Towards a Logic for Verification of Security Protocols

Vincent Bernat

LSV
ENS Cachan
61, avenue du President Wilson
94235 Cachan Cedex
France
bernat@lsv.ens-cachan.fr

1 Introduction

The verification of cryptographic protocols has been intensively studied these last years. It is a *model checking* problem: given a protocol P and a property ϕ, does P satisfies ϕ ?

Most well-known properties are *authentication* and *secrecy*, but *fairness* [1, 2], *anonymity* [3] and non-repudiation are other possible security properties we are willing to check on a protocol. However, such properties are not always formally defined.

For several years, cryptographic protocols were not rigorously proved and the research aimed at building protocols and searching flaws in them. The first step towards formal proofs of protocols was to define formal models like [4–8]. These models are used to express the semantics of the protocols and sometimes their properties.

There is a wide range of relevant authentication properties [9]. Secrecy may be declined in several versions. Common secrecy and authentication properties are listed in table 1.

Table 1. Common cryptographic properties

Secrecy	Authentication
from the intruder point of view	vivacity
temporary	weak agreement
partial	non-injective agreement
in a group of actors	injective agreement
etc.	full agreement
	etc.

All models use *ad hoc* methods to express and prove these properties. Properties are then tightened to the protocol (see the example below) or to the model itself: checking several properties on the same protocol often requires to build several models. The converse is also true: if one wishes to change the model used to express the protocol, the underlying properties must be modified. An intermediate layer between the description of the protocol and the properties would allow to build a language targeted to the expression of cryptographic properties, independently of any model used to express protocols.

Some models [5] use expressive formalisms like Horn clauses. It would be possible to build a language of properties using them but we could loose what is specific to cryptographic protocols and be unable to work efficiently with them.

Example of spi-calculus Spi-calculus [6] can be used to express secrecy. The protocol has to be written using this process calculus, then secrecy is specified using observational equivalence:

$$\forall M, M', M \simeq M' \Longrightarrow \mathrm{Inst}(M) \simeq \mathrm{Inst}(M')$$

Inst is the transcription of the protocol in the process algebra. It *does not* include an intruder process. M is a given parameter for the protocol and is generally the secret. This defines however *one* possible secrecy property. There are alternative possibilities (see table 1).

If we focus on authentication, the problem is even worse. In spi-calculus, authentication requires writing a new process, tailored to the process we want to check:

$$\forall M, \mathrm{Inst}(M) \simeq \mathrm{Inst}_{\mathrm{spec}}(M)$$

There is no automatic way to build the specification process $\text{Inst}_{\text{spec}}$ from an abstract authentication property and the protocol process. So each time one gets a new protocol, one must first write this specification process and then argue about its relevance. Another drawback of this approach is that it is not clear that we express the same properties for two different protocols.

If we look at what has been done for reactive systems, the verification of such systems uses three components:

- Many abstractions of the real system: automata networks, petri nets, process algebra, etc.
- One transition system: it is the glue between semantics of protocols and interpretations of formulas.
- Many (temporal) logics: LTL, CTL, their variants, etc.

The use of these three layers allow to change easily either the logic or the abstraction, minimizing the work to prove new properties and allowing to compare easily each layer. The comparison of different properties or models is also facilitated: several properties can be stated for a single model.

There is no such a distinction for cryptographic protocols. Our ultimate goal is to design a logic for security properties, which is independent of the protocols specification language. This implies, as in the reactive systems case, the definition of a class of models (transition systems in the reactive systems) which will serve for both the semantics of the protocols and the semantics of the properties.

We do not intend to define a new model for cryptographic protocols, but rather we try to point what is specific to security protocols, independently of the specification language. Indeed, we could simply stick to transition systems. However, we need unbounded branching (a session can be start at each state) and unbounded states (number of sessions not bounded, knowledge of the intruder not bounded, etc.); such a transition system does not handle specific features of security protocols. For instance, a given session which had been played once can be replayed later. These particularities lead us to define a new class of transition system.

