\}$ et que $\mathsf{Tags}(T, sid_2) = \{\tau_2\}$. De plus, $\sigma = \mathrm{mgu}(v_1, v_2)$ implique que $\tau_1\sigma = \tau_2\sigma$. D'où,

$$\begin{aligned}
\mathsf{Tags}(T\sigma, sid_1) &= \bigcup_{t^{sid_1} \in T\sigma} \mathsf{Tags}(t) \\
&= \bigcup_{t^{sid_1} \in T} \mathsf{Tags}(t\sigma) \\
&= \bigcup_{t^{sid_1} \in T} (\mathsf{Tags}(t))\sigma \ \cup \bigcup_{x \in (\mathcal{V}(T, sid_1) \cap \mathrm{dom}(\sigma))} \mathsf{Tags}(\sigma(x)) \\
&= \{\tau_1\sigma\} \ \cup \bigcup_{x \in (\mathcal{V}(T, sid_1) \cap \mathrm{dom}(\sigma))} \mathsf{Tags}(\sigma(x))
\end{aligned}$$

et de façon similaire

$$\mathsf{Tags}(T\sigma, sid_2) = \{\tau_2\sigma\} \ \cup \bigcup_{x \in (\mathcal{V}(T, sid_2) \cap \mathrm{dom}(\sigma))} \mathsf{Tags}(\sigma(x)).$$

Soit $x \in \mathrm{dom}(\sigma)$ et soit $t \in \mathsf{Est}(\sigma(x))$. D'après le lemme 5.1.4 nous savons qu'il existe un terme $t' \in \mathsf{Est}(\{v_1, v_2\})$ tel que $t = t'\sigma$. Supposons que $t' \in \mathsf{Est}(v_1)$ (le cas $t' \in v_2$ étant analogue nous ne le détaillerons pas ici). Sachant que T est bien formé et ayant établi que $\mathsf{Tags}(T, sid_1) = \{\tau_1\}$ nous déduisons que t' est de la forme $t' = g(\langle \tau_1, t_1 \rangle, t_2, \ldots, t_m)$ pour $g \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$, et certains termes $t_1, \ldots, t_m \in \mathcal{T}$. Mais alors $\mathsf{Tags}(\sigma(x)) \subseteq \{\tau_1\sigma\}$ et

$$\begin{aligned}
\mathsf{Tags}(T\sigma, sid_1) &= \{\tau_1\sigma\} \ \cup \bigcup_{x \in \mathrm{dom}(\sigma)} \mathsf{Tags}(\sigma(x)) \\
&= \{\tau_1\sigma\} \\
&= (\mathsf{Tags}(T, sid_1))\sigma.
\end{aligned}$$

De même,

$\mathsf{Tags}(T\sigma, sid_2) \quad = \quad (\mathsf{Tags}(T, sid_2))\sigma.$

Finalement, en faisant appel au fait établi $\tau_1\sigma = \tau_2\sigma$ nous déduisons que

$$\mathsf{Tags}(T\sigma, sid_1) = \mathsf{Tags}(T\sigma, sid_2) = (\mathsf{Tags}(T, sid_1))\sigma = (\mathsf{Tags}(T, sid_2))\sigma.$$

2. En combinant le fait établi en 1 avec le fait que T soit bien formé, et en particulier avec la condition 2 de la définition 6.4.1, nous déduisons

$$|\mathsf{Tags}(T\sigma, sid_1)| = |(\mathsf{Tags}(T, sid_1))\sigma| = 1,$$

et

$$|\mathsf{Tags}(T\sigma, sid_2)| = |(\mathsf{Tags}(T, sid_2))\sigma| = 1.$$

3. Etant donné que $\mathsf{Tags}(T\sigma, sid_1) = (\mathsf{Tags}(T, sid_1))\sigma \cup \bigcup_{x \in (\mathrm{dom}(\sigma) \cap \mathcal{V}(T, sid_1))} \mathsf{Tags}(\sigma(x))$, il est évident que pour toute variable $x \in \mathrm{dom}(\sigma)$,

$$\mathsf{Tags}(\sigma(x)) \subseteq \mathsf{Tags}(T\sigma, sid_1) \ (= \mathsf{Tags}(T\sigma, sid_2)).$$

4. Soient sid et sid' deux sessions telles que $\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid')$. Nous allons prouver ce point par contradiction, *i.e.* nous supposons que $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid')$. Nous distinguons deux cas.

   Cas $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_1)$ *et* $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_2)$. T étant bien formé nous pouvons faire appel aux conditions 1 et 3 de la définition 6.4.1 et déduire les quatre égalités suivantes :

$$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_1) = \emptyset \qquad \mathcal{V}(T, sid') \cap \mathcal{V}(T, sid_1) = \emptyset$$
$$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_2) = \emptyset \qquad \mathcal{V}(T, sid') \cap \mathcal{V}(T, sid_2) = \emptyset$$

   desquelles nous déduisons que

$$\mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma) = \emptyset \qquad \mathcal{V}(T, sid') \cap \mathrm{dom}(\sigma) = \emptyset.$$

   Mais alors nous nous trouvons face à la contradiction suivante :

$$\begin{array}{rcll} \mathsf{Tags}(T\sigma, sid) & = & (\mathsf{Tags}(T, sid))\sigma \ \cup \bigcup_{x \in (\mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma))} \mathsf{Tags}(\sigma(x)) & \\ & = & (\mathsf{Tags}(T, sid))\sigma & \mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma) = \emptyset \end{array}$$

   et de même

$$\begin{array}{rcll} \mathsf{Tags}(T\sigma, sid') & = & (\mathsf{Tags}(T, sid'))\sigma \ \cup \bigcup_{x \in (\mathcal{V}(T, sid') \cap \mathrm{dom}(\sigma))} \mathsf{Tags}(\sigma(x)) & \\ & = & (\mathsf{Tags}(T, sid'))\sigma & \mathcal{V}(T, sid') \cap \mathrm{dom}(\sigma) = \emptyset \end{array}$$

   sachant que $(\mathsf{Tags}(T, sid))\sigma = (\mathsf{Tags}(T, sid'))\sigma$. Nous concluons donc que

$$\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid').$$

   Cas $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid_1)$ *ou* $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid_2)$. Les deux cas étant analogues nous ne détaillerons ici que le premier. Il suffit pour conclure de noter la contradiction suivante

$$\begin{array}{rcll} \mathsf{Tags}(T\sigma, sid) & = & (\mathsf{Tags}(T, sid))\sigma \ \cup \bigcup_{x \in (\mathrm{dom}(\sigma) \cap \mathcal{V}(T, sid))} \mathsf{Tags}(\sigma(x)) & \\ & = & (\mathsf{Tags}(T, sid_1))\sigma \ \cup \bigcup_{x \in (\mathrm{dom}(\sigma) \cap \mathcal{V}(T, sid))} \mathsf{Tags}(\sigma(x)) & \text{hypothèse} \\ & = & (\mathsf{Tags}(T, sid_1))\sigma & \text{d'après 3} \end{array}$$

et

$$\begin{aligned}
\mathsf{Tags}(T\sigma, sid') &= (\mathsf{Tags}(T, sid'))\sigma \cup \bigcup_{x\in(\mathrm{dom}(\sigma)\cap\mathcal{V}(T,sid'))} \mathsf{Tags}(\sigma(x)) \\
&= (\mathsf{Tags}(T, sid))\sigma \cup \bigcup_{x\in(\mathrm{dom}(\sigma)\cap\mathcal{V}(T,sid'))} \mathsf{Tags}(\sigma(x)) && \text{hypothèse} \\
&= (\mathsf{Tags}(T, sid_1))\sigma \cup \bigcup_{x\in(\mathrm{dom}(\sigma)\cap\mathcal{V}(T,sid'))} \mathsf{Tags}(\sigma(x)) && \text{hypothèse} \\
&= (\mathsf{Tags}(T, sid_1))\sigma && \text{d'après 3.}
\end{aligned}$$

i.e. $\mathsf{Tags}(T\sigma, sid) = \mathsf{Tags}(T\sigma, sid')$. Nous concluons donc que

$$\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid').$$

5. Soit sid une session telle que $\mathsf{Tags}(T\sigma, sid) = \emptyset$. De 2 découle

$$\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid_1) \quad \text{et} \quad \mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid_2).$$

et d'après ce que nous venons de montrer en 4,

$$\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_1) \quad \text{et} \quad \mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T\sigma, sid_2).$$

T étant bien formé, nous pouvons alors faire appel aux conditions 1 et 3 de la définition 6.4.1 et déduire les deux égalités suivantes :

$$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_1) = \mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_2) = \emptyset$$

desquelles nous déduisons que

$$\mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma) = \emptyset.$$

Mais alors,

$$\begin{aligned}
\mathsf{Tags}(T\sigma, sid) &= (\mathsf{Tags}(T, sid))\sigma \cup \bigcup_{x\in(\mathcal{V}(T,sid)\cap\mathrm{dom}(\sigma))} \mathsf{Tags}(\sigma(x)) \\
&= (\mathsf{Tags}(T, sid))\sigma && \mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma) = \emptyset.
\end{aligned}$$

Nous concluons par hypothèse que $\mathsf{Tags}(T, sid) = \emptyset$.

Ayant démontré les 5 énoncés du lemme 6.4.5 notre preuve s'achève. □

Le lemme suivant stipule que l'ensemble de termes obtenu en appliquant à un ensemble de termes bien formé la substitution résultant de l'unification de deux de ses sous-termes chiffrés est bien formé.

Lemme 6.4.6. *Soient k un entier, et T un ensemble de termes k-taggés et traçables bien formé. Soient aussi deux termes $u_1^{sid_1}, u_2^{sid_2} \in T$ et deux termes $t_1 \in \mathsf{Est}(u_1)$ et $t_2 \in \mathsf{Est}(u_2)$. Si t_1 et t_2 sont unifiables avec $\sigma = \mathrm{mgu}(t_1, t_2)$, alors $T\sigma$ est un ensemble de termes k-taggés et traçables bien formé.*

Démonstration. Avant tout il nous faut établir que $T\sigma$ est un ensemble de termes k-taggés, *i.e.* que pour tout $v^{sid} \in T\sigma$ il existe un k-tag τ et un terme w tel que $v = \lfloor w \rfloor_\tau$. Cela revient à montrer qu'il existe un k-tag τ tel que pour tout $f(w_1, \ldots, w_n) \in \mathsf{Est}(v)$ avec $f \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$, w_1 est de la forme $w_1 = \langle \tau, w_1' \rangle$ pour un certain terme w_1'. Or, $v^{sid} \in T\sigma$ implique qu'il existe $v'^{sid} \in T$ tel que $v = v'\sigma$. Nous distinguons deux cas.

Cas $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_1)$ *et* $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_2)$.
Par hypothèses sur T nous savons que *(i)* v' est k-taggé, et que *(ii)* $\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_1) = \emptyset$ et $\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_2) = \emptyset$. De *(ii)* nous déduisons que $\mathcal{V}(v') \cap \mathrm{dom}(\sigma) = \emptyset$, et donc que $v = v'$. Finalement, en faisant appel à *(i)* nous déduisons que v est k-taggé.

Cas $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid_1)$ *ou* $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid_2)$.
Les deux cas étant analogues nous ne détaillons ici que le premier. D'après le lemme 6.4.5 (2), nous savons que $|\mathsf{Tags}(T\sigma, sid_1)| = |(\mathsf{Tags}(T, sid_1))\sigma| = 1$, et donc que $|\mathsf{Tags}(T, sid_1)| = |\mathsf{Tags}(T, sid)| = 1$. Posons $\{\varkappa\} = \mathsf{Tags}(T, sid)$. D'après le lemme 5.1.3, pour tout $f(w_1, \ldots, w_n) \in \mathsf{Est}(v'\sigma)$

$$f(w_1, \ldots, w_n) \in (\mathsf{Est}(v')\sigma) \ \vee \ \exists x \in \mathcal{V}(v').\ f(w_1, \ldots, w_n) \in \mathsf{Est}(\sigma(x)).$$

Dans le premier cas il existe $f(w'_1, \ldots, w'_n) \in \mathsf{Est}(v')$ tel que $f(w_1, \ldots, w_1) = f(w'_1\sigma, \ldots, w'_n\sigma)$. Or, $v'^{sid} \in T$ et T bien formé impliquent que w'_1 est de la forme $\langle \varkappa, w''_1 \rangle$ pour un certain terme w''_1, et donc $w_1 = \langle \varkappa\sigma, w''_1\sigma \rangle$.
Dans le second cas, nous savons d'après le lemme 5.1.4 qu'il existe un terme $f(w'_1, \ldots, w'_n) \in \mathsf{Est}(\{t_1, t_2\})$ tel que $f(w_1, \ldots, w_n) = f(w'_1\sigma, \ldots, w'_n\sigma)$. Or, $f(w'_1, \ldots, w'_n) \in \mathsf{Est}(\{t_1, t_2\}) \subseteq \mathsf{Est}(\{u_1, u_2\})$ et T bien formé impliquent que w'_1 est de la forme $w'_1 = \langle \varkappa', w''_1 \rangle$ pour un certain k-tag $\varkappa'$, et un certain terme w''_1, et donc $w_1 = \langle \varkappa'\sigma, w''_1\sigma \rangle$. Mais si $\varkappa' \in (\mathsf{Tags}(T, sid_1) \cup \mathsf{Tags}(T, sid_2))$, alors $\varkappa'\sigma \in \mathsf{Tags}(T\sigma, sid_1) \cup \mathsf{Tags}(T\sigma, sid_2)$; et d'après le lemme 6.4.5 (1)

$$\mathsf{Tags}(T\sigma, sid_1) = \mathsf{Tags}(T\sigma, sid_2) = (\mathsf{Tags}(T, sid_1))\sigma = (\mathsf{Tags}(T, sid_2))\sigma.$$

Rappelons maintenant que $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid_1) = \{\varkappa\}$ et que T est bien formé, et récapitulons

$$\begin{array}{lcll} \varkappa'\sigma & \in & \mathsf{Tags}(T\sigma, sid_1) \cup \mathsf{Tags}(T\sigma, sid_2) & \\ & = & \mathsf{Tags}(T\sigma, sid_1) & \text{lemme 6.4.5 (1)} \\ & = & (\mathsf{Tags}(T, sid_1))\sigma & \text{lemme 6.4.5 (1)} \\ & = & \{\varkappa\sigma\}. & \end{array}$$

Donc pour tout terme $v^{sid} \in T\sigma$, pour tout terme $f(w_1, \ldots, w_n) \in \mathsf{Est}(v)$, il existe $\tau = \varkappa$ avec $\{\varkappa\} = \mathsf{Tags}(T, sid)$ et il existe un terme w''_1 tels que $w_1 = \langle \varkappa\sigma, w''_1\sigma \rangle$, ce qui est précisément la définition d'un terme k-taggé.

Maintenant que nous avons établi que $T\sigma$ est un ensemble de termes k-taggés et traçables, il nous reste à prouver qu'il est bien formé. Nous procédons à la preuve de chacune des quatre conditions de la définition 6.4.1 séparément, et rappelons que d'après le lemme 6.4.5 (1) $\mathsf{Tags}(T\sigma, sid_1) = \mathsf{Tags}(T\sigma, sid_2)$.

1. Etant donné que $\mathcal{V}(T, 0) = \emptyset$, il est évident que $\mathcal{V}(T\sigma, 0) = \emptyset$ et que $\mathsf{Tags}(T\sigma, 0) = (\mathsf{Tags}(T, 0))\sigma$. Or $\mathsf{Tags}(T, 0) = \emptyset$, donc $\mathsf{Tags}(T\sigma, 0) = \emptyset$.
2. Soit un identificateur de session sid, montrons $|\mathsf{Tags}(T\sigma, sid)| \leq 1$. Pour ce faire, nous distinguons deux cas.

Cas $\mathsf{Tags}(T\sigma, sid) = \mathsf{Tags}(T\sigma, sid_1)$.
Le lemme 6.4.5 (2) nous permet de conclure immédiatement. En effet,

$$|\mathsf{Tags}(T\sigma, sid)| = |\mathsf{Tags}(T\sigma, sid_1)| = 1.$$

Cas $\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid_1)$.
D'après le lemme 6.4.5 (4) nous savons que $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_1)$, ainsi que $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_2)$; or, T étant bien formé nous savons d'après les conditions 1 et 3 de la définition 6.4.1 que $\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_1) = \emptyset$ et que $\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_2) = \emptyset$, ce qui implique que $\mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma) = \emptyset$. Mais alors pour tout terme $t^{sid} \in T\sigma$, $t^{sid} \in T$. Récapitulons à présent

$$|\mathsf{Tags}(T\sigma, sid)| = |\bigcup_{t^{sid} \in T\sigma} \mathsf{Tags}(t)| = |\bigcup_{t^{sid} \in T} \mathsf{Tags}(t)| = |\mathsf{Tags}(T, sid)| \leq 1.$$

3. Soient deux identificateurs de sessions $sid, sid' \in (\mathbb{N} \setminus \{0\})$, montrons que si sid et sid' sont k-taggées différemment dans $T\sigma$, *i.e.* $\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid')$, alors elles ne partagent ni variables ni nonces dans $T\sigma$. Pour ce faire, nous distinguons deux cas.

Cas $\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid_1)$ *et* $\mathsf{Tags}(T\sigma, sid') \neq \mathsf{Tags}(T\sigma, sid_1)$.
D'après le lemme 6.4.5 (1)

$$\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_1) \qquad \mathsf{Tags}(T, sid') \neq \mathsf{Tags}(T, sid_1)$$
$$\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_2) \qquad \mathsf{Tags}(T, sid') \neq \mathsf{Tags}(T, sid_2).$$

Mais nous avons déjà eu l'occasion de voir que de ces inégalités découle le fait que $\mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma) = \emptyset$, et de même que $\mathcal{V}(T, sid') \cap \mathrm{dom}(\sigma) = \emptyset$; ce qui à son tour implique que pour tout terme $t^{sid} \in T\sigma$, $t^{sid} \in T$, et de même que pour tout terme $u^{sid'} \in T\sigma$, $u^{sid'}$. Nous sommes donc en mesure de conclure aux deux égalités suivantes :

$$\begin{array}{lllll} \mathcal{V}(T\sigma, sid) & = \bigcup_{t^{sid} \in T\sigma} \mathcal{V}(t) & = \bigcup_{t^{sid} \in T} \mathcal{V}(t) & = \mathcal{V}(T, sid) \\ \mathcal{V}(T\sigma, sid') & = \bigcup_{u^{sid'} \in T\sigma} \mathcal{V}(u) & = \bigcup_{u^{sid'} \in T} \mathcal{V}(u) & = \mathcal{V}(T, sid') \end{array}$$

Il en va de même pour les nonces issus de sid et de sid' respectivement. Rappelons à présent que par hypothèse $\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid')$. En faisant à nouveau appel au lemme 6.4.5 (4) nous pouvons donc conclure que $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid')$. Mais par hypothèse sur T, et plus particulièrement grâce à la condition 3 de la définition 6.4.1, nous avons alors nécessairement :

$$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid') = \emptyset \qquad \mathcal{N}(T, sid) \cap \mathcal{N}(T, sid') = \emptyset.$$

Il ne nous reste plus qu'à résumer ce que nous venons de dire pour arriver aux conclusions visées, *i.e.*

$$\mathcal{V}(T\sigma, sid) \cap \mathcal{V}(T\sigma, sid') = \mathcal{V}(T, sid) \cap \mathcal{V}(T, sid') = \emptyset,$$

et

$$\mathcal{N}(T\sigma, sid) \cap \mathcal{N}(T\sigma, sid') = \mathcal{N}(T, sid) \cap \mathcal{N}(T, sid') = \emptyset.$$

Cas $\mathsf{Tags}(T\sigma, sid) = \mathsf{Tags}(T\sigma, sid_1)$ *ou* $\mathsf{Tags}(T, sid') = \mathsf{Tags}(T\sigma, sid_1)$. Les deux cas étant analogues, nous ne détaillerons ici que le premier, *i.e.* nous supposerons que $\mathsf{Tags}(T\sigma, sid) = \mathsf{Tags}(T\sigma, sid_1)$. Or,

$$\begin{aligned}\mathcal{V}(T\sigma, sid) &= \bigcup_{t^{sid} \in T\sigma} \mathcal{V}(t) \\ &\subseteq \bigcup_{t^{sid} \in T} \mathcal{V}(t) \cup \bigcup_{x \in \mathrm{dom}(\sigma)} \mathcal{V}(\sigma(x)) \\ &\subseteq \mathcal{V}(T, sid) \cup \mathcal{V}(T, sid_1) \cup \mathcal{V}(T, sid_2),\end{aligned}$$

et de même

$$\mathcal{N}(T\sigma, sid) \subseteq \mathcal{N}(T, sid) \cup \mathcal{N}(T, sid_1) \cup \mathcal{N}(T, sid_2).$$

Etant donné que $\mathsf{Tags}(T\sigma, sid) \neq \mathsf{Tags}(T\sigma, sid')$ et que $\mathsf{Tags}(T\sigma, sid) = \mathsf{Tags}(T\sigma, sid_1)$, nous pouvons établir d'après le lemme 6.4.5 (4) les trois inégalités suivantes :

$$\begin{aligned}\mathsf{Tags}(T, sid') &\neq \mathsf{Tags}(T, sid) \\ \mathsf{Tags}(T, sid') &\neq \mathsf{Tags}(T, sid_1) \\ \mathsf{Tags}(T, sid') &\neq \mathsf{Tags}(T, sid_2).\end{aligned}$$

Or T étant par hypothèse bien formé ces inégalités impliquent d'après la condition 3 de la définition 6.4.1 que :

(1)	$\mathcal{V}(T, sid') \cap \mathcal{V}(T, sid) = \emptyset$	(2)	$\mathcal{N}(T, sid') \cap \mathcal{N}(T, sid) = \emptyset$
(3)	$\mathcal{V}(T, sid') \cap \mathcal{V}(T, sid_1) = \emptyset$	(4)	$\mathcal{N}(T, sid') \cap \mathcal{N}(T, sid_1) = \emptyset$
(5)	$\mathcal{V}(T, sid') \cap \mathcal{V}(T, sid_2) = \emptyset$	(6)	$\mathcal{N}(T, sid') \cap \mathcal{N}(T, sid_2) = \emptyset$.

Des inégalités (3) et (5) il découle que $\mathcal{V}(T, sid') \cap \mathrm{dom}(\sigma) = \emptyset$, et donc que pour tout terme $u^{sid'} \in T\sigma$, $u^{sid'} \in T$. Ainsi,

$$\mathcal{V}(T\sigma, sid') = \bigcup_{u^{sid'} \in T\sigma} \mathcal{V}(u) = \bigcup_{u^{sid'} \in T} \mathcal{V}(u) = \mathcal{V}(T, sid')$$

et

$$\mathcal{N}(T\sigma, sid') = \bigcup_{u^{sid'} \in T\sigma} \mathcal{N}(u) = \bigcup_{u^{sid'} \in T} \mathcal{N}(u) = \mathcal{N}(T, sid').$$

En récapitulant tout ce qui vient d'être dit nous en arrivons donc aux conclusions escomptées, à savoir

$$\begin{aligned}\mathcal{V}(T\sigma, sid') \cap \mathcal{V}(T\sigma, sid) &= \mathcal{V}(T, sid') \cap \mathcal{V}(T\sigma, sid) \\ &\subseteq \mathcal{V}(T, sid') \cap (\mathcal{V}(T, sid) \cup \mathcal{V}(T, sid_1) \cup \mathcal{V}(T, sid_2)) \\ &= \emptyset \qquad \text{inégalités (1), (3) et (5),}\end{aligned}$$

et

$$\begin{aligned}\mathcal{N}(T\sigma, sid') \cap \mathcal{N}(T\sigma, sid) &= \mathcal{N}(T, sid') \cap \mathcal{N}(T\sigma, sid) \\ &\subseteq \mathcal{N}(T, sid') \cap (\mathcal{N}(T, sid) \cup \mathcal{N}(T, sid_1) \cup \mathcal{N}(T, sid_2)) \\ &= \emptyset \qquad \text{inégalités (2), (4) et (6),}\end{aligned}$$

4. Soient deux identificateurs de sessions $sid, sid' \in (\mathbb{N} \setminus \{0\})$ tels que $\mathsf{Tags}(T\sigma, sid) = \emptyset$ et $sid \neq sid'$. D'après le lemme 6.4.5 (5), nous savons que $\mathsf{Tags}(T, sid) = \emptyset$. T étant bien formé nous déduisons donc que :

$$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid') = \emptyset \qquad \text{et} \qquad \mathcal{N}(T, sid) \cap \mathcal{N}(T, sid') = \emptyset,$$

mais aussi que,

$$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_1) = \emptyset \qquad \text{et} \qquad \mathcal{N}(T, sid) \cap \mathcal{N}(T, sid_1) = \emptyset.$$

et,

$$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_2) = \emptyset \qquad \text{et} \qquad \mathcal{N}(T, sid) \cap \mathcal{N}(T, sid_2) = \emptyset.$$

D'où

$$\mathcal{V}(T\sigma, sid) = \bigcup_{t^{sid} \in T\sigma} \mathcal{V}(t) = \bigcup_{t^{sid} \in T} \mathcal{V}(t\sigma) = \bigcup_{t^{sid} \in T} \mathcal{V}(t) = \mathcal{V}(T, sid).$$

D'un autre coté,

$$\begin{aligned} \mathcal{V}(T\sigma, sid') &\subseteq \mathcal{V}(T, sid') \cup \mathcal{V}(T, sid_1) \cup \mathcal{V}(T, sid_2) \qquad \text{et} \\ \mathcal{N}(T\sigma, sid') &\subseteq \mathcal{N}(T, sid') \cup \mathcal{N}(T, sid_1) \cup \mathcal{N}(T, sid_2). \end{aligned}$$

Nous concluons donc que $\mathcal{V}(T\sigma, sid) \cap \mathcal{V}(T\sigma, sid') = \emptyset$ et de même que $\mathcal{N}(T, sid) \cap \mathcal{N}(T, sid') = \emptyset$.

Nous avons donc montré que l'ensemble $T\sigma$ est un ensemble de termes k-taggés et traçables qui vérifie les quatre conditions de la définition 6.4.1 ; $T\sigma$ est donc bien formé. □

Le prochain lemme stipule que substituer dans un ensemble de terme bien formé une variable par l'identité d'un participant maintient la propriété de « bonne formation ». Nous ne ferons qu'énoncer ce résultat, la preuve de ce dernier étant analogue à la preuve du précédent lemme.

Lemme 6.4.7. *Soit k un entier, et T un ensemble de termes k-taggés et traçables bien formé. Soit aussi une substitution σ. Si σ est de la forme $\sigma = \{x \mapsto t\}$ où t est un terme atomique clos,* i.e. *$t \in (\mathcal{P} \cup \mathcal{C} \cup \mathcal{N} \cup \mathcal{K}^{-}1)$, alors $T\sigma$ est un ensemble de termes k-taggés et traçables bien formé.*

D'après le lemme auquel nous nous attelons à présent, si les membres gauches et droits d'un système de contraintes forment un ensemble de termes bien formé, alors toute séquence de simplifications de ce système résulte en un système de contraintes dont les membres gauches et droits forment un ensemble de termes bien formé.

Lemme 6.4.8. *Soient k un entier, et T un ensemble de termes k-taggés et traçables bien formé. Soient aussi deux systèmes de contraintes C et D et une substitution σ, tels que* $\mathrm{lhs}(C) \subseteq T$, $\mathrm{rhs}(C) \subseteq \mathsf{St}(T)$, *et tels que D admette une solution. Si $C \rightsquigarrow_{\sigma}^{n} D$, alors*

- *$T\sigma$ est un ensemble de termes k-taggés et traçables bien formé,*
- *pour toute session sid,* $\mathsf{Tags}(T, sid) = (\mathsf{Tags}(T, sid))\sigma$, *et*
- $\mathrm{lhs}(D) \subseteq T\sigma$ *et* $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma)$.

Démonstration. Nous procédons par induction sur la longueur n de la dérivation.

Cas de base : $n = 0$. Alors $D = C$, $\sigma = \mathrm{id}$, et donc $t\sigma = T$. Nous concluons donc par hypothèses sur T et C.

Cas inductif : $n \geq 1$. Alors il existe un système de contraintes E, une règle R parmi les règles de simplification décrites à la figure 5.2, ainsi que deux substitutions σ_1 et σ_2 tels que

$$C \overset{\mathsf{R}}{\rightsquigarrow}_{\sigma_1} E \rightsquigarrow^{n-1}_{\sigma_2} D.$$

Pour pouvoir conclure par induction, il nous faut établir que $T\sigma_1$ est un ensemble de termes k-taggés et traçables bien formé, que $\mathsf{Tags}(T\sigma_1, sid) = (\mathsf{Tags}(T, sid))\sigma_1$ pour tout identificateur de session sid, que $\mathrm{lhs}(E) \subseteq T\sigma_1$ et que $\mathrm{rhs}(E) \subseteq \mathsf{St}(T\sigma_1)$. Pour ce faire nous procédons par analyse de cas sur la règle R impliquée dans le premier pas de notre dérivation.

Cas $\mathsf{R} = \mathsf{R}_1$. Il existe alors un système de contraintes C' ainsi qu'une contrainte $U \Vdash u^{sid}$ pour un certain sid tels que

$$C \;=\; (C' \wedge U \Vdash u^{sid}) \overset{\mathsf{R}_1}{\rightsquigarrow}_{\sigma_1} C' \;=\; E.$$

où $\sigma_1 = \mathrm{id}$. Il est alors aisé de conclure.

Cas $\mathsf{R} = \mathsf{R}_2$ *ou* $\mathsf{R} = \mathsf{R}_3$. Les deux cas étant analogues, nous ne détaillerons ici que le premier. Il existe alors un système de contraintes E, une contrainte $U \Vdash u_1^{sid_1}$, un terme traçable $u_2^{sid_2} \in U$, deux termes $v_1 \in \mathsf{Est}(u_1)$ et $v_2 \in \mathsf{Est}(u_2)$ tels que

$$C \;=\; C' \wedge U \Vdash u_1^{sid_1} \overset{\mathsf{R}_2}{\rightsquigarrow}_{\sigma_1} C\sigma_1 \;=\; E$$

où $\sigma_1 = \mathrm{mgu}(v_1, v_2)$.

- T, $u_1^{sid_1}$, $u_2^{sid_2}$, v_1 et v_2 vérifient les hypothèses du lemme 6.4.6, d'après lequel $T\sigma_1$ est un ensemble de termes k-taggés et traçables bien formé.
- Soit un identificateur de session sid. Afin de montrer que $\mathsf{Tags}(T\sigma_1, sid) = (\mathsf{Tags}(T, sid))\sigma_1$ nous distinguons deux cas.
  Cas $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid_1)$ *(ou* $\mathsf{Tags}(T, sid) = \mathsf{Tags}(T, sid_2)$*).* Rappelons que d'après le lemme 6.4.5 (1, 3) $\forall (x) \in \mathrm{dom}(\sigma_1)$, $\mathsf{Tags}(\sigma_1(x)) \subseteq \mathsf{Tags}(T\sigma_1, sid_1) = (\mathsf{Tags}(T, sid_1))\sigma_1$, d'où

  $$\begin{aligned} \mathsf{Tags}(T\sigma_1, sid) &= (\mathsf{Tags}(T, sid))\sigma_1 \cup \bigcup_{x \in (\mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma_1))} \mathsf{Tags}(\sigma_1(x)) \\ &= (\mathsf{Tags}(T, sid))\sigma_1 \end{aligned}$$

  Cas $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_1)$ *et* $\mathsf{Tags}(T, sid) \neq \mathsf{Tags}(T, sid_2)$. D'après les conditions 4 et 3 de la définition 6.4.1 nous avons alors les deux égalités suivantes :

  $$\mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_1) = \emptyset \qquad \text{et} \qquad \mathcal{V}(T, sid) \cap \mathcal{V}(T, sid_2) = \emptyset.$$

  Nous en déduisons donc que

  $$\mathcal{V}(T, sid) \cap \mathrm{dom}(\sigma) = \emptyset,$$

  d'où $\mathsf{Tags}(T\sigma_1, sid) = \mathsf{Tags}(T, sid)\sigma_1$.

– $\mathrm{lhs}(E) = \mathrm{lhs}(C\sigma_1) \subseteq T\sigma_1$ et $\mathrm{rhs}(E) = \mathrm{rhs}(C\sigma_1) \subseteq \mathsf{St}(T)\sigma_1 \subseteq \mathsf{St}(T\sigma_1)$.

Cas $\mathsf{R} = \mathsf{R}_4$. Il existe alors un système de contraintes C', une contrainte $U \Vdash u^{(r,sid)}$, deux termes $v_1^{(r_1,sid_1)} \in U$ et $v_2^{(r_2,sid_2)} \in (U \cup \{\mathsf{pvk}(\epsilon)\})$, ainsi que deux termes $\mathsf{aenc}(w_1, w_2) \in \mathsf{Est}(v_1)$ et $\mathsf{pvk}(w_3) \in \mathsf{Plaintext}(v_2)$ tels que $w_2 \neq w_3$ et

$$C \;=\; C' \wedge U \Vdash u^{(r,sid)} \;\overset{\mathsf{R}_4}{\rightsquigarrow}_{\sigma_1}\; C\sigma_1 \;=\; E$$

où $\sigma_1 = \mathrm{mgu}(w_2, w_3)$. Or, les clés privées considérées sont toutes de la forme $\mathsf{pvk}(p)$ avec $p \in \mathcal{P}$ (voir définition 2.3.1). Nous savons donc d'ores et déjà que $w_3 \in \mathcal{P}$, disons $w_3 = p$. De plus, $\mathrm{mgu}(w_2, w_3) \neq \bot$ et $w_2 \neq w_3$ impliquent que $w_2 \in \mathcal{V}$, disons $w_2 = x$, et donc que $\sigma_1 = \{x \mapsto p\}$.

– T et σ_1 vérifient les hypothèses du lemme 6.4.7 qui nous permet de conclure que $T\sigma_1$ est un ensemble de termes k-taggés et traçables bien formé.
– Soit sid' un identificateur de session.

$$\begin{aligned}
\mathsf{Tags}(T\sigma_1, sid') &= (\mathsf{Tags}(T, sid'))\sigma_1 \;\cup \bigcup_{y \in (\mathcal{V}(T,sid') \cap \mathrm{dom}(\sigma_1))} \mathsf{Tags}(\sigma_1(y)) \\
&= (\mathsf{Tags}(T, sid'))\sigma_1 \;\cup\; \mathsf{Tags}(p) && \mathrm{dom}(\sigma_1) = \{x\} \text{ et } \sigma_1(x) = p \\
&= (\mathsf{Tags}(T, sid'))\sigma_1 && \mathsf{Est}(p) = \emptyset \Rightarrow \mathsf{Tags}(p) = \emptyset
\end{aligned}$$

– $\mathrm{lhs}(E) = \mathrm{lhs}(C\sigma_1) \subseteq T\sigma_1$ et $\mathrm{rhs}(E) = \mathrm{rhs}(C\sigma_1) \subseteq \mathsf{St}(T)\sigma_1 \subseteq \mathsf{St}(T\sigma_1)$.

Cas $\mathsf{R} = \mathsf{R}_5$. Ce cas ne peut pas survenir. En effet, si $\mathsf{R} = \mathsf{R}_5$, alors $D = E = \bot$; ce qui vient contredire l'hypothèse faite selon laquelle D admet une solution, et selon laquelle $D \neq \bot$.

Cas $\mathsf{R} = \mathsf{R}_f$ *avec* $f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$. Il existe alors un système de contraintes C' et une contrainte $U \Vdash f(u_1, \ldots, u_n)^{sid}$ tels que

$$C \;=\; C' \wedge U \Vdash f(u_1, \ldots, u_n)^{sid} \;\overset{\mathsf{R}_f}{\rightsquigarrow}_{\sigma_1}\; C' \wedge U \Vdash u_1^{sid} \wedge \cdots \wedge U \Vdash u_n^{sid} \;=\; E$$

et $\sigma_1 = \mathrm{id}$.

– Sachant que T est par hypothèses bien formé, $T\sigma_1 = T$ est aussi un ensemble de termes k-taggés et traçables bien formé.
– Soit sid un identificateur de session, $\mathsf{Tags}(T\sigma_1, sid) = \mathsf{Tags}(T, sid) = (\mathsf{Tags}(T, sid))\sigma_1$.
– $\mathrm{lhs}(E) = \mathrm{lhs}(C) \subseteq T = T\sigma_1$ et $\mathrm{rhs}(E) = \mathrm{rhs}(C') \cup \{u_1, \ldots, u_n\}$. Or, $u_1, \ldots, u_n \in \mathsf{St}(f(u_1, \ldots, u_n)) \subseteq \mathsf{St}(\mathrm{rhs}(C)) \subseteq \mathsf{St}(\mathsf{St}(T)) = \mathsf{St}(T)$; donc $\mathrm{rhs}(E) \subseteq (\mathrm{rhs}(C') \cup \mathrm{rhs}(C)) \subseteq \mathrm{rhs}(C) \subseteq \mathsf{St}(T) = \mathsf{St}(T\sigma_1)$.

Ainsi, quelle que soit la règle R impliquée dans la dérivation $C \overset{\mathsf{R}}{\rightsquigarrow}_{\sigma_1} E$, nous avons montré que $T\sigma_1$ est un ensemble de termes k-taggés et traçables bien formé, que $\mathsf{Tags}(T\sigma_1, sid) = (\mathsf{Tags}(T, sid))\sigma_1$ pour tout identificateur de session sid, et que $\mathrm{lhs}(E) \subseteq T\sigma_1$ et $\mathrm{rhs}(E) \subseteq \mathsf{St}(T\sigma_1)$. Il suffit à présent de faire appel à notre hypothèse d'induction sur la dérivation $E \rightsquigarrow_{\sigma_2}^{n-1} D$ et l'ensemble de termes $T\sigma_1$ afin de conclure quant à D et $T\sigma_1\sigma_2$. □

Preuve de la proposition6.3.4

Nous en venons maintenant à la démonstration du résultat principal de cette section, à savoir que si $\Pi \in \mathcal{C}_1$ admet une attaque sur la propriété $\phi \in \Phi_1$, alors Π admet une attaque sur ϕ bien formée.

