Soundness of abstract cryptography
Lecture Notes

Hubert Comon-Lundh

November 19, 2007

These are preliminary notes, which have not been reviewed. Please comment, criticize, report mistakes to `h.comon-lundh@aist.go.jp`

1 Introduction

One of the simplest security question concerns confidentiality. The goal is to hide some confidential information, while using a public network.

This can be stated as follows: we assume that $m_1, \ldots, m_n$ is a message sequence and we ask whether an eavesdropper can retrieve some information from this sequence. Specifying this formally, we introduce another message sequence $m'_1, \ldots, m'_n$, which is identical to $m_1, \ldots, m_n$, but in which every confidential information has been replaced with random bitstrings (of appropriate length). Is it possible to distinguish the two message sequences ? By "distinguish" we mean: is there any predicate and an algorithm implementing this predicate, which holds true on one of the sequences and not on the other ?

For, we use encryption schemes, whose definitions and properties will be precised later. A message m, encrypted with k, using the algorithm a, will be denoted $\{m\}_k^a$. When a is not precised, it is assumed to be any symmetric encryption scheme. All encryption and decryption algorithms are assumed to be public, hence known to a potential attacker.

Example 1.1 *Assume for instance that the plaintext m is a confidential message written in English and the message sequence is $k, \{m\}_k$. Confidentiality can be stated as the indistinguishability of this sequence of two messages from $k, \{r\}_k$ where r is a random number.*

Of course, here, confidentiality is not achieved, as using the first message (k) to decrypt the second, we get m in the first case and r in the second. And m can be distinguished from r since English texts can be distinguished from random.

We have however to bound the computational power of an attacker. Otherwise, given simply $\{m\}_k$, he could distinguish it from $\{r\}_k$ by enumerating all possible keys and trying to decrypt the two messages. This is not realistic anyway, as enumerating all keys of lenght L requires 2^L computation steps. There are several possibilities for bounding the attacker power. Most of the time, it is assumed that the attacker is a polynomial time algorithm, being able however to guess and check: it will be a randomized polynomial time Turing machine, as it will be defined in section 2.1.1. The "polynomial" here is a function of a *security parameter*, in general the length of the keys. For instance considering as input the message sequences themselves, an attacker could distinguish two sequences if these sequences are long enough. But, again, this would require too much computational power. So, in general, the attacker will be a randomized Turing machine, which works in worse-case polynomial time.

There are other possibilities, such as defining an attacker as a non-uniform family of polynomial-size circuits, which would give him a much stronger power (as $\mathbf{BPP} \subset \mathbf{P}/Poly$). A seemingly more realistic model is to require a polynomial bound on *the average computation time* instead of the worse case. It is out of the scope of these notes to discuss these alternative models, which have all many drawbacks w.r.t. the standard choice of a randomized worse-case polynomial attacker.

Coming back to the simple security question, we are faced now with the, slightly more formalised, following question: given two message sequences, and a security parameter η, is there a polynomial P and a randomized Turing machine whose time complexity is bound by $P(\eta)$ such that the two message sequences can be distinguished with a significant probability ? Again "significant" will be made precise later. We cannot ask for non-zero probability since the attacker could simply guess a key, and, if it is the correct choice, will distinguish the sequences. This is however very unlikely to happen. Hence, we do not exclude this possibility, but we will say that the attacker is successful only if he can distinguish the sequences with probability larger than the inverse of a polynomial in η.

Now, assuming that all these notions are made precise, we want to decide if a given message sequence leaks some significant information. How do we proceed ? We cannot enumerate all possible attackers and simulate them on the input. (Since the attackers are randomized, we would even have to simulate all their possible behaviors !)

This decision problem has been considered in many research papers, starting maybe with [22]. The idea is to consider another model, more abstract than the above one. Messages are no-longer bitstrings but formal expressions and the attacker is no longer a randomized algorithm, but a proof system. For instance, in the model of [22], the author show that even a more advanced security issue can be decided in polynomial time. (It is more advanced since the attacker is no-longer passive, but can also interact with the network replacing messages by messages of his own and rely on multiple sessions of a given protocol).

Following this line of research, many automated provers were developed, aiming at finding attacks on security protocols (or prove that there is no attack!) There are so many such provers, that it is impossible to list all of them. Let us only mention the NRL prover [30], historically the first one and FDR, which allowed G. Lowe to find a simple attack on the Needham-Schroeder protocol [29].