The first part of this paper will introduce the class of transition systems and its (very few) invariants. The second part will discuss the logic, which will be used to express the properties of the transition system and give some examples.

2 The transition system

2.1 States

The transition system is a binary relation between states[1]. A state is a triple $< \phi, I, e >$ where:

- I is the knowledge which has leaked on the network: the secrecy property can easily be expressed
- ϕ associates a triple agent, session, role ($< A, s, r >$) to a tuple:
  - current step number of A (for session s and role r)
  - the agents that A remember of (for ...)
  - the other terms A remember of (for ...)
  - the last term A received (in session s and for role r)
- e is the last message sent on the network

What is a term, an agent, a step, a session, etc. is left to the protocol description.

The knowledge I which has leaked on the network is commonly seen as the intruder knowledge. Howebver, this does not mean that there is a single dishonest honest or intruder.

A same agent may play in different sessions and she may play in the same session different roles; to model the states of the agents, we have to distinguish sessions and roles. An agent plays a program, so she should know at what step she is. She usually plays with a given sets of agents and has to memorize who they are; this is also useful for agreement properties where different actors have to agree on their respective identities. An agent may have a bounded or unbounded memory for the other terms (for example, for nonces they received or they generated, but it is also possible to memorize some data from one session to another). At least, an actor has a short term memory where she puts the last term she received without putting it into her memory. The semantics is not clear if an agent may perform a step without receiving a message (or receiving several messages) or if many agents may move during the same transition. In most models, these behaviors are serializable: if an actor receives no message, we can say she receives an empty message, if she receives many messages, she may receive them one after the other. moment.

The last message sent e is useful to get the message sent by the actor which has just performed the last step. The same remarks than above about serialization applies: we can consider this message unique.

[1] We could also consider a family of relations, one for each communication channel, which would allow to consider properties of spi-calculus. These multiple relations can then be coded into one relation.

2.2 Inference rules

As it is, the transition system is too general for further analysis: we cannot prove useful properties on cryptographic protocols without adding rules.

We claim that cryptographic protocols specificities lie in two invariants: *honest replay* and *intruder attack*. We will only consider transition systems which are invariant by these two inference rules.

Honest replay Honest replay models the fact that if some transitions are possible in some state, they have to be possible in some other compatible state. This allows, for example, to force a set of actors to replay together a session they have already played, modulo some renaming.

Let us precise these notions. Let σ be a substitution whose domain is session numbers. We assume nonces to be written $f(A, s, n, \alpha)$ where A is the actor who generated the nonce, s the session in which he generated it, n the step and α an index to distinguish different nonces generated at the same step.

We define an operator $\oplus$ which works on partial states, i.e states which may not have all three components of the tuple $< \phi, I, e >$, e.g, $< \_, \_, e >$ is a partial state containing just the last message sent on the network. $\oplus$ is an associative and commutative operator such that $< \phi, I, e > = < \_, \_, e > \oplus < \_, I, \_ > \oplus < \phi, \_, \_ >$. Moreover $< \_, \_, e > \oplus < \_, \_, e' >$ is not allowed, $< \_, I, \_ > \oplus < \_, I', \_ > = < \_, I \cup I', \_ >$ and $< \phi, \_, \_ > \oplus < \phi', \_, \_ >$ is the union of relations defined by ϕ and ϕ' only if the result is still a function (there is no conflict).

The honest replay inference rule says that if one can go from state q to a state q', then, one must be able to go from state $q\sigma$ to $q'\sigma$ to which you can add "past context" q'' (because the replay takes place *later*) as long as there is no conflict with $q\sigma$ and $q'\sigma$:

$$\forall q, q', q'', \sigma, \frac{q \longrightarrow q'}{q'' \oplus q\sigma \longrightarrow q'' \oplus q'\sigma}$$

Intruder attack Intruder attack allows the intruder to interact with agents by providing them with some crafted terms matching what they expect. The targeted agent will play her rule as if the message originated from the expected agent. This rule may be composed with the previous one.