Proposition 6.3.4. *Soient un protocole $\Pi \in \mathcal{C}_1$, une formule $\phi \in \Phi_1$, et T_0 un ensemble de termes atomiques clos. Il existe un scénario sc de Π, et une substitution close σ tels que*

- *$tr\sigma$ soit une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0, et*
- *$\langle \mathsf{exec}, T_0 \rangle \models \neg\phi$,*

où tr est la trace symbolique associée à sc, si et seulement si il existe une substitution σ_{wf} telle que

- *$tr\sigma_{\mathsf{wf}}$ soit une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0,*
- *$tr\sigma_{\mathsf{wf}}$ soit bien formée,*
- *pour toute session sid, $\mathsf{Tags}(tr\sigma_{\mathsf{wf}}, sid) = (\mathsf{Tags}(tr, sid))\sigma_{\mathsf{wf}}$, et*
- *$\langle tr\sigma_{\mathsf{wf}}, T_0 \rangle \models \neg\phi$.*

Démonstration. L'une des implications est trivialement vraie. C'est à la preuve de l'autre sens de l'équivalence (*i.e.* de ($\Rightarrow$)) que nous nous attelons ici. Soient un protocole $\Pi \in \mathcal{C}_1$, un ensemble de termes atomiques clos $T_0 \subseteq (\mathcal{P} \cup \mathcal{C} \cup \mathcal{N} \cup \mathcal{K}^{-1})$, et une formule $\phi \in \Phi_1$. Soient aussi un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de Π, et une substitutions σ tels que

- $tr\sigma$ soit une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0, et
- $\langle tr\sigma, T_0 \rangle \models \neg\phi$

où tr est la trace symbolique associée à sc. En d'autres termes, soit $tr\sigma$ une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0, qui viole ϕ. Nous allons construire une substitution σ_{wf} telle que $tr\sigma_{\mathsf{wf}}$ soit une exécution valide bien formée de Π, au regard de la connaissance initiale de l'intrus T_0, violant ϕ.

Soit la formule élémentaire $\xi = \mathbf{T}'(\neg\phi, tr, \mathsf{interlvg}, T_0)$ avec $\mathbf{T}'$ telle que décrite à la section 6.4.1. D'après le lemme 5.4.4 nous savons que $\langle tr\sigma, T_0 \rangle \models \neg\phi$ si et seulement si $\sigma \models' \xi$. Nous savons aussi que ξ est de la forme $\xi = \exists x_1 \ldots \exists x_n.\ \xi'$, où ξ' est une formule élémentaire sans quantificateurs. Soit ψ la forme normale disjonctive associée à ξ', *i.e.* $\psi = \bigvee_{i \in [\![n]\!]} \psi_i$ où pour tout $i \in [\![n]\!]$, ψ_i est de la forme $\psi_i = C_i \wedge Eq_i \wedge Deq_i$ avec

- C_i un système de contraintes,
- Eq_i un ensemble d'égalités,
- Deq_i un ensemble de diségalités.

ψ et ξ' étant équivalentes, nous concluons que $\sigma \models' \exists x_1 \ldots \exists x_n.\ \xi'$ si et seulement si $\sigma \models' \exists x_1 \ldots \exists x_n.\ \psi$, et par là même que $\langle tr\sigma, T_0 \rangle \models \neg\phi$ si et seulement si $\sigma \models' \exists x_1 \ldots \exists x_n.\ \psi$.

Soit $C = \mathcal{C}'(tr, T_0)$ le système de contraintes associé à la trace symbolique tr et à la connaissance initiale de l'intrus T_0 tel que spécifié à la section 6.4.1.

D'après le lemme 5.5.3 de correction et de complétude de la procédure **D**, et étant donné que $\sigma \models' \exists x_1 \ldots \exists x_n.\ \psi$, nous déduisons qu'il existe un $i \in [\![n]\!]$ (posons i_{attack} un tel i), deux substitutions σ'_{wf} et σ''_{wf}, et deux systèmes de contraintes en forme résolue D et E tels que

- $C \rightsquigarrow^{\ell}_{\sigma_1} D$,
- $(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \rightsquigarrow^{h}_{\sigma_3} E$,
- $\sigma'_{\mathsf{wf}} = \sigma_1\sigma_2\sigma_3$,
- $\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}} \models' \exists x_1 \ldots \exists x_n.\ \psi$

où $\sigma_2 = \mathrm{mgu}((Eq_{i_{\mathsf{attack}}})\sigma_1)$. A nouveau en faisant appel au lemme 5.4.4 et au fait que ψ et ξ' sont équivalentes, nous déduisons que $\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}} \models' \neg\phi$. Posons

$$T = \mathrm{lhs}(C) \cup \mathrm{rhs}(C) \cup \bigcup_{t^{sid} = u^{sid^u_\phi} \in Eq_{i_{\mathsf{attack}}}} \{t^{sid} = u^{sid^u_\phi}\} \cup \mathrm{lhs}(C_{i_{\mathsf{attack}}} \cup \mathrm{rhs}(C_{i_{\mathsf{attack}}}).$$

En ayant annoté chaque variable de ϕ avec un « numéro de session » différent et les termes de T_0 (voir section 6.4.1), et par définition d'une trace symbolique, nous savons que T est un ensemble de termes k-taggés et traçables bien formé. Nous allons dans un premier temps montrer que $T\sigma'_{\mathsf{wf}}$ est un ensemble de termes k-taggés et traçables bien formé, et plus précisément nous allons montrer successivement que les ensembles $T\sigma_1$, $T\sigma_1\sigma_2$ puis $T\sigma_1\sigma_2\sigma_3$ sont des ensembles de termes k-taggés et traçables bien formés tels que pour toute session sid,

- $\mathsf{Tags}(T\sigma_1, sid) = (\mathsf{Tags}(T, sid))\sigma_1$,
- $\mathsf{Tags}(T\sigma_1\sigma_2, sid) = (\mathsf{Tags}(T, sid))\sigma_1\sigma_2$, et
- $\mathsf{Tags}(T\sigma_1\sigma_2\sigma_3, sid) = (\mathsf{Tags}(T, sid))\sigma_1\sigma_2\sigma_3$.

Puis nous construirons la substitution σ''_{wf} telle que $T\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}}$ aussi soit un ensemble de termes k-taggés et traçables bien formé tel que pour toute session sid, $\mathsf{Tags}(T\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}}, sid) = (\mathsf{Tags}(T, sid))\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}}$.

Par hypothèse C admet une solution et pas construction $\mathrm{lhs}(C) \subseteq T$ et $\mathrm{rhs}(C) \subseteq T \subseteq \mathsf{St}(T)$. De plus, nous venons de dire que T est bien formé. Donc C, D, T et σ_1 vérifient les hypothèses du lemme 6.4.8 qui nous permet de conclure que

- $T\sigma_1$ est un ensemble de termes k-taggés et traçables bien formé,
- pour toute session sid, $\mathsf{Tags}(T\sigma_1, sid) = (\mathsf{Tags}(T, sid))\sigma_1$, et
- $\mathrm{lhs}(D) \subseteq T\sigma_1$ et $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma_1)$.

A présent, rappelons que $\sigma_2 = \mathrm{mgu}((Eq_{i_{\mathsf{attack}}})\sigma_1)$ et posons $Eq_{i_{\mathsf{attack}}}\sigma_1 = \{t_j^{sid_{i_j}} = u_j^{sid^{u_j}_\phi}\}_{j \in [\![r]\!]}$. Donc $\sigma_2 = \theta_1 \ldots \theta_r$ avec pour tout $j \in [\![r]\!]$,

$$\theta_j = \mathrm{mgu}(t_j\theta_1 \ldots \theta_{j-1}, u_j\theta_1 \ldots \theta_{j-1}).$$

Mais d'après le lemme 5.4.6 pour tout $j \in [\![r]\!]$ $t_j \in \mathsf{St}(tr)$ et $u_j \in \mathsf{St}(\mathsf{SubForm}^+(\phi))$. Or nous avions imposé à la définition 6.2.1 que les sous-termes de ϕ soit atomiques et que chaque variable apparaissant négativement dans ϕ apparaisse sous au plus un prédicat. Ainsi d'après la transformation $\mathbf{T}'$ décrite à la section 6.4.1, si il existe $i, j \in [\![r]\!]$ tels que $t_i^{sid} = u_i^{sid^{u_i}_\phi}$ $t_j'^{sid'} = u_j^{sid^{u_j}_\phi}$ avec $u_i = u_j$ alors $sid = sid'$. Ainsi chaque variable apparaissant négativement dans ϕ, et donc positivement dans ϕ se voit unifiée par σ_2 avec la seule et même session de tr, préservant ainsi la bonne formation de $T\sigma_1$, *i.e.* $T\sigma_1\sigma_2$ est bien

formé. De plus, nous avons pris le soin de choisir des variables de ϕ disjointes de celles de tr ainsi les variables de ϕ n'apparaissent sous aucun sous-terme chiffré et donc dans aucun tag. Nous concluons donc que pour toute session sid, $\mathsf{Tags}(T\sigma_1\sigma_2, sid) = (\mathsf{Tags}(T\sigma_1, sid))\sigma_2 = (\mathsf{Tags}(T, sid))\sigma_1\sigma_2$.

Nous avons vu un peu plus haut dans la preuve que $\mathrm{lhs}(D) \subseteq T\sigma_1$, $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma_1)$. De plus, par construction $\mathrm{lhs}(C_{i_{\mathsf{attack}}}) \subseteq T$ et $\mathrm{rhs}(C_{i_{\mathsf{attack}}}) \subseteq T$. D'où

$$\mathrm{lhs}(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_1) = \mathrm{lhs}(D\sigma_2) \cup \mathrm{lhs}(C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \subseteq T\sigma_1\sigma_2$$

et

$$\begin{aligned} \mathrm{rhs}(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_1) &= \mathrm{rhs}(D\sigma_2) \cup \mathrm{rhs}(C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \\ &\subseteq \mathsf{St}(T\sigma_1)\sigma_2 \cup \mathsf{St}(T)\sigma_1\sigma_2 \\ &\subseteq \mathsf{St}(T\sigma_1\sigma_2). \end{aligned}$$

Et donc $T\sigma_1\sigma_2$, $(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_2)$, E et σ_3 vérifient les hypothèses du lemme 6.4.8 qui nous permet de conclure que

- $T\sigma_1\sigma_2\sigma_3$ est un ensemble de termes k-taggés et traçables bien formé,
- pour toute session sid, $\mathsf{Tags}(T\sigma_1\sigma_2\sigma_3, sid) = (\mathsf{Tags}(T, sid))\sigma_1\sigma_2\sigma_3$, et
- $\mathrm{lhs}(E) \subseteq T\sigma_1\sigma_2\sigma_3$ et $\mathrm{rhs}(E) \subseteq \mathsf{St}(T\sigma_1\sigma_2\sigma_3)$.

A ce stade nous avons montré que $\sigma'_{\mathsf{wf}} = \sigma_1\sigma_2\sigma_3$ est tel que $T\sigma'_{\mathsf{wf}}$ soit bien formé et pour toute session sid, $\mathsf{Tags}(T'_{\mathsf{wf}}\sigma'_{\mathsf{wf}}, sid) = (\mathsf{Tags}(T, sid))\sigma'_{\mathsf{wf}}$. Il nous reste donc à construire la substitution σ''_{wf}, avec pour seules contraintes :

1. pour toute contrainte $U \Vdash x^{sid} \in E$, $U \vdash \sigma''_{\mathsf{wf}}(x)$, et
2. pour toute diségalité $v \neq w \in (Deq_{i_{\mathsf{attack}}})\sigma'_{\mathsf{wf}}$, $v\sigma''_{\mathsf{wf}}\sigma''_{\mathsf{wf}} \neq w\sigma''_{\mathsf{wf}}$.

Posons $\mathrm{dom}(\sigma''_{\mathsf{wf}}) = \{x \mid U \Vdash x^{sid} \in E\}$. Il est évident que pour toute variable $x \in \mathrm{dom}(\sigma''_{\mathsf{wf}})$, il existe un agent $a \in \mathcal{A}$ tel que si $\sigma''_{\mathsf{wf}}(x) = a$ alors pour toute diségalité $v \neq w \in (Deq_{i_{\mathsf{attack}}})\sigma'_{\mathsf{wf}}$, $v\{x \mapsto a\} \neq w\{x \mapsto a\}$. Il suffit donc pour construire σ'', de choisir pour tout variable de son domaine un nom d'agent qui satisfasse les diségalités. Or, nous avons déjà eu l'occasion de voir que substituer une variable par un nom d'agent préserve la propriété de « bonne formation ». Ainsi construite σ''_{wf} implique donc que $T\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}}$ est un ensemble de termes k-taggés et traçables bien formé, et pour toute session sid, $\mathsf{Tags}(T\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}}, sid) = (\mathsf{Tags}(T, sid))\sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}}$.

Posons $\sigma_{\mathsf{wf}} = \sigma'_{\mathsf{wf}}\sigma''_{\mathsf{wf}}$. Nous avons donc comme annoncé plus haut $\sigma_{\mathsf{wf}} \models' \exists x_1 \ldots \exists x_n.\ \psi$ avec $T\sigma_{\mathsf{wf}}$ un ensemble de termes k-taggés et traçables bien formés tel que pour toute session sid, $\mathsf{Tags}(T\sigma_{\mathsf{wf}}, sid) = (\mathsf{Tags}(T, sid))\sigma_{\mathsf{wf}}$. Aussi, par correction et complétude de la procédure **D**, nous savons que $tr\sigma_{\mathsf{wf}}$ est une exécution valide de Π au regard de la connaissance initiale de l'intrus, et que $\langle tr\sigma_{\mathsf{wf}}, T_0\rangle \models \neg\phi$. Finalement, par construction de T nous avons $\mathsf{St}(tr\sigma_{\mathsf{wf}}) \subseteq \mathsf{St}(T\sigma_{\mathsf{wf}})$, et donc la trace $tr\sigma_{\mathsf{wf}}$ est une exécution valide de Π bien formée au regard de la connaissance initiale de l'intrus T_0 qui viole ϕ, *i.e.* $tr\sigma_{\mathsf{wf}}$ est une attaque bien formée sur la propriété ϕ. □

6.4.2 Preuve de la proposition 6.3.5

A la section 6.2 nous définissions la fonction $\mathsf{M}()$ sur les formules qui retourne le nombre de sessions de chaque rôle nécessaires pour monter une attaque. Dans cette section consacrée à la preuve de la proposition 6.3.5 nous

allons définir dans un premier temps quelles sont les sessions nécessaires d'une attaque, puis nous montrerons qu'il est possible de construire à partir d'une attaque, une attaque plus petite n'impliquant que ces sessions nécessaires.

Définitions et notations

Etant donnés une formule d'attaque ϕ et une exécution exec satisfaisant ϕ, la fonction $\mathsf{Witnesses}()$ retourne les sessions qui témoignent de l'attaque. Il y en a au plus $\mathsf{M}(\phi)$. La définition de $\mathsf{Witnesses}()$ repose sur les mêmes intuitions que celle de $\mathsf{M}()$.

Définition 6.4.9 (Sessions témoins, $\mathsf{Witnesses}()$)**.** *Soient* Π *un protocole,* ϕ *une formule close de* $\mathcal{PS}$*-LTL sans quantificateurs, et* T_0 *un ensemble de termes atomiques clos. Soient aussi* $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ *un scénario de* Π *de trace symbolique* tr*, avec* $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$*, et* σ *une substitution, tels que* $\mathsf{exec} = tr\sigma = [e_1; \ldots; e_\ell]$ *soit une exécution de valide de* Π*, au regard de la connaissance initiale de l'intrus* T_0 *qui satisfait* ϕ*,* i.e. $\langle \mathsf{exec}, T_0 \rangle \models \phi$*. Nous définissons l'ensemble* $\mathsf{Witnesses}(\mathsf{exec}, \phi)$ *de sessions témoins de l'attaque* exec *par*

$$\mathsf{Witnesses}(\mathsf{exec}, \phi) \stackrel{def}{=} \mathsf{Witnesses}^+(\mathsf{exec}, \phi) + \mathsf{LearnWitnesses}(\phi),$$

où

$$\mathsf{LearnWitnesses}(\phi) = \bigcup_{\substack{\mathsf{learn}(t) \in (\mathsf{SubForm}^+(\phi) \cup \mathsf{SubForm}^-(\phi)) \\ t \notin (\mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^{-1})}} \{sid_i \mid i = \max\{j \mid t \in \mathsf{St}(e_j)\}\},$$

et

$$\begin{array}{lcl}
\mathsf{Witnesses}^+(\mathsf{exec}, \mathsf{true}) & = & \emptyset \\
\mathsf{Witnesses}^+(\mathsf{exec}, \mathsf{Q}(t_1, \ldots, t_n)) & = & \{sid_\ell\} \\
\mathsf{Witnesses}^+(\mathsf{exec}, \mathsf{learn}(t)) & = & \emptyset \\
\mathsf{Witnesses}^+(\mathsf{exec}, \mathsf{C}(t)) & = & \emptyset \\
\mathsf{Witnesses}^+(\mathsf{exec}, \neg\phi) & = & \mathsf{Witnesses}^-(\mathsf{exec}, \phi) \\
\mathsf{Witnesses}^+(\mathsf{exec}, \phi_1 \vee \phi_2) & = & \begin{cases} \mathsf{Witnesses}^+(\mathsf{exec}, \phi_1) & si\ \langle \mathsf{exec}, T_0 \rangle \models \phi_1 \\ \mathsf{Witnesses}^+(\mathsf{exec}, \phi_2) & sinon \end{cases} \\
\mathsf{Witnesses}^+(\mathsf{exec}, \mathcal{Y}\phi) & = & \{sid_\ell\} \cup \mathsf{Witnesses}^+(\mathsf{exec}_{\ell-1}, \phi) \\
\mathsf{Witnesses}^+(\mathsf{exec}, \phi_1 \mathcal{S} \phi_2) & = & \mathsf{Witnesses}^+(\mathsf{exec}_i, \phi_2) \\
 & & \quad o\grave{u}\ i = \max\{j \in [\![\ell]\!] \mid \langle \mathsf{exec}_j, T_0 \rangle \models \phi_2\}
\end{array}$$

$$\begin{array}{lcl}
\mathsf{Witnesses}^-(\mathsf{exec}, \mathsf{true}) & = & \emptyset \\
\mathsf{Witnesses}^-(\mathsf{exec}, \mathsf{Q}(t_1, \ldots, t_n)) & = & \begin{cases} \{sid_\ell\} & si\ \ell > 0 \\ \emptyset & sinon \end{cases} \\
\mathsf{Witnesses}^-(\mathsf{exec}, \mathsf{learn}(t)) & = & \emptyset \\
\mathsf{Witnesses}^-(\mathsf{exec}, \mathsf{C}(t)) & = & \emptyset \\
\mathsf{Witnesses}^-(\mathsf{exec}, \neg\phi) & = & \mathsf{Witnesses}^+(\mathsf{exec}, \phi) \\
\mathsf{Witnesses}^-(\mathsf{exec}, \phi_1 \vee \phi_2) & = & \mathsf{Witnesses}^-(\mathsf{exec}, \phi_1) \cup \mathsf{Witnesses}^-(\mathsf{exec}, \phi_2) \\
\mathsf{Witnesses}^-(\mathsf{exec}, \mathcal{Y}\phi) & = & \begin{cases} \{sid_\ell\} \cup \mathsf{Witnesses}^-(\mathsf{exec}_{\ell-1}, \phi) & si\ \ell > 0 \\ \emptyset & sinon \end{cases} \\
\mathsf{Witnesses}^-(\mathsf{exec}, \phi_1 \mathcal{S} \phi_2) & = & \begin{cases} \emptyset & si\ \forall i \in [\![\ell]\!].\ \langle \mathsf{exec}_i, T_0 \rangle \models \neg\phi_2 \\ \mathsf{Witnesses}^-(\mathsf{exec}_i, \phi_1) & sinon \\ \quad o\grave{u}\ i = \max\{j \in [\![\ell]\!] \mid \langle \mathsf{exec}_j, T_0 \rangle \models \neg\phi_1\} & \end{cases}
\end{array}$$

Définition 6.4.10. *Soient* Π *un protocole,* $\mathsf{sc} = (\mathsf{interlvg}, initagts)$ *d'entrelacement* $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$ *et de trace symbolique associée* $tr = [e_1; \ldots; e_\ell]$*. Soient aussi* σ *une substitution et* S *un ensemble de sessions. La restriction de* $tr\sigma$ *aux sessions de* S *est définie*

$$tr\sigma|_S = \begin{cases} [] & si\ \ell = 0 \\ [e_1\sigma]@(tr'\sigma|_S) & si\ sid_1 \in S \\ (tr'\sigma|_S) & sinon \end{cases}$$

où $tr' = [e_2; \ldots; e_\ell]$*.*

Lemmes intermédiaires

Le premier lemme stipule, comme nous l'annoncions, que la fonction $\mathsf{M}()$ majore le nombre de sessions retournées par la fonction $\mathsf{Witnesses}()$ pour une même formule donnée. Ceci n'a rien de surprenant, étant donné que la définition de la première est calquée sur celle de la seconde. D'ailleurs, nous ne donnerons pas la preuve de ce lemme ici précisément pour cette raison.

Lemme 6.4.11. *Soient* Π *un protocole,* ϕ *une formule close de* $\mathcal{PS}$*-LTL sans quantificateurs, et* T_0 *un ensemble de termes atomiques clos. Soient aussi* $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ *un scénario de* Π *de trace symbolique* tr*, avec* $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$*, et* σ *une substitution, tels que* $\mathsf{exec} = tr\sigma$ *soit une exécution valide de* Π*, au regard de la connaissance initiale de l'intrus* T_0 *qui satisfait* ϕ*,* i.e. $\langle \mathsf{exec}, T_0 \rangle \models \phi$*.*

$$|\mathsf{Witnesses}(\mathsf{exec}, \phi)| \leq \mathsf{M}(\phi)$$

Lemme 6.4.12. *Soient un protocole* $\Pi \in \mathcal{C}_1$*, et* T_0 *un ensemble de termes atomiques clos. Soient aussi* exec *une exécution valide et bien formée de* Π*, au regard de la connaissance initiale de l'intrus* T_0*, qui ne révèle aucune clé long-terme de déchiffrement. Soit un ensemble de sessions vérifiant*

> *pour toutes sessions* sid_1 *et* sid_2 *telles que* $\mathsf{Tags}(\mathsf{exec}, sid_1) \neq \emptyset$ *et* $\mathsf{Tags}(\mathsf{exec}, sid_1) = \mathsf{Tags}(\mathsf{exec}, sid_2)$*,* $sid_1 \in S$ *si et seulement si* $sid_2 \in S$*.*

Pour toute session $sid \in S$ *et tout terme* $t \in \mathsf{St}(\mathsf{exec}, sid)$*, si* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t$*, alors* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_\mathsf{S}) \vdash t$*.*

Démonstration. Posons $\mathsf{exec} = [e_1; \ldots; e_\ell]$. Etant donné que exec est une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0, il existe par définition un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de Π, avec un certain entrelacement $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, ainsi qu'une substitution close σ tels que $\mathsf{exec} = tr\sigma$, où tr est la trace symbolique associée à sc.

Considérons dans un premier temps le cas où $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) = \emptyset$. Dans ce cas t n'admet pas de sous-terme chiffré, *i.e.* $\mathsf{Est}(t) = \emptyset$. Supposons le contraire, *i.e.* supposons qu'il existe $f(t_1, \ldots, t_n) \in \mathsf{Est}(t)$. exec étant bien formée nous savons que tout sous-terme chiffré de exec est k-taggé, et en particulier donc $f(t_1, \ldots, t_n)$. Il existe donc un k-tag τ tel que t_1 soit de la forme $\langle \tau, t'_1 \rangle$ pour un

certain terme t'_1. Mais alors par définition de la fonction $\mathsf{Tags}()$ nous savons que $\tau \in \mathsf{Tags}(\mathsf{exec}, sid')$. De plus, la condition 3 de la définition 6.3.3 d'exécution bien formée, implique qu'il existe un k-tag $\tau' \in \mathsf{Tags}(tr, sid')$ tel que $\tau = \tau'\sigma$. Or, nous savons par construction de Π et par définition de tr que tout k-tag apparaissant dans tr contient au moins un nonce non dans $\mathcal{N}_\epsilon(\Pi)$, et donc en particulier $(\mathcal{N}(\tau') \setminus \mathcal{N}_\epsilon(\Pi)) \neq \emptyset$. Mais ceci vient contredire l'hypothèse faite selon laquelle $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) = \emptyset$, et ce car $\mathcal{N}(\tau') \subseteq \mathcal{N}(\tau'\sigma) = \mathcal{N}(\tau) \subseteq \mathcal{N}(t)$.

Nous concluons donc que t est un n-uplet. Remarquons aussi que pour toute clé long-terme de déchiffrement $k \in \mathcal{K}^{-1}$, si $k \in \mathsf{St}(t)$, alors $T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash k$. En effet, dans le cas contraire, sachant que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t$, nous serions en situation de contradiction avec l'hypothèse faite selon laquelle exec ne révèle aucune clé long-terme de déchiffrement.

A ce stade nous savons que t est un n-uplet dont les parties atomiques sont toutes dérivables par l'intrus à partir de T_0 et de $\mathcal{N}_\epsilon(\Pi)$. Nous concluons donc que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash t$, et donc que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t$.

Plaçons nous à présent dans le cas où $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) \neq \emptyset$. Nous procédons par induction sur la taille de la preuve minimale π témoignant de $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t$, et distinguons quatre cas en fonction de la dernière règle appliquée dans π.

Cas R=*(A).*
Dans ce cas nous savons que $t \in T_0 \cup \mathsf{K}(\mathsf{exec})$. Si $t \in T_0$, il est évident que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t$. Si $t \in \mathsf{K}(\mathsf{exec})$, alors il existe une session sid'' telle que $t \in \mathsf{K}(\mathsf{exec}|_{sid''})$. Or, nous avons supposé que $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) \neq \emptyset$; donc $\mathcal{N}(\mathsf{exec}, sid') \cap \mathcal{N}(\mathsf{exec}, sid'') \neq \emptyset$. Mais comme exec est bien formée, nous concluons que $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et donc $sid'' \in S$. Ainsi nous savons que $t \in \mathsf{K}(\mathsf{exec}|_S)$ et concluons finalement que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t$.

Cas R=*(C) ou* R=*(Ltk).*
Ce cas ne peut pas survenir car nous avons supposé que $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) \neq \emptyset$.

Cas R *est une règle de composition.*
Dans ce cas, notre arbre de dérivation est de la forme

$$\dfrac{\dfrac{\pi_1}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_1} \quad \ldots \quad \dfrac{\pi_n}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_n}}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t}\ (\mathsf{R})$$

pour certains messages $t_1, \ldots, t_n \in \mathcal{M}$. Pour chacun des t_i (avec $i \in [\![n]\!]$) il nous faut établir que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t_i$. Soit $i \in [\![n]\!]$. Etant donné que $t_i \in \mathsf{St}(t) \subseteq \mathsf{St}(\mathsf{exec})$, le cas où $(\mathcal{N}(t_i) \setminus \mathcal{N}_\epsilon(\Pi)) = \emptyset$ peut être traité de manière analogue au cas $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) = \emptyset$. Nous concluons donc que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t_i$.

Plaçons nous à présent dans le cas où $(\mathcal{N}(t_i) \setminus \mathcal{N}_\epsilon(\Pi)) \neq \emptyset$. Aux vues des règles de composition, nous savons que $t_i \in \mathsf{St}(t) \subseteq \mathsf{St}(\mathsf{exec}, sid')$. Nous pouvons donc faire appel à notre hypothèse d'induction et conclure que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_W) \vdash t_i$.

Cas R *est une règle de décomposition.*

Dans ce cas, notre arbre de dérivation est de la forme

$$\dfrac{\dfrac{\pi_1}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_1} \quad \ldots \quad \dfrac{\pi_n}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_n}}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t}\ (\mathsf{R})$$

pour certains messages $t_1, \ldots, t_n \in \mathcal{M}$, avec $n \in \{1, 2\}$. Pour chacun des t_i (avec $i \in [\![n]\!]$) il nous faut établir que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{\mathsf{S}}) \vdash t_i$. Ayant supposé que π est minimale, nous pouvons faire appel au lemme 2.5.5 et conclure que pour tout $i \in [\![n]\!]$ $t_i \in (\mathsf{St}(\mathsf{K}(\mathsf{exec}))T_0 \cup \cup\mathcal{N}_\epsilon(\Pi) \cup \cup\mathcal{C}\mathcal{K}_\epsilon(\Pi))$. Ainsi pour tout $i \in [\![n]\!]$. Si $t_i \in (T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$ nous concluons immédiatement. Le cas $t_i \in \mathsf{St}(\mathsf{K}(\mathsf{exec}))$ et $(\mathcal{N}(t_i) \setminus \mathcal{N}_\epsilon(\Pi)) = \emptyset$ peut être traité de manière analogue au cas $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) = \emptyset$. Nous concluons donc dans ce cas que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_W) \vdash t_i$.

Plaçons nous à présent dans le cas où $(\mathcal{N}(t_i) \setminus \mathcal{N}_\epsilon(\Pi)) \neq \emptyset$ et $t_i \in \mathsf{St}(\mathsf{K}(\mathsf{exec}),$ alors il existe une session sid'' telle que $t_i \in \mathsf{St}(\mathsf{exec}, sid'')$. Or, nous avons supposé que $(\mathcal{N}(t) \setminus \mathcal{N}_\epsilon(\Pi)) \neq \emptyset$; comme exec est bien formée, et que $\mathcal{N}(\mathsf{exec}, sid') = \mathcal{N}(\mathsf{exec}, sid'') \neq \emptyset$, nous concluons que $\mathsf{Tags}(\mathsf{exec}, sid') \cap \mathsf{Tags}(\mathsf{exec}, sid'')$. Mais alors par construction de S, $sid'' \in S$, et donc en faisant appel à notre hypothèse d'induction nous pouvons conclure que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t_i$. □

Lemme 6.4.13. *Soient un protocole $\Pi \in \mathcal{C}_1$, un ensemble de termes atomiques clos T_0. Soit aussi exec une exécution valide et bien formée de Π, au regard de la connaissance initiale de l'intrus T_0, qui ne révèle aucune clé long-terme de déchiffrement. Pour tout ensemble de sessions S vérifiant :*

> *pour toutes sessions sid_1 et sid_2 telles que $\mathsf{Tags}(\mathsf{exec}, sid_1) \neq \emptyset$ et $\mathsf{Tags}(\mathsf{exec}, sid_1) = \mathsf{Tags}(\mathsf{exec}, sid_2)$, $sid_1 \in S$ si et seulement si $sid_2 \in S$.*

$\mathsf{exec}|_S$ aussi est une exécution valide et bien formée de Π, au regard de la connaissance initiale de l'intrus T_0.

Démonstration. Posons $\mathsf{exec} = [e_1; \ldots; e_\ell]$. Etant donné que exec est une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0, il existe par définition un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de Π, avec un certain entrelacement $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, ainsi qu'une substitution close σ tels que $\mathsf{exec} = tr\sigma$, où tr est la trace symbolique associée à sc.

Posons $1 \leq i_1 < \cdots < i_n \leq \ell$ tels que $\mathsf{exec}|_S = [e_{i_1}; \ldots; e_{i_n}]$. Nous procédons par induction sur la longueur n de $\mathsf{exec}|_S$.

Cas de base : $n = 0$. Par définition $\mathsf{exec}|_S = [\,]$ est une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0.

Cas inductif : $n \geq 1$. Par hypothèse d'induction, nous savons que $[e_{i_1}; \ldots; e_{i_{n-1}}]$ est une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0. Si e_{i_n} est une émission ou un status event, alors $\mathsf{exec}|_S = [e_{i_1}; \ldots; e_{i_{n-1}}; e_{i_n}]$ est par définition une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0. Si par contre e_{i_n} est une réception, *i.e.* $\exists p_1, p_2 \in \mathcal{P}$. $\exists m \in \mathcal{M}$. $e_{i_n} = \mathsf{rcv}(p_1, p_2, m)$, il faut nous assurer que

$$T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}([e_{i_1}; \ldots; e_{i_{(n-1)}}]) \vdash m,$$

sachant que

$$T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}([e_1; \ldots; e_{(i_n)-1}]) \vdash m.$$

Or, d'après le lemme 6.4.12

$$T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_\mathsf{S}) \vdash m$$

Il suffit à présent de noter que par définition de l'opérateur $\mathsf{K}()$, e_{i_n} étant un événement de réception, $\mathsf{K}([e_{i_1}; \ldots; e_{i_{(n-1)}}]) = \mathsf{K}([e_{i_1}; \ldots; e_{i_n}]) = \mathsf{K}(\mathsf{exec}|_S)$. Ainsi nous déduisons

$$T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}([e_{i_1}; \ldots; e_{i_{(n-1)}}]) \vdash m.$$

Nous avons donc montré que $\mathsf{exec}|_S$ est une exécution valide de Π, au regard de la connaissance initiale de l'intrus. De plus, exec étant bien formée, il est évident que $\mathsf{exec}|_S$ l'est aussi. □

Lemme 6.4.14. *Soient un protocole $\Pi \in \mathcal{C}_1$, un ensemble de termes atomiques clos T_0, et une formule sans quantificateurs ϕ dans $\mathcal{PS}$-LTL telles qu'aucune modalité ni aucun $\mathsf{learn}(t)$ n'apparaisse sous la modalité $\mathcal{S}$ dans ϕ. Soit aussi exec une exécution bien formée de Π, qui ne révèle aucune clé de déchiffrement long-terme, et telle que pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}(\phi)$[5], $t \in (\mathsf{St}(\mathsf{exec}) \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^{-1})$.*

Si exec satisfait ϕ au regard de T_0 i.e. $\langle \mathsf{exec}, T_0 \rangle \models \phi$, *alors pour tout ensemble de sessions S vérifiant :*

- *pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,*
- *$\mathsf{Witnesses}^+(\mathsf{exec}, \phi) \subseteq S$, et*
- *pour toutes sessions sid' et sid''* t.q. $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ *et* $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, *$sid' \in S$ si et seulement si $sid'' \in S$,*

$\mathsf{exec}|_S$ *aussi est une exécution de Π qui satisfait ϕ,* i.e. $\langle \mathsf{exec}|_S, T_0 \rangle \models \phi$.

Si exec viole ϕ au regard de T_0, i.e. $\langle \mathsf{exec}, T_0 \rangle \models \neg\phi$, *alors pour tout ensemble de sessions S vérifiant :*

- *pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)$, si $t \notin \mathcal{P}$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,*
- *$\mathsf{Witnesses}^-(\mathsf{exec}, \phi) \subseteq S$, et*
- *pour toutes sessions sid' et sid''* t.q. $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ *et* $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, *$sid' \in S$ si et seulement si $sid'' \in S$,*

$\mathsf{exec}|_S$ *aussi est une exécution de Π qui viole ϕ,* i.e. $\langle \mathsf{exec}|_S, T_0 \rangle \models \neg\phi$.

Démonstration. Posons $\mathsf{exec} = [e_1; \ldots; e_\ell]$. Etant donné que exec est une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0, il existe par définition un scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de Π, avec un certain entrelacement $\mathsf{interlvg} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$, ainsi qu'une substitution close σ tels que $\mathsf{exec} = tr\sigma$, où tr est la trace symbolique associée à sc. Nous procédons par induction sur la taille de ϕ.

Cas $\phi = \mathsf{true}$.

[5] $\mathsf{SubForm}(\phi) = \mathsf{SubForm}^+(\phi) \cup \mathsf{SubForm}^-(\phi)$.

Dans ce cas, par définition $\langle \mathsf{exec}, T_0\rangle \models \phi$ et pour tout ensemble de sessions S, $\langle \mathsf{exec}|_S, T_0\rangle \models \phi$.

Cas $\phi = \mathsf{Q}(t_1, \ldots, t_n)$.
Si $\langle \mathsf{exec}, T_0\rangle \models \phi$, alors $e_\ell = \mathsf{Q}(t_1, \ldots, t_n)$ et $\mathsf{Witnesses}^+(\mathsf{exec}, \phi) = \{sid_\ell\}$. Soit un ensemble S de sessions tel que $\mathsf{Witnesses}^+(\mathsf{exec}, \phi) \subseteq S$, alors $sid_\ell \in S$ et donc $\mathsf{exec}|_S = (\mathsf{exec}_{\ell-1})|_S @ [e_\ell]$. Nous concluons donc que $\langle \mathsf{exec}_\ell|_S, T_0\rangle \models \phi$.
Si par contre $\langle \mathsf{exec}, T_0\rangle \models \neg\phi$, nous distinguons alors deux cas. Supposons en premier que $\ell = 0$, *i.e.* $\mathsf{exec} = [\,]$. Dans ce cas, pour tout ensemble de sessions S, $\mathsf{exec}|_S = [\,]$ et donc $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$. Supposons à présent que $\ell > 0$, *i.e.* $e_\ell \neq \mathsf{Q}(t_1, \ldots, t_n)$. Par définition nous avons $\mathsf{Witnesses}^-(\mathsf{exec}, \phi) = \{sid_\ell\}$. Soit un ensemble S de sessions tel que $\mathsf{Witnesses}^-(\mathsf{exec}, \phi) \subseteq S$, alors $sid_\ell \in S$ et donc $\mathsf{exec}|_S = (\mathsf{exec}_{\ell-1})|_S @ [e_\ell]$. Nous concluons donc que $\langle \mathsf{exec}_\ell|_S, T_0\rangle \models \neg\phi$.

Cas $\phi = \mathsf{learn}(t)$.
Si $\langle \mathsf{exec}, T_0\rangle \models \phi$, *i.e.* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t$. Si $t \in \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$, alors sachant que exec ne révèle aucune clé long-terme de déchiffrement, il est évident pour tout ensemble de sessions S, $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t$; d'où $\langle \mathsf{exec}|_S, T_0\rangle \models \mathsf{learn}(t)$. Par contre, si $t \in \mathsf{St}(\mathsf{exec})$, pour tout ensemble de sessions S vérifiant
- il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$

d'après le lemme 6.4.13 $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash t$, et donc $\langle \mathsf{exec}|_S, T_0\rangle \models \phi$.
Si par contre $\langle \mathsf{exec}, T_0\rangle \models \neg\mathsf{learn}(t)$, *i.e.* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \not\vdash t$, alors il est évident que pour tout ensemble de sessions S, $\mathsf{K}(\mathsf{exec}|_S) \subseteq \mathsf{K}(\mathsf{exec})$ et donc que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \not\vdash t$, *i.e.* $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$.

Cas $\phi = \mathsf{C}(t)$.
Si $\langle \mathsf{exec}, T_0\rangle \models \mathsf{C}(t)$, alors par définition $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \models \mathsf{pvk}(t)$ (respectivement $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \models \mathsf{shk}(t, t')$ pour un certain $t' \in (\mathcal{P} \setminus \{\epsilon\})$). Mais ayant supposé que exec ne révèle aucune clé long-terme de déchiffrement, il est nécessaire que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash \mathsf{pvk}(t)$ (respectivement $T_0 \cup \mathcal{N}_\epsilon(\Pi) \vdash \mathsf{shk}(t, t')$). Ainsi, pour tout ensemble de sessions S, $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash \mathsf{pvk}(t)$ (respectivement $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_S) \vdash \mathsf{shk}(t, t')$) et donc $\langle \mathsf{exec}|_S, T_0\rangle \models \mathsf{C}(t)$.
Si $\langle \mathsf{exec}, T_0\rangle \models \neg\mathsf{C}(t)$, alors sachant que pour tout S, $\mathsf{K}(\mathsf{exec}|_S) \subseteq \mathsf{K}(\mathsf{exec})$, il est vident que pour tout ensemble de sessions S, $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$.