While such tools were very satisfactory when they found attacks, the conclusion was less clear when they did not find attacks. One of the reasons why the security proofs in such models are questionable is the abstraction level of the models. Typically, until 2003, all automated tools rely on a "perfect cryptography assumption", which allows to consider messages as members of a free term algebra.

Example 1.2 *With perfect cryptography, as long as k is an unknown key, for any m, m', $\{m\}_k$ and $\{m'\}_k$ are indistinguishable in the abstract world. This is not however the case in practice if the length of messages m, m' are distinct.*

Other typical examples come from the use of arithmetic or Boolean functions, which satisfy some algebraic identities. This is the case for exclusive or, modular exponentiation,... Such algebraic identities (as well as the length of messages in the previous example) can be expressed at the term algebra level, thus refining the abstract model. We are however faced with the main question: *is our abstraction sound ?* In other words, do we loose some attacks when abstracting both the messages and the attacker ?

This is the main question, which is addressed in these notes. It was first considered by M. Abadi and P. Rogaway in [4] in a particular setting. In this paper, the authors show that, whenever the encryption scheme satisfies a certain level of security, called *type-0* in [4], then indistinguishability of messages sequences is implied by a notion of indistinguishability defined at the term level and which can be easily decided. Their result was further extended to other security primitives than symmetric encryption with atomic keys, for instance in [27, 5, 13], to more powerful attackers, for instance in [10, 31] and with weaker assumptions on encryption schemes, for instance in [12, 32].

The stake in these results is to get precise security assumptions on the cryptographic primitives, under which the abstractions are sound, therefore allowing to reason at an abstract level without missing attacks. Actually, not only soundness is important, but also completeness. Otherwise, a trivial abstract model, in which the intruder is omnipotent, is sound but does not help since security goals are never achieved in this model.

In a first part, we consider soundness of symbolic equivalence for a passive eavesdropper. This will include the following chapters:

Definitions . We will start from scratch, providing with all definitions. Only a minimal knowledge in discrete probabilities and on computability/complexity is required.

This will include the definition of PPT (randomized polynomial time Turing machines), which will represent the adversary, the definition of indistinguishability and some notions of security for encryption schemes. At this stage, we only consider symmetric encryption for simplicity and will later consider public-key encryption, one-way functions, signatures, exclusive-or etc.

On the symbolic side, we will define a general notion of symbolic equivalence. We use here basic notions on term rewriting. The decision problem for symbolic equivalence is left out (might be considered in another chapter)

A soundness result We prove an analog of [4]. The main difference is that we will consider various security levels, whereas [4] only consider the strongest assumption. On the symbolic side, this implies a more complex notion of symbolic equivalence. A similar result is proved in [12].

Many points can be discussed here:

- What is the most suitable security level ?
- We assume the absence of so-called "key-cycle".Is it realistic ? How can we model it ? This issue has already been discussed in many papers, see for instance [5, 9] for two recent papers.
- Is typing or tagging relevant ?
- How much does this result depend on the attacker model ?

Completeness We show the converse of the previous theorem: the abstraction is not only sound but also faithful. (Another advantage over [4]). In general, the question is to capture algebraically all the identities, which hold (or rather may hold) in the computational interpretations. This is explained in [13] through the *computational algebras*.

Other security primitives We consider other security primitives as well as the corresponding algebras at the symbolic level. This includes for instance public key cryptography, password encryption, hash functions. The proofs are often similar to the previous ones, while there are some subtle differences and additional difficulties. This is especially true for public key encryption: some of the research papers published on this subject overlooked difficulties.

Algebraic properties Many security primitives, including exclusive or and modular exponentiation, satisfy algebraic properties. In that case, such properties have to be considered in the symbolic equivalence. For instance $x \oplus y$ is indistinguishable from $y \oplus x$, hence we have to capture all such identities at the symbolic level.

In a second part, we will consider

Active attackers: So far, we only considered the simple problem of indistinguishability of message sequences, while the true question is the security of protocols in a hostile environment. There are several possibilities, both on the computational and on the symbolic sides.

The most satisfactory notion is probably *observational equivalence* [3, 2] on the symbolic side, though it is difficult to automate. The corresponding(?) satisfactory notion on the computational side is *universal composability* [17]. In both cases, the protocol is secure if, embedded in an arbitrary environment, it does not leak any confidential information.