Let q be any partial state, e_1, e_2, e_3 terms (representing messages sent on the network), I_1 and I_2 two sets of terms (representing the evolution of the intruder knowledge), A, B two actors $q_{\Omega}^{x} : (\Omega, s_{\Omega}, r_{\Omega}) \mapsto < n_{\Omega} + x, \ldots >$ a state function (with $x = 1$ or $x = 2$, $\Omega = A$ or $\Omega = B$, n_{Ω} a possible step, r_{Ω} a possible role, s_{Ω} a session number). Intruder attack states that if e_2 is included in the intruder knowledge of q, then an intruder may spoof B, who plays before A, because he can send the message A would have sent. Then A plays, allowing to compose this rule several times:

$$\frac{q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \longrightarrow q \oplus \mathbf{q_A^2} \oplus \mathbf{q_B^1} \oplus e_2 \oplus I_1 \longrightarrow q \oplus q_A^2 \oplus q_B^2 \oplus e_3 \oplus I_2}{q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \longrightarrow q \oplus \mathbf{q_A^1} \oplus \mathbf{q_B^2} \oplus e_3 \oplus I_2 \longrightarrow q \oplus q_A^2 \oplus q_B^2 \oplus e_2 \oplus I_2} \text{ if } e_2 \in q_{|I}$$

Example Our transition system includes the three states below, with transition between the first and the second, then between the second and the third: first A plays, then B.

$$\text{state}_1 \begin{cases} I : A, B, \{f(B, s_2, 3)\}_{K_A}, \ldots \\ \phi : (A, s_1, \text{init}) \mapsto \langle \mathbf{n-1}, < A, B >, f(A, s_1, 1), \{f(B, s_2, 3)\}_{K_A} \rangle \\ \quad (B, s_2, \text{resp}) \mapsto \langle \mathbf{m-1}, < A, B >, f(B, s_2, 1), \{f(B, s_2, 3), A, f(A, s_1, 1)\}_{K_B} \rangle \\ e : \{f(B, s_2, 3), A, 4\}_{K_A} \end{cases}$$

$$\text{state}_2 \begin{cases} I : A, B, \{f(B, s_2, 3)\}_{K_A}, \{f(B, s_2, 3), A, 4\}_{K_A}, \ldots \\ \phi : (A, s_1, \text{init}) \mapsto \langle \mathbf{n}, < A, B >, f(A, s_1, 1), \{f(B, s_2, 3), A, 4\}_{K_A} \rangle \\ \quad (B, s_2, \text{resp}) \mapsto \langle \mathbf{m-1}, < A, B >, f(B, s_2, 1), \{f(B, s_2, 3), A, f(A, s_1, 1)\}_{K_B} \rangle \\ e : \{B, A\}_{K_B} \end{cases}$$

$$\text{state}_3 \begin{cases} I : A, B, \{f(B, s_2, 3)\}_{K_A}, \{f(B, s_2, 3), A, 4\}_{K_A}, \{B, A\}_{K_B}, \ldots \\ \phi : (A, s_1, \text{init}) \mapsto \langle \mathbf{n}, < A, B >, f(A, s_1, 1), \{f(B, s_2, 3), A, 4\}_{K_A} \rangle \\ \quad (B, s_2, \text{resp}) \mapsto \langle \mathbf{m}, < A, B >, f(B, s_2, 1), \{B, A\}_{K_B} \rangle \\ e : \{B, A, f(B, s_2, 1)\}_{K_A} \end{cases}$$

Suppose we have some state like the following one:

$$\text{state}'_1 \begin{cases} I: A, B, \{f(B, s'_2, 3)\}_{K_A}, \{\mathbf{B}, \mathbf{A}\}_{\mathbf{K_B}}, \ldots \\ \phi: (A, s'_1, \text{init}) \mapsto \langle \mathbf{n-1}, < A, B >, f(A, s'_1, 1), \{f(B, s'_2, 3)\}_{K_A} \rangle \\ \quad (B, s'_2, \text{resp}) \mapsto \langle \mathbf{m-1}, < A, B >, f(B, s'_2, 1), \{f(B, s'_2, 3), A, f(A, s'_1, 1)\}_{K_B} \rangle \\ e: \{f(B, s'_2, 3), A, 4\}_{K_A} \end{cases}$$