Cas $\phi = \neg\psi$.
Si $\langle \mathsf{exec}, T_0\rangle \models \neg\psi$. Soit S un ensemble de session tel que
- pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,
- $\mathsf{Witnesses}^+(\mathsf{exec}, \phi) = \mathsf{Witnesses}^-(\mathsf{exec}, \psi) \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$.

Or par définition $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi) \Rightarrow \mathsf{learn}(t) \in \mathsf{SubForm}^-(\psi)$. Nous pouvons donc faire appel à notre hypothèse d'induction afin de conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\psi$, *i.e.* $\langle \mathsf{exec}|_S, T_0\rangle \models \phi$.
Si par contre, $\langle \mathsf{exec}, T_0\rangle \models \neg\neg\psi$. Soit S un ensemble de session tel que
- pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,

- $\mathsf{Witnesses}^-(\mathsf{exec}, \phi) = \mathsf{Witnesses}^+(\mathsf{exec}, \psi) \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$.

Or par définition $\mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi) \Rightarrow \mathsf{learn}(t) \in \mathsf{SubForm}^+(\psi)$. Nous pouvons donc faire appel à notre hypothèse d'induction afin de conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \psi$, *i.e.* $\langle \mathsf{exec}|_S, T_0\rangle \models \phi$.

Cas $\phi = \phi_1 \vee \phi_2$.
Si $\langle \mathsf{exec}, T_0\rangle \models \phi_1 \vee \phi_2$, alors ou bien $\langle \mathsf{exec}, T_0\rangle \models \phi_1$ ou encore $\langle \mathsf{exec}, T_0\rangle \models \phi_2$. Nous ne détaillerons ici que le cas où $\langle \mathsf{exec}, T_0\rangle \models \phi_1$, le second pouvant être traité de la même manière. Dans ce cas, nous savons par définition que $\mathsf{Witnesses}^+(\mathsf{exec}, \phi) = \mathsf{Witnesses}^+(\mathsf{exec}, \phi_1)$. Soit un ensemble de sessions S tel que

- pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,
- $\mathsf{Witnesses}^+(\mathsf{exec}, \phi) = \mathsf{Witnesses}^+(\mathsf{exec}, \phi_1) \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid_1 \in S$ si et seulement si $sid'' \in S$.

Or par définition, $\mathsf{learn}(t) \in \phi_1 \Rightarrow \mathsf{learn}(t) \in \phi$. Nous pouvons donc faire appel à notre hypothèse d'induction afin de conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \phi_1$ et donc que $\langle \mathsf{exec}|_S, T_0\rangle \models \phi$.
Si par contre $\langle \mathsf{exec}, T_0\rangle \models \neg(\phi_1 \vee \phi_2)$, alors $\langle \mathsf{exec}, T_0\rangle \models \neg\phi_1$, $\langle \mathsf{exec}, T_0\rangle \models \neg\phi_2$ et par définition $\mathsf{Witnesses}^-(\phi_1 \vee \phi_2) = \mathsf{Witnesses}^-(\mathsf{exec}, \phi_1) \cup \mathsf{Witnesses}^-(\mathsf{exec}, \phi_2)$. Soit un ensemble de sessions tel que

- pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,
- $\mathsf{Witnesses}^-(\mathsf{exec}, \phi) = \mathsf{Witnesses}^-(\mathsf{exec}, \phi_1) \cup \mathsf{Witnesses}^-(\mathsf{exec}, \phi_2) \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$.

Or, $\mathsf{learn}(t) \in (\mathsf{SubForm}^-(\phi_1) \cup \mathsf{SubForm}^-(\Phi_2) \Rightarrow \mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)$. Nous pouvons donc faire appel à notre hypothèse d'induction et conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi_1$ et que $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi_2$ et que $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$.

Cas $\phi = \mathcal{Y}\psi$.
Si $\langle \mathsf{exec}, T_0\rangle \models \mathcal{Y}\psi$, alors $\langle \mathsf{exec}|_{\ell-1}, T_0\rangle \models \psi$. Soit S un ensemble de sessions tel que

- pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,
- $\mathsf{Witnesses}^+(\mathsf{exec}, \phi) = \mathsf{Witnesses}^+(\mathsf{exec}_{\ell-1}, \psi) \cup \{sid_\ell\} \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$.

Or, $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\psi) \Rightarrow \mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi)$. Nous pouvons donc faire appel à notre hypothèse d'induction et conclure que $\langle \mathsf{exec}_{\ell-1}|_S, T_0\rangle \models \psi$. De plus, par construction de S, nous savons que $sid_\ell \in S$; d'où $\mathsf{exec}|_S = \mathsf{exec}_{\ell-1}|_S @ [e_\ell]$. Nous sommes donc en droit de conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \phi$.
Si par contre $\langle \mathsf{exec}, T_0\rangle \models \neg\mathcal{Y}\psi$, nous distinguons alors deux cas. Supposons que $\ell = 0$, dans ce cas, il est évident que pour tout ensemble de sessions S, $\mathsf{exec}|_S = [\,]$, et donc que $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\mathcal{Y}\psi$, *i.e.* $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$. Supposons à présent que $\ell > 0$. Alors $\langle \mathsf{exec}_{\ell-1}, T_0\rangle \models \neg\psi$. Soit S un ensemble de sessions

tel que
- pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,
- $\mathsf{Witnesses}^-(\mathsf{exec}, \phi) = \mathsf{Witnesses}^-(\mathsf{exec}, \psi) \cup \{sid_\ell\} \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$.

Or, $\mathsf{learn}(t) \in \mathsf{SubForm}^-(\psi) \Rightarrow \mathsf{learn}(t) \in \mathsf{SubForm}^-(\phi)$. Nous pouvons donc faire appel à notre hypothèse d'induction et conclure que $\langle \mathsf{exec}_{\ell-1}|_S, T_0\rangle \models \neg\psi$. De plus, par construction de S, nous savons que $sid_\ell \in S$; d'où $\mathsf{exec}|_S = \mathsf{exec}_{\ell-1}|_S @ [e_\ell]$. Nous sommes donc en droit de conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$.

Cas $\phi = \phi_1 \mathcal{S} \phi_2$.
Si $\langle \mathsf{exec}, T_0\rangle \models \phi_1 \mathcal{S} \phi_2$, alors il existe i tel que $0 \leq i \leq \ell$ et $\langle \mathsf{exec}_i, T_0\rangle \models \phi_2$ et pour tout j tel que $i < j \leq \ell$ $\langle \mathsf{exec}_j, T_0\rangle \models \phi_1$. Posons $i_{max} = \max\{i \mid \langle \mathsf{exec}_i, T_0\rangle \models \phi_2\}$. Soit un ensemble de sessions S tel que
- pour tout $\mathsf{learn}(t) \in \mathsf{SubForm}^+(\phi)$, si $t \notin \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-1$ alors il existe $sid \in S$ telle que $t \in \mathsf{St}(\mathsf{exec}, sid)$,
- $\mathsf{Witnesses}^+(\mathsf{exec}, \phi) = \mathsf{Witnesses}^+(\mathsf{exec}, \phi_2) \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$.

Posons $\mathsf{exec}' = \mathsf{exec}|_S = [e_{i_1}; \ldots; e_{i_n}]$ avec $1 \leq i_1 < \cdots < i_n \leq \ell$. Par hypothèse sur ϕ nous savons qu'aucun $\mathsf{learn}(t)$ ni aucune modalité n'apparaît dans ϕ_2. Nous pouvons donc faire appel à notre hypothèse d'induction et conclure que $\langle \mathsf{exec}_{i_{max}}|_S, T_0\rangle \models \phi_2$. Or, $\mathsf{exec}' = \mathsf{exec}_{i_{max}}|_S @ ([e_{i_{max}+1}; \ldots; e_\ell])|_S$, implique qu'il existe $j \in [\![n]\!]$ tel que $\langle \mathsf{exec}'_j, T_0\rangle \models \phi_2$. D'un autre coté, sachant qu'aucun $\mathsf{learn}(t)$ ni aucune modalité n'apparaît dans ϕ_1, nous savons que pour tout $k \in [\![\ell]\!]$ si $\langle \mathsf{exec}_k, T_0\rangle \models \phi_1$ alors $\mathsf{Witnesses}(\mathsf{exec}_k, \phi_1) \subseteq \{sid_k\}$. Donc pour tout k tel que $j < k \leq n$ sachant que $\langle \mathsf{exec}_{i_k}, T_0\rangle \models \phi_1$ nous concluons que $\langle \mathsf{exec}'_k, T_0\rangle \models \phi_1$. Finalement, nous sommes donc en mesure de conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \phi$.
Si $\langle \mathsf{exec}, T_0\rangle \models \neg(\phi_1 \mathcal{S} \phi_2)$, nous distinguons alors deux cas. Supposons que pour tout $i \in [\![\ell]\!]$, $\langle \mathsf{exec}_i, T_0\rangle \models \neg\phi_2$, sachant qu'aucun $\mathsf{learn}(t)$ ni aucune modalité n'apparaît dans ϕ_2 nous pouvons immédiatement déduire que pour tout ensemble de sessions S, $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$. Supposons à présent qu'il existe $i \in [\![\ell]\!]$ tel que $\langle \mathsf{exec}_i, T_0\rangle \models \phi_2$. Il existe alors j tel que $i < j \leq \ell$ et $\langle \mathsf{exec}_j, T_0\rangle \models \neg\phi_1$. Soit $j_{max} = \max\{j \mid \langle \mathsf{exec}_j, T_0\rangle \models \neg\phi_1\}$. Soit un ensemble de sessions, tel que
- $\mathsf{Witnesses}^-(\mathsf{exec}, \phi) = \mathsf{Witnesses}^-(\mathsf{exec}, \phi_1) \subseteq S$, et
- pour toutes sessions sid' et sid'' *t.q.* $\mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid'')$ et $\mathsf{Tags}(\mathsf{exec}, sid') \neq \emptyset$, $sid' \in S$ si et seulement si $sid'' \in S$.

Posons $\mathsf{exec}' = \mathsf{exec}|_S = [e_{i_1}; \ldots; e_{i_n}]$ avec $1 \leq i_1 < \cdots < i_n \leq \ell$. Par hypothèse sur ϕ nous savons qu'aucun $\mathsf{learn}(t)$ ni aucune modalité n'apparaît dans ϕ_1. Nous pouvons donc faire appel à notre hypothèse d'induction et conclure que $\langle \mathsf{exec}_{j_{max}}|_S, T_0\rangle \models \neg\phi_1$. Or, $\mathsf{exec}' = \mathsf{exec}_{j_{max}}|_S @ ([e_{j_{max}+1}; \ldots; e_\ell])|_S$, implique qu'il existe $j \in [\![n]\!]$ tel que $\langle \mathsf{exec}'_j, T_0\rangle \models \phi_2$. D'un autre coté, sachant qu'aucun $\mathsf{learn}(t)$ ni aucune modalité n'apparaît dans ϕ_2, nous savons que pour tout $k \in [\![\ell]\!]$ si $\langle \mathsf{exec}_k, T_0\rangle \models \neg\phi_2$ alors $\mathsf{Witnesses}^-(\mathsf{exec}_k, \phi_2) \subseteq \{sid_k\}$. Donc pour tout k tel que $j_{max} < k \leq n$ sachant que $\langle \mathsf{exec}_{i_k}, T_0\rangle \models \neg\phi_2$ nous concluons que $\langle \mathsf{exec}'_k, T_0\rangle \models \neg\phi_2$. Finalement, nous sommes donc en mesure de conclure que $\langle \mathsf{exec}|_S, T_0\rangle \models \neg\phi$. □

Lemme 6.4.15. *Soient* Π *un protocole de* $\mathcal{C}_1$*,* T_0 *un ensemble de termes atomiques et clos. Si* Π *révèle des clés de déchiffrement long-terme,* i.e. *il existe une exécution* exec *valide de* Π*, au regard de la connaissance initiale de l'intrus, et une clé long-terme de déchiffrement* $k \in \mathcal{K}^{-1}$ *telle que* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \not\vdash k$ *mais* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash k$*, alors il existe une exécution valide et bien formée de* Π*, au regard de la connaissance initiale de l'intrus* T_0 *qui révèle* k *et implique au plus une session de chaque rôle de* Π*.*

Démonstration. Soient une exécution exec bien formée de Π. Introduisons la notation suivante : pour toute session sid

$$W(\mathsf{exec}, sid) = \begin{cases} \{sid\} & \text{si } \mathsf{Tags}(\mathsf{exec}, sid) = \emptyset \\ \{sid' \mid \mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid)\} & \text{sinon.} \end{cases}$$

Nous allons dans un premier temps, montrer que pour tout terme $t \in \mathsf{St}(\mathsf{exec})$ si $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t$, alors

- il existe une session sid telle que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid)}) \vdash t$,
- ou bien il existe sid et une clé long-terme de déchiffrement $k \in \mathcal{K}^{-1}$ telles que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \not\vdash k$ et $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid)}) \vdash k$.

Nous procédons par induction sur la taille de la preuve minimale π témoignant de $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t$.

Cas où la dernière règle appliquée dans π *est l'axiome.*
Dans ce cas $t \in (T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}))$. Si $t \in (T_0 \cup \mathcal{N}_\epsilon(\Pi))$, alors pour n'importe laquelle des sessions sid, nous avons que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid)}) \vdash t$. Supposons à présent que $t \notin (T_0 \cup \mathcal{N}_\epsilon(\Pi))$. Dans ce cas, il existe une session sid telle que $t \in \mathsf{K}(\mathsf{exec}|_{\{sid\}})$, et donc $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid)}) \vdash t$.

Cas où la dernière règle appliquée dans π *est (C) ou (Ltk).*
Dans ce cas, pour n'importe laquelle des sessions sid, nous avons que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid)}) \vdash t$.

Cas où la dernière règle appliquée dans π *est une règle de composition.*
Dans ce cas π est de la forme

$$\frac{\dfrac{\pi_1}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_1} \quad \ldots \quad \dfrac{\pi_n}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_n}}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t}$$

avec $n \in \{1, 2\}$, et $t = f(t_1, \ldots, t_n)$ pour $f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$. Supposons qu'il n'existe pas de session sid telle que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid)}) \vdash t$. Par hypothèse d'induction nous savons pourtant que pour tout $i \in [\![n]\!]$

- il existe une session sid_i telle que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid_i)}) \vdash t_i$,
- ou bien il existe sid'_i et une clé long-terme de déchiffrement $k_i \in \mathcal{K}^{-1}$ telles que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \not\vdash k_1$ et $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid'_i)}) \vdash k_i$.

Le seul cas qu'il faille analyser pour conclure est celui où $n = 2$, et

- il existe une session sid_1 telle que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid_1)}) \vdash t_1$, et
- il existe une session sid_2 telle que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec},sid_2)}) \vdash t_2$,

mais que $sid_2 \neq sid_1$ et $sid_1 \notin W(\mathsf{exec}, sid_2)$. Mais alors exec étant bien formée nous savons que $\mathcal{N}(\mathsf{exec}, sid_1) \cap \mathcal{N}(\mathsf{exec}, sid_2) = \emptyset$, et par là même que $\mathsf{Est}(t_i) = \emptyset$ pour au moins un des $i \in \{1, 2\}$. Supposons que $\mathsf{Est}(t_1) = \emptyset$. Donc t_1 est un n-uplet dont tous les atomes ne sont pas initialement connus de l'intrus, auquel cas nous pourrions prendre $sid_1 = sid_2$. Mais alors il existe $k \in \mathsf{St}(t_1) \cap \mathcal{K}^{-1}$ tel que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \not\vdash k$. Or nous nous nous sommes placés dans le cas où $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec}, sid_1)}) \vdash t_1$, et donc $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec}, sid_1)}) \vdash k$. Ce qui nous permet de conclure.

Cas où la dernière règles est une règle de décomposition.
Dans ce cas π est de la forme

$$\frac{\dfrac{\pi_1}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_1} \quad \ldots \quad \dfrac{\pi_n}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t_n}}{T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash t}$$

D'après le lemme 2.5.5 de localité, nous savons que pour tout $i \in [\![n]\!]$ $t_i \in (\mathsf{St}(\mathsf{K}(\mathsf{exec}) \cup \{t\}) \cup T_0 \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$. Les cas $t_i \in (T_0 \cup \mathcal{C} \cup \mathcal{K}_\epsilon)$ (pour $i \in [\![n]\!]$) peut être trivialement traité. Le cas $t_i \in \mathsf{St}(\mathsf{K}(\mathsf{exec}) \cup \{t\}) \subseteq \mathsf{St}(\mathsf{exec})$ peut être traité de façon analogue au cas où la dernière règle appliquée dans π est une règle de composition.

Maintenant que nous avons établi ce résultat intermédiaire nous pouvons nous atteler à la preuve à proprement parler du lemme 6.4.15. Pour ce faire, il nous faut montrer qu'il existe une exécution de Π bien formée, valide au regard de la connaissance initiale de l'intrus T_0 qui révèle une clé long-terme de déchiffrement. Soit exec une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0, qui révèle la clé long-terme de déchiffrement $k \in \mathcal{K}^{-1}$, *i.e.* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \not\vdash k$ mais par contre $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash k$. Considérons la formule $\phi = \neg\mathsf{learn}(k)$. Il s'agît là d'une formule de Φ_1. D'après la proposition 6.3.4, nous savons donc qu'il existe une exécution $\mathsf{exec}^{\mathsf{wf}} = [e_1; \ldots; e_\ell]$ valide et bien formée de Π, au regard de la connaissance initiale de l'intrus T_0 qui viole ϕ, *i.e.* $\langle \mathsf{exec}^{\mathsf{wf}}, T_0 \rangle \models \neg\phi$, *i.e.* $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}^{\mathsf{wf}}) \vdash k$. Posons $\mathsf{sc}^{\mathsf{wf}}$ le scénario de Π d'entrelacement $\mathsf{interlvg}^{\mathsf{wf}} = [(r_1, sid_1); \ldots; (r_\ell, sid_\ell)]$ et de trace symbolique associée tr^{wf}, ainsi que la substitution σ^{wf} tels que $\mathsf{exec}^{\mathsf{wf}} = tr^{\mathsf{wf}}\sigma^{\mathsf{wf}}$. Soit k_0 la première clé long-terme dévoilée à l'intrus dans $\mathsf{exec}^{\mathsf{wf}}$ et soit $i_{min} \in [\![\ell]\!]$ le moment où la clé k_0 est dévoilée à l'intrus, *i.e.* pour tout $j \in [\![i_{min} - 1]\!]$, $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}_j) \not\vdash k_0$ mais par contre $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}_{i_{min}}) \vdash k_0$. Donc $\mathsf{exec}^{\mathsf{wf}}_{i_{min}}$ est une exécution valide et bien formée de Π qui révèle une clé long-terme de déchiffrement, et $\mathsf{exec}^{\mathsf{wf}}_{i_{min}-1}$ est une exécution valide et bien formée de Π qui ne révèle aucune clé long-terme. D'après notre résultat préliminaire nous savons que

- il existe une session sid telle que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}^{\mathsf{wf}}|_{W(\mathsf{exec}^{\mathsf{wf}}, sid)}) \vdash k_0$,
- ou bien il existe sid' et une clé long-terme de déchiffrement $k' \in \mathcal{K}^{-1}$ telles que $T_0 \cup \mathcal{N}_\epsilon(\Pi) \not\vdash k$ et $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}|_{W(\mathsf{exec}^{\mathsf{wf}}, sid')}) \vdash k'$.

Mais si le deuxième cas est vrai le premier aussi, sinon k_0 ne serait pas la première clé long-terme de déchiffrement dévoilée dans $\mathsf{exec}^{\mathsf{wf}}$. Plaçons nous dans le premier cas. Alors nous savons que l'événement $e_{i_{min}}$ est une émission initiée par la session $sid_{i_{min}} \in W(\mathsf{exec}^{\mathsf{wf}}, sid)$. Considérons l'exécution $(\mathsf{exec}^{\mathsf{wf}}_{i_{min}-1})|_{W(\mathsf{exec}^{\mathsf{wf}}, sid)}$. Comme $\mathsf{exec}^{\mathsf{wf}}_{i_{min}-1}$ est bien formée et ne révèle aucune

clé long-terme de déchiffrement, nous pouvons faire appel au lemme 6.4.14 et déduire que $(\mathsf{exec}^{\mathsf{wf}}_{i_{min}-1})|_{W(\mathsf{exec}^{\mathsf{wf}},sid)}$, mais comme $e_{i_{min}}$ est une émission nous concluons que $(\mathsf{exec}^{\mathsf{wf}}_{i_{min}})|_{W(\mathsf{exec}^{\mathsf{wf}},sid)} = (\mathsf{exec}^{\mathsf{wf}}_{i_{min}-1})|_{W(\mathsf{exec}^{\mathsf{wf}},sid)}@[e_{i_{min}}]$ aussi est une exécution valide et bien formée, qui d'après ce que nous avons vu révèle la clé long-terme de déchiffrement k_0.

Finalement, nous devons montrer que $|W(\mathsf{exec}^{\mathsf{wf}}, sid)| < k$ où $k = |\mathsf{Roles}(\Pi)|$. Si $\mathsf{Tags}(\mathsf{exec}, sid) = \emptyset$, alors par définition $W(\mathsf{exec}^{\mathsf{wf}}, sid) = \{sid\}$. Nous pouvons donc conclure aisément. Si par contre $\mathsf{Tags}(\mathsf{exec}, sid) \neq \emptyset$, $W(\mathsf{exec}^{\mathsf{wf}}, sid) = \{sid' \mid \mathsf{Tags}(\mathsf{exec}, sid') = \mathsf{Tags}(\mathsf{exec}, sid)\}$. Mais par définition d'une exécution bien formée (voir définition 6.3.3) et plus particulièrement de la condition 3 de cette définition, nous savons que pour tout $sid' \in W(\mathsf{exec}^{\mathsf{wf}}, sid)$, $\mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid') = (\mathsf{Tags}(tr^{\mathsf{wf}}, sid'))\sigma^{\mathsf{wf}}$. Or par construction de Π et donc de tr^{wf}, deux sessions du mêmes rôles ne peuvent pas être taggées de la même manière puisqu'elles introduisent toutes deux un nonce différent dans la même composante de leur tag. Ainsi si sid' et sid'' sont deux sessions du même rôle et que $\mathsf{Tags}(tr^{\mathsf{wf}}, sid') = \langle t_1, \ldots, t_k\rangle$ et $\mathsf{Tags}(tr^{\mathsf{wf}}, sid'') = \langle u_1, \ldots, u_k\rangle$, il existe $i \in [\![k]\!]$ tel que $t_i, u_i \in \mathcal{N}$ et $t_i \neq u_i$. Nous concluons donc que $W(\mathsf{exec}^{\mathsf{wf}}, sid)$ contient au plus une session de chaque rôle de Π. □

Preuve de la proposition 6.3.5

Nous avons à présent tous les résultats dont nous avions besoin pour démontrer la proposition principale de cette section.

Proposition 6.3.5. *Soient Π un protocole dans $\mathcal{C}_1$, une formule $\phi \in \Phi_1$, et T_0 un ensemble de termes atomiques clos. Si Π admet une exécution valide au regard de la connaissance initiale de l'intrus T_0, bien formée et violant ϕ, alors Π admet une exécution valide au regard de la connaissance initiale de l'intrus T_0*

- *qui viole ϕ en impliquant au plus $\mathsf{M}(\phi)$ sessions de chaque rôle de Π,*
- *ou bien, qui révèle une clé long-terme de déchiffrement en impliquant au plus une session de chaque rôle de Π.*

Démonstration. Soit Π un protocole de $\mathcal{C}_1$, ϕ une propriété de Φ_1, et T_0 un ensemble de termes clos. Supposons que Π viole ϕ au regard de la connaissance initiale de l'intrus T_0, *i.e* il existe une exécution exec valide de Π, au regard de la connaissance initiale de l'intrus T_0 telle que $\langle \mathsf{exec}, T_0\rangle \models \neg\phi$. D'après la proposition 6.3.4, nous savons qu'il existe une exécution $\mathsf{exec}^{\mathsf{wf}}$ valide et bien formée de Π, au regard de la connaissance initiale de l'intrus T_0 qui viole ϕ, *i.e.* $\langle \mathsf{exec}^{\mathsf{wf}}, T_0\rangle \models \neg\phi$. Si $\mathsf{exec}^{\mathsf{wf}}$ révèle une clé long-terme de déchiffrement, il suffit de faire appel au lemme 6.4.15 pour conclure qu'il existe une exécution valide de Π au regard de la connaissance initiale de l'intrus, qui implique tout au plus une session par rôle de Π et révèle une clé long-terme de déchiffrement.

Plaçons-nous à présent dans le cas où $\mathsf{exec}^{\mathsf{wf}}$ ne révèle aucune clé long-terme de déchiffrement. ϕ étant dans Φ_1, nous savons que $\neg\phi$ est de la forme $\exists x_1 \ldots \exists x_n.\psi$ où ψ est une formule sans quantificateurs. De plus, étant donné que $\mathsf{exec}^{\mathsf{wf}}$ satisfait $\neg\phi$ nous savons d'après la sémantique de $\mathcal{PS}$-LTL qu'il existe n messages $m_1, \ldots, m_n \in \mathcal{M}$ *t.q.* $\langle \mathsf{exec}^{\mathsf{wf}}, T_0\rangle \models \psi\{x_1 \mapsto m_1, \ldots, x_n \mapsto m_n\}$. Au regard de la procédure **D** et de la construction de $\mathsf{exec}^{\mathsf{wf}}$ (voir preuve

de la proposition 6.3.4), nous savons qui plus est qu'il existe $m'_1, \ldots, m'_n \in (\mathsf{St}(\mathsf{exec}^{\mathsf{wf}}) \cup \mathcal{P})$ tels que $\langle \mathsf{exec}^{\mathsf{wf}}, T_0 \rangle \models \psi\{x_1 \mapsto m'_1, \ldots, x_n \mapsto m'_n\}$. Or par hypothèse sur les formules de la classe Φ_1 (voir définition 6.2.1) nous savons que $\mathsf{St}(\mathsf{SubForm}(\phi)) \subseteq (\mathcal{V} \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^-)$. En combinant ces deux dernières remarques, nous concluons que pour tout $\mathsf{learn}(t) \in \psi\{x_1 \mapsto m'_1, \ldots, x_n \mapsto m'_n\}$, $t \in (\mathsf{St}(\mathsf{exec}^{\mathsf{wf}}) \cup \mathcal{P} \cup \mathcal{C} \cup \mathcal{K}^- 1)$. Posons $\psi' = \psi\{x_1 \mapsto m'_1, \ldots, x_n \mapsto m'_n\}$, et construisons l'ensemble de sessions

$$S = \bigcup_{sid \in \mathsf{Witnesses}(\mathsf{exec}^{\mathsf{wf}}, \psi')} W(\mathsf{exec}^{\mathsf{wf}}, sid)$$

avec pour toute session sid

$$W(\mathsf{exec}^{\mathsf{wf}}, sid) = \begin{cases} \{sid\} & \text{si } \mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid) = \emptyset \\ \{sid' \mid \mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid') = \mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid)\} & \text{sinon} \end{cases}$$

Ainsi Π, T_0, ψ', $\mathsf{exec}^{\mathsf{wf}}$ et S vérifient les hypothèses du lemme 6.4.14 qui nous permet d'affirmer que $\langle \mathsf{exec}^{\mathsf{wf}}|_S, T_0 \rangle \models \psi'$ et donc que $\langle \mathsf{exec}^{\mathsf{wf}}|_S, T_0 \rangle \models \neg\phi$.

Montrons à présent que pour toute session sid $|W(\mathsf{exec}^{\mathsf{wf}}, sid)| \leq k$ pour $k = |\mathsf{Roles}(\Pi)|$. Nous avons déjà eu l'occasion d'établir ce fait lors de la preuve du lemme 6.4.15. En effet, si $\mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid) = \emptyset$, alors par définition $W(\mathsf{exec}^{\mathsf{wf}}, sid) = \{sid\}$, et donc la conclusion est immédiate. Si par contre $\mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid) \neq \emptyset$ il suffit de constater que deux sessions sid' et sid'' du même rôle ne peuvent pas apparaître à la fois dans $W(\mathsf{exec}^{\mathsf{wf}}, sid)$. Détaillons. Soit tr^{wf} la trace symbolique de Π et σ^{wf} la substitution telles que $\mathsf{exec}^{\mathsf{wf}} = tr^{\mathsf{wf}}\sigma^{\mathsf{wf}}$. Par construction de Π et donc de tr^{wf}, nous savons que $\mathsf{Tags}(tr^{\mathsf{wf}}, sid') \neq \mathsf{Tags}(tr^{\mathsf{wf}}, sid'')$. Posons $\{\langle t_1, \ldots, t_k \rangle\} = \mathsf{Tags}(tr^{\mathsf{wf}}, sid')$ et $\{\langle u_1, \ldots, u_k \rangle\} = \mathsf{Tags}(tr^{\mathsf{wf}}, sid'')$. Nous savons qu'il existe $i \in [\![k]\!]$ tel que $t_i, u_i \in \mathcal{N}$ (les nonces engendrés par sid' et sid'' respectivement, au cours de la phase d'initialisation) tels que $t_i \neq u_i$. Or $\mathsf{exec}^{\mathsf{wf}}$ est bien formée implique (voir condition 3 de la définition 6.3.3) que

$$\mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid') = (\mathsf{Tags}(tr^{\mathsf{wf}}, sid'))\sigma^{\mathsf{wf}}$$
$$\mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid'') = (\mathsf{Tags}(tr^{\mathsf{wf}}, sid''))\sigma^{\mathsf{wf}}$$

et donc que $\mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid') \neq \mathsf{Tags}(\mathsf{exec}^{\mathsf{wf}}, sid'')$. Ainsi nous concluons que pour toute session sid, $|W(\mathsf{exec}^{\mathsf{wf}}, sid)| \leq k$. Mais alors sachant d'après le lemme 6.4.9 que $|\mathsf{Witnesses}(\mathsf{exec}^{\mathsf{wf}}, \psi')| \leq \mathsf{M}(\psi')$, nous concluons par construction de S que S contient au plus $\mathsf{M}(\psi')$ sessions de chaque rôle, et que donc $\mathsf{exec}^{\mathsf{wf}}|_S$ implique au plus $\mathsf{M}(\psi')$ sessions de chaque rôle. Finalement, pour conclure nous recourrons à la définition de la fonction $\mathsf{M}()$ selon laquelle

$$\mathsf{M}(\psi') = \mathsf{M}(\psi) = \mathsf{M}(\neg\phi) = \mathsf{M}(\phi).$$

Donc $\mathsf{exec}^{\mathsf{wf}}|_S$ est une exécution valide et bien formée de Π, au regard de la connaissance initiale de l'intrus T_0 qui viole ϕ et qui implique tout au plus $\mathsf{M}(\phi)$ sessions par rôle de Π □

6.5 La propriété du secret : une session suffit

Dans cette section, nous revenons sur la propriété du secret. Cette propriété est de loin la propriété de trace la plus recherchée sur les protocoles,

et mérite qu'on s'y attarde. D'autant plus, qu'un simple raisonnement supplémentaire permet de restreindre encore le nombre de sessions à considérer pour construire une attaque sur cette propriété. Commençons par rappeler la propriété du secret présentée au chapitre 3.

Soit $\Pi = [e_1; \ldots; e_\ell]$ un protocole de $\mathcal{C}_1$, et Π' le protocole construit pour spécifier le nonce n, engendré par le participant p, en tant que secret de Π :

$$\Pi' = [e_1; \ldots; e_i; \mathsf{Secret}(p, a_1, \ldots, a_k, n); e_{i+1}; \ldots; e_\ell].$$

Nous énoncions au même chapitre, la propriété du secret pour n à l'aide de la formule

$$\phi_{\mathsf{S}} = \left\{ \begin{array}{c} \forall y_1 \ldots \forall y_k . \forall z. \\ [\mathcal{O}(\mathsf{Secret}(y_1, \ldots, y_k, z)) \ \wedge \bigwedge_{i \in [\![k]\!]} \mathsf{NC}(y_i) \ \wedge \bigwedge_{s \in (\mathcal{P}(\Pi) \cap \mathcal{S})} \mathsf{NC}(s)] \\ \Rightarrow \\ \neg \mathsf{learn}(z). \end{array} \right\} \psi$$

Lorsque nous appliquons, tel quel, le résultat de réduction de la section 6.3 à un protocole qui ne révèle pas de clé long-terme de déchiffrement, nous obtenons que le nombre de sessions à considérer pour trouver une attaque sur ϕ_{S} est de $\mathsf{M}(\phi_{\mathsf{S}}) = 2$. Regardons d'un peu plus près.

Supposons que le protocole Π' viole ϕ_{S} au regard de la connaissance initiale de l'intrus T_0, *i.e.* il existe une exécution exec valide de Π', au regard de la connaissance initiale de l'intrus T_0 telle que $\langle \mathsf{exec}, T_0 \rangle \models \neg\phi_{\mathsf{S}}$, sans révéler aucune clé long-terme de déchiffrement. Supposons de plus que exec soit bien formée (*cf.* définition 6.3.3 et proposition 6.3.4). D'après la sémantique de $\mathcal{PS}$-LTL cela implique qu'il existe $i \in [\![|\mathsf{exec}|]\!]$ tel que $\mathsf{exec}[i] = \mathsf{Secret}(a_{i_1}, \ldots, a_{i_k}, m)$ et $T_0 \cup \mathcal{N}_\epsilon(\Pi) \cup \mathsf{K}(\mathsf{exec}) \vdash m$, où m est un nonce et les a_{i_j} (pour $j \in [\![k]\!]$) des agents non-compromis.

Si $\mathsf{M}(\phi_{\mathsf{S}}) = 2$ c'est parce que la fonction $\mathsf{Witnesses}()$ appliquée à exec et à $\neg\phi$ sélectionne dans exec la session sid qui témoigne de l'événement $\mathsf{Secret}(a_{i_1}, \ldots, a_{i_k}, m)$ et une session sid' qui admet m parmi ses sous-termes (*cf.* définition 6.4.9). Distinguons deux cas. Supposons dans un premier temps que $\mathsf{Tags}(\mathsf{exec}, sid) = \emptyset$. Alors, par définition d'une exécution bien formée, nous avons nécessairement que $sid = sid'$, et donc la taille de $|\mathsf{Witnesses}(\mathsf{exec}, \phi_{\mathsf{S}})| = 1$.

Si nous supposons, au contraire, que $\mathsf{Tags}(\mathsf{exec}, sid) \neq \emptyset$, alors exec étant bien formée nous savons que $\mathsf{Tags}(\mathsf{exec}, sid) = \mathsf{Tags}(\mathsf{exec}, sid')$. Partant, l'exécution exhibée dans la preuve de la proposition 6.3.5 n'implique que des sessions taggées avec $\mathsf{Tags}(\mathsf{exec}, sid)$. Là encore nous aurions donc pu nous restreindre à une session par rôle de Π'.

Dans tous les cas, afin de vérifier la propriété du secret telle que formulée par ϕ_{S}, il suffit donc de considérer une session par rôle (et non pas deux comme le préconise notre résultat de réduction). Ceci est bien entendu lié au fait que le théorème 6.3.2 vise non-pas la seule propriété du secret, mais bien toute la classe de propriétés Φ_1. Notons finalement, que ce raisonnement tient pour tout sous-terme de Π' spécifié comme le secret [4].

6.6 Comparaisons

Le genre de compilateurs proposé ici a également été exploré dans le modèle calculatoire, en particulier pour la conception de protocoles de groupe visant l'établissement de clés. Par exemple Katz et Young proposent dans [38] un compilateur qui transforme un protocole sûr face à un intrus passif en un protocole sûr face à un intrus actif. Dans la même veine, de précédents travaux [9, 42] proposent des compilateurs pour des protocoles à deux participants.

Dans le modèle symbolique, des travaux récents [8, 26] présentent de telles transformations. B. Beauquier et F. Gauche dans [8] proposent une transformation qui appliquée à un protocole sûr pour une session de chaque rôle et un intrus passif, résulte en un protocole sûr pour un nombre non-borné de sessions face à un intrus actif. Néanmoins, cette transformation ne s'applique qu'à des protocoles dont la profondeur de chiffrement dans leur spécification se borne à 1. Par ailleurs les auteurs n'autorisent que le chiffrement symétrique.

D'un autre côté, V. Cortier et *al.* dans [26] proposent un compilateur qui à un protocole « exécutable » (notion plus faible que 'sûr face à un intrus actif') associe un protocole sûr quel que soit le nombre de sessions considéré, et ce face à un intrus actif. Les auteur n'introduisent alors aucune restriction syntaxique particulière sur le protocole d'origine.

L'intérêt de notre travail réside essentiellement dans le fait que nous n'ayons pas fait appel à la cryptographie pour établir notre résultat. En effet, tous les travaux mentionnés ci-dessus partagent le même inconvénient, à savoir le recours intensif et « onéreux » aux opérations de chiffrement. Ceci est en grande partie dû aux faibles conditions posées quant à la sécurité des protocoles d'origine. Par exemple, pour le secret ou l'authentification nous demandons pour notre part que le protocole initial soit sûr pour une session par rôle face à un intrus actif, quand V. Cortier et *al.* demandent juste qu'il soit excutable. Ceci n'est néanmoins pas une lourde contrepartie compte tenu de l'existence d'outils efficaces pour la vérification d'un nombre borné de sessions.

Considérons par exemple le protocole correspondant dans le modèle Alice et Bob à la seule émission :

$$a \rightarrow b : \mathsf{aenc}(s, b)$$

et appliquons la transformation proposée dans [26]. Le protocole résultant est alors le suivant :

$$\begin{aligned} a &\rightarrow b : n_1, a \\ b &\rightarrow a : n_2, b \\ a &\rightarrow b : \mathsf{aenc}(\langle \mathsf{aenc}(s, b), \mathsf{sign}(\langle \mathsf{aenc}(s, b), \tau \rangle, a) \rangle, b) \end{aligned}$$

où $\tau = \langle n_1, n_2, a_1, a_2 \rangle$. Le message initial est concaténé a la signature de ce dernier concatnéné au numéro de session établi à l'issue de la première étape, le tout chiffré asymétriquement. Aux vues de cet exemple, la simplicité de notre solution justifie à notre sens le choix que nous avons fait de relaxer les conditions sur le protocole d'origine.