A trend of research, following M. Backes and B. Pfitzmann, aims at achieving universal composability in a stepwise manner (simulatability). This yielded soundness results in presence of an active attacker [10, 7]. However, their soundness result is not a lifting to the active case of the soundness results described in the first part of these notes. This subtle changes, as well as the additional hypotheses required in this framework will be presented and discussed.

A weaker notion of security relies on trace semantics: a protocol is secure if, when embedded in a hostile environment, all (resp. almost all) its execution traces do not leak any confidential information. In [32], the authors show a soundness result, assuming a IND-CCA2 secure encryption scheme, by lifting computational traces to symbolic traces. We will also review this result and other results, which extend this "trace mapping property" such as [19].

The content (especially in the second part) may evolve, depending on the feedback on the first lectures.

2 Basic definitions

We discuss and give the definitions of the computational models in section 2.1, consider the question of security of encryption schemes in section 2.2 and define the symbolic model in section 2.4.

2.1 Computational models

$\mathbf{Pol}_k$ is the set of non-zero polynomials with positive integer coefficients and k variables.

Computational adversaries will have a limited computational power. Though, classically, their computation time is bounded by a polynomial (in a security parameter), this model is sometimes disputed; alternative models considered in various situations include (non uniform) polynomial size circuits families, expected polynomial time, subexponential time. We will try to abstract out the computational power of the attacker, setting the security assumptions accordingly.

Nevertheless, the most common model for an attacker is called "PPT" and corresponds to randomized Turing machines, running in (strict) polynomial time. We first recall some basic definitions concerning randomized Turing machines and PPT in section 2.1.1. Next, we precise the model we use for oracle randomized Turing machines.

2.1.1 Randomized Turing machines and Probabilistic Polynomial time Turing machines

In order to make precise statements, we will always state, when considering probabilities, what is the sample space and probability distribution:

$$\mathbf{Pr}\{x_1 \overset{R}{\longleftarrow} D_1, \ldots, x_n \overset{R}{\longleftarrow} D_n : P(x_1, \ldots, x_n)\}$$

is the probability of the predicate $P(x_1, \ldots, x_n)$ to hold true when $x_1, \ldots, x_n$ are drawn according to distributions $D_1, \ldots D_n$ respectively. By default, the sample space for each x_i is $\{0,1\}^*$, but the support of D_i will always be finite. In such a definition, D_i may depend on $x_1, \ldots, x_{i-1}$, but not on $x_j, j \geq i$. We will also write U_n the uniform distribution of the random input strings of length n or simply U when the length is irrelevant (i.e. any sufficiently large n will yield the same result).

A randomized Turing machine is a 4 tapes Turing Machine: one tape is the *input tape*, one tape is the *output tape*, one tape is the *working tape* and the last one is the *random tape*. Input and random tapes are read-only. Output tape is write-only.

The initial configuration of the machine is given by the input string x on the input tape and a random string $r \in \{0,1\}^\omega$ on the random tape. It is convenient to consider r as an infinite string, though only a prefix depending on the computation time of the machine is relevant; when we will set bounds on the computation time, the random input will be bound in length accordingly.

Moves of the Turing Machine are defined as usual. Complexity notions are defined in terms of number of moves, or space used on the working space and are considered as functions of the length of the input tape only. For instance a machine running in time $f(n)$ will read at most (half of) n bits on the random tape. A *PPT machine* (*Probabilistic Polynomial time Turing machine*) $\mathcal{A}$ is a randomized Turing machine working in (worst case) polynomial time: there is a polynomial $P_{\mathcal{A}} \in \mathbf{Pol}_1$ such that, for all inputs of size at most n and all possible contents of the random tape, $\mathcal{A}$ halts after at most $P_{\mathcal{A}}(n)$ steps.

The machines that will be considered may always be viewed as deterministic, possibly with a random input tape. To distinguish arguments on the input tape and random inputs, we separate them with a $|$, as in $\mathcal{A}(x_1, x_2 \mid r)$.

When $\mathcal{A}$ is a PPT, this is one of the classical definitions of the **BPP** class.

The use of an infinite string on the input tape, is only a convenient way for not specifying its input length. Of course, in practice, this string will be computed lazzyly, tossing a coin each time a new bit of the random string is accessed. Moreover, for every $r \in \{0,1\}^\omega$ and for every $r_0 \in \{0,1\}^k$,

$$\mathbf{Pr}\{r \xleftarrow{R} U : \exists r_1 . r = r_0 \cdot r_1\} = \frac{1}{2^k}$$

hence when only a prefix of length k of r is accessed by the machine, we can replace the uniform distribution on infinite strings with a uniform distribution on strings of length k. For instance, if $S_k(x)$ is the subset of $\{0,1\}^k$ of strings r_0 such that, on input x, with the random tape $r = r_0 \cdot r_1$, the machine halts with success after at most k steps. M accepts x with probability p if $\lim_{k\to\infty} \frac{|S_k(x)|}{2^k} = p$. (Note that the limit exists, as it is an increasing function of k and is bounded by 1).