state'_1 is almost identical to state_1: if we define σ as the substitution which replaces s_1 by s'_1 and s_2 by s'_2, we can write $\text{state}'_1 = q'' \oplus \text{state}_1\sigma$ where q'' is a partial state containing only the I component: $\{B, A\}_{K_B}$[2]. We can use the *honest replay* rule twice to get the two following states where $\text{state}'_1 \rightarrow \text{state}'_2 \rightarrow \text{state}'_3$ are valid transitions:

$$\text{state}'_2 \begin{cases} I: A, B, \{f(B, s'_2, 3)\}_{K_A}, \{f(B, s'_2, 3), A, 4\}_{K_A}, \{\mathbf{B}, \mathbf{A}\}_{\mathbf{K_B}}, \ldots \\ \phi: (A, s'_1, \text{init}) \mapsto \langle \mathbf{n}, < A, B >, f(A, s'_1, 1), \{f(B, s'_2, 3), A, 4\}_{K_A} \rangle \\ \quad (B, s'_2, \text{resp}) \mapsto \langle \mathbf{m-1}, < A, B >, f(B, s'_2, 1), \{f(B, s'_2, 3), A, f(A, s'_1, 1)\}_{K_B} \rangle \\ e: \{\mathbf{B}, \mathbf{A}\}_{\mathbf{K_B}} \end{cases}$$

$$\text{state}'_3 \begin{cases} I: A, B, \{f(B, s'_2, 3)\}_{K_A}, \{f(B, s'_2, 3), A, 4\}_{K_A}, \{B, A\}_{K_B}, \ldots \\ \phi: (A, s'_1, \text{init}) \mapsto \langle \mathbf{n}, < A, B >, f(A, s'_1, 1), \{f(B, s'_2, 3), A, 4\}_{K_A} \rangle \\ \quad (B, s'_2, \text{resp}) \mapsto \langle \mathbf{m}, < A, B >, f(B, s'_2, 1), \{B, A\}_{K_B} \rangle \\ e: \{B, A, f(B, s'_2, 1)\}_{K_A} \end{cases}$$

The message sent in state'_2 is $\{B, A\}_{K_B}$ which is included in I part of state'_1. We are allowed to use the *intruder attack* rule to build state''_2:

$$\text{state}''_2 \begin{cases} I: A, B, \{f(B, s'_2, 3)\}_{K_A}, \{f(B, s'_2, 3), A, 4\}_{K_A}, \{\mathbf{B}, \mathbf{A}\}_{\mathbf{K_B}}, \ldots \\ \phi: (A, s'_1, \text{init}) \mapsto \langle \mathbf{n-1}, < A, B >, f(A, s'_1, 1), \{f(B, s'_2, 3)\}_{K_A} \rangle \\ \quad (B, s'_2, \text{resp}) \mapsto \langle \mathbf{m}, < A, B >, f(B, s'_2, 1), \{B, A\}_{K_B} \rangle \\ e: \{B, A\}_{K_B} \end{cases}$$

The *intruder attack* rule allows us to say that $\text{state}'_1 \rightarrow \text{state}''_2 \rightarrow \text{state}'_3$ are valid transitions of the transition system.

3 A logic

Let us consider now what could be a possible syntax for security properties. We cannot just choose one of standard temporal logics: they are usually limited to a finite number of atomic propositions while the information contained in each state is not bounded.

Many logics may be relevant. Hence we will not define a precise one: we only put forward some necessary ingredients.

The first step is to choose a logic to express properties on each state. For example, we want to be able to say that for a state $< \phi, I, e >$, there exists a session s where $\phi(A, s, \text{init})$ equals to a given value: it will be useful to express that A has played a given step.

The second step is to add a temporal dimension to this logic. Past operators like X^{-1} (*previous*), S (*weak since*) seem mandatory: we need them, for example, to check if, in the past, a given actor has played a given step with some given parameters. Operators derived from branching temporal logics are welcome to express more complex properties. Authentication properties like injective agreement (B receives his last message less often then A sends it, see [9]) needs a way to count the number of times a property is satisfied between two states. This has to be reflected at the syntactic level such as in [10].