Chapitre 7

Réduction de l'espace de recherche à des exécutions « bien typées »

Ce chapitre présente un résultat de réduction de l'espace de recherche justifiant dans de nombreux cas le recours à l'abstraction dite de « typage » (*typing abstraction*). Cette abstraction, à l'origine de plusieurs résultats de décidabilité dans le cadre d'un nombre non-borné de sessions et de nonces [40, 46, 32, 15], suppose qu'il est possible pour un participant de déterminer le type (et donc la taille) de tout message. Néanmoins, une telle abstraction n'est pas correcte dans le cas général, comme en témoigne d'ailleurs l'exemple Π'_{Toy} qui nous accompagne tout au long de cet exposé et son attaque.

Dans ce chapitre, nous verrons que pour la classe de protocoles $\mathcal{C}_2$ vérifiant le critère de « non-unifiabilité des sous-termes » énoncé dans [1], ladite abstraction est correcte au regard de nombreuses propriétés de trace (*e.g.* le secret ou l'authentification). Un tel résultat présente alors un intérêt théorique immédiat, mais également pratique aux vues des résultats de décidabilité ainsi que des nombreux outils de vérification [6, 25, 14, 22] qui en dépendent.

Dans un premier temps (section 7.1), nous définirons le système de types sur lequel repose notre critère de réduction de l'espace de recherche. Puis, aux sections 7.2 et 7.3, nous définirons respectivement la classe de protocoles $\mathcal{C}_2$ et la classe de propriétés Φ_2 pour lesquelles notre résultat de réduction tient. Ce dernier est énoncé et prouvé à la section 7.4.

7.1 Le système de types

Le but de cette section est de définir la relation de typage, dénotée $\mathcal{E} \vdash t : \tau$, qui dans l'*environnement de typage* $\mathcal{E}$ associe le type τ au terme t. C'est sur cette relation de typage que sera par la suite définie la notion d'*exécution bien typée*, et donc notre critère de réduction de l'espace de recherche.

Définition 7.1.1 (Environnement de typage)**.** *Un environnement de typage $\mathcal{E}$ est un triplet $\langle \Pi, \mathsf{sc}, \phi \rangle$ où :*

– Π *est un protocole,*
– sc *est un scénario de* Π*, et*
– ϕ *est une propriété de sécurité de* $\mathcal{PS}$*-LTL*$^-$

Π et ϕ vérifient certaines conditions, mais celles-ci ne seront introduites qu'au fur et à mesure de l'exposé, et ce au moment précis où elles deviendront nécessaires à la cohérence de la définition de notre relation de typage. Nous avons fait ce choix car l'énonciation de ces conditions nécessite de faire appel à des notions que nous ne pouvons encore définir, risquant de rendre la lecture de cette section difficile.

Nous étendons les fonctions $\mathcal{N}()$, $\mathcal{C}()$, et $\mathcal{V}()$ aux environnements de typage de la façon suivante :

$$\begin{aligned}\mathcal{N}(\mathcal{E}) &\stackrel{def}{=} \mathcal{N}(\Pi) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{N}(tr),\\ \mathcal{C}(\mathcal{E}) &\stackrel{def}{=} \mathcal{C}(\Pi) \cup \mathcal{C}(tr), \text{ et}\\ \mathcal{V}(\mathcal{E}) &\stackrel{def}{=} \mathcal{V}(\Pi) \cup \mathcal{V}(tr) \cup \mathcal{V}(\phi),\end{aligned}$$

où tr est la trace symbolique associée au scénario sc.

Exemple 7.1.2. *Soient le protocole* Π'_{Toy}*, le scénario* $\mathsf{sc}'_{\mathsf{Toy}}$*, et la formule* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *définis à l'exemple 3.2.1. Nous les rappelons ici, car nous nous y référerons tout au long de ce chapitre.*

Le protocole Π'_{Toy} *est défini par la séquence d'événements suivante*

$$\begin{aligned}\Pi'_{\mathsf{Toy}} = [\ & \mathsf{snd}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,b),a\rangle,b));\\ & \mathsf{Secret}(a,a,b,n);\\ & \mathsf{rcv}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,b),a\rangle,b));\\ & \mathsf{snd}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,a),b\rangle,a));\\ & \mathsf{rcv}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,a),b\rangle,a))]\end{aligned}$$

et les rôles de Π'_{Toy} *sont rappelons le*

$$\begin{aligned}r'_a = [\ & \mathsf{snd}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,b),a\rangle,b));\\ & \mathsf{Secret}(a,b,n);\\ & \mathsf{rcv}(a,b,\mathsf{aenc}(\langle\mathsf{aenc}(n,a),b\rangle,a))]\end{aligned}\qquad \begin{aligned}r'_b = [\ & \mathsf{rcv}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,b),a\rangle,b));\\ & \mathsf{snd}(b,a,\mathsf{aenc}(\langle\mathsf{aenc}(x,a),b\rangle,a))]\end{aligned}$$

La formule $\phi^{\mathsf{S}}_{\mathsf{Toy}}$

$$\phi^{\mathsf{S}}_{\mathsf{Toy}} = \left\{ \begin{array}{c} \forall y_a.\forall y_b.\forall y_n.\\ [\mathcal{O}(\mathsf{Secret}(y_a,y_b,y_n)) \ \wedge\ \mathsf{NC}(y_a) \ \wedge\ \mathsf{NC}(y_b)] \ \Rightarrow\ \neg\mathsf{learn}(y_n). \end{array} \right.$$

spécifie n *comme étant le secret de* Π'_{Toy}*.*

Nous considérerons le scénario $\mathsf{sc}'_{\mathsf{Toy}} = (\mathsf{interlvg}'_{\mathsf{Toy}}, \mathsf{initagts}'_{\mathsf{Toy}})$ *avec :*

$$\mathsf{interlvg}'_{\mathsf{Toy}} = [(r'_a,1);(r'_a,1);(r'_b,2);(r'_b,2);(r'_b,3);(r'_b,3);(r'_a,1)]$$

$\mathsf{initagts}'_{\mathsf{Toy}}(1) = (a, b)$, $\mathsf{initagts}'_{\mathsf{Toy}}(2) = (\epsilon, b)$, *et* $\mathsf{initagts}'_{\mathsf{Toy}}(3) = (\epsilon, b)$.

La trace symbolique tr'_{Toy} *associée à ce scénario est la séquence d'événements suivante :*

$$\begin{aligned} tr'_{\mathsf{Toy}} = [\ & \mathsf{snd}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, b), a\rangle, b)); \mathit{Secret}(a, b, n^1); \\ & \mathsf{rcv}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, b), \epsilon\rangle, b)); \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, \epsilon), b\rangle, \epsilon)); \\ & \mathsf{rcv}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^3, b), \epsilon\rangle, b)); \mathsf{snd}(b, \epsilon, \mathsf{aenc}(\langle \mathsf{aenc}(x^3, \epsilon), b\rangle, \epsilon)); \\ & \mathsf{rcv}(a, b, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, a), b\rangle, a))] \end{aligned}$$

D'après la définition 2.6.4, l'intrus est muni d'un nonce n^ϵ, i.e. $\mathcal{N}_\epsilon(\Pi'_{\mathsf{Toy}}) = \{n^\epsilon\}$. *Considérons le triplet* $\mathcal{E}_{\mathsf{Toy}} = \langle \Pi'_{\mathsf{Toy}}, \mathsf{sc}'_{\mathsf{Toy}}, \phi^{\mathsf{S}}_{\mathsf{Toy}}\rangle$. *Nous verrons par la suite (exemple 7.1.10) que* $\mathcal{E}_{\mathsf{Toy}}$ *est un environnement de typage,* i.e. Π'_{Toy} *et* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *vérifient les conditions que nous avons tue pour l'instant. D'après la définition 7.1.1,*

$$\mathcal{N}(\mathcal{E}_{\mathsf{Toy}}) = \{n, n^\epsilon, n^1\}, \qquad \mathcal{C}(\mathcal{E}_{\mathsf{Toy}}) = \emptyset, \ \textit{et} \qquad \mathcal{V}(\mathcal{E}_{\mathsf{Toy}}) = \{x, x^1, y_a, y_b, y_n\}.$$

Afin d'éviter de trop nombreuses répétitions, nous fixons (pour cette section 7.1) l'environnement de typage $\mathcal{E} = \langle \Pi, \mathsf{sc}, \phi\rangle$ avec le protocole $\Pi = [e_1; \ldots; e_\ell]$ et le scénario $\mathsf{sc} = (\mathsf{interlvg}, \mathsf{initagts})$ de Π, ainsi que $[n_1, \ldots, n_k] = \mathsf{Nces}(\Pi)$ la séquence de nonces apparaissant dans Π, et $[c_1, \ldots, c_m] = \mathsf{Csts}(\Pi)$ la séquence de constantes apparaissant dans Π. Nous fixons aussi $\mathsf{Nces}_\epsilon(\Pi) = [n^\epsilon_1; \ldots; n^\epsilon_k]$ la séquence de nonces engendrés par l'intrus, ainsi que la séquence ordonnées et sans répétitions $\mathsf{Sess}(\mathsf{interlvg}) = [(r_1, sid_1); \ldots; (r_h, sid_h)]$ de sessions apparaissant dans $\mathsf{interlvg}$, et tr la trace symbolique associée à sc. Précisons que nous considérons que les variables de ϕ sont convenablement renommées, de sorte à ce que $\mathcal{V}(\Pi) \cap \mathcal{V}(\phi) = \emptyset$ et $\mathcal{V}(tr) \cap \mathcal{V}(\phi) = \emptyset$.

La relation de typage que nous visons est quelque peu complexe, et c'est pour cette raison que nous procédons à sa définition en plusieurs étapes : *(1)* nous définirons dans un premier temps le type des termes atomiques n'apparaissant ni dans la trace tr, ni dans la formule ϕ (et en particulier celui des atomes apparaissant dans Π), ainsi que la règle de typage des termes composés, *(2)* nous définirons ensuite le type de chaque atome apparaissant dans la trace symbolique tr associée à sc, *(3)* nous en viendrons finalement à la définition du type des atomes (et plus précisément des variables) apparaissant dans ϕ.

Mais avant de nous atteler à la définition de la relation de typage $\vdash$, il nous faut déterminer l'ensemble de types sur lequel celle-ci est définie. Cet ensemble est déterminé à partir de la spécification du protocole Π.

Définition 7.1.3 (Types induits par Π)**.** *Pour chaque nonce* n_i *de* Π *(*$i \in [\![k]\!]$*) nous introduisons un nouveau type* ν_i *qui lui sera associé ; pour chaque constante* c_i *de* Π *(*$i \in [\![m]\!]$*) nous introduisons un nouveau type* γ_i *qui lui sera associé. Nous introduisons aussi un type* α *qui sera attribué aux agents et aux serveurs, ainsi qu'un type indéterminé* ω*. L'ensemble des types induits par* Π *est engendré par la grammaire suivante :*

$$
\begin{array}{rcll}
\tau, \tau_1, \ldots & ::= & \alpha & \\
& | & \nu_i & i \in [\![k]\!] \\
& | & \gamma_i & i \in [\![m]\!] \\
& | & \omega & \\
& | & \mathsf{pvk}(\alpha) & \\
& | & \mathsf{shk}(\alpha, \alpha) & \\
& | & f(\tau_1, \tau_2) & f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\} \\
& | & \mathsf{h}(\tau) &
\end{array}
$$

Soit τ un type parmi ceux induits par un protocole Π. La taille $|\tau|$ de τ est définie de la même manière que celle d'un terme. Il en est de même pour l'ensemble $\mathsf{SubType}(\tau)$ des sous-types de τ.

Exemple 7.1.4. *Les types atomiques induits par Π'_{Toy} sont α et ω, ainsi que ν pour le nonce n engendré par a.*

(Etape 1) A présent, nous sommes en mesure de donner les premières règles de typage de notre système de types. Elles sont décrites à la figure 7.1, et correspondent à la première étape mentionnée un peu plus haut.

$$\frac{}{\mathcal{E} \vdash t : \alpha}\ t \in \mathcal{P} \quad (1) \qquad \frac{}{\mathcal{E} \vdash n_i : \nu_i}\ i \in [\![k]\!] \quad (2) \qquad \frac{}{\mathcal{E} \vdash n_i^{\epsilon} : \nu_i}\ i \in [\![k]\!] \quad (3) \qquad \frac{}{\mathcal{E} \vdash c_i : \gamma_i}\ i \in [\![m]\!] \quad (4)$$

$$\frac{}{\mathcal{E} \vdash t : \omega}\ t \in (\mathcal{N} \cup \mathcal{C} \cup \mathcal{V}) \setminus (\mathcal{N}(\mathcal{E}) \cup \mathcal{C}(\mathcal{E}) \cup \mathcal{V}(\mathcal{E})) \quad (5)$$

$$\frac{\mathcal{E} \vdash t_1 : \tau_1 \ \ldots \ \Pi \vdash t_n : \tau_n}{\mathcal{E} \vdash f(t_1, \ldots, t_n) : f(\tau_1, \ldots, \tau_n)}\ f \in \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\} \quad (6)$$

$$\frac{\mathcal{E} \vdash \delta_{\Pi}(x) : \tau}{\mathcal{E} \vdash x : \tau}\ x \in \mathcal{V}(\Pi) \quad (7)$$

Fig. 7.1: Règles de typage (1)

A la définition 7.1.3 nous introduisions le type α pour les entités de $\mathcal{P}$. C'est ce que décrit la première règle de la figure 7.1. Elle attribue le type α à toute entité $t \in \mathcal{P}$, *i.e.* aux agents et serveurs. De même, pour tout $i \in [\![k]\!]$ (respectivement $i \in [\![m]\!]$) nous introduisions le type ν_i (respectivement γ_i) pour le nonce n_i (respectivement la constante c_i). La $2^{ème}$ (respectivement la $4^{ème}$) règle associe alors le type ν_i à n_i (respectivement γ_i à c_i). Notons, que le système de types ainsi défini associe un type différent à chaque nonce ainsi qu'à chaque constante dans Π. La $3^{ème}$ règle revient à munir l'intrus d'un

nonce de chaque type. En effet, d'après la définition 2.6.4, à chaque nonce n_i^ϵ de $\mathsf{Nces}_\epsilon(\Pi)$ correspond le nonce n_i de $\mathsf{Nces}(\Pi)$ (avec $i \in [\![k]\!]$), et d'après les règles 2 et 3 n_i^ϵ et n_i se voient attribuer le même type. A la définition 7.1.3 nous introduisions aussi le type ω, celui-ci est attribué aux nonces, variables et constantes n'apparaissant ni dans Π, ni dans tr, ni dans ϕ, comme le décrit la $5^{ème}$ règle de notre système de types. La $6^{ème}$ règle définit le type composé associé à un terme composé de la manière usuelle. La dernière règle stipule que le type associé à une variable x apparaissant dans Π, est le type du terme $\delta_\Pi(x)$ qui est spécifié à la définition 2.4.2 comme la substitution honnête de x. Intuitivement, cela revient à attribuer à x le type du terme qu'un agent honnête s'attend à recevoir pour instantiation de cette variable. Le type du terme $\delta_\Pi(x)$ est bien défini puisque $\delta_\Pi(x)$ est clos pour tout $x \in \mathcal{V}(\Pi)$, et que les règles de typage des participants, nonces, et constantes apparaissant dans $\delta_\Pi(x)$, et donc dans Π sont déjà données.

Exemple 7.1.5. *Soit l'environnement de typage $\mathcal{E}_{\mathsf{Toy}}$ défini à l'exemple 7.1.2. D'après la deuxième règle de la figure 7.1, le type du nonce n est ν, le type introduit à cet effet (voir exemple 7.1.4),* i.e. *$\mathcal{E}_{\mathsf{Toy}} \vdash n : \nu$. D'après la dernière règle de typage de la figure 7.1 et sachant que $\delta_{\Pi'_{\mathsf{Toy}}}(x) = n$ (vu à l'exemple 2.4.4), nous pouvons aussi typer la variable x comme suit*

$$\frac{\mathcal{E}_{\mathsf{Toy}} \vdash \delta_{\Pi'_{\mathsf{Toy}}}(x) : \nu}{\mathcal{E}_{\mathsf{Toy}} \vdash x : \nu}$$

(Etape 2) Il s'agît à présent de déterminer le type qui sera attribué aux atomes de la trace tr. Nous avons déjà vu le type associé aux entités de $\mathcal{P}$, ainsi que celui associé à chacune des constantes apparaissant dans tr (en effet, selon la définition 2.6.5 $\mathcal{C}(tr) \subseteq \mathcal{C}(\Pi)$). Il nous faut encore définir le type associé aux nonces et aux variables de tr. Intuitivement, nous souhaiterions que toute instance d'un nonce $n \in \mathcal{N}(\Pi)$ de type ν se voit attribuer ce même type ν. Et de même, nous souhaiterions que toute instance d'une variable $x \in \mathcal{V}(\Pi)$ de type τ se voit attribuer ce même type τ.

Illustrons notre intention à l'aide de notre exemple courant. Dans la trace tr'_{Toy}, le nonce n^1 est une instance du nonce n apparaissant dans Π. Or, à l'exemple 7.1.5 nous avons vu que le type de n est ν, *i.e.* $\mathcal{E}_{\mathsf{Toy}} \vdash n : \nu$. Aussi, nous souhaiterions que le type attribué à n^1, en tant qu'instance de n, soit ν. De même, les variables x^2 et x^3 de la trace tr'_{Toy} sont des instances de la variable x de Π. Or, nous avons vu à l'exemple 7.1.5 que le type de x est ν, *i.e.* $\mathcal{E}_{\mathsf{Toy}} \vdash x : \nu$. Aussi, nous souhaiterions que le type attribué à x^2 ainsi qu'à x^3, en tant qu'instances de x, soit ν.

Afin de modéliser formellement ces considérations, nous étendons la relation de typage décrite par les règles de la figure 7.1, avec celles de la figure 7.2.

$$\frac{\mathcal{E} \vdash n : \tau}{\mathcal{E} \vdash \mathsf{init}_{r_i,sid_i}(n) : \tau} \begin{cases} \mathsf{init}_{r_i,sid_i}(n) \in \mathcal{N}(tr) \\ n \in \mathcal{N}(\Pi(r_i)) \\ i \in [\![h]\!] \end{cases} \qquad \frac{\mathcal{E} \vdash x : \tau}{\mathcal{E} \vdash \mathsf{init}_{r_i,sid_i}(x) : \tau} \begin{cases} \mathsf{init}_{r_i,sid_i}(x) \in \mathcal{V}(tr) \\ x \in \mathcal{V}(\Pi(r_i)) \\ i \in [\![h]\!] \end{cases}$$

FIG. 7.2: Règles de typage (2)

D'après la définition 2.6.5, pour tout $i \in [\![h]\!]$, un nonce n apparaissant dans le corps du rôle r_i, *i.e.* $n \in \mathcal{N}(\Pi(r_i))$, sera instancié par le nonce frais $\mathsf{init}_{r_i,sid_i}(n)$, où $\mathsf{init}_{r_i,sid_i}$ est la fonction d'initialisation de sid_i jouant le rôle r_i associée à $\mathsf{initagts}$, V_i et N_i, avec

$$\begin{cases} V_1 = \mathcal{V}(\Pi) \\ V_i = V_{i-1} \cup \bigcup\limits_{x \in \mathcal{V}(\Pi(r'_{i-1}))} \{\mathsf{init}_{r_{i-1},sid_{i-1}}(x)\}, \text{ et} \end{cases}$$

$$\begin{cases} N_1 = \mathcal{N}(\Pi) \cup \mathcal{N}_\epsilon(\Pi) \\ N_i = N_{i-1} \cup \bigcup\limits_{n \in \mathcal{N}(\Pi(r'_{i-1}))} \{\mathsf{init}_{r_{i-1},sid_{i-1}}(n)\}, \end{cases}$$

Aussi, le type que nous souhaiterions voir associé à $\mathsf{init}_{r_i,sid_i}(n)$ est le type de n, *i.e.* τ tel que $\mathcal{E} \vdash n : \tau$. C'est exactement ce que décrit la première règle de la figure 7.2. De même, pour tout $i \in [\![h]\!]$, une variable x apparaissant dans le corps du rôle r_i, *i.e.* $x \in \mathcal{V}(\Pi(r_i))$ sera instanciée par la variable fraîche $\mathsf{init}_{r_i,sid_i}(x)$. Et le type que nous souhaiterions voir associé à $\mathsf{init}_{r_i,sid_i}(x)$ est le type de x, *i.e.* τ tel que $\mathcal{E} \vdash x : \tau$. La seconde règle de la figure 7.2 décrit bien cette relation de typage.

Notons que $V_1 = \mathcal{V}(\Pi)$ et $N_1 = \mathcal{N}(\Pi) \cup \mathcal{N}_\epsilon(\Pi)$ impliquent que $\mathcal{V}(\Pi) \cap \mathcal{V}(tr) = \emptyset$, ainsi que $\mathcal{N}(\Pi) \cap \mathcal{N}(tr) = \emptyset$ et, $\mathcal{N}_\epsilon(\Pi) \cap \mathcal{N}(tr) = \emptyset$, ce qui nous assure du fait que les règles de typage deux et trois de la figure 7.1 et la première règle de typage de la figure 7.2 ont des domaines d'application disjoints. Il en est de même pour la dernière règle de typage de la figure 7.1 et la seconde règle de typage de la figure 7.2.

Vérifions que les types que nous escomptions pour les nonces et variables de tr'_{Toy} sont bien ceux attribués par notre système de type.

Exemple 7.1.6. *Soit l'environnement de typage $\mathcal{E}_{\mathsf{Toy}}$ défini à l'exemple 7.1.2. D'après la trace tr'_{Toy} associée à $\mathsf{sc}'_{\mathsf{Toy}}$ nous savons que $x^2 = \mathsf{init}_{r'_b,sid_2}(x)$ et que $x^3 = \mathsf{init}_{r'_b,sid_3}(x)$. Aussi, nous avons vu à l'exemple 7.1.5 que $\mathcal{E}_{\mathsf{Toy}} \vdash x : \nu$. Ainsi d'après la seconde règle de typage de la figure 7.2 nous déduisons que $\mathcal{E}_{\mathsf{Toy}} \vdash x^2 : \nu$ et que $\mathcal{E}_{\mathsf{Toy}} \vdash x^3 : \nu$. De même, nous savons que $n^1 = \mathsf{init}_{r'_a,sid_1}(n)$ et que $\mathcal{E}_{\mathsf{Toy}} \vdash n : \nu$, d'où d'après la première règle de typage de la figure 7.2 $\mathcal{E}_{\mathsf{Toy}} \vdash n^1 : \nu$.*

(Etape 3) Nous en venons maintenant à la partie la plus délicate de la définition de notre relation de typage, dont l'objectif est de définir les types attribués aux atomes apparaissant dans ϕ. Pour ce faire, il nous faut avant tout introduire les conditions sur Π et ϕ que nous remettions à plus tard en début de section.

Condition 1 : Π est bien typé dans l'environnement $\mathcal{E}$ Π vérifie la propriété suivante :

> $\forall i, j \in [\![\ell]\!]$, si $e_i = \mathsf{Q}(p, u_1, \ldots, u_q)$ et $e_j = \mathsf{Q}(p', v_1, \ldots, v_r)$, alors $q = r$ et pour tout $k \in [\![q]\!]$ il existe un type τ_k tel que $\mathcal{E} \vdash u_k : \tau_k$ et $\mathcal{E} \vdash v_k : \tau_k$.

Cette condition revient à associer à chaque prédicat Q apparaissant dans un status event de Π un type $\tau_1 \times \cdots \times \tau_p$[1], et à demander que ce type soit respecté dans Π, *i.e.*

$$\forall i \in [\![\ell]\!].\ e_i = \mathsf{Q}(p, t_1, \ldots, t_q) \Rightarrow (\mathcal{E} \vdash t_1 : \tau_1 \ \wedge \ \ldots \ \wedge \ \mathcal{E} \vdash t_p : \tau_p).$$

Nous dirons alors que Q est de type $\tau_1 \times \cdots \times \tau_p$, ce que nous dénoterons par l'expression $\mathcal{E} \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_p$.

Quoique la définition de notre système de types ne soit pas complète, la condition « Π est bien typé dans l'environnement $\mathcal{E}$ » est bien définie. En effet, tous les atomes de Π se voient attribuer un type à l'aide des règles de la figure 7.1. Et la règle de typage des termes composés est elle aussi définie à la figure 7.1. Ainsi, pour tout événement de Π de la forme $\mathsf{Q}(p, t_1, \ldots, t_n)$, chacun des t_i ($i \in [\![n]\!]$) peut être typé à l'aide de ces premières règles de typage.

Exemple 7.1.7. *Le seul status event de* Π'_{Toy} *est* $\mathsf{Secret}(a, a, b, n)$. Π'_{Toy} *est donc nécessairement bien typé. De plus, nous avons déjà vu que* a *et* b *sont de type* α*, et* n *est de type* ν *dans l'environnement* $\mathcal{E}_{\mathsf{Toy}}$*, donc le prédicat* Secret *est de type* $\alpha \times \alpha \times \nu$ *dans l'environnement* $\mathcal{E}_{\mathsf{Toy}}$, i.e. $\mathcal{E}_{\mathsf{Toy}} \vdash \mathsf{Secret} : \alpha \times \alpha \times \nu$.

Condition 2 : ϕ est bien typée dans l'environnement $\mathcal{E}$ Intuitivement, nous cherchons à définir une condition sur ϕ qui nous permettra d'attribuer des types aux variables de ϕ cohérents avec les types des status events de Π. Plus précisément, nous voulons définir une condition sur ϕ telle que si nous posons $\{x_1, \ldots, x_p\} = \mathcal{V}(\phi)$ l'ensemble de variables de ϕ, nous pourrons alors définir un ensemble de types $\{\tau_1, \ldots, \tau_p\}$, tel qu'en étendant la relation de typage décrite par les règles des figures 7.1 et 7.2 avec la règle de typage

$$\frac{}{\mathcal{E} \vdash x_i : \tau_i}\ i \in [\![p]\!]$$

ϕ vérifie la propriété selon laquelle

> pour tout $\mathsf{Q}(t_1, \ldots, t_r) \in \mathsf{SubForm}^-(\phi)$ et pour tout $k \in [\![r]\!]$, $\mathcal{E} \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_r \ \Rightarrow \ \mathcal{E} \vdash t_k : \tau_k$.

Pour rendre notre propos plus intuitif nous l'illustrons à l'aide de la propriété $\phi^{\mathsf{S}}_{\mathsf{Toy}}$. Nous venons de voir à l'exemple 7.1.7 que le type du prédicat Secret est $\alpha \times \alpha \times \nu$. Nous cherchons donc un type pour les variables $y_a, y_b, y_n \in \mathcal{V}(\phi^{\mathsf{S}}_{\mathsf{Toy}})$ tel que le type de Secret soit respecté dans $\phi^{\mathsf{S}}_{\mathsf{Toy}}$. Il est évident que le seul moyen d'y parvenir est d'attribuer le type α à y_a, le type α à y_b, et le type ν à y_n.

Néanmoins, sans aucune restriction sur ϕ, il ne sera pas nécessairement possible d'inférer un type pour ses variables cohérent avec les types des status events de Π. La condition de « bon typage » que nous cherchons donc à définir, est une condition sur ϕ qui nous assure qu'il nous sera possible d'attribuer des types cohérents aux variables de ϕ. Pour y parvenir, nous introduisons quelques notions supplémentaires.

Définition 7.1.8 (Problème de typage)**.** *Un problème de typage* Γ *dans l'environnement* $\mathcal{E}$ *est un ensemble d'expressions de la forme* $t \overset{?}{:} \tau$ *où* t *est un terme et* τ *un type parmi ceux induits par* Π *(*cf. *définition 7.1.3).*

Un problème de typage $\Gamma = \{t_i \overset{?}{:} \tau_i\}_{i \in [\![p]\!]}$ *est dit* en forme résolue *si :*

[1] Notons que le premier argument de tout prédicat Q disparaissant dans les exécutions de Π nous n'avons pas inclus son type (*i.e.* α) dans le type de Q.

– $\forall i \in [\![p]\!].\ t_i \in \mathcal{V}$, *et*
– $\forall t \overset{?}{:} \tau, t' \overset{?}{:} \tau' \in \Gamma.\ t = t' \Rightarrow \tau = \tau'$.

Nous allons à présent définir une procédure de résolution de problèmes de typage reposant sur l'ensemble de règles de la figure 7.3. Remarquons que l'environnement de typage $\mathcal{E}$ est une entrée de la procédure, au même titre que Γ (voir règle R_2 de la figure 7.3). Pour tout $n \geq 1$ nous notons $\Gamma_0 \rightsquigarrow^n \Gamma_n$ pour la dérivation $\Gamma_0 \rightsquigarrow \Gamma_1 \rightsquigarrow \cdots \rightsquigarrow \Gamma_n$. Nous introduisons aussi la notation $\Gamma \rightsquigarrow^* \Delta$ pour dénoter que $\Gamma \rightsquigarrow^n \Delta$ pour un certain $n \geq 1$, ou que $\Delta = \Gamma$. Un problème de typage Γ est dit *admettre une solution* s'il existe un problème de typage Δ en forme résolue tel que $\Gamma \rightsquigarrow^* \Delta$. Δ est une *solution* de Γ, et comme nous allons le voir maintenant (*c.f.* lemme 7.1.9) elle est unique.

$$
\begin{array}{lll}
\mathsf{R}_1 : & \Gamma \cup \{t : \alpha\} \rightsquigarrow \Gamma & \text{si } t \in \mathcal{P} \\
\mathsf{R}_2 : & \Gamma \cup \{c \overset{?}{:} \tau\} \rightsquigarrow \Gamma & \text{si } c \in \mathcal{C} \text{ et } \mathcal{E} \vdash c : \tau \\
\mathsf{R}_3 : & \Gamma \cup \{f(t_1, \ldots, t_n) \overset{?}{:} f(\tau_1, \ldots, \tau_n)\} \rightsquigarrow \Gamma \cup \{t_1 \overset{?}{:} \alpha, \ldots, t_n \overset{?}{:} \alpha\} & \\
 & & \text{si } f \in \{\mathsf{pvk}, \mathsf{shk}\} \text{ et } \forall i \in [\![n]\!].\ \tau_i = \alpha \\
\mathsf{R}_4 : & \Gamma \cup \{f(t_1, \ldots, t_n) \overset{?}{:} f(\tau_1, \ldots, \tau_n)\} \rightsquigarrow \Gamma \cup \{t_1 \overset{?}{:} \tau_1, \ldots, t_n \overset{?}{:} \tau_n\} & \\
 & & \text{si } f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}
\end{array}
$$

Fig. 7.3: Règles de simplification

Le lemme suivant stipule que l'ensemble des règles de simplification de la figure 7.3 termine et est confluent. Ainsi, si un problème de typage admet une solution Δ, alors Δ est l'unique solution de Γ.

Lemme 7.1.9 (Unicité de la solution d'un problème de typage)**.** *Le système de simplification décrit à la figure 7.3 termine et est confluent.*

Démonstration. Soit Γ un problème de typage. Définissons la mesure sur Γ, $M(\Gamma) = \sum_{t \overset{?}{:} \tau} |t|$. Soit Δ un problème de typage tel que $\Gamma \overset{\mathsf{R}}{\rightsquigarrow} \Delta$. Nous allons montrer que quelle que soit la règle R impliquée dans cette dérivation, $M(\Delta) < M(\Gamma)$. La relation $<$ sur les entiers étant un beau préordre, nous pourrons alors conclure qu'il n'existe pas de dérivation infinie $\Gamma \rightsquigarrow \Gamma_1 \rightsquigarrow \Gamma_2 \rightsquigarrow \ldots$. Il suffit, en fait de remarquer que pour toute règle R, $M(\Delta) = M(\Gamma) - 1 < M(\Gamma)$. Ce qui nous permet de conclure quant à la terminaison de notre système de simplification.

Passons maintenant à la preuve de confluence. Nous allons plus précisément montrer que notre système de simplification vérifie la propriété du diamant, *i.e.* pour tous problèmes de typage Γ, Γ_1 et Γ_2, si $\Gamma \rightsquigarrow \Gamma_1$, $\Gamma \rightsquigarrow \Gamma_2$, et $\Gamma_1 \neq \Gamma_2$, alors il existe un problème de typage Δ tel que $\Gamma_1 \rightsquigarrow \Delta$ et $\Gamma_2 \rightsquigarrow \Delta$.
Soient Γ, Γ_1 et Γ_2 ainsi que deux règle R et R' parmi celles décrites à ma figure 7.3, tels que $\Gamma \overset{\mathsf{R}}{\rightsquigarrow} \Gamma_1$, $\Gamma \overset{\mathsf{R}'}{\rightsquigarrow} \Gamma_2$, et $\Gamma_1 \neq \Gamma_2$. Nous distinguons deux cas :

$\mathsf{R} \neq \mathsf{R}'$. Les quatre règles de la figure 7.3 s'appliquant à des ensembles disjoints d'expressions de la forme $t \overset{?}{:} \tau$, il existe nécessairement deux expressions distinctes $t_1 \overset{?}{:} \tau_1$, $t_2 \overset{?}{:} \tau_2$ $((t_1 \overset{?}{:} \tau_1) \neq (t_2 \overset{?}{:} \tau_2))$, ainsi que trois problèmes de typage Γ', Γ'', Γ''', tels que $\Gamma = \Gamma' \cup \{t_1 \overset{?}{:} \tau_1\} \cup \{t_2 \overset{?}{:} \tau_2\}$, $\Gamma_1 = \Gamma' \cup \{t_1 \overset{?}{:} \tau_1\} \cup \Gamma'''$ et $\Gamma_2 = \Gamma' \cup \Gamma'' \cup \{t_2 \overset{?}{:} \tau_2\}$. Il est évident que pour $\Delta = \Gamma' \cup \Gamma'' \cup \Gamma'''$, les simplifications $\Gamma_1 \overset{\mathsf{R}'}{\rightsquigarrow} \Gamma' = \Delta$, et $\Gamma_2 \overset{\mathsf{R}}{\rightsquigarrow} \Gamma' = \Delta$ sont possibles.

$\mathsf{R} = \mathsf{R}'$. Comme $\Gamma_1 \neq \Gamma_2$, alors R a nécessairement été appliquée à deux expressions distinctes $t_1 \overset{?}{:} \tau_1, t_2 \overset{?}{:} \tau_2$ $((t_1 \overset{?}{:} \tau_1) \neq (t_2 \overset{?}{:} \tau_2))$ de Γ. Ainsi, il doit exister trois problèmes de typage Γ', Γ'', Γ''', tels que $\Gamma = \Gamma' \cup \{t_1 \overset{?}{:} \tau_1\} \cup \{t_2 \overset{?}{:} \tau_2\}$, $\Gamma_1 = \Gamma' \cup \{t_1 \overset{?}{:} \tau_1\} \cup \Gamma'''$ et $\Gamma_2 = \Gamma' \cup \Gamma'' \cup \{t_2 \overset{?}{:} \tau_2\}$. Il est évident que pour $\Delta = \Gamma' \cup \Gamma'' \cup \Gamma'''$ les simplifications $\Gamma_1 \overset{\mathsf{R}}{\rightsquigarrow} \Gamma' = \Delta$, et $\Gamma_2 \overset{\mathsf{R}}{\rightsquigarrow} \Gamma' = \Delta$ sont possibles. □

Nous sommes à présent en mesure de définir la condition de « bon typage ». ϕ est bien typée si elle satisfait les propriétés suivantes :

1. $\forall\, \mathsf{Q}(t_1, \ldots, t_q) \in \mathsf{SubForm}^-(\phi). \exists i \in [\![\ell]\!].\ e_i = \mathsf{Q}(p, u_1, \ldots, u_q)$,
2. $\forall\, \mathsf{learn}(t) \in \mathsf{SubForm}(\phi).\ \forall x \in \mathcal{V}(t).\ \exists \mathsf{Q}(t_1, \ldots, t_q) \in \mathsf{SubForm}^-(\phi). \exists i \in [\![q]\!].\ x \in \mathcal{V}(t_i)$, et
3. le problème de typage

$$\Gamma_\phi = \bigcup_{\substack{\mathsf{Q}(t_1,\ldots,t_q) \in \mathsf{SubForm}^-(\phi) \\ \mathcal{E} \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_q}} \{t_1 \overset{?}{:} \tau_1, \ldots, t_q \overset{?}{:} \tau_q\} \cup \bigcup_{\mathsf{C}(t) \in \mathsf{SubForm}^-(\phi)} \{t \overset{?}{:} \alpha\}$$

admet une solution Σ_ϕ.

La première condition stipule que tout prédicat apparaissant négativement dans ϕ, doit apparaître dans Π. En effet, d'après la condition de « bon typage » de Π, seuls les prédicats dans Π se voient attribuer un type, ce sans quoi Γ_ϕ risquerait de ne pas être bien défini. La deuxième condition restreint les variables apparaissant sous un learn, et ce pour s'assurer qu'elles se verront par la suite attribuer des types cohérents. Si la troisième condition impose que Γ_ϕ admette une solution Σ_ϕ c'est parce que, comme nous allons le voir juste après l'exemple 7.1.10, les types associés aux variables de ϕ sont déterminés à partir de la solution Σ_ϕ.

Quoique la définition de notre système de types ne soit pas encore complète, la condition « ϕ est bien typée dans l'environnement $\mathcal{E}$ » est bien définie. En effet, ayant imposé que les prédicats apparaissant dans des sous-formules négatives de ϕ apparaissent aussi dans Π, le type de ces derniers est uniquement déterminé à l'aide des règles de la figure 7.1.

Exemple 7.1.10. *Revenons sur notre protocole* Π'_{Toy} *et la formule* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$. *Le seul status event apparaissant négativement dans* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *est* $\mathsf{Secret}(y_a, y_b, y_n)$, *et il apparaît aussi dans* Π'_{Toy}, *la première condition de la propriété définie ci-dessus est donc vérifiée. Il en est de même pour la deuxième, puisque* $y_n \in \mathcal{V}(\mathsf{Secret}(y_a, y_b, y_n))$. *Aussi, comme nous l'avons vu à l'exemple 7.1.7* $\mathcal{E}_{\mathsf{Toy}} \vdash$

$\mathsf{Q} : \alpha \times \alpha \times \nu$. *Donc le problème de typage associé à* Π'_{Toy} *et* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *est le suivant :*

$$\Gamma_{\phi^{\mathsf{S}}_{\mathsf{Toy}}} = \{y_a \overset{?}{:} \alpha, y_b \overset{?}{:} \alpha, y_n \overset{?}{:} \nu\}.$$

$\Gamma_{\phi^{\mathsf{S}}_{\mathsf{Toy}}}$ *est en forme résolue, et admet donc une solution. Ainsi nous pouvons conclure que* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *est bien typée dans l'environnement* $\mathcal{E}_{\mathsf{Toy}}$.