For PPTs, the relevant prefix of r (hence r itself) can be a priori bounded by a polynomial in the length of the input string, hence can be considered as a finite (bounded length) string.

If $\mathcal{A}$ is a randomized machine, which always halts, and D is a distribution of a random variable (ranging over a finite subset of $\{0,1\}^*$), we define the image $\mathcal{A}(D)$ of D by $\mathcal{A}$, as the distribution on strings defined by: for every $a \in \{0,1\}^*$,

$$\mathbf{Pr}\{x \xleftarrow{R} \mathcal{A}(D) : x = a\} \stackrel{\text{def}}{=} \mathbf{Pr}\{y \xleftarrow{R} D, r \xleftarrow{R} U : \mathcal{A}(y \mid r) = a\}$$

When $\mathcal{A}$ is a PPT, $\mathcal{A}(D)$ is said *polynomially computable* from D. More generally, an ensemble $\{D'_n\}_{n\in\mathbb{N}}$, that is a family of distributions, one for each string length n is *polynomially computable* from $\{D_n\}_{n\in\mathbb{N}}$ if there is a PPT $\mathcal{A}$ with two inputs such that, for each n, $D'_n = \mathcal{A}(0^n, D_n)$. We also say simply that the ensemble $\{D_n\}_{n\in\mathbb{N}}$ is polynomially computable if it can be polynomially computed from the uniform distribution (or, which is equivalent, from a pointwise distribution). This way we may (an will sometimes) confuse a randomized machine $\mathcal{A}$ and the ensemble $\{\mathcal{A}_\eta\}_{\eta\in\mathbb{N}}$ it defines:

$$\forall Q \subseteq \{0,1\}^*, \quad \mathbf{Pr}\{x \xleftarrow{R} \mathcal{A}_\eta : Q(x)\} \stackrel{\text{def}}{=} \mathbf{Pr}\{r \xleftarrow{R} U : Q(\mathcal{A}(0^\eta \mid r))\}$$

This is also generalized to machines with several input/output strings.

2.1.2 Oracle machines

Usually, oracle machines are just Turing machines with an additional instruction, which is a call to an oracle. The machine is equipped with two additional tapes (for each oracle): one input tape (write only) and one output tape (read only). In a single move of the machine, the configuration changes by setting $\mathcal{O}(i)$ on the oracle output-tape, if i is the content of the oracle input tape. In case the oracle is randomized, there are actually 3 additional tapes for each oracle: the additional one is a random input to the oracle, which is now a mapping from

$\{0,1\}^* \times \{0,1\}^\omega$ to $\{0,1\}^*$. Each time the oracle is called, a new random input string is drawn on the oracle random tape[1] . Note however that the oracle random tape cannot be accessed by the machine itself.

Now, the time complexity of an oracle machine is measured in terms of number of steps and number of calls to the oracle. That is the total number of instructions executed on the input x by the oracle machine.

We distinguish particular classes of oracle machines, for which the number and ordering of the calls is fixed.

We will write $\mathcal{A}^{\mathcal{O}_1(\cdot|\cdot),\ldots,\mathcal{O}_n(\cdot|\cdot)}$ for machines which are allowed exactly one call to each randomized oracle $\mathcal{O}_1, \ldots \mathcal{O}_n$, in this order. Formally, $\mathcal{A}$ is a randomized Turing machine, equipped with one special instruction $\mathcal{O}$ (oracle call), the machine input tapes, $n+1$ random input tapes (one is read-only, the others are cannot be read to written), n oracle input tapes (write only), n oracle output tapes (read only) and finally a counter tape, initially set to 1. Each instruction $\mathcal{O}$ is interpreted as:

1. If the value i on the counter tape is strictly larger than n, then fail. (Move to a trash state)

2. Otherwise, move to a configuration in which the counter tape is incremented and the ith oracle output tape is set to $\mathcal{O}_i(x \mid r_i)$ where x is the current content of the ith oracle input tape and r_i is the ith random input tape.