As an example, suppose we have a first order logic with fixed domain describing the properties of each state. We add the operators X^{-1} and G (*always*) with their usual semantics and an operator which binds variables to the number of times a property is satisfied since they were instantiated. These variables may then be used to describe properties of each state. Given a property P, we can express the fact that P is true on even states:

$$[x : \mathsf{true}].\mathsf{G}((\exists k.2k = x) \Rightarrow P)$$

With such a logic, we can express properties like secrecy (even temporary one) and injective agreement without any nonces (there is agreement on the identity of actors only):

[2] In the real world, we would have state_3 included in q'' since this state happens "later". I would then contain $\{B, A\}_{K_B}$.

$$\forall A, B$$

$$[x : \exists s\ \phi(A, s, \mathrm{init}) = < n, < A, B, \cdots >, \cdots > \\ \mathsf{X}^{-1}\phi(A, s, \mathrm{init}) = < n-1, < A, B, \cdots >, \cdots >]$$

$$[y : \exists s\ \phi(B, s, \mathrm{resp}) = < m, < B, A, \cdots >, \cdots > \\ \mathsf{X}^{-1}\phi(B, s, \mathrm{resp}) = < m-1, < B, \cdots >, \cdots >]$$

$$\mathsf{G}(y \leq x)$$

We count the number of times the actor A playing the initiator role terminates her part of the protocol (assuming n is the last step) with B and the number of time the actor B playing the responder role terminates his part (assuming m is the last step) with A. The property is true if and only if the first number is always greater or equal to the second.

This is the same agreement as in [9], without any agreement on specific terms (like nonces). From this simple example, we can see that basic properties require complex formulas, which indicates that this logic may be not well suited here: it is too verbose.

4 Future works

Setting the transition system and a minimal set of inference rules on it is only a first step towards a logic for cryptographic protocols. Future works include:

- design of a logic based either on temporal logics;
- expression of common (and less common) properties using this logic: all authentication properties given in [9], variants of secrecy, more advanced properties like fairness, etc;
- showing that all models, when expressed as transition systems, satisfy the invariants of the transition system;
- use this framework to prove some results, like, for injective agreement, the necessity of fresh data in the "last message" or the reduction of authentication to secrecy.

References

1. Chadha, R., Kanovich, M., Scedrov, A.: Inductive methods and contract-signing protocols. In: Proceedings of the 8th ACM conference on Computer and Communications Security, ACM Press (2001) 176–185
2. Shmatikov, V., Mitchell, J.C.: Analysis of abuse-free contract signing. Lecture Notes in Computer Science **1962** (2001)
3. Schneider, S., Sidiropoulos, A.: CSP and anonymity. In: ESORICS. (1996) 198–218
4. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security **6** (1998) 85–128
5. Comon-Lundh, H., Cortier, V.: Security properties: two agents are sufficient. In: Proceedings of the 12th European Symposium On Programming (ESOP'03). Volume 2618 of Lecture Notes in Computer Science., Warsaw, Poland, Springer Verlag (2003) 99–113
6. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. In: Fourth ACM Conference on Computer and Communications Security, ACM Press (1997) 36–47
7. Millen, J.K., Ruess, H.: Protocol-independent secrecy. In: IEEE Symposium on Security and Privacy. (2000) 110–209
8. Thayer, J., Herzog, J., Guttman, J.: Strand spaces: Proving security protocols correct. Journal of Computer Security (1999)
9. Lowe, G.: A hierarchy of authentication specifications. In: PCSFW: Proceedings of The 10th Computer Security Foundations Workshop, IEEE Computer Society Press (1997)
10. Bouajjani, A., Echahed, R., Habermehl, P.: On the verification of nonregular properties for nonregular processes. In Press, I.C.S., ed.: Proceedings of the tenth annual IEEE Symposium on Logic in Computer Science. (1995) 123–133