De plus, nous avons vu à l'exemple 7.1.7 que Π'_{Toy} *est bien typé dans ce même environnement. Nous vérifions donc bien ce qui était annoncé à l'exemple 7.1.2, à savoir que* $\mathcal{E}'_{\mathsf{Toy}}$ *est un environnement de typage.*

Maintenant que nous avons énoncé les conditions que Π et ϕ doivent satisfaire, nous pouvons revenir à la dernière étape de la définition de notre relation de typage, ayant pour but de définir le type qui sera attribué aux variables de ϕ. Rappelons que nous considérons les ensembles de variables dans Π, tr, et ϕ disjoints deux à deux. Nous étendons donc la relation de typage décrite par les règles des figures 7.1 et 7.2 avec celles de la figure 7.4. Celles-ci ne s'appliquent qu'aux variables de ϕ, car les règles trois et quatre de la figure 7.1 couvrent les constantes de ϕ, et par définition (3.1.1) aucun nonce n'apparaît dans ϕ (*i.e.* $\mathcal{N}(\phi) = \emptyset$). La première règle de la figure 7.4 attribue à une variable x de ϕ, le

$$\frac{}{\mathcal{E} \vdash x : \tau} \begin{cases} x \in \mathcal{V}(\phi) \\ x \overset{?}{:} \tau \in \Sigma_\phi \end{cases} \qquad \frac{}{\mathcal{E} \vdash x : \omega} \begin{cases} x \in \mathcal{V}(\phi) \\ x \overset{?}{:} \tau \notin \Sigma_\phi \text{ pour tout type } \tau \end{cases}$$

Fig. 7.4: Règles de typage (3)

type qui lui est associé dans la solution Σ_ϕ, à condition que x apparaisse dans Σ_ϕ. Ainsi, comme nous le verrons au lemme 7.1.15, le type de x sera cohérent avec ceux des status events de Π. La deuxième règle associe aux variables n'apparaissant pas dans Σ_ϕ, le type indéterminé ω.

Exemple 7.1.11. *Il est facile de voir à partir du problème de typage* $\Gamma_{\phi^{\mathsf{S}}_{\mathsf{Toy}}}$ *vu à l'exemple 7.1.10 associé à* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *que les variables de* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *se voient attribuer les types que nous annoncions. En effet, comme nous le voulions* $\mathcal{E}_{\mathsf{Toy}} \vdash y_a : \alpha$ *car* $y_a \overset{?}{:} \alpha \in \Gamma_{\phi^{\mathsf{S}}_{\mathsf{Toy}}}$. *Et de même pour* y_b *et* y_n *qui se voient attribuer le type* α *et le type* ν *respectivement.*

Définition 7.1.12 ($\mathcal{E} \vdash t : \tau$)**.** *Soient* τ *un type induit par* Π *et* t *un terme.* t *est de type* τ *dans l'environnement* $\mathcal{E}$*, noté* $\mathcal{E} \vdash t : \tau$*, s'il existe un arbre dont les nœuds sont étiquetés par des expressions de la forme* $\mathcal{E} \vdash t' : \tau'$ *et tels que :*

- *la racine est étiquetée par le séquent* $\mathcal{E} \vdash t : \tau$,
- *pour tout nœud intermédiaire étiqueté* $\mathcal{E} \vdash t' : \tau'$ *avec pour fils des nœuds étiquetés* $\mathcal{E} \vdash t'_1 : \tau'_1 \ \ldots \ \mathcal{E} \vdash t'_n : \tau'_n$*, il existe une substitution* σ *et une règle des Figures 7.1, 7.2, ou 7.4* $\dfrac{\mathcal{E} \vdash t''_1 : \tau''_1 \ \ldots \ \mathcal{E} \vdash t''_n : \tau''_n}{\mathcal{E} \vdash t'' : \tau''}$ *tels que* $t'_i = t''_i\sigma$ *et* $\tau'_i = \tau''_i\sigma$ *pour tout* $i \in [\![n]\!]$*, ainsi que* $t' = t''\sigma$ *et* $\tau' = \tau''\sigma$,
- *toute feuille est étiquetée par un séquent* $\mathcal{E} \vdash t' : \tau'$ *avec*
  - *soit* $t' \in \mathcal{P}$ *et* $\tau' = \alpha$,
  - *soit* $t' = n_i$ *et* $\tau' = \nu_i$ *avec* $i \in [\![k]\!]$,

- *soit* $t' = n_i^{\epsilon}$ *et* $\tau' = \nu_i$ *avec* $i \in [\![k]\!]$,
- *soit* $t' = c_i$ *et* $\tau' = \gamma_i$ *avec* $i \in [\![m]\!]$,
- *soit* $t' \in (\mathcal{N} \cup \mathcal{C} \cup \mathcal{V}) \setminus (\mathcal{N}(\mathcal{E}) \cup \mathcal{C}(\mathcal{E}) \cup \mathcal{V}(\mathcal{E}))$ *et* $\tau' = \omega$,
- *soit* $t' \in \mathcal{V}(\phi)$ *et* $x \overset{?}{:} \tau' \in \Sigma_\phi$,
- *ou encore* $t' \in \mathcal{V}(\phi)$ *mais pour tout* $y \overset{?}{:} \tau'' \in \Sigma_\phi,\ x \neq y$ *et* $\tau' = \omega$.

Exemple 7.1.13. *Comme en témoigne l'arbre de typage ci-dessous le terme* $\mathsf{aenc}(\langle \mathsf{aenc}(x,b),a\rangle,b)$ *est de type* $\mathsf{aenc}(\langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle,\alpha)$ *dans l'environnement* $\mathcal{E}_{\mathsf{Toy}}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash \delta_{\Pi'_{\mathsf{Toy}}}(x) : \nu}}{\mathcal{E}_{\mathsf{Toy}} \vdash x : \nu} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash b : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \mathsf{aenc}(x,b) : \mathsf{aenc}(\nu,\alpha)} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash a : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \langle \mathsf{aenc}(x,b),a\rangle : \langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash b : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \mathsf{aenc}(\langle \mathsf{aenc}(x,b),a\rangle,b) : \mathsf{aenc}(\langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle,\alpha)}$$

De même, l'arbre de typage ci-dessous témoigne du fait que dans l'environnement $\mathcal{E}_{\mathsf{Toy}}$ *le terme* $\mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1,b),a\rangle,b),\epsilon\rangle,b)$ *se voit attribuer le type* $\mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle,\alpha),\alpha\rangle,\alpha)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash n : \nu}}{\mathcal{E}_{\mathsf{Toy}} \vdash n^1 : \nu} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash b : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \mathsf{aenc}(n^1,b) : \mathsf{aenc}(\nu,\alpha)} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash a : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \langle \mathsf{aenc}(n^1,b),a\rangle : \langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash b : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \mathsf{aenc}(\langle \mathsf{aenc}(n^1,b),a\rangle,b) : \mathsf{aenc}(\langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle,\alpha)} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash \epsilon : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1,b),a\rangle,b),\epsilon\rangle : \langle \mathsf{aenc}(\langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle,\alpha),\alpha\rangle} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash b : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(n^1,b),a\rangle,b),\epsilon\rangle,b) : \mathsf{aenc}(\langle \mathsf{aenc}(\langle \mathsf{aenc}(\nu,\alpha),\alpha\rangle,\alpha),\alpha\rangle,\alpha)}$$

La définition de la relation de typage $\vdash$ arrive ainsi à son terme. Il nous reste néanmoins à vérifier que cette relation est consistante, et en particulier, qu'il s'agît bien là d'une relation de typage, *i.e.* qu'étant donné un terme t, la relation de typage $\vdash$ dans l'environnement $\mathcal{E}$ attribue un et un seul type à t.

Lemme 7.1.14. *Soit* t *un terme. Il existe un et un seul type* τ *tel que* $\mathcal{E} \vdash t : \tau$.

Démonstration. Nous procédons par induction sur la mesure $M(t) = (|\mathcal{V}(t)|, |t|)$, en considérant l'ordre lexicographique.

Cas $t \in \mathcal{P}$.
D'après la première règle de la figure 7.1, pour $\tau = \alpha$ nous avons bien $\mathcal{E} \vdash t : \tau$. Aussi, en inspectant les autres règles de typage il est aisé de voir que c'est là l'unique règle qui puisse être appliquée afin d'attribuer un type à t et que donc τ est unique.

Cas $t \in \mathcal{V}$.
Nous distinguons alors quatre cas disjoints.

- $t \in \mathcal{V}(\Pi)$. Or, $\forall \tau.\ \mathcal{E} \vdash t : \tau \Leftrightarrow \mathcal{E} \vdash \delta_\Pi(t) : \tau$. Aussi, nous savons que $\delta_\Pi(t)$ est unique (voir définition 2.4.2 de δ_Π), ainsi que $0 = |\mathcal{V}(\delta_\Pi(t))| < |\mathcal{V}(t)| = 1$. Donc $M(\delta_\Pi(t)) <_{\mathsf{lex}} M(t)$ et par hypothèse d'induction nous savons qu'il existe un unique τ tel que $\mathcal{E} \vdash \delta_\Pi(t) : \tau$. Pour conclure, il ne reste plus qu'à noter que la règle sept de la figure 7.1 est l'unique règle qui puisse être appliquée afin d'attribuer un type à t, et que donc il existe un unique type τ (le même que pour $\delta_\Pi(t)$) tel que $\mathcal{E} \vdash t : \tau$.

- $t \in \mathcal{V}(tr)$. Alors, il existe $i \in [\![h]\!]$ et une variable $x \in \mathcal{N}(\Pi(r_i))$, tels que $t = \mathsf{init}_{r_i,sid_i}(x)$, *i.e.* t est une instance d'une variable x d'un rôle r_i de Π. Or, nous venons de voir que pour tout $y \in \mathcal{V}(\Pi)$, il existe un unique type τ' tel que $\mathcal{E} \vdash y : \tau'$, ainsi pour x en particulier, il existe un unique τ tel que $\mathcal{E} \vdash x : \tau$. Or, d'après l'unique règle qui puisse être appliquée (*i.e* la règle 2 de la figure 7.2), $\mathcal{E} \vdash t : \tau$.

- $t \in \mathcal{V}(\phi)$. Si t apparaît dans Σ_ϕ nous savons d'après le lemme 7.1.9 qu'il existe un unique type τ tel que $x \overset{?}{:} \tau \in \Sigma_\phi$ (car Σ_ϕ est en forme résolue), et par la première règle de la figure 7.4 il en découle qu'il existe un unique τ tel que $\mathcal{E} \vdash t : \tau$. Si t n'apparaît pas dans Σ_ϕ, alors l'unique règle que nous puissions appliquer afin d'attribuer un type à t est la deuxième règle de la figure 7.4. Nous concluons donc que dans ce cas le seul type qui puisse être attribué à t est ω.

- Si t ne rentre dans aucun des trois cas précédents, alors l'unique règle qui puisse être invoquée pour lui attribuer un type est la règle cinq de la figure 7.1. De plus, celle-ci ne s'applique qu'aux variables de $\mathcal{V} \setminus \mathcal{V}(\mathcal{E})$, avec $\mathcal{V}(\mathcal{E}) = \mathcal{V}(\Pi) \cup \mathcal{V}(tr) \cup \mathcal{V}(\phi)$. Ainsi, le seul type qui puisse être attribué à t est ω.

Cas $t \in \mathcal{N}$.
Nous distinguons alors quatre cas disjoints. En effet, par définition (voir définition 2.6.5) nous savons que les ensembles $\mathcal{N}(\Pi)$, $\mathcal{N}_\epsilon(\Pi)$ et $\mathcal{N}(tr)$ sont deux à deux disjoints. Nous rappelons aussi que d'après la définition 3.1.1, $\mathcal{N}(\phi) = \emptyset$.

- $t \in \mathcal{N}(\Pi)$. Alors il existe $i \in [\![k]\!]$ tel que $t = n_i$, et d'après la deuxième règle de la figure 7.1, pour $\tau = \nu_i$ nous avons bien $\mathcal{E} \vdash t : \tau$. Aussi, en inspectant les autres règles de typage il est aisé de voir que c'est là l'unique règle qui puisse être invoquée pour attribuer un type à t et que donc τ est unique.

- $t \in \mathcal{N}_\epsilon(\Pi)$. Alors il existe $i \in [\![k]\!]$ tel que $t = n_i^\epsilon$, et d'après la troisième règle de la figure 7.1, pour $\tau = \nu_i$ nous avons bien $\mathcal{E} \vdash t : \tau$. Aussi, en inspectant les autres règles de typage il est aisé de voir que c'est là l'unique règle qui puisse être invoquée pour attribuer un type à t et que donc τ est unique.

- $t \in \mathcal{N}(tr)$. Alors, il existe $i \in [\![h]\!]$ et un nonce $n \in \mathcal{N}(\Pi(r_i))$, tels que $t = \mathsf{init}_{r_i,sid_i}(n)$, *i.e.* t est une instance d'un nonce n d'un rôle r_i de Π. Or, nous venons de voir que pour tout $n' \in \mathcal{N}(\Pi)$, il existe un unique type τ' tel que $\mathcal{E} \vdash n' : \tau'$, ainsi pour n en particulier, il existe un unique

τ tel que $\mathcal{E} \vdash n : \tau$. Or, d'après l'unique règle qui puisse être appliquée (*i.e* la règle un de la figure 7.2), $\mathcal{E} \vdash t : \tau$.

– Si t ne rentre dans aucun des trois cas précédents, alors l'unique règle qui puisse être invoquée pour lui attribuer un type est la règle cinq de la figure 7.1. De plus, celle-ci ne s'applique qu'aux nonces de $\mathcal{N} \setminus \mathcal{N}(\mathcal{E})$, avec $\mathcal{N}(\mathcal{E}) = \mathcal{N}(\Pi) \cup \mathcal{N}_\epsilon(\Pi) \cup \mathcal{N}(tr)$. Ainsi, le seul type qui puisse être attribué à t est ω.

Cas $t \in \mathcal{C}$.
Nous distinguons alors deux cas disjoints.

– $t \in \mathcal{C}(\Pi)$. Alors il existe $i \in [\![m]\!]$ tel que $t = c_i$, et d'après la quatrième règle de la figure 7.1, pour $\tau = \gamma_i$ nous avons bien $\mathcal{E} \vdash t : \tau$. Aussi, en inspectant les autres règles de typage il est aisé de voir que c'est là l'unique règle qui puisse être invoquée afin d'attribuer un type à t et que donc τ est unique.

– $t \notin \mathcal{C}(\Pi)$. Alors l'unique règle qui puisse être appliquée pour lui attribuer un type est la règle cinq de la figure 7.1. De plus, celle-ci ne s'applique qu'aux constantes de $\mathcal{C} \setminus \mathcal{C}(\mathcal{E})$, avec $\mathcal{C}(\mathcal{E}) = \mathcal{C}(\Pi) \cup \mathcal{C}(tr)$ ($= \mathcal{C}(\Pi)$ comme nous avons déjà eu l'occasion de le voir). Ainsi, le seul type qui puisse être attribué à t est ω.

Cas $t = f(t_1, \ldots, t_n)$ avec $f \in \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$.
Pour chacun des t_i ($i \in [\![n]\!]$) nous avons $|\mathcal{V}(t_i)| \leq |\mathcal{V}(t)|$ et $|t_i| < |t|$, soit $M(t_i) <_{\mathsf{lex}} M(t)$. Nous pouvons donc appliquer notre hypothèse d'induction à chacun des t_i et conclure que pour chacun d'eux il existe un et un seul type τ_i tel que $\mathcal{E} \vdash t_i : \tau_i$. Or, la seule règle qui puisse être appliquée pour typer t étant la sixième règle de la figure 7.1, il existe un unique $\tau = f(\tau_1, \ldots, \tau_n)$ tel que $\Pi \vdash t : \tau$. □

Finalement nous vérifions que les types attribués aux variables de ϕ sont cohérents avec ceux des status events de Π.

Lemme 7.1.15. *Soit* $\mathsf{Q}(u_1, \ldots, u_r) \in \mathsf{SubForm}^-(\phi)$. *Si* $\mathcal{E} \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_r$, *alors pour tout* $k \in [\![r]\!]$, $\mathcal{E} \vdash u_k : \tau_k$.

Démonstration. Soit $\mathsf{Q}(u_1, \ldots, u_r) \in \mathsf{SubForm}^-(\phi)$ tel que $\mathcal{E} \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_r$. Par construction de Γ_ϕ nous savons que $\{u_1 \stackrel{?}{:} \tau_1, \ldots, u_r \stackrel{?}{:} \tau_r\} \subseteq \Gamma_\phi$ et par hypothèse sur ϕ, nous savons que Γ_ϕ admet la solution Σ_ϕ, et plus précisément qu'il existe une dérivation $\Gamma_\phi = \Gamma_0 \rightsquigarrow \Gamma_1 \rightsquigarrow \cdots \rightsquigarrow \Gamma_m = \Sigma_\phi$. Afin de pouvoir conclure que pour tout $k \in [\![r]\!]$, $\mathcal{E} \vdash u_k : \tau_k$, nous allons montrer le résultat suivant : $\forall i \in [\![m]\!]. \forall t \stackrel{?}{:} \tau \in \Gamma_i.\ \mathcal{E} \vdash t : \tau$. Nous procédons par induction sur $m - i$.

Cas de base : $m - i = 0$. Dans ce cas, $i = m$ et $\Gamma_i = \Sigma_\phi$, et donc pour tout $t \stackrel{?}{:} \tau \in \Gamma_i$, nous savons que $t \in \mathcal{V}(\phi)$ et $t \stackrel{?}{:} \tau \in \Sigma_\phi$. Ainsi, d'après la première règle de la figure 7.4 $\mathcal{E} \vdash t : \tau$.

Cas inductif : $m - i \geq 1$. Nous procédons par analyse de cas sur la règle R impliquée dans la simplification $\Gamma_i \overset{\mathsf{R}}{\rightsquigarrow} \Gamma_{i+1}$.

Cas $\mathsf{R} = \mathsf{R}_1$.
Il existe alors un problème de typage Γ et une entité $p \in \mathcal{P}$ tels que $\Gamma_i = \Gamma \cup \{p \overset{?}{:} \alpha\}$ et $\Gamma_{i+1} = \Gamma$. Soit $t \overset{?}{:} \tau \in \Gamma_i$, nous distinguons deux cas :
- $t \overset{?}{:} \tau \in \Gamma = \Gamma_{i+1}$. Nous savons alors par hypothèse d'induction que $\mathcal{E} \vdash t : \tau$.
- $t \overset{?}{:} \tau \notin \Gamma$. Alors nous savons que $t = p(avec p \in \mathcal{P})$ et $\tau = \alpha$. Or d'après la première règle de typage de la figure 7.1 $\mathcal{E} \vdash p : \alpha$, et donc $\mathcal{E} \vdash t : \tau$.

Cas $\mathsf{R} = \mathsf{R}_2$.
Il existe alors un problème de typage Γ, une constante $c \in \mathcal{C}$ et un type τ' tels que $\mathcal{E} \vdash c : \tau'$, $\Gamma_i = \Gamma \cup \{c \overset{?}{:} \tau'\}$ et $\Gamma_{i+1} = \Gamma$. Soit $t \overset{?}{:} \tau \in \Gamma_i$, nous distinguons deux cas :
- $t \overset{?}{:} \tau \in \Gamma = \Gamma_{i+1}$. Nous savons alors par hypothèse d'induction que $\mathcal{E} \vdash t : \tau$.
- $t \overset{?}{:} \tau \notin \Gamma$. Alors nous savons que $t = c(avec c \in \mathcal{C})$ et que $\tau = \tau'$. Or par hypothèse d'application de la règle R_2 nous savons que $\mathcal{E} \vdash c : \tau'$, et donc que $\mathcal{E} \vdash t : \tau$.

Cas $\mathsf{R} = \mathsf{R}_3$.
Il existe alors un problème de typage Γ, un terme $f(t_1, \ldots, t_n)$, et un type $f(\tau_1, \ldots, \tau_n)$ avec $f \in \{\mathsf{pvk}, \mathsf{shk}\}$, tels que $\tau_1 = \alpha, \ \ldots, \tau_n = \alpha$, $\Gamma_i = \Gamma \cup \{f(t_1, \ldots, t_n) \overset{?}{:} f(\tau_1, \ldots, \tau_n)\}$, et $\Gamma_{i+1} = \Gamma \cup \{t_1 \overset{?}{:} \alpha, \ldots, t_n \overset{?}{:} \alpha)\}$. Soit $t \overset{?}{:} \tau \in \Gamma_i$, nous distinguons deux cas :
- $t \overset{?}{:} \tau \in \Gamma \subseteq \Gamma_{i+1}$. Nous savons alors par hypothèse d'induction que $\mathcal{E} \vdash t : \tau$.
- $t \overset{?}{:} \tau \notin \Gamma$. Alors nous savons que $t = f(t_1, \ldots, t_n)$, et que $\tau = f(\alpha, \ldots, \alpha)$. Or pour tout $k \in [\![n]\!]$, $t_i \overset{?}{:} \alpha \in \Gamma_{i+1}$ et par hypothèse d'induction nous concluons que $\mathcal{E} \vdash t_i : \alpha$. En appliquant la règle six de la figure 7.1 nous concluons que $\mathcal{E} \vdash f(t_1, \ldots, t_n) : f(\alpha, \ldots, \alpha)$, et donc que $\mathcal{E} \vdash t : \tau$.

Cas $\mathsf{R} = \mathsf{R}_4$.
Il existe alors un problème de typage Γ, un terme $f(t_1, \ldots, t_n)$, et un type $f(\tau_1, \ldots, \tau_n)$ avec $f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$, tels que $\Gamma_i = \Gamma \cup \{f(t_1, \ldots, t_n) \overset{?}{:} f(\tau_1, \ldots, \tau_n)\}$, et $\Gamma_{i+1} = \Gamma \cup \{t_1 \overset{?}{:} \tau_1, \ldots, t_n \overset{?}{:} \tau_n)\}$. Soit $t \overset{?}{:} \tau \in \Gamma_i$, nous distinguons deux cas :
- $t \overset{?}{:} \tau \in \Gamma \subseteq \Gamma_{i+1}$. Nous savons alors par hypothèse d'induction que $\mathcal{E} \vdash t : \tau$.
- $t \overset{?}{:} \tau \notin \Gamma$. Alors nous savons que $t = f(t_1, \ldots, t_n)$, et que $\tau = f(\tau_1, \ldots, \tau_n)$. Or pour tout $k \in [\![n]\!]$, $t_i \overset{?}{:} \tau_i \in \Gamma_{i+1}$ et par hypothèse d'induction nous concluons que $\mathcal{E} \vdash t_i : \tau_i$. En appliquant la règle six de la figure 7.1 nous concluons que $\mathcal{E} \vdash f(t_1, \ldots, t_n) : f(\tau_1, \ldots, \tau_n)$, et donc que $\mathcal{E} \vdash t : \tau$.

Ainsi, nous avons montré que pour tout $i \in [\![m]\!]$ et pour tout $t \overset{?}{:} \tau \in \Gamma_i$, $\mathcal{E} \vdash t :$

τ. En particulier, ceci est vrai pour $i = 0$, *i.e.* pour tout $t \overset{?}{:} \tau \in \Gamma_0 = \Gamma_\phi$, $\mathcal{E} \vdash t : \tau$. Or nous avons vu que par construction de Γ_ϕ pour tout $k \in [\![r]\!]$. $u_k \overset{?}{:} \tau_k \in \Gamma_\phi$, et donc $\mathcal{E} \vdash u_k : \tau_k$. □

7.2 La classe de protocoles $\mathcal{C}_2$

Le résultat de réduction qui sera énoncé à la section 7.4 n'est pas vrai en général, et comme son titre l'indique, cette section est dédiée à la définition des protocoles pour lesquels notre résultat est vrai. En effet, soient Π un protocole, ϕ une propriété de $\mathcal{PS}$-LTL$^-$, et k un entier, la question « est-ce que Π admet une exécution qui viole ϕ et n'implique que des messages de taille inférieure ou égale à k » est indécidable dans le cas général. Si nous considérons le protocole Π'_{Toy}, il n'est à priori (*i.e.* sans déjà connaître l'attaque) pas possible de savoir la profondeur de messages à considérer pour trouver l'attaque sur le secret. Rien dans sa spécification n'indique que si nous nous restreignons à des messages de taille inférieure ou égale à 7 alors ce protocole préserve la confidentialité, mais qu'en autorisant des messages de taille inférieure ou égale à 11 il divulgue son secret. Ici, nous définissons une classe de protocoles, que nous appellerons $\mathcal{C}_2$, pour laquelle si nous considérons certaines « bonnes » propriétés (que nous définirons juste après à la section 7.3), alors (comme nous le verrons à la section 7.4) il suffit de se restreindre à des messages dont la taille est bornée par une constante k pour la trouver.

Définition 7.2.1 (La classe $\mathcal{C}_2$)**.** *La classe de protocoles $\mathcal{C}_2$ est l'ensemble des protocoles $\Pi = [e_1; \ldots; e_\ell]$ vérifiant les trois propriétés suivantes :*

1. *Pour tout scénario* sc *de* Π *et toute formule* ϕ, Π *est bien typé dans l'environnement* $\langle \Pi, \mathsf{sc}, \phi \rangle$, i.e.

   $$\forall i, j \in [\![\ell]\!],\ \textit{si}\ e_i = \mathsf{Q}(p, u_1, \ldots, u_q)\ \textit{et}\ e_j = \mathsf{Q}(p', v_1, \ldots, v_r),$$
   $$\textit{alors}\ q = r\ \textit{et pour tout}\ k \in [\![q]\!]\ \textit{il existe un type}\ \tau_k\ \textit{tel que}$$
   $$\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u_k : \tau_k\ \textit{et}\ \langle \Pi, \mathsf{sc}, \phi \rangle \vdash v_k : \tau_k.$$

2. *Pour tout scénario* sc *de* Π *et toute formule* ϕ, Π *vérifie*

   $$\forall \mathsf{aenc}(u, v) \in \mathsf{Est}(\Pi).\ \langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \alpha.$$

3. *Pour tout scénario* sc *de* Π *et toute formule* ϕ, Π *vérifie*

   $$\forall u, v \in \mathsf{Est}(tr).\ \mathrm{mgu}(u, v) \neq \bot \Rightarrow \exists \tau.\ \langle \Pi, \mathsf{sc}, \phi \rangle \vdash u : \tau \wedge \langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \tau$$

   avec tr *la trace symbolique associée à* sc.

La première condition nous permettra comme nous l'avons vu à la section 7.1 précédente d'associer un type aux prédicats apparaissant dans les status events de Π. La deuxième condition stipule que tous les chiffrements asymétrique se font avec des clés du même type, et plus particulièrement du type α. La troisième condition restreint l'unifiabilité entre sous-termes chiffrés de la trace symbolique tr associée au scénario sc, à des termes du même type, *i.e.* si u et v sont unifiables alors ils sont du même type. Quoique quantifiées universellement sur l'ensemble infini des scenari de Π et l'ensemble infini de formules, ces trois conditions sont, comme le montre le lemme 7.2.4, décidables. Plus précisément, nous montrons pour tout protocole Π il est possible

de construire un scénario « témoin » $\mathsf{sc}^{\Pi}_{\mathsf{witness}}$ de Π tel qu'il suffise de vérifier que Π satisfait les conditions de la définition 7.2.1 dans le seul environnement de typage $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle$.

Définition 7.2.2 (Scénario témoin)**.** *Soit* Π *un protocole tel que* $\mathsf{Roles}(\Pi) = \{r_1, \ldots, r_\rho\}$*. Le scénario témoin de* Π *est dénoté* $\mathsf{sc}^{\Pi}_{\mathsf{witness}}$ *et est défini par* $\mathsf{sc}^{\Pi}_{\mathsf{witness}} = (\mathsf{interlvg}^{\Pi}_{\mathsf{witness}}, \mathsf{initagts}^{\Pi}_{\mathsf{witness}})$ *où*

$$\mathsf{interlvg}^{\Pi}_{\mathsf{witness}} = [\underbrace{(r_1, sid_1); \ldots; (r_1, sid_1)}_{|r_1|\ fois}; \ldots; \underbrace{(r_\rho, sid_\rho); \ldots; (r_\rho, sid_\rho)}_{|r_\rho|\ fois}]$$

et où pour tout $i \in [\![\rho]\!]$

$$\mathsf{initagts}^{\Pi}_{\mathsf{witness}}(sid_i) = (\underbrace{a, \ldots, a}_{|\mathcal{A}(\Pi)|\ fois})$$

avec $a \in \mathcal{A}$*.*

Nous avons ainsi défini un scénario dont la trace symbolique associée contiendra une instance de chaque événement spécifié dans Π. Nous dénoterons $tr^{\Pi}_{\mathsf{witness}}$ la trace symbolique associée à $\mathsf{sc}^{\Pi}_{\mathsf{witness}}$.

Informellement, l'intérêt d'avoir défini ce scénario est le suivant. Soient u et v deux sous-termes chiffrés de Π, et sc un scénario quelconque de Π, de trace symbolique associée tr. Supposons qu'il existe dans tr une instance u_1 de u ainsi qu'une instance v_1 de v. Par construction de $\mathsf{sc}^{\Pi}_{\mathsf{witness}}$ nous savons que dans $tr^{\Pi}_{\mathsf{witness}}$ aussi il existe une instance u_2 de u et une instance v_2 de v. Mais en ayant initialisé tous les agents de Π au seul et même agent a, et sachant que si u et v sont des sous-termes du même rôle alors u_2 et v_2 sont des sous-termes de la même sessions nous aurons nécessairement que, si u_1 et v_1 sont unifiables alors u_2 et v_2 le sont nécessairement aussi. Ainsi pour tester la non-unifiabilité des sous-termes chiffrés de tr, il suffira de tester la non-unifiabilité des sous-termes chiffrés correspondants de $tr^{\Pi}_{\mathsf{witness}}$.

Exemple 7.2.3. *Le scénario témoin* $\mathsf{sc}^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}}$ *de* Π'_{Toy} *est*

$$\mathsf{sc}^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}} = [(r'_a, sid_1); (r'_a, sid_1); (r'_a, sid_1); (r'_b, sid_2); (r'_b, sid_2)].$$

La trace symbolique associée à ce scénario $\mathsf{sc}^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}}$ *est la suivante :*

$$\begin{aligned} tr^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}} = [\ & \mathsf{snd}(a, a, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, a), a \rangle, a)); \\ & \mathsf{Secret}(a, a, n^1); \\ & \mathsf{rcv}(a, a, \mathsf{aenc}(\langle \mathsf{aenc}(n^1, a), a \rangle, a)); \\ & \mathsf{rcv}(a, a, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, a), a \rangle, a)); \\ & \mathsf{snd}(a, a, \mathsf{aenc}(\langle \mathsf{aenc}(x^2, a), a \rangle, a))] \end{aligned}$$

Notons que les instances dans $tr^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}}$*,*

$$\mathsf{aenc}(\langle \mathsf{aenc}(n^1, a), a \rangle, a) \quad \textit{et} \quad \mathsf{aenc}(\langle \mathsf{aenc}(x^2,), a \rangle, a)$$

des sous-termes

$$\mathsf{aenc}(\langle \mathsf{aenc}(n, b), a \rangle, b) \quad \textit{et} \quad \mathsf{aenc}(\langle \mathsf{aenc}(x, a), b \rangle, a)$$

respectivement de Π'_{Toy} *sont unifiables, alors que les instances de ces deux mêmes termes dans une session où les agents* a *et* b *sont initialisés à des agents distincts ne le sont pas.*

Lemme 7.2.4. *Soit* $\Pi = [e_1; \ldots; e_\ell]$ *un protocole,*

- Π *vérifie la condition 1 de la définition 7.2.1 si et seulement si* Π *vérifie la condition suivante :*

  1'. $\forall i, j \in [\![\ell]\!]$, *si* $e_i = \mathsf{Q}(p, u_1, \ldots, u_q)$ *et* $e_j = \mathsf{Q}(p', v_1, \ldots, v_r)$, *alors* $q = r$ *et pour tout* $k \in [\![q]\!]$ *il existe un type* τ_k *tel que*
  $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash u_k : \tau_k$ *et* $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash v_k : \tau_k$.
- Π *vérifie la condition 2 de la définition 7.2.1 si et seulement si* Π *vérifie la propriété suivante :*

  2'. $\forall \mathsf{aenc}(u, v) \in \mathsf{Est}(\Pi).\ \langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash v : \alpha$.
- Π *vérifie la condition 3 de la définition 7.2.1 si et seulement si* Π *vérifie la propriété suivante :*

  3'. $\forall u, v \in \mathsf{Est}(tr^{\Pi}_{\mathsf{witness}}).\ \mathrm{mgu}(u, v) \neq \bot \Rightarrow$
  $\exists \tau.\ \langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash u : \tau \wedge \langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash v : \tau$.

Démonstration. Avant de procéder aux preuves de ces équivalences nous allons établir que pour tous scenari sc_1, sc_2 de Π, pour toutes formules ϕ_1, ϕ_2 de $\mathcal{PS}$-LTL$^-$, ainsi que pour tout terme $t \in \mathsf{St}(\Pi)$ et tout type τ,

$$\langle \Pi, \mathsf{sc}_1, \phi_1 \rangle \vdash t : \tau \text{ si et seulement si } \langle \Pi, \mathsf{sc}_2, \phi_2 \rangle \vdash t : \tau. \qquad (\dagger)$$

Nous procédons par induction sur la taille du terme t.
Cas $t \in \mathcal{A}$, $t \in \mathcal{N}$, $t \in \mathcal{C}$ *ou* $t \in \mathcal{V}$. Nous détaillons le cas $t \in \mathcal{A}$, les autres pouvant être traités de manière analogue. La seule règle qui puisse être appliquée ne dépend ni du scénario, ni de la formule de l'environnement de typage (il s'agît de la règle un de la figure 7.1), ainsi dans chacun des deux environnements de typage le type attribué a t est α.
Cas $t = f(t_1, \ldots, t_n)$ *avec* $f \in \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$ *pour certains* $t_1, \ldots, t_n \in \mathcal{T}$. Par hypothèse d'induction nous savons que pour tout type τ_i, $\langle \Pi, \mathsf{sc}_1, \phi_1 \rangle \vdash t_i : \tau_i$ si et seulement si $\langle \Pi, \mathsf{sc}_2, \phi_2 \rangle \vdash t_i : \tau_i$. Etant donné que indépendemment du scénario et de la formule de l'environnement de typage la seule règle qui puisse être appliquée est la règle six de la figure 7.1, nous pouvons conclure que pour tous types $\tau_1, \ldots, \tau_n$, $\langle \Pi, \mathsf{sc}_1, \phi_1 \rangle \vdash t : f(\tau_1, \ldots, \tau_n)$ si et seulement si $\langle \Pi, \mathsf{sc}_2, \phi_2 \rangle \vdash t : f(\tau_1, \ldots, \tau_n)$.

Les implications (***1*** $\Rightarrow$ ***1'***), (***2*** $\Rightarrow$ ***2'***), et (***3*** $\Rightarrow$ ***3'***) sont évidentes. En effet, *1'* n'est qu'une instance de *1*, *2'* n'est qu'une instance de *2*, et *3'* n'est qu'une instance de *3*. Donc, si Π vérifie *1*, *2* ou *3*, alors nécessairement Π vérifie *1'*, *2'* ou *3'* respectivement.

(***1'*** $\Rightarrow$ ***1***). Supposons que Π satisfasse *1'* mais pas *1*, *i.e.* qu'il existe un scénario sc de Π, et une formule $\phi \in \mathcal{PS}$-LTL$^-$, ainsi que $i, j \in [\![\ell]\!]$, tels que $e_i = \mathsf{Q}(p, u_1, \ldots, u_n)$ et $e_j = \mathsf{Q}(p', v_1, \ldots, v_n)$ mais que $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u_k : \tau_k$ et $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash v_k : \tau'_k$ avec $\tau_k \neq \tau'_k$ pour un certain $k \in [\![n]\!]$ et certains types τ_k et τ'_k. Or, l'énoncé ($\dagger$) que nous avons établi au tout début de cette preuve implique que $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash u_k : \tau_k$ et $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash v_k : \tau'_k$, ce qui vient contredire l'hypothèse faite selon laquelle Π vérifie *1'*. Nous concluons donc que (*1'* $\Rightarrow$ *1*).

(***2'*** $\Rightarrow$ ***2***). Supposons que Π vérifie *2'* mais pas *2*, *i.e.* qu'il existe un scénario sc de Π, une formule ϕ de $\mathcal{PS}$-LTL$^-$ et un terme $\mathsf{aenc}(u, v) \in \mathsf{Est}(\Pi)$ tels que

$\langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \tau$ avec $\tau \neq \alpha$ pour un certain type τ. Or, d'après l'énoncé (†) établi en début de preuve. Cela implique alors que $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash v : \tau$, ce qui vient contredire l'hypothèse faite selon laquelle Π satisfait *2'*. Nous concluons donc que (*2'* $\Rightarrow$ *2*).

(***3'*** $\Rightarrow$ ***3***). Supposons que Π vérifie *3'* mais pas *3*, *i.e.* qu'il existe un scénario sc de Π, une formule ϕ de $\mathcal{PS}$-LTL$^-$ et deux termes $u, v \in \mathsf{Est}(tr)$, où tr est la trace symbolique associée à sc, tels que $\mathrm{mgu}(u, v) \neq \bot$ et tels que $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u : \tau$ et $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \tau'$ avec $\tau \neq \tau'$ pour certains types τ et τ'. Mais comme nous l'expliquions un peu plus haut, cela implique qu'il existe deux termes $u', v' \in \mathsf{Est}(\Pi)$ tels que $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u' : \tau$ et $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash v' : \tau'$. De plus par construction de $\mathsf{sc}^{\Pi}_{\mathsf{witness}}$ nous savons qu'il existe alors deux termes $u'', v'' \in \mathsf{St}(tr^{\Pi}_{\mathsf{witness}})$, tels que $\mathrm{mgu}(u'', v'') \neq \bot$, et tels que $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash u'' : \tau$ et $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash v'' : \tau'$, ce qui vient contredire l'hypothèse faite selon laquelle Π satisfait *3'*. Nous concluons donc que (*3'* $\Rightarrow$ *3*). □

Il est clair que les conditions *1'*, *2'* et *3'* sont quant à elles décidables. Nous sommes donc en mesure de conclure à la décidabilité de l'appartenance d'un protocole Π à la classe $\mathcal{C}_2$. De plus, l'appartenance d'un protocole à la classe $\mathcal{C}_2$ est indépendante de la propriété de sécurité considérée.

De la preuve du lemme 7.2.4 (et en particulier de la preuve de l'implication (*1'* $\Rightarrow$ *1*)) découle aussi que la propriété de « bon typage » d'un protocole, ne dépend que de la spécification du protocoles, et aucunement du scénario ou de la formule de l'environnement de typage considéré. Nous pouvons donc dire qu'un protocole est bien typé dans l'absolu, *i.e.* sans préciser d'environnement de typage.

Exemple 7.2.5. *Clairement notre protocole* Π'_{Toy} *n'est pas dans* $\mathcal{C}_2$. *Soient les deux sous-termes chiffrés de la trace* $tr^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}}$ *définie à l'exemple 7.2.3,* $u = \mathsf{aenc}(x^2, a)$ *et* $v = \mathsf{aenc}(\langle \mathsf{aenc}(x^2, a), a \rangle, a)$. *Ces deux termes sont unifiables* ($\mathrm{mgu}(u, v) = \{x^2 \mapsto \langle \mathsf{aenc}(x^2, a), a \rangle\} \neq \bot$). *Or,*

$$\langle \Pi'_{\mathsf{Toy}}, \mathsf{sc}^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash u : \mathsf{aenc}(\nu, \alpha)$$

mais,

$$\langle \Pi'_{\mathsf{Toy}}, \mathsf{sc}^{\Pi'_{\mathsf{Toy}}}_{\mathsf{witness}}, \mathsf{true} \rangle \vdash v : \mathsf{aenc}(\langle \mathsf{aenc}(\nu, \alpha), \alpha \rangle, \alpha).$$

7.3 La classe de propriétés Φ_2

Comme nous l'annoncions en introduction de la section 7.2, le résultat de réduction présenté à la section 7.4 n'est pas vrai en général. De plus, la classe de propriétés pour laquelle ce résultat tient est fonction du protocole considéré. En particulier, il se peut qu'une propriété soit dans la classe des propriétés pour laquelle le résultat de réduction est vrai pour un certain protocole, mais pas pour un autre. C'est à la définition de ces « bonnes » propriétés qu'est consacrée cette section.

Définition 7.3.1 (La classe $\Phi_2(\Pi)$)**.** *Soit* $\Pi = [e_1; \ldots; e_\ell]$ *un protocole bien typé*[2]*. La classe de formules* $\Phi_2(\Pi) \subseteq \mathcal{PS}\text{-}LTL^-$ *est l'ensemble des formules* ϕ *vérifiant les cinq propriétés suivantes :*

1. *Pour tout scénario* sc *de* Π*,* ϕ *est bien typée dans l'environnement* $\langle \Pi, \mathsf{sc}, \phi \rangle$, i.e. ϕ *satisfait les propriétés suivantes :*
   a) $\forall \mathsf{Q}(t_1, \ldots, t_q) \in \mathsf{SubForm}^-(\phi).\ \exists i \in [\![\ell]\!].\ e_i = \mathsf{Q}(p, s_1, \ldots, s_q)$,
   b) $\forall \mathsf{learn}(t) \in \mathsf{SubForm}(\phi).\ \forall x \in \mathcal{V}(t).\ \exists \mathsf{Q}(t_1, \ldots, t_p) \in \mathsf{SubForm}^-(\phi).\ \exists\ i \in [\![p]\!].\ x \in \mathcal{V}(t_i)$*, et*
   c) le problème de typage

$$\Gamma_\phi \quad = \bigcup_{\substack{\mathsf{Q}(t_1,\ldots,t_r) \in \mathsf{SubForm}^-(\phi) \\ \langle \Pi, \mathsf{sc}, \phi \rangle \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_r}} \{t_1 \overset{?}{:} \tau_1, \ldots, t_r \overset{?}{:} \tau_r\}$$
$$\cup \bigcup_{\mathsf{C}(t) \in \mathsf{SubForm}^-(\phi)} \{t \overset{?}{:} \alpha\}$$

   admet une solution Σ_ϕ*.*

2. *Pour tout scénario* sc *de* Π*,* $\forall \mathsf{aenc}(u, v) \in \mathsf{Est}^-(\phi).\ \langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \alpha$.
3. *Pour tout scénario* sc *de* Π*,* ϕ *vérifie :*
   $\forall u, v \in \mathsf{Est}^-(\phi).\ \mathrm{mgu}(u, v) \neq \bot \Rightarrow \exists \tau. \langle \Pi, \mathsf{sc}, \phi \rangle \vdash u : \tau \wedge \langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \tau$.
4. *Pour tout scénario* sc *de* Π
   $\forall u \in \mathsf{Est}(tr). \forall v \in \mathsf{Est}^-(\phi).\ \mathrm{mgu}(u, v) \neq \bot \Rightarrow \exists \tau.\ \langle \Pi, \mathsf{sc}, \phi \rangle \vdash u : \tau \wedge \langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \tau$
   avec tr *la trace symbolique associée au scénario* sc*.*
5. *Pour tout scénario* sc *de* Π*,*
   $\forall \mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{SubForm}^+(\phi)$*, s'il existe* $\mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{Elmts}(\Pi)$ *et que* $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_n$*, alors pour tout* $i \in [\![n]\!]$ $\mathsf{pvk}(\alpha) \notin \mathsf{SubType}(\tau_i)$ *et pour tout type constant* $\gamma \neq \omega$*,* i.e. $c \in \mathcal{C}(\Pi)$ *telle que* $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash c : \gamma$*,* $\gamma \notin \mathsf{SubType}(\tau_i)$

Comme au chapitre précédent, la plupart de ces conditions porte sur les sous-formules négatives de ϕ ainsi que sur les sous-termes chiffrés apparaissant dans ces dernières ; le but étant là encore de contrôler les unifications calculées au cours d'un appel à la procédure **D**.

Les conditions 1, 2, 3, et 4 sont quantifiées universellement sur l'ensemble infini de scenari de Π. Il est néanmoins possible, ici aussi, d'énoncer des conditions équivalentes et décidables. Définissons cette fois un ensemble de scénarios témoins

$$\mathsf{Scenarios}_{\mathsf{W}}(\Pi, \phi) = \{(\mathsf{interlvg}^{\Pi}_{\mathsf{witness}}, \mathsf{initagts}) \mid \forall sid.\ \mathsf{initagts}(sid) \in (\mathcal{A}(\phi) \cup \{a\})^{|\mathcal{A}(\Pi)|}\}.$$

Des noms d'agents pouvant apparaître dans ϕ, il n'est pas suffisant de tester les conditions de la définition 7.3.1 sur le seul scénario $\mathsf{sc}_{\mathsf{witness}}$. Par contre, il est clairement suffisant de se restreindre à des scénarios dans lesquels les agents sont initialisés à des agents apparaissant dans ϕ. Nous avons tout de

[2]Rappelons que la propriété de bon typage pour un protocole ne dépend que de sa spécification, et que donc dire d'un protocole qu'il est bien typé sans préciser dans quel environnement de typage est cohérent.

même considéré l'agent a pour traiter les formules dans lesquelles aucun agent n'apparaît. C'est le cas par exemple des formules présentées à la section 3.2.

Lemme 7.3.2. *Soient* $\Pi = [e_1; \ldots; e_\ell]$ *un protocole bien typé, et* ϕ *une formule de* $\mathcal{PS}$*-LTL*$^-$ *tels que*

1. $\forall \mathsf{Q}(t_1, \ldots, t_q) \in \mathsf{SubForm}^-(\phi). \exists i \in [\![\ell]\!].\ e_i = \mathsf{Q}(p, s_1, \ldots, s_q)$,
2. $\forall \mathsf{learn}(t) \in \mathsf{SubForm}(\phi).\ \forall x \in \mathcal{V}(t).\ \exists \mathsf{Q}(t_1, \ldots, t_p) \in \mathsf{SubForm}^-(\phi).\ \exists i \in [\![p]\!].\ x \in \mathcal{V}(t_i)$,

alors

– ϕ *vérifie la condition 1 de la définition 7.3.1 si et seulement si* ϕ *vérifie la condition suivante :*

1'. le problème de typage

$$\Gamma_\phi \quad = \bigcup_{\substack{\mathsf{Q}(t_1,\ldots,t_r) \in \mathsf{SubForm}^-(\phi) \\ \langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \phi\rangle \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_r}} \{t_1 \overset{?}{:} \tau_1, \ldots, t_r \overset{?}{:} \tau_r\}$$
$$\cup \bigcup_{\mathsf{C}(t) \in \mathsf{SubForm}^-(\phi)} \{t \overset{?}{:} \alpha\}$$

admet une solution Σ_ϕ.

– ϕ *vérifie la condition 2 de la définition 7.3.1 si et seulement si* ϕ *vérifie la condition suivante :*

2'. $\forall \mathsf{aenc}(u, v) \in \mathsf{Est}(\phi).\ \langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \phi\rangle \vdash v : \alpha$.

– ϕ *vérifie la condition 3 de la définition 7.3.1 si et seulement si* ϕ *vérifie la condition suivante :*

3'. $\forall u, v \in \mathsf{Est}^-(\phi).\ \mathrm{mgu}(u, v) \neq \bot \Rightarrow \exists \tau.\ \langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \phi\rangle \vdash u : \tau \wedge \langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \phi\rangle \vdash v : \tau$.

– ϕ *vérifie la condition 4 de la définition 7.3.1 si et seulement si* ϕ *vérifie la condition suivante. Pour tout scénario* $\mathsf{sc} \in \mathsf{Scenarios}_{\mathsf{W}}(\Pi, \phi)$ *de trace symbolique associée tr*

4'. $\forall u \in \mathsf{Est}(tr).\ \forall v \in \mathsf{Est}^-(\phi).\ \mathrm{mgu}(u, v) \neq \bot \Rightarrow \exists \tau.\ \langle \Pi, \mathsf{sc}, \phi\rangle \vdash u : \tau \wedge \langle \Pi, \mathsf{sc}, \phi\rangle \vdash v : \tau$

– ϕ *vérifie la condition 5 de la définition 7.3.1 si et seulement si* ϕ *vérifie la condition suivante :*

5'. $\forall \mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{SubForm}^+(\phi)$, *s'il existe* $\mathsf{Q}(u_1, \ldots, u_n) \in \mathsf{Elmts}(\Pi)$ *et que* $\langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \phi\rangle \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_n$, *alors pour tout* $i \in [\![n]\!]$ $\mathsf{pvk}(\alpha) \notin \mathsf{SubType}(\tau_i)$ *et pour tout type constant* $\gamma \neq \omega$, i.e. $c \in \mathcal{C}(\Pi)$ *telle que* $\langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \phi\rangle \vdash c : \gamma$, $\gamma \notin \mathsf{SubType}(\tau_i)$

Démonstration.
Les implications (***1*** $\Rightarrow$ ***1'***), (***2*** $\Rightarrow$ ***2'***), (***3*** $\Rightarrow$ ***3'***), (***4*** $\Rightarrow$ ***4'***), et (***5*** $\Rightarrow$ ***5'***) sont évidentes. En effet, *1'* n'est qu'une instance de *1*, *2'* une instance de *2*, *3'* une instance de *3*, *4'* une instance de *4*, et *5'* une instance de *5*. Donc, si Π vérifie *1*, *2*, *3*, *4* , ou *5* alors nécessairement Π vérifie *1'*, *2'*, *3'*, *4'*, ou *5'* respectivement.

(***1'*** $\Rightarrow$ ***1***). Supposons que ϕ satisfasse *1'* et que Γ_ϕ soit le problème de typage correspondant, mais qu'il existe un scénario sc de Π et une formule ϕ tels que le problème de typage

$$\begin{aligned}\Gamma'_\phi \quad = \quad & \bigcup_{\substack{\mathsf{Q}(t_1,\ldots,t_r) \in \mathsf{SubForm}^-(\phi) \\ \langle \Pi, \mathsf{sc}, \phi\rangle \vdash \mathsf{Q} : \tau_1 \times \cdots \times \tau_r}} \{t_1 \overset{?}{:} \tau_1, \ldots, t_r \overset{?}{:} \tau_r\} \\ \cup \quad & \bigcup_{\mathsf{C}(t) \in \mathsf{SubForm}^-(\phi)} \{t \overset{?}{:} \alpha\}\end{aligned}$$

n'admette pas de solution, *i.e.* il n'existe pas de Δ en forme résolue tel que $\Gamma_\phi \rightsquigarrow^* \Delta$. Or, de l'énoncé (†) établi dans la preuve du lemme 7.2.4 découle que pour tout prédicat Q et tout type τ, $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash \mathsf{Q} : \tau$ si et seulement si $\langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \mathsf{true}\rangle \vdash \mathsf{Q} : \tau$; et donc que $\Gamma_\phi = \Gamma'_\phi$. Aussi, en observant les règles de simplification décrites à la figure 7.3 nous constatons que toute séquence de réductions dans l'environnement $\langle \Pi, \mathsf{sc}, \phi\rangle$ est aussi une séquence de simplifications dans l'environnement de typage $\langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \mathsf{true}\rangle$. En effet, l'environnement de typage n'intervient que lorsque l'expression sélectionnée est de la forme $c : \tau$ avec $c \in \mathcal{C}$, or le type de c ne dépend que de Π et aucunement du scénario ou de la formule de l'environnement de typage considéré. Ainsi le fait que Γ_ϕ n'admette pas de solution dans l'environnement $\langle \Pi, \mathsf{sc}, \phi\rangle$ vient contredire le fait qu'il en admette une dans l'environnement de typage $\langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \mathsf{true}\rangle$, et donc vient contredire l'hypothèse faite selon laquelle ϕ satisfait *1'*. Nous concluons donc que (*1'* $\Rightarrow$ *1*).

Avant de continuer avec la preuve à proprement parler nous allons montrer que pour tous scenari sc_1, sc_2 de Π, pour tout terme $t \in \mathsf{St}(\phi)$, et pour tout type τ,

$$\langle \Pi, \mathsf{sc}_1, \phi\rangle \vdash t : \tau \text{ si et seulement si } \langle \Pi, \mathsf{sc}_2, \phi\rangle \vdash t : \tau. \qquad (\ddagger)$$

Nous procédons par induction sur la taille du terme t. Si $t \in \mathcal{A}$ ou $t \in \mathcal{C}$, le type de t ne dépend pas du scénario de l'environnement de typage et donc l'équivalence est trivialement vraie. Supposons que $t \in \mathcal{V}(\phi)$, nous venons de voir que le problème de typage associé à ϕ est le même que nous considérions le scénario sc_1 ou le scénario sc_2. De plus, nous venons de voir que quel que soit le scénario considéré sc_1 ou sc_2, la solution est la même. Nous pouvons donc conclure que pour $t \in \mathcal{V}(\phi)$, le type que t se voit attribuer est le même indifféremment du scénario dans l'environnement de typage. Le cas $t \in \mathcal{N}$ ne pouvant par définition pas survenir, il nous reste à étudier le cas où t est un terme composé. Or, dans ce cas nous pouvons conclure par hypothèse d'induction.

(***2'*** $\Rightarrow$ ***2***). Supposons que Π vérifie *2'* mais pas *2*, *i.e.* qu'il existe un scénario sc de Π et un terme $\mathsf{aenc}(u, v) \in \mathsf{Est}(\phi)$ tels que $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash v : \tau$ avec $\tau \neq \alpha$ pour un certain type τ. Alors d'après l'équivalence (‡) que nous venons de montrer $\langle \Pi, \mathsf{sc}^\Pi_{\mathsf{witness}}, \phi\rangle \vdash v : \tau$. Or, cela vient contredire l'hypothèse faite selon laquelle Π satisfait *2'*. Nous concluons donc que (*2'* $\Rightarrow$ *2*).

(***3'*** $\Rightarrow$ ***3***). Supposons que ϕ satisfasse la condition *3'* mais pas la *3*, *i.e.* qu'il existe un scénario sc de Π ainsi que deux termes $u, v \in \mathsf{Est}(\phi)$ et deux types τ,

τ' tels que $\tau \neq \tau'$, $\mathrm{mgu}(u, v) \neq \bot$, $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u : \tau$, et $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash v : \tau'$. Mais là encore d'après l'équivalence (‡) établie un peu plus haut, cela implique alors que $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \phi \rangle \vdash u : \tau$, et $\langle \Pi, \mathsf{sc}^{\Pi}_{\mathsf{witness}}, \phi \rangle \vdash v : \tau'$, ce qui vient contredire l'hypothèse faite selon laquelle ϕ satisfait *3'*. Nous concluons donc que (*3'* ⇒ *3*).

(***4'*** ⇒ ***4***). Supposons que ϕ satisfasse *4'* mais pas *4*, *i.e.* qu'il existe un scénario de Π $\mathsf{sc} \notin \mathsf{Scenarios}_{\mathsf{W}}(\Pi, \phi)$, ainsi qu'un terme $u \in \mathsf{Est}(tr)$ où tr est la trace symbolique associée à sc, et un terme $v \in \mathsf{Est}^{-}(\phi)$ tels que $\mathrm{mgu}(u, v) \neq \bot$, $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u : \tau$ et $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u : \tau'$ avec $\tau \neq \tau'$ pour certains types τ et τ'. Or, par définition de notre système de types, et en particulier des types associés aux atomes de tr cela implique qu'il existe un terme $u' \in \mathsf{Est}(\Pi)$ tel que $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash u' : \tau$. De plus, par construction de l'ensemble de scenari $\mathsf{Scenarios}_{\mathsf{W}}(\Pi, \phi)$ nous savons qu'il existe alors un scénario $\mathsf{sc}' \in \mathsf{Scenarios}_{\mathsf{W}}(\Pi, \phi)$ de trace symbolique associée tr' ainsi qu'un terme $u'' \in \mathsf{Est}(tr)$ tels que $\langle \Pi, \mathsf{sc}', \phi \rangle \vdash u'' : \tau$ et $\mathrm{mgu}(u'', v) \neq \bot$, ce qui vient contredire l'hypothèse faite selon laquelle ϕ satisfait *4'*. Nous concluons donc que (*4'* ⇒ *4*).

(***5'*** ⇒ ***5***). Supposons que ϕ vérifie *5'* mais pas *5*, *i.e.* qu'il existe un scénario sc de Π, ainsi que $\mathsf{Q}(t_1, \ldots, t_n) \in \mathsf{SubForm}^{+}(\phi)$ tel que Q apparaisse dans Π tel que $\langle \Pi, \mathsf{sc}, \phi \rangle \vdash \mathsf{Q}(\tau_1, \ldots, t_n)$ et qu'il existe $i \in [\![n]\!]$ tel que $\mathsf{pvk}(\alpha) \in \mathsf{SubType}(\tau_i)$ (ou $\gamma \in \mathsf{SubType}(\tau_i)$ avec $\gamma \neq \omega$ le type associé à une constante). En mimant les arguments exhibés pour établir que ***(1')*** ⇒ ***(1)***, à savoir que le type d'un prédicat est le même quelque soit le scénario de l'environnement de typage considéré, on arrive à un contradiction quant à l'hypothèse faite selon laquelle ϕ satisfait *5'*. Nous concluons donc que (*5'* ⇒ *5*). □

Exemple 7.3.3. *D'après l'exemple 7.1.7 nous savons que* Π'_{Toy} *est bien typé. De même, d'après l'exemple 7.1.10 nous savons que* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *est bien typée. De plus,* $\mathsf{Est}(\phi^{\mathsf{S}}_{\mathsf{Toy}}) = \emptyset$ *implique trivialement que les conditions 2, 3, et 4 de la définition 7.3.1 sont satisfaites. Finalement, du fait qu'aucun status event n'apparaît dans* $\mathsf{SubForm}^{+}(\phi^{\mathsf{S}}_{\mathsf{Toy}})$*, nous déduisons que* $\phi^{\mathsf{S}}_{\mathsf{Toy}}$ *vérifie la condition 5 de la définition 7.3.1. Nous concluons donc que* $\phi^{\mathsf{S}}_{\mathsf{Toy}} \in \Phi_2(\Pi'_{\mathsf{Toy}})$.

7.4 Réduction à des exécutions « bien typées »

A ce stade du chapitre, nous avons aussi bien défini le système de types considéré, que les classes de protocoles et de formules pour lesquelles le résultat de réduction que nous visons tient. Il nous manque néanmoins la définition d'une exécution « bien typée », notion qui comme l'indique le titre de cette section n'en demeure pas moins cruciale. Rappelons le, une exécution est une instantiation d'une trace symbolique associée à un scénario du protocole considéré. Informellement, elle sera bien typée si toute variable de la trace symbolique sous-jacente, et le terme par lequel elle est instanciée dans l'exécution en question sont du même type. Une fois ces définitions posées formellement nous serons en mesure d'énoncer notre résultat de réduction ainsi que de procéder à sa preuve.

Définition 7.4.1 (Substitution bien typée)**.** *Soit* $\mathcal{E} = \langle \Pi, \mathsf{sc}, \phi \rangle$ *un environnement de typage. Une substitution* σ *est* bien typée dans l'environnement $\mathcal{E}$

si :

$$\forall x \in \mathrm{dom}(\sigma).\ \forall \tau.\ \mathcal{E} \vdash x : \tau \ \Leftrightarrow \mathcal{E} \vdash \sigma(x) : \tau.$$

Définition 7.4.2 (Trace bien typée)**.** *Soient* Π *un protocole,* sc *un scénario,* T_0 *un ensemble de termes atomiques clos, et* ϕ *une formule. Soit* $\mathcal{E} = \langle \Pi, \mathsf{sc}, \phi \rangle$ *l'environnement de typage associé.* tr *est une trace bien typée dans l'environnement* $\mathcal{E}$*, s'il existe une substitution* σ *telle que* $tr = tr'\sigma$*, où* tr' *est la trace symbolique associée à* sc*, et si sa restriction aux variables de* tr'*, dénotée* $\sigma|_{\mathcal{V}(tr')}$*, et définie comme suit :*

- $\mathrm{dom}(\sigma|_{\mathcal{V}(tr')}) = \mathcal{V}(tr')$*, et*
- $\forall x \in \mathcal{V}(tr').\ \sigma|_{\mathcal{V}(tr')}(x) = \sigma(x)$

est une substitution bien typée. Nous dirons que tr *est une* trace valide bien typée (respectivement une exécution valide bien typée) dans l'environnement $\mathcal{E}$*, si* tr *est une trace bien typée dans l'environnement* $\mathcal{E}$ *et si* tr *est une trace valide (respectivement une exécution valide) de* Π *au regard de la connaissance initiale de l'intrus* T_0*.*

Exemple 7.4.3. *Reprenons la substitution* $\sigma = \{x^2 \mapsto \langle \mathsf{aenc}(n^1, b), a \rangle, x^3 \mapsto n^1\}$ *définie à l'exemple 2.6.9. Nous avons déjà vu à l'exemple 3.2.1 que la trace* $tr_{\mathsf{Toy}}\sigma$ *était une exécution valide qui viole la propriété du secret* ϕ_{Toy}*. Nous avons aussi vu à l'exemple 7.1.6 que* $\mathcal{E}_{\mathsf{Toy}} \vdash x^2 : \nu$ *et que* $\mathcal{E}_{\mathsf{Toy}} \vdash x^3 : \nu$*. Or, l'arbre suivant*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash n : \nu}}{\mathcal{E}_{\mathsf{Toy}} \vdash n^1 : \nu} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash b : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \mathsf{aenc}(n^1, b) : \mathsf{aenc}(\nu, \alpha)} \quad \dfrac{}{\mathcal{E}_{\mathsf{Toy}} \vdash a : \alpha}}{\mathcal{E}_{\mathsf{Toy}} \vdash \langle \mathsf{aenc}(n^1, b), a \rangle : \langle \mathsf{aenc}(\nu, \alpha), \alpha \rangle}$$

montre que $\mathcal{E} \vdash \sigma(x^2) : \langle \mathsf{aenc}(\nu, \alpha), \alpha \rangle$*. D'après la définition 7.4.2, l'attaque* $tr_{\mathsf{Toy}}\sigma$ *n'est pas bien typée.*

Informellement, le résultat que nous visions s'énonce comme suit : « Un protocole Π de $\mathcal{C}_2$ admet une attaque sur une propriété de sécurité ϕ de $\Phi_2(\Pi)$ si et seulement si il admet une attaque sur ϕ bien typée ». Formellement,

Théorème 7.4.4. *Soient un protocole* $\Pi \in \mathcal{C}_2$*, un ensemble de termes atomiques clos* T_0*, et une propriété de sécurité* $\phi \in \Phi_2(\Pi)$*. Il existe un scénario* sc *de* Π*, et une substitution close* σ *tels que :*

1. $tr\sigma$ *soit une exécution valide de* Π *au regard de la connaissance initiale de l'intrus* T_0*, et*
2. $\langle tr\sigma, T_0 \rangle \models \neg\phi$*,*

où tr *est la trace symbolique associée à* sc*, si et seulement si il existe une substitution close* σ_{wt} *telle que :*

1. $tr\sigma_{\mathsf{wt}}$ *soit une exécution valide bien typée dans l'environnement de typage* $\mathcal{E} = \langle \Pi, \mathsf{sc}, \phi \rangle$ *de* Π *au regard de la connaissance initiale de l'intrus* T_0*, et*
2. $\langle tr\sigma_{\mathsf{wt}}, T_0 \rangle \models \neg\phi$*.*

La preuve de ce théorème fait appel à une notion supplémentaire, à savoir celle d'*ensemble de termes bien typé*, ainsi qu'à des résultats intermédiaires que nous ne ferons qu'énoncer dans un premier temps afin de passer au plus vite à la preuve du théorème 7.4.4. La preuve de ces lemmes est remise à la section 7.4.1 suivante.

Définition 7.4.5 (Ensembles de termes bien typés)**.** *Soient $\mathcal{E}$ un environnement de typage, et T un ensemble de termes. T est dit être bien typé dans l'environnement $\mathcal{E}$, si T vérifie*

1. $\forall u, v \in \mathsf{Est}(T).\ \mathrm{mgu}(u, v) \neq \bot \ \Rightarrow\ \exists \tau.\ \mathcal{E} \vdash u : \tau \ \wedge\ \mathcal{E} \vdash v : \tau$, *et*
2. $\forall \mathsf{aenc}(u, v) \in \mathsf{Est}(T).\ \mathcal{E} \vdash v : \alpha$.

Le premier lemme dont nous aurons besoin établit que l'unification de deux termes unifiables et du même type (dans le même environnement de typage) résulte en une substitution bien typée.

Lemme 7.4.6. *Soient un environnement de typage $\mathcal{E}$, et deux termes $u, v \in \mathcal{T}$ du même type,* i.e. $\exists \tau.\ \mathcal{E} \vdash u : \tau \ \wedge\ \mathcal{E} \vdash v : \tau$. *Alors ou bien la substitution* $\mathrm{mgu}(u, v)$ *est bien typée, ou* $\mathrm{mgu}(u, v) = \bot$.

Nous ferons aussi appelle au lemme suivant qui stipule qu'un ensemble de termes bien typé reste bien typé après application d'une substitution bien typée.

Lemme 7.4.7. *Soient $\mathcal{E}$ un environnement de typage, et T un ensemble de termes bien typé dans l'environnement $\mathcal{E}$. Soient aussi deux termes $v, w \in \mathsf{St}(T)$ tels que v et w soient du même type,* i.e. $\exists \tau.\ \mathcal{E} \vdash v : \tau \wedge \mathcal{E} \vdash w : \tau$ *; posons* $\sigma = \mathrm{mgu}(v, w)$. *Si $\sigma \neq \bot$, alors $T\sigma$ est lui aussi bien typé dans l'environnement $\mathcal{E}$.*

Finalement, nous aurons besoin de la proposition suivante selon laquelle la procédure de simplification de contraintes présentée à la section 5.3 appliquée à un système de contraintes dont les membres gauches et droits constituent un ensemble de termes bien typé, résulte en une substitution bien typée.

Proposition 7.4.8. *Soient $\mathcal{E}$ un environnement de typage, et T un ensemble de termes bien typé dans l'environnement $\mathcal{E}$. Soient aussi deux systèmes de contraintes C et D et une substitution σ, tels que* $\mathrm{lhs}(C) \subseteq T$, $\mathrm{rhs}(C) \subseteq \mathsf{St}(T)$, *et tels que D admette une solution. Si $C \rightsquigarrow^n_\sigma D$, alors*

- *σ est bien typée dans l'environnement $\mathcal{E}$,*
- $\mathrm{lhs}(D) \subseteq T\sigma$ *et* $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma)$, *et*
- *$T\sigma$ est bien typé dans l'environnement $\mathcal{E}$.*

Preuve du théorème 7.4.4 Nous avons désormais tous les ingrédients nécessaires pour mener à bien notre preuve.

Démonstration. L'une des implications est trivialement vraie. C'est à la preuve de l'autre sens (*i.e.* de ($\Rightarrow$)) de l'équivalence que nous nous attelons ici. Soient un protocole $\Pi \in \mathcal{C}_2$, un ensemble de termes atomiques clos T_0, une propriété de sécurité $\phi \in \Phi_2(\Pi)$. Soient aussi un scénario sc de Π, et une substitution σ tels que

1. $tr\sigma$ soit une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0, et
2. $\langle tr\sigma, T_0\rangle \models \neg\phi$,

où tr est la trace symbolique associée à sc. En d'autres termes, soit $tr\sigma$ une exécution valide de Π, au regard de la connaissance initiale de l'intrus T_0, qui viole la propriété ϕ. Posons $\mathcal{E}$ l'environnement de typage correspondant, *i.e.* $\mathcal{E} = \langle \Pi, \mathsf{sc}, \phi\rangle$. Nous allons construire une substitution σ_{wt} bien typée dans l'environnement $\mathcal{E}$ telle que $tr\sigma_{\mathsf{wt}}$ soit un exécution valide bien typée de Π, au regard de la connaissance initiale de l'intrus T_0, violant ϕ.

Soit la formule élémentaire $\xi = \mathbf{T}(\neg\phi, tr, T_0)$. D'après le lemme 5.4.4 nous savons que $\langle tr\sigma, T_0\rangle \models \neg\phi$ si et seulement si $\sigma \models' \xi$. Nous savons aussi que ξ est de la forme $\xi = \exists x_1 \ldots \exists x_k.\ \xi'$, où ξ' est une formule élémentaire sans quantificateurs. Soit ψ la forme normale disjonctive associée à ξ', *i.e.* $\psi = \bigvee_{i\in[\![n]\!]} \psi_i$ avec pour tout $i \in [\![n]\!]$, $\psi_i = C_i \wedge Eq_i \wedge Deq_i$ où
- C_i est un système de contraintes,
- Eq_i est un ensemble d'égalités,
- Deq_i est un ensemble de diségalités.

ψ et ξ' étant équivalentes ; d'où $\sigma \models' \exists x_1 \ldots \exists x_n.\ \xi'$ si et seulement si $\sigma \models' \exists x_1 \ldots \exists x_n.\ \psi$, et par là même $\langle tr\sigma\rangle \models \neg\phi$ si et seulement si $\sigma \models' \exists x_1 \ldots \exists x_n.\ \psi$.

Soit $C = \mathcal{C}(tr, T_0)$ le système de contraintes associé à la trace symbolique tr et à la connaissance initiale de l'intrus T_0 tel que spécifié à la définition 5.2.2. D'après le lemme 5.5.3 de correction et de complétude de la procédure $\mathbf{D}$, et étant donné que $\sigma \models' \exists x_1 \ldots \exists x_n.\ \psi$, nous déduisons qu'il existe un $i \in [\![n]\!]$ (posons i_{attack} un tel i), deux substitutions σ'_{wt} et σ''_{wt}, ainsi que deux systèmes de contraintes en forme résolue D et E tels que
- $C \rightsquigarrow^{\ell}_{\sigma_1} D$,
- $(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \rightsquigarrow^{h}_{\sigma_3} E$,
- $\sigma'_{\mathsf{wt}} = \sigma_1\sigma_2\sigma_3$,
- $\sigma'_{\mathsf{wt}}\sigma''_{\mathsf{wt}} \models' \exists x_1 \ldots \exists x_n.\ \psi$

avec $\sigma_2 = \mathrm{mgu}((Eq_{i_{\mathsf{attack}}})\sigma_1)$. A nouveau en faisant appel au lemme 5.4.4 et au fait que ξ' et ψ soient équivalentes, nous déduisons que $\sigma'_{\mathsf{wt}}\sigma''_{\mathsf{wt}} \models' \neg\phi$. Nous allons dans un premier temps montrer que σ'_{wt} est bien typée. Les domaines des substitutions σ_1, σ_2 et σ_3 étant disjoints et chacune de ces substitutions étant appliquée à $C \wedge C_{i_{\mathsf{attack}}}$, $Eq_{i_{\mathsf{attack}}}$ et $Deq_{i_{\mathsf{attack}}}$ au moment où elles est calculées, il suffit d'établir que chacune des substitutions σ_1, σ_2 et σ_3 séparément est bien typée. Puis nous construirons la substitution σ''_{wt} de façon à ce qu'elle soit bien typée aussi. Posons

$$T = \mathrm{lhs}(C) \cup \mathrm{rhs}(C) \ \cup \bigcup_{t=u\in Eq_{i_{\mathsf{attack}}}} \{t, u\} \ \cup \ \mathrm{lhs}(C_{i_{\mathsf{attack}}}) \cup \mathrm{rhs}(C_{i_{\mathsf{attack}}}).$$

Par construction de C nous savons que

$$\mathsf{Est}(\mathrm{lhs}(C) \cup \mathrm{rhs}(C)) \subseteq \mathsf{Est}(tr)$$

Par définition de la transformation $\mathbf{T}$ nous savons aussi que

$$\mathsf{Est}(\mathrm{lhs}(C_{i_{\mathsf{attack}}})) \subseteq \mathsf{Est}(tr) \quad \text{et} \quad \mathsf{Est}(\mathrm{rhs}(C_{i_{\mathsf{attack}}})) \subseteq \mathsf{Est}^{-}(\phi)$$

et d'après le lemme 5.4.6 que pour tout $t = u \in Eq_{i_{\mathsf{attack}}}$

$$\mathsf{Est}(t) \subseteq \mathsf{Est}^{-}(tr) \quad \text{et} \quad \mathsf{Est}(u) \subseteq \mathsf{Est}(\phi).$$

En récapitulant nous obtenons

$$\mathsf{Est}(T) \subseteq \mathsf{Est}(tr) \cup \mathsf{Est}^{-}(\phi).$$

Soient $u, v \in \mathsf{Est}(T)$ tels que $\mathrm{mgu}(u, v) \neq \bot$. La condition 3 de la définition 7.2.1, et les conditions 3, et 4 de la définition 7.3.1 impliquent qu'il existe τ tel que $\mathcal{E} \vdash u : \tau$ et $\mathcal{E} \vdash v : \tau$. De même, soit $\mathsf{aenc}(u, v) \in \mathsf{Est}(T)$. La condition 2 de la définition 7.2.1, et la condition 2 de la définition 7.3.1 impliquent que $\mathcal{E} \vdash v : \alpha$. Donc T est bien typé.

σ_1 et $T\sigma_1$ sont bien typés.
Par hypothèse C admet une solution (le système de contraintes D par exemple) et par construction $\mathrm{lhs}(C) \subseteq T$ et $\mathrm{rhs}(C) \subseteq T \subseteq \mathsf{St}(T)$. De plus, nous venons d'établir que est T bien typé. Donc C et T vérifient les hypothèses de la proposition 7.4.8 qui nous permet de conclure que σ_1 et $T\sigma_1$ sont bien typés, et que $\mathrm{lhs}(D) \subseteq T\sigma_1$ et $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma_1)$.

σ_2 et $T\sigma_1\sigma_2$ sont bien typés.
Rappelons que $\sigma_2 = \mathrm{mgu}((Eq_{i_{\mathsf{attack}}})\sigma_1)$ et posons $(Eq_{i_{\mathsf{attack}}})\sigma_1 = \{t_j = u_j\}_{j \in [\![r]\!]}$. Donc $\sigma_2 = \theta_1 \dots \theta_r$ pour

$$\theta_j = \mathrm{mgu}(t_j\theta_1 \dots \theta_{j-1}, u_j\theta_1 \dots \theta_{j-1}),$$

T, pour tout $j \in [\![r]\!]$, $t_j, u_j \in T\sigma_1$. Nous allons prouver par induction sur r que
- σ_2 est bien typée, et que
- $T\sigma_1\sigma_2$ est bien typé.

Cas de base : $r = 0$. Dans ce cas, $\sigma_2 = \mathrm{id}$ est trivialement bien typée. De plus, sachant que $T\sigma_1$ est bien typé, $T\sigma_1\sigma_2 = T\sigma_1$ l'est nécessairement aussi.

Cas inductif : $r \geq 1$. Dans ce cas, nous savons par hypothèse d'induction que $\theta_1 \dots \theta_{r-1}$ est bien typée, et que $T\sigma_1\theta_1 \dots \theta_{r-1}$ est bien typé. De plus, $t_r = u_r \in Eq_{i_{\mathsf{attack}}}\sigma_1$ implique qu'il existe $t'_r = u'_r \in Eq_{i_{\mathsf{attack}}}$ tel que $t_r = t'_r\sigma_1$ et $u_r = u'_r\sigma_1$. Mais du lemme 5.4.6 et de la condition 1 de la définition 7.3.1 de la classe Φ_2, découle que t'_r et u'_r sont du même type, *i.e.* $\exists\tau.\ \mathcal{E} \vdash t'_r : \tau \wedge \mathcal{E} \vdash u'_r : \tau$. Et comme σ_1 et $(\theta_1 \dots \theta_{k-1})$ sont deux substitutions bien typées nous déduisons que $t'_r\sigma_1(\theta_1 \dots \theta_{r-1})$ et $u'_r\sigma_1(\theta_1 \dots \theta_{r-1})$ sont eux aussi du même type, *i.e.* $\mathcal{E} \vdash t'_r\sigma_1\theta_1 \dots \theta_{r-1} : \tau$ et $\mathcal{E} \vdash u'_r\sigma_1\theta_1 \dots \theta_{r-1} : \tau$. D'après le lemme 7.4.6, la substitution

$$\theta_r = \mathrm{mgu}(t'_r\sigma_1\theta_1 \dots \theta_{k-1}, u'_r\sigma_1\theta_1 \dots \theta_{k-1}) = \mathrm{mgu}(t_r\theta_1 \dots \theta_{k-1}, u_r\theta_1 \dots \theta_{k-1})$$

est donc bien typée. Son domaine étant disjoint de celui de $\theta_1 \dots \theta_{r-1}$ nous pouvons conclure que $\sigma_2 = \theta_1 \dots \theta_{r-1}\theta_r$ est bien typée. Finalement, notons que $t_r, u_r \in T\sigma_1$ implique que $t_r\theta_1 \dots \theta_{r-1}, u_r\theta_1 \dots \theta_{r-1} \in T\sigma_1\theta_1 \dots \theta_{r-1}$. Ainsi toutes les conditions du lemme 7.4.7 sont réunies, ce qui nous permet de conclure que $T\sigma_1\theta_1 \dots \theta_r = T\sigma_1\sigma_2$ est bien typé et achève notre preuve

par induction. Nous avons donc montré que σ_2 et $T\sigma_1\sigma_2$ sont bien typés dans l'environnement $\mathcal{E}$.

σ_3 est bien typée.
Sachant que $\mathrm{lhs}(D) \subseteq T\sigma_1$ et $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma_1)$ (voir un peu plus haut dans la preuve), nous déduisons que $\mathrm{lhs}(D\sigma_2) \subseteq T\sigma_1\sigma_2$ et $\mathrm{rhs}(D\sigma_2) \subseteq (\mathsf{St}(T\sigma_1))\sigma_2 \subseteq \mathsf{St}(T\sigma_1\sigma_2)$. De même, sachant que $\mathrm{lhs}(C_{i_{\mathsf{attack}}}) \subseteq T$ et que $\mathrm{rhs}(C_{i_{\mathsf{attack}}}) \subseteq T$ (par construction de T), nous déduisons que $\mathrm{lhs}(C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \subseteq T\sigma_1\sigma_2$ et que $\mathrm{rhs}(C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \subseteq (\mathsf{St}(T))\sigma_1\sigma_2 \subseteq \mathsf{St}(T\sigma_1\sigma_2)$. Donc

$$\mathrm{lhs}(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_1) = \mathrm{lhs}(D\sigma_2) \cup \mathrm{lhs}(C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \subseteq T\sigma_1\sigma_2$$

et

$$\mathrm{rhs}(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) = \mathrm{rhs}(D\sigma_2) \cup \mathrm{rhs}(C_{i_{\mathsf{attack}}}\sigma_1\sigma_2) \subseteq \mathsf{St}(T\sigma_1\sigma_2).$$

$T\sigma_1\sigma_2$ et $(D\sigma_2 \wedge C_{i_{\mathsf{attack}}}\sigma_1\sigma_2)$ vérifient donc les hypothèses de la proposition 7.4.8 ce qui nous permet de conclure que σ_3 est bien typée.

A ce stade nous avons montré que $\sigma'_{\mathsf{wt}} = \sigma_1\sigma_2\sigma_3$ est bien typée. Il nous reste donc à construire la substitution σ''_{wt} telle que σ''_{wt} soit bien typée, et telle que son domaine de définition inclut toutes les variables de tr non encore instanciées, *i.e.* les membres droits de E. Rappelons aussi que les seules contraintes concernant la construction de σ''_{wt} sont les suivantes :

- pour toute variable $x \in \mathrm{dom}(\sigma''_{\mathsf{wt}})$, $U \cup \mathcal{N}_\epsilon(\Pi) \vdash \sigma''_{\mathsf{wt}}(x)$, où $U \Vdash x \in E$.
- $t\sigma''_{\mathsf{wt}} \neq u\sigma''_{\mathsf{wt}}$ pour tout $t \neq u \in (Deq_{i_{\mathsf{attack}}})\sigma'_{\mathsf{wt}}$.

Nous construisons σ''_{wt} de la manière suivante : $\mathrm{dom}(\sigma') = \mathrm{rhs}(E)$, et pour tout $x \in \mathrm{dom}(\sigma''_{\mathsf{wt}})$, $\sigma''_{\mathsf{wt}}(x) = \mathsf{fake}(\tau)$ où τ est le type de x dans l'environnement $\mathcal{E} = \langle \Pi, \mathsf{sc}, \phi \rangle$, *i.e.* $\mathcal{E} \vdash x : \tau$. La fonction $\mathsf{fake}()$ est une fonction qui appliquée à un type τ retourne un terme t tel que $\mathcal{E} \vdash t : \tau$. Elle est définie inductivement comme suit :

$$\mathsf{fake}(\alpha) = a \qquad \text{avec } a \in \mathcal{A} \setminus (\mathcal{A}(\phi) \cup \bigcup_{x \in \mathcal{V}(\phi)} \mathcal{A}(\sigma'_{\mathsf{wt}}(x)))$$

$$\mathsf{fake}(\gamma) = c \qquad \text{si } c \in \mathcal{C} \text{ et } \mathcal{E} \vdash c : \gamma$$

$$\mathsf{fake}(\nu) = n \qquad \text{si } n \in \mathcal{N}_\epsilon(\Pi) \text{ et } \mathcal{E} \vdash n : \nu$$

$$\mathsf{fake}(\mathsf{pvk}(\alpha)) = \mathsf{pvk}(\epsilon)$$

$$\mathsf{fake}(\mathsf{shk}(\alpha, \alpha)) = \mathsf{shk}(a, \epsilon) \qquad \text{avec } a \in \mathcal{A} \setminus (\mathcal{A}(\phi) \bigcup_{x \in \mathcal{V}(\phi)} \mathcal{A}(\sigma'_{\mathsf{wt}}(x)))$$

$$\mathsf{fake}(f(\tau_1, \ldots, \tau_n)) = f(\mathsf{fake}(\tau_1), \ldots, \mathsf{fake}(\tau_n)) \quad \text{pour } f \in \{\langle\rangle, \mathsf{aenc}, \mathsf{senc}, \mathsf{sign}, \mathsf{h}\}$$

Il est facile de voir que pour tout type τ induit par Π, $\mathcal{N}_\epsilon(\Pi) \vdash \mathsf{fake}(\tau)$, et ayant muni l'intrus d'un nonce de chaque type, que σ''_{wt} est bien typée. De plus, pour tout $t \neq u \in Deq_{i_{\mathsf{attack}}}\sigma'_{\mathsf{wt}}$, il existe $t' \neq u' \in Deq_{i_{\mathsf{attack}}}$ tel que $t = t'\sigma'_{\mathsf{wt}}$ et $u = u'\sigma'_{\mathsf{wt}}$ avec $t \in \mathsf{St}(tr)$ et $u \in \mathsf{St}(\mathsf{SubForm}^+(\phi))$. Or d'après la condition 5 nous savons t n'admet aucun sous-type constant ni clé privée. Alors, ayant pris le soin de sélectionner des agents n'apparaissant pas dans u pour $\mathsf{fake}(\alpha)$ et $\mathsf{fake}(\mathsf{shk}(\alpha, \alpha))$ nous avons la garantie que les diségalités sont satisfaites.

Nous avons donc comme annoncé plus haut $\sigma'_{\mathsf{wt}}\sigma''_{\mathsf{wt}} \models' \exists x_1 \ldots \exists x_n . \psi$ avec $\sigma_{\mathsf{wt}} = \sigma'_{\mathsf{wt}}\sigma''_{\mathsf{wt}}$ bien typée. Donc $tr\sigma_{\mathsf{wt}} \models \neg\phi$. Aussi par correction et complétude de la procédure **D** nous savons que $tr\sigma_{\mathsf{wt}}$ est une exécution valide de Π au regard de la connaissance initiale T_0. Ce qui vient achever notre preuve. □

Nous énonçons maintenant un corollaire du théorème 7.4.4. Informellement, il s'énonce comme suit : « Un protocole Π de $\mathcal{C}_2$ admet une attaque sur une propriété de sécurité ϕ de $\Phi_2(\Pi)$ si et seulement si il admet une attaque sur ϕ dont tous les messages sont de taille bornée ». Formellement,

Corollaire 7.4.9. *Soient un protocole $\Pi \in \mathcal{C}_2$, un ensemble de termes atomiques clos T_0, et une propriété de sécurité $\phi \in \Phi_2(\Pi)$. Il existe un scénario sc de Π, et une substitution close σ tels que :*

1. *$tr\sigma$ soit une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0, et*
2. *$\langle tr\sigma, T_0\rangle \models \neg\phi$,*

où tr est la trace symbolique associée à sc, si et seulement si il existe une substitution close σ' telle que :

1. *$tr\sigma'$ soit une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0, et*
2. *$\forall m \in \mathsf{St}(tr\sigma').\ |m| \leq \mathsf{B}(\Pi)$, et*
3. *$\langle tr\sigma', T_0\rangle \models \neg\phi$.*

Démonstration. A nouveau l'une des deux implications est triviale. Nous allons dans un premier temps montrer que pour tout terme t, et tout type τ si $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash t : \tau$, alors $|t| \leq |\tau|$. Nous procédons par induction sur la taille de τ.

Si $\tau = \alpha$, alors $|\tau| = 1$ et $t \in \mathcal{P} \cup \mathcal{V}$; donc $|t| = 1$ et l'inégalité est bien satisfaite.

Si $\tau = \gamma$ et γ est le type d'une constante, *i.e.* il existe $c \in \mathcal{C}(\Pi)$ telle que $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash c : \gamma$, alors $|\tau| = 1$ et $t \in \mathcal{C} \cup \mathcal{V}$; donc $|t| = 1$ et l'inégalité est bien vérifiée.

Si $\tau = \nu$ et ν est le type d'un nonce, *i.e.* il existe $n \in \mathcal{N}(\Pi)$ tel que $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash n : \nu$, alors $|\tau| = 1$ et $t \in \mathcal{N} \cup \mathcal{V}$; donc $|t| = 1$ et l'inégalité est bien vérifiée.

Si $\tau = \omega$, alors $|\tau| = 1$ et $t \in \mathcal{C} \cup \mathcal{N} \cup \mathcal{V}$; et donc $|t| = 1$ et l'inégalité est bien vérifiée.

Si $\tau = f(\tau_1, \ldots, \tau_n)$, alors nous distinguons deux cas
- Si $t \in \mathcal{V}$. Par définition nous savons que $|\tau| \geq 1$, or $|t| = 1$, donc l'inégalité est bien vérifiée.
- Si $t \notin \mathcal{V}$, alors $t = f(t_1, \ldots, t_n)$ et pour tout $i \in [\![n]\!]$ $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash t_i : \tau_i$. Nous pouvons donc appliquer notre hypothèse d'induction et conclure que pour tout $i \in [\![n]\!]$, $|t_i| \leq |\tau_i|$. Mais par définition nous savons que $|t| = 1 + \sum_{i \in [\![n]\!]} |t_i|$ et $|\tau| = 1 + \sum_{i \in [\![n]\!]} |\tau_i|$, d'où

$$|t| = 1 + \sum_{i \in [\![n]\!]} |t_i| \leq 1 + \sum_{i \in [\![n]\!]} |\tau_i| = |\tau|.$$

Soient un protocole $\Pi \in \mathcal{C}_2$, un ensemble de termes atomiques clos T_0, et une propriété de sécurité $\phi \in \Phi_2(\Pi)$. Soient aussi un scénario sc de Π de trace symbolique associée tr, et une substitution close σ. Supposons que

1. $tr\sigma$ est une exécution valide de Π au regard de la connaissance initiale de l'intrus T_0, et
2. $\langle tr\sigma, T_0\rangle \models \neg\phi$,

D'après le théorème 7.4.4 nous savons qu'il existe une substitution close σ_{wt} telle que :

1. $tr\sigma_{\mathsf{wt}}$ soit une exécution valide bien typée de Π au regard de la connaissance initiale de l'intrus T_0, et
2. $\langle tr\sigma_{\mathsf{wt}}, T_0\rangle \models \neg\phi$.

Soit $m \in \mathsf{St}(tr\sigma_{\mathsf{wt}})$ avec $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash m : \tau$ pour un certain type τ, alors

1. soit il existe un terme $t \in \mathsf{St}(tr)$ tel que $m = t\sigma_{\mathsf{wt}}$ et comme σ_{wt} est bien typée $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash t : \tau$,
2. soit il existe une variable $x \in \mathcal{V}(tr)$ et un type τ' telle que $m \in \mathsf{St}(\sigma_{\mathsf{wt}}(x))$ et $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash x : \tau'$, mais alors $|\tau'| \geq |\tau|$.

Supposons que nous sommes dans le premier cas. Alors par construction d'une trace symbolique associée à un scénario de Π nous savons qu'il existe un terme $u \in \mathsf{St}(\Pi)$ tel que $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash u : \tau$, et donc que $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash \delta_\Pi(u) : \tau$. De plus, il est facile de voir que $|\tau| = |\delta_\Pi(u)|$. Or par définition de la mesure sur les protocoles $\mathsf{B}()$, nous savons que $|\tau| = |\delta_\Pi(u)| \leq \mathsf{B}(\Pi)$. En faisant à présent appel à ce que nous établissions en tout début de preuve nous avons $|m| \leq |\tau| = |\delta_\Pi(u)| \leq \mathsf{B}(\Pi)$.

Plaçons nous maintenant dans le second cas. Par construction d'une trace symbolique associée à un scénario de Π nous savons qu'il existe une variable $y \in \mathcal{V}(\Pi)$ tel que $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash y : \tau'$, et que $\langle \Pi, \mathsf{sc}, \phi\rangle \vdash \delta_\Pi(y) : \tau'$. De plus, il est facile de voir que $|\tau'| = |\delta_\Pi(y)|$. Or par définition de la mesure sur les protocoles $\mathsf{B}()$, nous savons que $|\tau'| = |\delta_\Pi(y)| \leq \mathsf{B}(\Pi)$. En faisant à présent appel au fait que $|\tau| \leq |\tau'|$ et à $|m| \leq |\tau|$ d'après le résultat établit en tout début de cette preuve, nous savons que $|m| \leq |\tau| \leq |\tau'| = |\delta_\Pi(y)| \leq \mathsf{B}(\Pi)$. □

7.4.1 Preuves des lemmes intermédiaires

Comme nous l'annoncions cette section est consacrée à la démonstration des résultats intermédiaires sur lesquels repose notre résultat de réduction énoncé au théorème 7.4.4.

Ce lemme, rappelons-le, établit que l'unification de deux termes unifiables et du même type (dans le même environnement de typage) résulte en une substitution bien typée.

Lemme 7.4.6. *Soient un environnement de typage $\mathcal{E}$, et deux termes $u, v \in \mathcal{T}$ du même type,* i.e. $\exists\tau.\ \mathcal{E} \vdash u : \tau \ \wedge\ \mathcal{E} \vdash v : \tau$. *Alors ou bien la substitution* $\mathrm{mgu}(u, v)$ *est bien typée, ou* $\mathrm{mgu}(u, v) = \bot$.

Démonstration. Supposons que $\mathrm{mgu}(u, v) \neq \bot$. La substitution $\mathrm{mgu}(u, v)$ peut être calculée à l'aide de l'algorithme d'unification vu à la section 5.1 du chapitre 5, l'ensemble initial d'équations étant $E_0 = \{u \stackrel{?}{=} v\}$. Soit $E_0 \rightsquigarrow_{\sigma_1} E_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_k} E_k$ une séquence de simplifications aboutissant à la substitution $\mathrm{mgu}(u, v)$, *i.e.* $\mathrm{mgu}(u, v) = \sigma_1\sigma_2 \ldots \sigma_k$.

Un ensemble d'équations $\{t_i \stackrel{?}{=} u_i\}_{i\in[\![n]\!]}$ sera dit être bien typé dans l'environnement $\mathcal{E}$ si pour tout $i \in [\![n]\!]$, t_i et u_i sont du même type, *i.e.* $\exists \tau_i.\ \mathcal{E} \vdash t_i : \tau_i \wedge \mathcal{E} \vdash u_i : \tau_i$.

Nous allons montrer par induction sur i que l'ensemble d'équations E_i et la substitution σ_i sont bien typés dans l'environnement $\mathcal{E}$.

Cas de base : $i = 0$. Par hypothèse nous savons que u et v sont du même type dans l'environnement $\mathcal{E}$, et donc que E_0 est bien typé dans l'environnement $\mathcal{E}$. De plus, $\sigma_0 = \mathrm{id}$ est trivialement bien typée.

Cas inductif : $i \geq 1$. Nous savons par hypothèse d'induction que E_j et σ_j, pour tout $j \in [\![i-1]\!]$, sont bien typés dans l'environnement $\mathcal{E}$. Afin d'établir que E_i et σ_i sont eux aussi bien typés dans l'environnement $\mathcal{E}$, nous procédons par analyse de cas sur la règle R impliquée dans la réduction $E_{i-1} \overset{\mathsf{R}}{\rightsquigarrow}_{\sigma_i} E_i$.

Cas $\mathsf{R} = (a)$. Il existe alors nécessairement un ensemble d'équations E ainsi qu'une équation $t = u$ tels que $E_{i-1} = E \cup \{t \stackrel{?}{=} u\}$, et $E_i = E \cup \{u \stackrel{?}{=} t\}$. Il est évident que si E_{i-1} est bien typé, alors E_i l'est aussi. De plus, $\sigma_i = \mathrm{id}$ est trivialement bien typée.

Cas $\mathsf{R} = (b)$. Alors $E_i \subseteq E_{i-1}$. Sachant que E_{i-1} est bien typé, nous concluons que E_i l'est nécessairement aussi. De plus, $\sigma_i = \mathrm{id}$ est trivialement bien typée.

Cas $\mathsf{R} = (c)$. Nous distinguons deux cas :

1. $E_i \subseteq E_{i-1}$, alors sachant que E_{i-1} est bien typé, nous concluons que E_i l'est nécessairement aussi.
2. $E_i \not\subseteq E_{i-1}$, et sachant que $\mathrm{mgu}(u, v) \neq \bot$, nous déduisons qu'il existe nécessairement un ensemble d'équations E et une équation $f(t_1, \ldots, t_r) \stackrel{?}{=} f(u_1, \ldots, u_r)$ avec $f \in \{\mathsf{pvk}, \mathsf{shk}, \langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$, tels que $E_{i-1} = E \cup \{f(t_1, \ldots, t_r) \stackrel{?}{=} f(u_1, \ldots, u_r)\}$ et $E_i = E \cup \{t_1 \stackrel{?}{=} u_1, \ldots, t_r \stackrel{?}{=} u_r)\}$. Aussi, d'après notre système de types et plus particulièrement d'après la règle six de la figure 7.1, il existe nécessairement des types $\tau_1, \ldots, \tau_r$ tels que $\mathcal{E} \vdash f(t_1, \ldots, t_r) : f(\tau_1, \ldots, \tau_r)$ et $\mathcal{E} \vdash f(u_1, \ldots, u_r) : f(\tau_1, \ldots, \tau_r)$, mais aussi tels que $\mathcal{E} \vdash t_i : \tau_i$ et $\mathcal{E} \vdash u_i : \tau_i$ pour tout $i \in [\![r]\!]$. Ce qui nous permet de conclure que $\{t_1 \stackrel{?}{=} u_1, \ldots, t_r \stackrel{?}{=} u_r\}$ est bien typé. De plus, comme $E \subset E_{i-1}$ nous savons que E est bien typé. Ainsi, $E_i = E \cup \{t_1 \stackrel{?}{=} u_1, \ldots, t_r \stackrel{?}{=} u_r\}$, en temps qu'union de deux ensembles d'équations bien typés, est lui aussi bien typé dans l'environnement $\mathcal{E}$.

Dans les deux cas $\sigma_i = \mathrm{id}$, et donc dans les deux cas σ_i est bien typée.

Cas $\mathsf{R} = (d)$. Ayant supposé que $\mathrm{mgu}(u, v) \neq \bot$, il existe nécessairement un ensemble d'équations E et une équation $x \stackrel{?}{=} t$, tels que $x \in \mathcal{V}$, $t \neq x$, x apparaît dans E, $E_{i-1} = E \cup \{x \stackrel{?}{=} t\}$, $E_i = E\sigma_i \cup \{x \stackrel{?}{=} t\}$, et $\sigma_i = \{x \mapsto t\}$. Puisque $x \stackrel{?}{=} t \in E_{i-1}$ et E_{i-1} est bien typé, la substitution σ_i est bien typée, aussi puisque $E \subseteq E_{i-1}$, E est lui aussi bien typé. Comme l'application d'une substitution bien typée à un ensemble d'équations bien typées résulte en un en-

semble d'équations bien typées, $E\sigma_i$ est bien typé, et donc $E_i = E\sigma_i \cup \{x \stackrel{?}{=} t\}$ est bien typé.

Ainsi, nous avons montré que chacune des σ_i ($i \in [\![k]\!]$) était bien typé. Le domaine de chacune de ces substitution étant disjoint (ceci est dû à l'application de la substitution σ_i à E_{i-1} dans le cas $\mathsf{R} = (d)$), nous pouvons conclure que la substitution $\mathrm{mgu}(u, v) = \sigma_1\sigma_2 \dots \sigma_k$ est bien typée. □

Le lemme suivant stipule qu'un ensemble de termes bien typé reste bien typé après application d'une substitution bien typée.

Lemme 7.4.7. *Soient $\mathcal{E}$ un environnement de typage, et T un ensemble de termes bien typé dans l'environnement $\mathcal{E}$. Soient aussi deux termes $v, w \in \mathsf{St}(T)$ tels que v et w soient du même type,* i.e. $\exists\tau.\ \mathcal{E} \vdash v : \tau \wedge \mathcal{E} \vdash w : \tau$ *; posons $\sigma = \mathrm{mgu}(v, w)$. Si $\sigma \neq \bot$, alors $T\sigma$ est lui aussi bien typé dans l'environnement $\mathcal{E}$.*

Démonstration. D'après la définition 7.4.5, pour montrer que $T\sigma$ est bien typé dans l'environnement $\mathcal{E}$, il faut établir que

1. $\forall t, u \in \mathsf{Est}(T\sigma).\ \mathrm{mgu}(t, u) \neq \bot \ \Rightarrow\ \exists\tau.\ \mathcal{E} \vdash t : \tau \ \wedge\ \mathcal{E} \vdash u : \tau$, et que
2. $\forall \mathsf{aenc}(t, u) \in \mathsf{Est}(T\sigma).\ \mathcal{E} \vdash u : \alpha$.

Nous procédons à la preuve de chacun de ces deux points séparément.

1. Soient deux termes $t, u \in \mathsf{Est}(T\sigma)$ tels que $\mathrm{mgu}(t, u) \neq \bot$. Par définition, nous savons qu'il existe deux termes $t', u' \in T$ tels $t \in \mathsf{Est}(t'\sigma)$ et $u \in \mathsf{Est}(u'\sigma)$. D'après le lemme 5.1.3, nous savons alors que

$$t \in \mathsf{Est}(t')\sigma \ \vee\ (\exists x \in \mathcal{V}(t').\ t \in \mathsf{Est}(\sigma(x))),$$

   et que

$$u \in \mathsf{Est}(u')\sigma \ \vee\ (\exists x \in \mathcal{V}(u').\ u \in \mathsf{Est}(\sigma(x))).$$

   Nous procédons par analyse de cas sur les combinaisons possibles.

   Cas $t \in \mathsf{Est}(t')\sigma$ et $u \in \mathsf{Est}(u')\sigma$.
   Notons tout d'abord que $t \in \mathsf{Est}(t')\sigma$ est équivalent à $\exists t'' \in \mathsf{Est}(t')$ tel que $t = t''\sigma$. De même que $u \in \mathsf{Est}(u')\sigma$ est équivalent à $\exists u'' \in \mathsf{Est}(u')$ tel que $u = u''\sigma$. Or $\mathrm{mgu}(t, u) \neq \bot$ (posons $\theta = \mathrm{mgu}(t, u)$), donc $t''\sigma\theta = u''\sigma\theta$, et donc $\sigma\theta$ est un unificateur de t'' et u''. Mais alors t'' et u'' admettent unificateur plus général, *i.e.* $\mathrm{mgu}(t'', u'') \neq \bot$. Notons maintenant que $t'', u'' \in \mathsf{Est}(T)$ et que donc par hypothèse sur T (T est bien typé dans l'environnement $\mathcal{E}$), t'' et u'' sont du même type dans l'environnement $\mathcal{E}$, *i.e.* $\exists\tau.\ \mathcal{E} \vdash t'' : \tau \ \wedge\ \mathcal{E} \vdash u'' : \tau$. Aussi, étant donné que v et w sont du même type, nous savons d'après le lemme 7.4.6 que σ est bien typée et que donc $t(= t''\sigma)$ et t'' sont du même type, et que $u = (u''\sigma)$ et u'' sont du même type, ce qui nous permet de conclure que $\mathcal{E} \vdash t : \tau$ et $\mathcal{E} \vdash u : \tau$.

   Cas $t \in \mathsf{Est}(t')\sigma$ et $(\exists x \in \mathcal{V}(u').\ u \in \mathsf{Est}(\sigma(x)))$.
   Notons tout d'abord que $t \in \mathsf{Est}(t')\sigma$ est équivalent à $\exists t'' \in \mathsf{Est}(t')$ tel que $t = t''\sigma$. De l'autre coté, nous savons d'après le lemme 5.1.4 que $\exists u'' \in \mathsf{Est}(v, w),\ u = u''\sigma$. Supposons que $u'' \in \mathsf{Est}(v)$ (le cas $u'' \in \mathsf{Est}(w)$

peut être traité de la même manière). Par hypothèse nous savons que $\mathrm{mgu}(t,u) \neq \bot$ (posons $\theta = \mathrm{mgu}(t,u)$), et donc que $t''\sigma\theta = u''\sigma\theta$, et $\sigma\theta$ est un unificateur de t'' et u''. Comme nous l'avons vu à la section 5.1 du chapitre 5, si t'' et u'' admettent un unificateur, alors ils admettent un unificateur plus général, *i.e.* $\mathrm{mgu}(t'',u'') \neq \bot$. Notons maintenant que $t'', u'' \in \mathsf{Est}(T)$ et que donc par hypothèse sur T (T est bien typé dans l'environnement $\mathcal{E}$), t'' et u'' sont du même type dans l'environnement $\mathcal{E}$, *i.e.* $\exists\tau.\ \mathcal{E} \vdash t'' : \tau \ \wedge\ \mathcal{E} \vdash u'' : \tau$. Aussi, étant donné que v et w sont du même type, nous savons d'après le lemme 7.4.6 que σ est bien typée et que donc $t(= t''\sigma)$ et t'' sont du même type, et que $u = (u''\sigma)$ et u'' sont du même type, ce qui nous permet de conclure que $\mathcal{E} \vdash t : \tau$ et $\mathcal{E} \vdash u : \tau$.

Cas $(\exists x \in \mathcal{V}(t').\ t \in \mathsf{Est}(\sigma(x)))$ *et* $u \in \mathsf{Est}(u')\sigma$.
Ce cas est analogue au précédent.

Cas $(\exists x \in \mathcal{V}(t').\ t \in \mathsf{Est}(\sigma(x)))$ *et* $(\exists x \in \mathcal{V}(u').\ u \in \mathsf{Est}(\sigma(x)))$.
Notons tout d'abord que d'après le lemme 5.1.4 $\exists t'' \in \mathsf{Est}(v,w),\ t = t''\sigma$. Supposons que $t'' \in \mathsf{Est}(v)$ (le cas $t'' \in \mathsf{Est}(w)$ peut être traité de la même manière). De même, nous savons que $\exists u'' \in \mathsf{Est}(v,w),\ u = u''\sigma$. Supposons que $u'' \in \mathsf{Est}(v)$ (le cas $u'' \in \mathsf{Est}(w)$ peut être traité de la même manière). Par hypothèse nous savons que $\mathrm{mgu}(t,u) \neq \bot$ (posons $\theta = \mathrm{mgu}(t,u)$), et donc que $t''\sigma\theta = u''\sigma\theta$, et $\sigma\theta$ est un unificateur de t'' et u''. Mais alors t'' et u'' admettent un unificateur plus général, *i.e.* $\mathrm{mgu}(t'',u'') \neq \bot$. Notons maintenant que $t'', u'' \in \mathsf{Est}(T)$ et que donc par hypothèse sur T (T est bien typé dans l'environnement $\mathcal{E}$), t'' et u'' sont du même type dans l'environnement $\mathcal{E}$, *i.e.* $\exists\tau.\ \mathcal{E} \vdash t'' : \tau \ \wedge\ \mathcal{E} \vdash u'' : \tau$. Aussi, étant donné que v et w sont du même type, nous savons d'après le lemme 7.4.6 que σ est bien typée et que donc $t(= t''\sigma)$ et t'' sont du même type, et que $u = (u''\sigma)$ et u'' sont du même type, ce qui nous permet de conclure que $\mathcal{E} \vdash t : \tau$ et $\mathcal{E} \vdash u : \tau$.

2. Soit $\mathsf{aenc}(t,u) \in \mathsf{Est}(T\sigma)$. Par définition nous savons qu'il existe un terme $t_1 \in T$ tel que $\mathsf{aenc}(t,u) \in \mathsf{Est}(t_1\sigma)$. D'après le lemme 5.1.3, nous savons alors que

$$\mathsf{aenc}(t,u) \in \mathsf{Est}(t_1)\sigma \ \vee\ (\exists x \in \mathcal{V}(t_1).\ \mathsf{aenc}(t,u) \in \mathsf{Est}(\sigma(x))).$$

Si $\mathsf{aenc}(t,u) \in \mathsf{Est}(t_1)\sigma$, alors il existe $t_2 \in \mathsf{Est}(t) \subseteq \mathsf{Est}(T)$ tel que $\mathsf{aenc}(t,u) = t_2\sigma$. Donc, nécessairement t_2 est de la forme $t_2 = \mathsf{aenc}(t_3,u')$ pour certains termes t_3 et u' et donc $\mathsf{aenc}(t,u) = \mathsf{aenc}(t_3\sigma,u'\sigma)$. De plus, nous savons que T est bien typé, et comme $t_2 = \mathsf{aenc}(t_3,u') \in \mathsf{Est}(T)$, nécessairement $\mathcal{E} \vdash u' : \alpha$. A présent, nous faisons appel au lemme 7.4.6 selon lequel σ est bien typée pour déduire qu'étant donné que $\mathcal{E} \vdash u' : \alpha$, nécessairement $\mathcal{E} \vdash u'\sigma : \alpha$ et donc $\mathcal{E} \vdash u : \alpha$.
Supposons maintenant que $\exists x \in \mathcal{V}(t_1).\ \mathsf{aenc}(t,u) \in \mathsf{Est}(\sigma(x))$. Alors, d'après le lemme 5.1.4, $\exists t_2 \in \mathsf{Est}(\{v,w\}),\ \mathsf{aenc}(t,u) = t_2\sigma$. Supposons que $t_2 \in \mathsf{Est}(v)$ (le cas $t_2 \in \mathsf{Est}(w)$ peut être traité de manière analogue). Donc t_2 est de la forme $t_2 = \mathsf{aenc}(t',u')$, $\mathsf{aenc}(t,u) = \mathsf{aenc}(t'\sigma,u'\sigma)$, et $t_2 \in \mathsf{Est}(T)$. Or nous savons par hypothèse sur T que T est bien typé dans l'environnement $\mathcal{E}$. Donc $\mathcal{E} \vdash u' : \alpha$. A présent, nous faisons appel au lemme 7.4.6 selon lequel σ est bien typée pour déduire que étant donné

que $\mathcal{E} \vdash u' : \alpha$ et que σ est bien typée, alors nécessairement $\mathcal{E} \vdash u'\sigma : \alpha$ et donc $\mathcal{E} \vdash u : \alpha$.

Nous avons donc montré que $T\sigma$ vérifie les deux conditions de la définition 7.4.5, ce qui nous permet de conclure que $T\sigma$ est bien typé. □

Le principale proposition sur laquelle repose la preuve du théorème 7.4.4 stipule que la procédure de simplification de contraintes présentée à la section 5.3 appliquée à un système de contraintes dont les membres gauches et droits forment un ensemble bien typé, résulte en une substitution bien typée.

Proposition 7.4.8. *Soient $\mathcal{E}$ un environnement de typage, et T un ensemble de termes bien typé dans l'environnement $\mathcal{E}$. Soient aussi deux systèmes de contraintes C et D et une substitution σ, tels que* $\mathrm{lhs}(C) \subseteq T$, $\mathrm{rhs}(C) \subseteq \mathsf{St}(T)$, *et tels que D admette une solution. Si $C \rightsquigarrow_\sigma^n D$, alors*

- *σ est bien typée dans l'environnement $\mathcal{E}$,*
- $\mathrm{lhs}(D) \subseteq T\sigma$ *et* $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma)$, *et*
- *$T\sigma$ est bien typé dans l'environnement $\mathcal{E}$.*

Démonstration. Nous procédons par induction sur la longueur n de la dérivation.

Cas de base : $n = 0$. Alors $D = C$, $\sigma = \mathrm{id}$, et $T\sigma = T$. Il est alors évident que $\mathrm{lhs}(D) = \mathrm{lhs}(C) \subseteq T = T\sigma$ et $\mathrm{rhs}(D) = \mathrm{rhs}(C) \subseteq \mathsf{St}(T) = \mathsf{St}(T\sigma)$. Aussi, comme $\sigma = \mathrm{id}$, σ est trivialement bien typée. Finalement, puisque $T\sigma = T$ nous concluons par hypothèse sur T que $T\sigma$ est bien typé dans l'environnement $\mathcal{E}$.

Cas inductif : $n \geq 1$. Il existe alors un système de contraintes E, une règle R parmi celles décrites à la figure 5.2, ainsi que deux substitutions σ_1 et σ_2 tels que

$$C \overset{\mathsf{R}}{\rightsquigarrow}_{\sigma_1} E \rightsquigarrow_{\sigma_2}^{n-1} D \qquad \text{et} \qquad \sigma = \sigma_1\sigma_2.$$

Nous allons montrer que σ_1 est bien typée, que $\mathrm{lhs}(E) \subseteq T\sigma_1$ et $\mathrm{rhs}(E) \subseteq \mathsf{St}(T\sigma_1)$, ainsi que $T\sigma_1$ est bien typé dans l'environnement $\mathcal{E}$. Nous procédons par analyse de cas sur la règle R.

Cas $\mathsf{R} = \mathsf{R}_1$.
Il existe donc un système de contraintes C', un ensemble de termes U, ainsi qu'un terme u tels que $C = C' \wedge U \Vdash u$ et $E = C'$.

- σ_1 est donc la substitution identité et est trivialement bien typée.
- $\mathrm{lhs}(E) = \mathrm{lhs}(C') \subseteq \mathrm{lhs}(C) \subseteq T$ et $\mathrm{rhs}(E) = \mathrm{rhs}(C') \subseteq \mathrm{rhs}(C) \subseteq \mathsf{St}(T)$.
- Du fait que $T\sigma_1 = T\,\mathrm{id} = T$, nous concluons par hypothèse sur T que $T\sigma_1$ est bien typé dans l'environnement $\mathcal{E}$.

Cas $\mathsf{R} = \mathsf{R}_2$.
Il existe donc un système de contraintes C', un ensemble de termes U, ainsi que deux termes u et v tels que $C = C' \wedge U \Vdash u$, $\sigma_1 = \mathrm{mgu}(u, v)$, u et v ne soient ni des paires, ni des variable, $u \neq v$, et $E = C\sigma_1$.

- Du fait que $u \neq v$ nous déduisons immédiatement que u et v sont des chiffrés, *i.e.* $v \in \mathsf{Est}(U) \subseteq \mathsf{Est}(\mathrm{lhs}(C)) \subseteq \mathsf{Est}(T)$ et $u \in \mathsf{Est}(\mathrm{rhs}(C)) \subseteq \mathsf{Est}(\mathsf{St}(T)) = \mathsf{Est}(T)$. De plus, u et v étant unifiables et T étant bien

typé, il existe un type τ tel que $\mathcal{E} \vdash u : \tau$ et $\mathcal{E} \vdash v : \tau$. Nous pouvons alors appliquer le lemme 7.4.6 et conclure que σ_1 est bien typée.
- Comme $\text{lhs}(E) = \text{lhs}(C\sigma_1)$ et $\text{lhs}(C) \subseteq T$, nous concluons que $\text{lhs}(E) = \text{lhs}(C\sigma_1) \subseteq T\sigma_1$. De même, comme $\text{rhs}(E) = \text{rhs}(C\sigma_1)$ et $\text{rhs}(C) \subseteq \mathsf{St}(T)$, nous concluons que $\text{rhs}(E) = \text{rhs}(C\sigma_1) \subseteq (\mathsf{St}(T))\sigma_1 \subseteq \mathsf{St}(T\sigma_1)$.
- Finalement, notons que $v \in \mathsf{St}(\text{lhs}(C)) \subseteq \mathsf{St}(T)$ et $u \in \text{rhs}(C) \subseteq \mathsf{St}(T)$ et rappelons nous que u et v sont du même type dans l'environnement $\mathcal{E}$. Ainsi, T, u et v vérifient les hypothèses du lemme 7.4.7 qui nous permet de conclure que $T\sigma_1$ est bien typé.

Cas $\mathsf{R} = \mathsf{R}_3$.
Il existe donc un système de contraintes C', un ensemble de termes U, ainsi que trois termes u, v et w tels que $C = C' \wedge U \Vdash u$, $\sigma_1 = \text{mgu}(v, w)$, $v, w \in \mathsf{St}(U)$, v et w ni variables ni paires, $v \neq w$, et $E = C\sigma_1$.
- Du fait que $v \neq w$ nous déduisons immédiatement que v et w sont des chiffrés, *i.e.* $v, w \in \mathsf{Est}(U) \subseteq \mathsf{Est}(\text{lhs}(C)) \subseteq \mathsf{Est}(T)$. De plus, v et w étant unifiables et T étant bien typé, il existe un type τ tel que $\mathcal{E} \vdash v : \tau$ et $\mathcal{E} \vdash w : \tau$. Nous pouvons alors appliquer le lemme 7.4.6 et conclure que σ_1 est bien typée.
- Comme $\text{lhs}(E) = \text{lhs}(C\sigma_1)$ et $\text{lhs}(C) \subseteq T$, nous concluons que $\text{lhs}(E) = \text{lhs}(C\sigma_1) \subseteq T\sigma_1$. De même, comme $\text{rhs}(E) = \text{rhs}(C\sigma_1)$ et $\text{rhs}(C) \subseteq \mathsf{St}(T)$, nous concluons que $\text{rhs}(E) = \text{rhs}(C\sigma_1) \subseteq (\mathsf{St}(T))\sigma_1 \subseteq \mathsf{St}(T\sigma_1)$.
- Finalement, notons que $v, w \in \mathsf{St}(\text{lhs}(C)) \subseteq \mathsf{St}(T)$ et rappelons nous que v et w sont du même type dans l'environnement $\mathcal{E}$. Ainsi, T, v et w vérifient les hypothèses du lemme 7.4.7 qui nous permet de conclure que $T\sigma_1$ est bien typé.

Cas $\mathsf{R} = \mathsf{R}_4$.
Il existe donc un système de contraintes C', un ensemble de termes U, ainsi que quatre termes u, v_1, v_2 et v_3 tels que $C = C' \wedge U \Vdash u$, $\sigma_1 = \text{mgu}(v_2, v_3)$, $\mathsf{aenc}(v_1, v_2) \in \mathsf{St}(U)$, $\mathsf{pvk}(v_3) \in \mathsf{Plaintext}(U) \cup \{\mathsf{pvk}(\epsilon)\}$, $v_2 \neq v_3$ et $E = C\sigma_1$.
- Comme T est bien typé, nous savons que $\mathcal{E} \vdash v_2 : \alpha$. Aussi, d'après la grammaire de nos termes nous savons que $v_3 \in \mathcal{P}$ et donc que $\mathcal{E} \vdash v_3 : \alpha$. Donc v_2 et v_3 sont du même type dans l'environnement $\mathcal{E}$, et donc d'après le lemme 7.4.6 σ_1 est bien typée.
- Comme $\text{lhs}(E) = \text{lhs}(C\sigma_1)$ et $\text{lhs}(C) \subseteq T$, nous concluons que $\text{lhs}(E) = \text{lhs}(C\sigma_1) \subseteq T\sigma_1$. De même, comme $\text{rhs}(E) = \text{rhs}(C\sigma_1)$ et $\text{rhs}(C) \subseteq \mathsf{St}(T)$, nous concluons que $\text{rhs}(E) = \text{rhs}(C\sigma_1) \subseteq (\mathsf{St}(T))\sigma_1 \subseteq \mathsf{St}(T\sigma_1)$.
- Comme T est bien typé, nous savons que $\mathcal{E} \vdash v_2 : \alpha$. Aussi, d'après la grammaire de nos termes nous savons que $v_3 \in \mathcal{P}$ et donc que $\mathcal{E} \vdash v_3 : \alpha$. Donc $\sigma_1 = \{v_2 \mapsto v_3\}$ est bien typée et $T\sigma_1$ est clairement un ensemble de terme bien typé.

Cas $\mathsf{R} = \mathsf{R}_5$.
Ce cas ne peut pas survenir car il viendrait contredire l'hypothèse selon laquelle D admettrait une solution.

Cas $\mathsf{R} = \mathsf{R}_f$ *avec* $f \in \{\langle\rangle, \mathsf{senc}, \mathsf{aenc}, \mathsf{sign}, \mathsf{h}\}$.
Il existe donc un système de contraintes C', un ensemble de termes U, ainsi qu'un terme $f(u_1, \ldots, u_r)$ tels que $C = C' \wedge U \Vdash f(u_1, \ldots, u_r)$ et $E = C' \wedge U \Vdash$

$u_1 \wedge \cdots \wedge U \Vdash u_r$.

- σ_1 est donc la substitution identité, et est donc trivialement bien typée.
- Comme $\mathrm{lhs}(E) = \mathrm{lhs}(C) \subseteq T$ et $T\sigma_1 = T$, nous concluons que $\mathrm{lhs}(E) \subseteq T\sigma_1$. De l'autre coté, nous avons $\mathrm{rhs}(E) = \mathrm{rhs}(C') \cup \{u_1, \ldots, u_r\}$, avec $\mathrm{rhs}(C') \subseteq \mathrm{rhs}(C) \subseteq \mathsf{St}(T)$ et $f(u_1, \ldots, u_r) \in \mathrm{rhs}(C) \subseteq \mathsf{St}(T)$. Or, par définition $u_1, \ldots, u_r \in \mathsf{St}(f(u_1, \ldots, u_r)) \subseteq \mathsf{St}(T)$, ce qui nous permet de conclure que $\mathrm{rhs}(E) \subseteq \mathsf{St}(T) = \mathsf{St}(T\sigma_1)$.
- Du fait que $T\sigma_1 = T\,\mathrm{id} = T$, nous concluons par hypothèse sur T que $T\sigma_1$ est bien typé.

Nous pouvons donc appliquer notre hypothèse d'induction à E et conclure que

- σ_2 est bien typée. Et comme σ_1 l'est aussi, nécessairement $\sigma = \sigma_1\sigma_2$ est bien typée.
- $\mathrm{lhs}(D) \subseteq T\sigma_1\sigma_2 = T\sigma$ et $\mathrm{rhs}(D) \subseteq \mathsf{St}(T\sigma_2\sigma_2) = \mathsf{St}(T\sigma)$, et
- $T\sigma_1\sigma_2 = T\sigma$ est bien typé.

□

7.5 Construction de protocoles dans $\mathcal{C}_2$

Bien des protocoles existants ne rentrent malheureusement pas dans la classe $\mathcal{C}_2$. Nous allons voir dans cette section qu'il est possible de transformer la plupart des protocoles en des protocoles de $\mathcal{C}_2$, tout en gardant la « sémantique » du protocole d'origine. Cette transformation consiste à annoter[3] avec des constantes toute application d'une primitive cryptographique tel que spécifié à la définition suivante.

Définition 7.5.1 (La classe de protocoles $\mathcal{C}_2'$)**.** *La classe de protocoles $\mathcal{C}_2'$ est l'ensemble des protocoles $\Pi = [e_1; \ldots; e_\ell]$ vérifiant les quatre conditions suivantes :*

1. *tout prédicat apparaît dans au plus un status event du protocole,* i.e. *pour tout $i, j \in [\![\ell]\!]$, si $i \neq j$, $e_i = \mathsf{Q}(t_1, \ldots, t_n)$ et $u_j = \mathsf{Q}'(u_1, \ldots, u_m)$, alors $\mathsf{Q} \neq \mathsf{Q}'$,*
2. *pour tout $\mathsf{aenc}(u, v) \in \mathsf{Est}(\Pi)$, $\delta_\Pi(v) \in \mathcal{P}$,*
3. *pour tout $f(u_1, \ldots, u_n) \in \mathsf{Est}(\Pi)$, il existe une constante $c \in \mathcal{C}$ ainsi qu'un terme $u_1' \in \mathcal{T}$ tels que $u_1 = \langle c, u_1' \rangle$, et*
4. *pour tous $f(\langle c, u_1 \rangle, \ldots, u_n), f(\langle d, v_1 \rangle, \ldots, v_n) \in \mathsf{Est}(\Pi)$*

$$(f(\langle c, u_1 \rangle, \ldots, u_n))\delta_\Pi \neq (f(\langle d, v_1 \rangle, \ldots, v_n))\delta_\Pi \;\Rightarrow\; c \neq d.$$

La condition 1 force le protocole Π à être bien typé et implique donc la condition 1 de la définition 7.2.1. La condition 2 quant à elle impose que tout chiffrement asymétrique dans une exécution honnête de Π se fasse avec la clé privée d'un participant, impliquant donc la condition 2 de la définition 7.2.1. La conditions 3 stipule que toute application d'une primitive cryptographique soit marquée (ou encore *statiquement taggée*[4]) par une constante c ; alors que la condition 4 implique que deux sous-termes chiffrés ne correspondant pas au même message dans l'exécution honnête de Π se voient marqués différemment.

[3] Attention. Il ne s'agît pas exactement de la même opération de marquage qu'au chapitre précédent.

[4] Là encore le terme tag ne correspond à celui du chapitre précédent.

L'idée étant qu'en marquant ainsi les sous-termes chiffrés de Π nous restreignons l'unifiabilité des sous-termes de Π dans le sens de la condition 3 de la définition 7.2.1.

En effet, soient t et u deux sous-termes chiffrés de Π statiquement taggés différemment, t et u sont alors non-unifiables et de types différents. De plus, soit sc un scénario de Π de trace symbolique associée tr, si $t' \in \mathsf{Est}(tr)$ est une instance de t et $u' \in \mathsf{Est}(tr)$ est une instance de u, alors t' et u' sont eux aussi non-unifiables et de types différents. Inversement, soient v et w deux sous-termes chiffrés de la trace tr, et soient v' le sous-terme chiffré de Π qu'instancie v, et w' le sous-terme chiffré de Π qu'instancie w. Si v et w sont unifiables alors ils sont nécessairement statiquement taggés avec la même constante (sinon l'unification échouerait). La condition 3 de la définition 7.5.1 nous assure alors que v' et w' correspondent au même message dans l'exécution type du protocole, et donc qu'ils sont du même type. Mais alors v et w aussi sont du même type. Nous retrouvons bien là la non-unifiabilité entre sous-termes chiffrés d'une trace symbolique de Π de types différents. Les conditions 3 et 4 de la définition 7.5.1 combinées impliquent donc la condition 3 de la définition 7.2.1.

L'intérêt d'un tel fragment de la classe $\mathcal{C}_2$ est double. Le premier réside dans le fait qu'il est possible de transformer un protocole satisfaisant les conditions 1 et 2 de la définition 7.5.1 en un protocole de $\mathcal{C}'_2$, simplement en marquant chaque application d'une primitive cryptographique dans la spécification du dit protocole avec une constante. Or, la plupart des protocoles satisfont la condition 2 de la définition 7.5.1. En ce qui concerne la condition 1 de la définition 7.5.1, rappelons juste que les status events sont ajoutés à un protocole dans le but de le vérifier. Les status events introduits dans la spécification d'un protocole dépendent donc de la formule considérée. Pour les propriétés usuelles de sécurité telles que celles présentées à la section 3.2 les status events ajoutés dans un protocole sont tous différents. Ainsi, si nous nous concentrons sur ces propriétés il est possible de transformer un grand nombre de protocoles en des protocoles de $\mathcal{C}'_2$. De plus, le coup de cette transformation est très faible. En effet, l'opération de marquage statique ne fait qu'augmenter de quelques bits la taille des messages du protocole d'origine. Elle n'augmente ni le nombre de messages échangés, ni le nombre d'applications de primitives cryptographiques.

Le second intérêt réside dans le fait que le protocole ainsi construit est au moins aussi sûre que le protocole d'origine. Expliquons-nous. Un grand nombre d'attaques repose sur le fait qu'un agent déchiffrera un message en croyant en déchiffrer un d'un autre type. Ces attaques reposent donc sur des confusions concernant les types des messages échangés au cours d'une exécution d'un protocole. Comme le montre l'exemple ci-dessous il arrivera que le protocole construit soit plus sûr que le protocole d'origine, précisément parce que le type de chaque message est rendu explicite grâce aux tags statiques utilisés pour marquer les applications de primitives de chiffrement. C'est d'ailleurs une des techniques préconisées par M. Abadi et R. Needham dans [1] pour la conception de protocoles « sûrs ».

Exemple 7.5.2. *Illustrons notre propos en construisant à partir de* Π'_{Toy} *un*

protocole $\widehat{\Pi'_{\mathsf{Toy}}}$ *dans* $\mathcal{C}'_2$.

$$\begin{aligned}\widehat{\Pi'_{\mathsf{Toy}}} = [\quad & \mathsf{snd}(a, b, \mathsf{aenc}(\langle 1, \langle \mathsf{aenc}(\langle 2, n\rangle, b), a\rangle\rangle, b));\\ & \mathsf{Secret}(a, a, b, n);\\ & \mathsf{rcv}(b, a, \mathsf{aenc}(\langle\langle 1, \mathsf{aenc}(\langle 2, x\rangle, b)\rangle a, , \rangle b));\\ & \mathsf{snd}(b, a, \mathsf{aenc}(\langle 3, \langle \mathsf{aenc}(\langle 4, x\rangle, a), b\rangle\rangle, a));\\ & \mathsf{rcv}(a, b, \mathsf{aenc}(\langle 3, \langle \mathsf{aenc}(\langle 4, n\rangle, a), b\rangle\rangle, a))]\end{aligned}$$

Remarquons qu'ainsi marqués les sous-termes chiffrés

$$\mathsf{aenc}(\langle 1, \langle \mathsf{aenc}(\langle 2, n\rangle, b), a\rangle\rangle, b)) \text{ et } \mathsf{aenc}(\langle 2, x\rangle, b)$$

ne sont plus unifiables, et l'attaque sur le secret exhibée à l'exemple 3.2.1 ne peut pas être montée sur le protocole $\widehat{\Pi'_{\mathsf{Toy}}}$.

7.6 Comparaisons

Le travail le plus proche, à notre connaissance, de celui présenté dans ce chapitre est celui de J. Heather et *al.* [37]. Les auteurs y présentent eux aussi un résultat de réduction de l'espace de recherche à des exécutions bien typées. Ils montrent en effet qu'il est possible de renforcer les protocoles par un étiquetage des sous-termes apparaissant dans leur spécification. Néanmoins, leur approche présente deux faiblesses : l'une liée au modèle de protocoles considéré, l'autre au schéma d'étiquetage.

Plus précisément, les auteurs se placent dans le modèle des strand spaces [52, 36], qui exclut d'emblée les protocoles mettant en œuvre des copies en aveugle, *i.e.* des variables de type composé, ou des clés composées.

D'autre part, [37] impose un schéma d'étiquetage plus lourd que celui introduit dans ce travail (*cf.* définition 7.5.1). Là où nous proposons de ne tagger que les applications de primitives cryptographiques, les auteurs proposent quant à eux de marquer tous les sous-termes apparaissant dans la spécification des protocoles considérés. Ainsi, si nous appliquons le schéma de marquage proposé par J. Heather et *al.* à notre protocole Π_{Toy}, nous obtenons la séquence d'événements suivante :

$$\begin{aligned}\Pi_{\mathsf{Toy}} = [\quad & \mathsf{snd}(a, b, \langle 1, \mathsf{aenc}(\langle 2, \langle\langle 3, \mathsf{aenc}(\langle 4, n\rangle, b)\rangle\langle 5, a\rangle, \rangle\rangle, b)\rangle));\\ & \mathsf{rcv}(b, a, \langle 1, \mathsf{aenc}(\langle 2, \langle\langle 3, \mathsf{aenc}(\langle 4, x\rangle, b)\rangle\langle 5, a\rangle, \rangle\rangle, b)\rangle));\\ & \mathsf{snd}(b, a, \langle 1, \mathsf{aenc}(\langle 2, \langle\langle 3, \mathsf{aenc}(\langle 4, x\rangle, a)\rangle\langle 5, b\rangle, \rangle\rangle, a)\rangle));\\ & \mathsf{rcv}(a, b, \langle 1, \mathsf{aenc}(\langle 2, \langle\langle 3, \mathsf{aenc}(\langle 4, n\rangle, a)\rangle\langle 5, b\rangle, \rangle\rangle, a)\rangle)\quad]\end{aligned}$$

En sommes, nous parvenons à étendre la classe des protocoles concernés par un tel résultat de réduction, tout en diminuant la taille des messages émis par les protocoles taggés.

7.7 Conclusion

La classe de protocoles $\mathcal{C}_2$ a déjà été étudiée par le passé, et plus particulièrement le fragment $\mathcal{C}'_2$. Aux prix de certaines abstractions ou restrictions

supplémentaires, des résultats de décidabilité ont été établis pour cette classe de protocoles. Ainsi, B. Blanchet et A. Podelski dans [12] ont montré que dans le cadre d'un nombre borné de nonces, la propriété du secret et certaines versions de l'authentification sont décidables pour la classe de protocoles $\mathcal{C}'_2$. D'un autre coté, R. Ramanujam et S. P. Suresh ainsi que G. Lowe dans [47] et [40, 37] respectivement ont montré que sous réserve de restrictions supplémentaires la propriété du secret est décidable pour la classe de protocoles $\mathcal{C}'_2$. Ces restrictions consistent à interdire des protocoles admettant des secrets temporaires ou des copies en aveugles (*i.e.* des variables de type composé).

Si nous revenons maintenant aux preuves d'indécidabilité existantes (voir [19, 32] entre autres ou se reporter au chapitre 4), il apparaît que les protocoles mis en œuvre dans ces preuves ne tombent pas dans la classe de protocoles $\mathcal{C}'_2$, et que les attaques exhibées dans ces preuves ne sont pas bien typées au regard de notre système de types (section 7.1). Qui plus est, la correction des codages exhibés dans ces preuves repose sur le fait qu'il soit possible de monter des attaques mal typées.

Ces quelques remarques pourraient être interprétées comme allant dans le sens d'une réponse positive à la question de la décidabilité, du moins de la propriété du secret, pour la classe de protocoles $\mathcal{C}'_2$ tout entière. Néanmoins, la réintroduction des secrets temporaires complique dramatiquement le problème. Alors qu'il suffit de considérer une session par rôle pour trouver une attaque sur les protocole de la classe [47, 40, 37], tel n'est pas le cas pour la classe $\mathcal{C}'_2$ tout entière. En effet, l'exemple suivant montre que, pour tout entier k, il est possible de construire un protocole Π_k de $\mathcal{C}'_2$ qui admet une attaque sur la propriété ϕ_k du secret nécessitant $2k+1$ sessions pour être montée. C'est donc bien dans la réintroduction des secrets temporaires que réside la difficulté à trancher quant à la décidabilité du problème de la vérification pour la classe $\mathcal{C}'_2$.

Pour $k = 1$, le protocole Π_1 construit correspond à la séquence d'événements suivante :

$$\begin{array}{lll}
\Pi_1 = [& \mathsf{snd}(a,b, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle n_1, n_2\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, n_2\rangle\rangle, b)\rangle, a)\rangle\rangle; \\
& \mathsf{rcv}(b,a, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle x_1, x_2\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, x_2\rangle\rangle, b)\rangle, a)\rangle\rangle; \\
& \mathsf{snd}(b,a, & \langle x_2, \mathsf{senc}(\langle 4, n_3\rangle, x_1)\rangle\rangle; \\
& \mathsf{Secret}_1(b,a,b,n_3); & \\
& \mathsf{rcv}(a,b, & \langle n_2, \mathsf{senc}(\langle 4, x_3\rangle, n_1)\rangle\rangle)]
\end{array}$$

et la propriété du secret s'énonce à l'aide de la formule :

$$\phi_1 = \forall y_a.\ \forall y_b.\ \forall y_n.\ \mathcal{O}(\mathsf{Secret}_1(y_a, y_b, y_n)) \ \Rightarrow\ \neg \mathsf{learn}(y_n).$$

L'exécution exec_1 décrite ci-dessous est une exécution valide de Π_1 au regard de n'importe quelle connaissance initiale de l'intrus T_0, qui viole la propriété du secret ϕ_1 :

$$\begin{array}{lll}
\mathsf{exec}_1 = [& \mathsf{snd}(a,b, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle n_1^1, n_2^1\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, n_2^1\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{rcv}(b,a, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle n_1^1, n_2^1\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, n_2^1\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{snd}(b,a, & \langle \underline{n_2^1}, \mathsf{senc}(\langle 4, n_3^2\rangle, n_1^1)\rangle); \\
& \mathsf{rcv}(b,a, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle \underline{\underline{n_1^\epsilon}}, \underline{n_2^1}\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, \underline{n_2^1}\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{snd}(b,a, & \langle n_2^1, \mathsf{senc}(\langle 4, n_3^3\rangle, \underline{\underline{n_1^\epsilon}})\rangle); \\
& \mathsf{Secret}_1(a,b,n_3^3)] &
\end{array}$$

En apprenant le secret temporaire n_2^1, l'intrus a réussit à introduire son nonce n_1^ϵ en position de clé dans le message contenant le secret n_3^3. Il lui a fallu pour y arriver ouvrir 3 sessions.

Cette construction peut se généraliser à tout k inductivement de la manière suivante. Supposons que k soit pair. Soit le protocole Π_{k-1} obtenu selon ce même procédé et tel que $\mathsf{Secret}_{k-1}(b,a,b,n) \in \mathsf{Elmts}(\Pi_{k-1})$ et $x \in \mathcal{V}(\Pi_{k-1})$ telle que $\delta_{\Pi_{k-1}}(x) = n$. Soient aussi 4 nouvelles constantes $c_1, c_2, c_3, c_4 \in (\mathcal{C} \setminus \mathcal{C}(\Pi_{k-1}))$, deux nouveaux nonces $m_1, m_2 \in (\mathcal{N} \setminus \mathcal{N}(\Pi_{k-1}))$ et deux nouvelles variables $y_1, y_2 \in (\mathcal{V} \setminus \mathcal{V}(\Pi_{k-1}))$. $\Pi_k = \Pi'_k \mathbin{@} \Pi_{k-1}$ où

$$\begin{array}{lll}
\Pi'_k = [& \mathsf{snd}(b,a, & \langle \mathsf{aenc}(\langle c_1, \langle b, \langle m_1, n\rangle\rangle\rangle, a), \mathsf{sign}(\langle c_2, \mathsf{aenc}(\langle c_3, \langle b, n\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{rcv}(a,b, & \langle \mathsf{aenc}(\langle c_1, \langle b, \langle y_1, x\rangle\rangle\rangle, a), \mathsf{sign}(\langle c_2, \mathsf{aenc}(\langle c_3, \langle b, x\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{snd}(a,b, & \mathsf{senc}(\langle c_4, m_2\rangle, y_1)); \\
& \mathsf{Secret}_k(a,a,b,m_2); & \\
& \mathsf{rcv}(b,a, & \mathsf{senc}(\langle c_4, y_2\rangle, m_1))]
\end{array}$$

La propriété à présent considérée est

$$\phi_k = \forall y_a.\ \forall y_b.\ \forall y_n.\ \mathcal{O}(\mathsf{Secret}_k(y_a, y_b, y_n)) \ \Rightarrow\ \neg\mathsf{learn}(y_n).$$

Pour monter une attaque sur Π_k l'intrus devra d'abord apprendre le secret n de Π_{k-1} (n est l'analogue du secret temporaire n_2^1 de Π_1) puis mimer l'attaque faite sur Π_1. Le cas où k est impair est analogue à l'exception que le secret de Π_{k-1} est engendré par a et qu'il faut donc inverser dans la construction les noms de a et de b.

Illustrons nos propos en construisant le protocole Π_2 à partir du protocole

Π_1.

$$\begin{array}{lll}
\Pi_2 = [& \mathsf{snd}(b,a, & \langle \mathsf{aenc}(\langle 5, \langle b, \langle n_4, n_3\rangle\rangle\rangle, a), \mathsf{sign}(\langle 6, \mathsf{aenc}(\langle 7, \langle b, n_3\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{rcv}(a,b, & \langle \mathsf{aenc}(\langle 5, \langle b, \langle y_4, x_3\rangle\rangle\rangle, a), \mathsf{sign}(\langle 6, \mathsf{aenc}(\langle 7, \langle b, x_3\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{snd}(a,b, & \mathsf{senc}(\langle 8, n_5\rangle, y_4)); \\
& \mathsf{Secret}_2(a,a,b,n_5); & \\
& \mathsf{rcv}(b,a, & \mathsf{senc}(\langle 8, y_5\rangle, n_4)); \\
& & \\
& \mathsf{snd}(a,b, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle n_1, n_2\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, n_2\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{rcv}(b,a, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle x_1, x_2\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, x_2\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{snd}(b,a, & \langle x_2, \mathsf{senc}(\langle 4, n_3\rangle, x_1)\rangle); \\
& \mathsf{Secret}_1(b,a,b,n_3); & \\
& \mathsf{rcv}(a,b, & \langle n_2, \mathsf{senc}(\langle 4, x_3\rangle, n_1)\rangle)]
\end{array}$$

L'exécution exec_2 décrite ci-dessous est une exécution valide de Π_2 au regard de n'importe quelle connaissance initiale de l'intrus T_0, qui viole la propriété du secret ϕ_2.

$$\begin{array}{lll}
\mathsf{exec}_2 = [& \mathsf{snd}(b,a, & \langle \mathsf{aenc}(\langle 5, \langle b, \langle n_4^1, n_3^1\rangle\rangle\rangle, a), \mathsf{sign}(\langle 6, \mathsf{aenc}(\langle 7, \langle b, n_3^1\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{rcv}(a,b, & \langle \mathsf{aenc}(\langle 5, \langle b, \langle n_4^1, n_3^1\rangle\rangle\rangle a), \mathsf{sign}(\langle 6, \mathsf{aenc}(\langle 7, \langle b, n_3^1\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{snd}(a,b, & \mathsf{senc}(\langle 8, n_5^2\rangle, n_4^1)); \\
& \mathsf{Secret}_2(a,b,n_5^2); & \\
& \mathsf{rcv}(b,a, & \mathsf{senc}(\langle 8, n_5^2\rangle, n_4^1)); \\
& \mathsf{snd}(a,b, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle n_1^2, n_2^2\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, n_2^2\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{rcv}(b,a, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle n_1^2, n_2^2\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, n_2^2\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{snd}(b,a, & \langle \underline{n_2^2}, \mathsf{senc}(\langle 4, n_3^1\rangle, n_1^2)\rangle); \\
& & \\
& \mathsf{snd}(b,a, & \langle \mathsf{aenc}(\langle 5, \langle b, \langle n_4^3, n_3^3\rangle\rangle\rangle, a), \mathsf{sign}(\langle 6, \mathsf{aenc}(\langle 7, \langle b, n_3^3\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{rcv}(a,b, & \langle \mathsf{aenc}(\langle 5, \langle b, \langle n_4^3, n_3^3\rangle\rangle\rangle, a), \mathsf{sign}(\langle 6, \mathsf{aenc}(\langle 7, \langle b, n_3^3\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{snd}(a,b, & \mathsf{senc}(\langle 8, n_5^4\rangle, n_4^3)); \\
& \mathsf{rcv}(b,a, & \mathsf{senc}(\langle 8, n_5^4\rangle, n_4^3)); \\
& \mathsf{rcv}(b,a, & \langle \mathsf{aenc}(\langle 1, \langle a, \langle \underline{\underline{n_1^\epsilon}}, \underline{n_2^2}\rangle\rangle\rangle, b), \mathsf{sign}(\langle 2, \mathsf{aenc}(\langle 3, \langle a, \underline{n_2^2}\rangle\rangle, b)\rangle, a)\rangle); \\
& \mathsf{snd}(b,a, & \langle n_2^2, \mathsf{senc}(\langle 4, \underline{n_3^3}\rangle, \underline{\underline{n_1^\epsilon}})\rangle); \\
& & \\
& \mathsf{rcv}(a,b, & \langle \mathsf{aenc}(\langle 5, \langle b, \langle \underline{\underline{n_4^\epsilon}}, \underline{n_3^3}\rangle\rangle\rangle, a), \mathsf{sign}(\langle 6, \mathsf{aenc}(\langle 7, \langle b, \underline{n_3^3}\rangle\rangle, a)\rangle, b)\rangle); \\
& \mathsf{snd}(a,b, & \mathsf{senc}(\langle 8, n_5^5\rangle, \underline{\underline{n_4^\epsilon}})); \\
& \mathsf{Secret}_2(a,b,n_5^5) &]
\end{array}$$

En apprenant le secret temporaire n_2^2, l'intrus a réussit à introduire son nonce n_1^ϵ en position de clé dans le message contenant le « secret temporaire » n_3^3. Il introduit ainsi sa clé dans le message contenant le secret n_5^5. Il lui a fallu pour

y arriver ouvrir 5 sessions.

Il semblerait aux vues de ces quelques considérations que la classe $\mathcal{C}_2'$ soit aux frontières entre décidabilité et indécidabilité.

Chapitre 8

Conclusion et perspectives

Revenons brièvement sur les résultats principaux obtenus au cours de cette thèse, et essayons de dégager les perspectives de recherche ouvertes par ceux-ci.

Récapitulatif des résultats de la thèse

Au **chapitre 4**, nous avons présenté un premier résultat négatif. Nous avons fourni une preuve formelle de l'indécidabilité du problème de la vérification des protocoles de sécurité dans le cadre de messages de taille bornée. Certes, ce résultat était attendu, plusieurs codages de problèmes indécidables vers le problème de la vérification des protocoles de sécurité ayant déjà été proposés. Néanmoins, la correction de ces codages n'était pas formellement établie, et reposait par ailleurs sur des propriétés de la modélisation que les « vrais » protocoles ne satisfont pas. Or, en proposant un codage plus réaliste, nous avons établi que l'indécidabilité n'est pas un simple artefact du modèle.

Au **chapitre 6**, nous avons présenté la classe de protocoles $\mathcal{C}_1$ ainsi que la classe de propriétés associées Φ_1 pour lesquelles le problème de la vérification est décidable. Plus précisément, nous avons montré qu'il suffit de considérer un nombre borné de sessions afin de décider si un protocole dans $\mathcal{C}_1$ vérifie une propriété donnée dans Φ_1. Il s'est avéré que pour la propriété du secret ou de l'authentification en particulier il suffit de considérer une session par rôle du protocole. Notons que tout protocole peut être transformé comme nous l'avons vu (section 6.1) en un protocole de $\mathcal{C}_1$. Ce résultat est d'autant plus intéressant qu'il existe, à l'heure actuelle, de nombreux outils de vérification pour un nombre borné de sessions (Avispa entre autres [7]).

Enfin, au **chapitre 7**, nous avons établi que pour la classe de protocoles $\mathcal{C}_2$ vérifiant le célèbre critère dit de « non-unifiabilité des sous-termes » (énoncé par M. Abadi et R. Needham dans [1]) : il suffit de considérer des exécutions bien

typées afin de construire une attaque sur des propriétés de traces telles que le secret ou l'authentification. Notons que ce résultat de réduction légitime, pour la classe $\mathcal{C}_2$, l'abstraction sur laquelle reposent plusieurs outils de vérification. Ainsi, outre l'intérêt théorique de ce résultat, celui-ci présente d'importantes applications pratiques en justifiant l'utilisation d'outils tels que [6, 25, 14, 22] pour la vérification des protocoles de $\mathcal{C}_2$.

Perspectives

Les deux résultats de réduction (chapitres 6 et 7) ont été obtenus dans le même cadre, à savoir sous l'hypothèse du chiffrement parfait en ne considérant pas les propriétés algébriques des primitives cryptographiques (modèle par rôles avec filtrage), et ce, uniquement pour des propriétés de trace et de déduction (exprimables dans $\mathcal{PS}$-LTL$^-$).

Une poursuite naturelle de ces travaux consiste, dans un premier temps, à regarder comment l'affaiblissement de l'hypothèse du chiffrement parfait affecterait nos deux résultats. L'intérêt étant que, un nombre non-négligable de protocoles font appel à des primitives cryptographiques vérifiant des propriétés algébriques (*e.g.* « ou » exclusif, exponentiation modulaire, *etc.*), et se retrouvent donc d'emblée exclus des classes $\mathcal{C}_1$ et $\mathcal{C}_2$.

Dans la même veine, il serait légitime de chercher à étendre la classe des propriétés considérées jusqu'ici. En effet, nous n'avons, pour le moment, envisagé que des propriétés de trace et de déduction. Or, il existe toute une famille d'importantes propriétés reposant sur la notion d'indistinguabilité et exprimables en termes d'équivalences observationnelles (*e.g.* anonymat, secret fort, *etc.*), et qui par là même ne rentrent pas dans le cadre des classes Φ_1 et Φ_2.

A ce stade, il n'est pas clair du tout comment procéder afin d'établir de telles extensions, à supposer bien sûr qu'elles soient possibles. Ce qui est certain en revanche, c'est qu'il sera nécessaire de nous abstraire, dans un premier temps, de la procédure de vérification sur laquelle reposent les preuves fournies dans le présent travail. En d'autres termes, nous devrons établir à un niveau « sémantique » la propriété de conservativité que nous assure la dite procédure à un niveau syntaxique.

Enfin, une autre direction de recherche intéressante mais ambitieuse concerne la classe de protocoles $\mathcal{C}_2$. En conclusion du chapitre 7, nous avons exhibé des arguments aussi bien en faveur de la décidabilité que de l'indécidabilité du problème de la vérification pour cette classe de protocoles. Or, comme nous l'avons vu, tout protocole peut être transformé à moindre coût en un protocole « sémantiquement équivalent » de $\mathcal{C}_2$. Cela explique l'importance accordée à cette classe (voir [40, 12, 46, 47, 5]), et justifie que les recherches visant à répondre à la difficile question de sa décidabilité soient poursuivies.

Bibliographie

[1] M. Abadi and R. M. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1) :6–15, 1996.

[2] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *TCS '00 : Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics*, pages 3–22, London, UK, 2000. Springer-Verlag.

[3] R. M. Amadio and W. Charatonik. On name generation and set-based analysis in dolev-yao model. Technical Report 4379, INRIA, 2002.

[4] M. Arapinis, S. Delaune, and S. Kremer. From one session to many : Dynamic tags for security protocols. In I. Cervesato, editor, *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, Lecture Notes in Artificial Intelligence, Doha, Qatar, Nov. 2008. Springer. To appear.

[5] M. Arapinis and M. Duflot. Bounding messages for free in security protocols. In V. Arvind and S. Prasad, editors, *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 376–387, New Delhi, India, Dec. 2007. Springer.

[6] A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The aviss security protocol analysis tool. In *CAV '02 : Proceedings of the 14th International Conference on Computer Aided Verification*, pages 349–353. Springer-Verlag, 2002.

[7] A. Armando et al. The Avispa tool for the automated validation of internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, pages 281–285. Springer, 2005.

[8] D. Beauquier and F. Gauche. How to guarantee secrecy for cryptographic protocols. *CoRR*, abs/cs/0703140, 2007.

[9] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *Proc. 30th Annual ACM Symposium on the Theory of Computing (STOC'98)*, pages 419–428. ACM Press, 1998.

[10] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen. *Systems and Software Verification. Model-Checking Techniques and Tools*. Springer, 2001.

[11] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW '01 : Proceedings of the 14th IEEE workshop on Computer Security Foundations*, pages 82–96, Washington, DC, USA, 2001. IEEE Computer Society.

[12] B. Blanchet and A. Podelski. Verification of Cryptographic Protocols : Tagging Enforces Termination. *Theoretical Computer Science*, 333(1-2) :67–90, Mar. 2005. Special issue FoSSaCS'03.

[13] M. Boreale and M. G. Buscemi. Experimenting with sta, a tool for automatic analysis of security protocols. In *SAC '02 : Proceedings of the 2002 ACM symposium on Applied computing*, pages 281–285, New York, NY, USA, 2002. ACM.

[14] L. Bozga, Y. Lakhnech, and M. Périn. Pattern-based abstraction for verifying secrecy in protocols. *International Journal on Softwate Tools for Technology Transfer*, 8(1) :57–76, 2006.

[15] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Extending the Dolev-Yao intruder for analyzing an unbounded number of sessions. In *Proc. 17th Int. Work. on Computer Science Logic (CSL03)*, volume 2803 of *LNCS*, pages 128–141. Springer, 2003.

[16] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, January 2000.

[17] E. M. Clarke, S. Jha, and W. Marrero. Verifying security protocols with brutus. *ACM Trans. Softw. Eng. Methodol.*, 9(4) :443–487, 2000.

[18] H. Comon. Résolution de contraintes et recherche d'attaques pour un nombre borné de sessions. Available at http ://www.lsv.ens-cachan.fr/ comon/CRYPTO/bounded.ps.

[19] H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. Research Report LSV-01-13, Laboratoire Spécification et Vérification, ENS Cachan, France, dec 2001. 98 pages.

[20] R. Corin. *Analysis Models for Security Protocols*. Phd thesis, University of Twente, 2006.

[21] R. Corin, S. Etalle, and A. Saptawijaya. A logic for constraint-based security protocol analysis. In *SP '06 : Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 155–168, Washington, DC, USA, 2006. IEEE Computer Society.

[22] V. Cortier. A guide for SECURIFY. Technical Report 13, projet RNTL EVA, 2003. 9 pages.

[23] V. Cortier. Vérifier les protocoles cryptographiques. *Technique et Science Informatiques*, 24(1) :115–140, 2005.

[24] V. Cortier, J. Delaitre, and S. Delaune. Safely composing security protocols. In *Proc. 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'07)*, volume 4855 of *LNCS*. Springer, 2007.

[25] V. Cortier, J. Millen, and H. Rueβ. Proving secrecy is easy enough. In *CSFW '01 : Proceedings of the 14th IEEE workshop on Computer Security Foundations*, pages 97–108. IEEE Computer Society, 2001.

[26] V. Cortier, B. Warinschi, and E. Zălinescu. Synthesizing secure protocols. In *Proc. 12th European Symposium On Research In Computer Security (ESORICS'07)*, volume 4734 of *LNCS*, pages 406–421. Springer, 2007.

[27] V. Cortier and E. Zalinescu. Deciding key cycles for security protocols. In *Proc. 13th Inter. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *LNCS*, pages 317–331. Springer, 2006.

[28] S. Delaune. Note : Constraint solving procedure. Available at http ://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/CDD-fsttcs07-addendum.pdf.

[29] D. Dolev and A. C. Yao. On the security of public key protocols. *Symposium on Foundations of Computer Science*, 0 :350–357, 1981.

[30] D. Dolev and A. C.-C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2) :198–207, 1983.

[31] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Multiset rewriting and the complexity of bounded security protocols. *J. Comput. Secur.*, 12(2) :247–311, 2004.

[32] N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *proceedings of the Workshop on Formal Methods and Security Protocols-FMSP*, 1999.

[33] S. B. Fröschle. The insecurity problem : Tackling unbounded data. In *20th IEEE Computer Security Foundations Symposium, CSF 2007, 6-8 July 2007, Venice, Italy*, pages 370–384. IEEE Computer Society, 2007.

[34] T. Genet and F. Klay. Rewriting for cryptographic protocol verification. In *CADE-17 : Proceedings of the 17th International Conference on Automated Deduction*, pages 271–290, London, UK, 2000. Springer-Verlag.

[35] J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *IPDPS '00 : Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 977–984, London, UK, 2000. Springer-Verlag.

[36] J. Guttman and J. Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2) :333–380, 2002.

[37] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. *Journal of Computer Security*, 11(2) :217–244, 2003.

[38] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proc. 23rd Annual International Cryptology Conference (CRYPTO'03)*, volume 2729 of *LNCS*, pages 110–125. Springer, 2003.

[39] G. Lowe. A hierarchy of authentication specifications. In *CSFW '97 : Proceedings of the 10th IEEE workshop on Computer Security Foundations*, page 31, Washington, DC, USA, 1997. IEEE Computer Society.

[40] G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(1), 1999.

[41] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Trans. Program. Lang. Syst.*, 4(2) :258–282, 1982.

[42] A. J. Mayer and M. Yung. Secure protocol transformation via "expansion" : From two-party to groups. In *Proc. 6th ACM Conference on Computer and Communications Security (CCS'99)*, pages 83–92. ACM Press, 1999.

[43] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*. ACM Press, 2001.

[44] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using mur/spl phi/. In *SP '97 : Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 141–151, Washington, DC, USA, 1997. IEEE Computer Society.

[45] D. Monniaux. Abstracting cryptographic protocols with tree automata. *Sci. Comput. Program.*, 47(2-3) :177–202, 2003.

[46] R. Ramanujam and S. P. Suresh. A decidable subclass of unbounded security protocols. In *Proc. Work. on Issues in the Theory of Security (WITS'03)*, 2003.

[47] R. Ramanujam and S. P. Suresh. Tagging makes secrecy decidable with unbounded nonces as well. *FSTTCS : Foundations of Software Technology and Theoretical Computer Science*, 2914, 2003.

[48] R. Ramanujam and S. P. Suresh. Undecidability of the secrecy problem for security protocols, 2003. Available at http ://www.imsc.res.in/ jam/TR-undec.ps.gz.

[49] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is np-complete. In *CSFW '01 : Proceedings of the 14th IEEE workshop on Computer Security Foundations*, page 174, Washington, DC, USA, 2001. IEEE Computer Society.

[50] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science*, 299(1-3) :451–475, 2003.

[51] M. Sipser. *Introduction to the Theory of Computation*. PWS, first edition, 1996.

[52] J. Thayer, J. Herzog, and J. Guttman. Strand spaces : proving security protocols correct. *IEEE Journal of Computer Security*, 7(2-3) :191–230, 1999.

[53] C. Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *CADE-16 : Proceedings of the 16th International Conference on Automated Deduction*, pages 314–328, London, UK, 1999. Springer-Verlag.