Note that the machine is not required to complete all the oracle calls and may stop before (without failure) before the counter reaches n.

Also note that "exactly one call" or "at most one call" will yield equivalent definitions, when univerally quantifying over the machines, as a machine can always be completed with dummy calls to oracles.

These definitions are generalized in a straightforward way to oracles which have more than one input, in which case we write $\mathcal{O}_1(\cdot, \cdot \mid r_1)$ for instance.

For a fixed n, we define the probability of $\mathcal{M} = \mathcal{A}^{\mathcal{O}_1(\cdot|r_1),\ldots,\mathcal{O}_n(\cdot|r_n)}$ to accept x as follows:

$$p_{\mathcal{M}}(x) \stackrel{\text{def}}{=} \mathbf{Pr}\{r \stackrel{R}{\longleftarrow} U, r_1 \stackrel{R}{\longleftarrow} U, \ldots r_n \stackrel{R}{\longleftarrow} U : \mathcal{A}^{\mathcal{O}_1(\cdot|r_1),\ldots,\mathcal{O}_n(\cdot|r_n)}(x \mid r) = 1\}$$

Now, if we do not restrict the number of possible accesses to the oracle $\mathcal{O}$ (let us write this $\mathcal{M} = \mathcal{A}^{\mathcal{O}^*(\cdot|\cdot)}$),

$$p_{\mathcal{M}}(x) \stackrel{\text{def}}{=} \lim_{n \to +\infty} \mathbf{Pr}\{r \stackrel{R}{\longleftarrow} U, r_1 \stackrel{R}{\longleftarrow} U, \ldots r_n \stackrel{R}{\longleftarrow} U : \mathcal{A}^{\mathcal{O}(\cdot|r_1),\ldots\mathcal{O}(\cdot|r_n)}(x \mid r) = 1\}$$

Note that this is well defined as the probability is non-decreasing with n and is bounded by 1.

Example 2.1 *A typical example will be an oracle, which encrypts with a secret key: the adversary $\mathcal{A}$ will be allowed to query the oracle, which is itself a PPT.*

[1] If we follow however the idea that all random inputs to the machine, including oracle calls, are drawn beforehand, which allows for much more clean and precise statements, we have to assume an unbounded number of additional tapes, each containing a random input.

2.2 Security of Encryption schemes

2.2.1 Encryption schemes

I rely on [24, 25] for instance, however not requiring specifically that the key-generation algorithm runs in polynomial time. The encryption and decryption algorithms are still required to be PPT computable: it seems more reasonable from a user point of view. Also, some results, or even definitions, in what follows would not work properly otherwise. We will need to consider machines with oracle calls to encryption/decryption. For such oracle machines in the complexity class C we need to remain in C when querying the oracles. In other words, C should be closed by oracle calls to encryption/decryption, which can be ensured in all models, only when encryption/decryption are PPT computable.

Definition 2.1 *An encryption scheme is a triple $(G, \mathcal{E}, \mathcal{D})$ where $\mathcal{E}, \mathcal{D}$, respectively the encryption algorithm and the decryption algorithm are PPT and G, the key generation algorithm is a randomized Turing machine. G, on input 0^n, outputs a pair (k, k') of bitstrings. $\mathcal{E}$ and $\mathcal{D}$ must satisfy, for all $x \in \{0,1\}^*$, for all $r, r' \stackrel{R}{\longleftarrow} U$*

$$\mathcal{D}(\mathcal{E}(x, k \mid r), k' \mid r') = x$$

Exercise 1 (2)
Show that we get an equivalent definition if we replace the above identity by

$$\mathbf{Pr}\{r, r' \stackrel{R}{\longleftarrow} U : \mathcal{D}(\mathcal{E}(x, k \mid r), k' \mid r') = x\} = 1$$

Exercise 2 (2)
Show that the above definition can be simplified, requiring $\mathcal{D}$ to be deterministic (and not randomized): if there is a PPT implementing $\mathcal{D}$, then there is also a deterministic polynomial time algorithm implementing $\mathcal{D}$.

Let us recall that, according to the defintion of section 2.1, G can also be viewed as an ensemble, and G_η is a distribution defined by the machine G on input 0^η:

$$\mathbf{Pr}\{k \stackrel{R}{\longleftarrow} G_\eta : Q(k)\} \stackrel{\mathrm{def}}{=} \mathbf{Pr}\{r \stackrel{R}{\longleftarrow} U : Q(G(0^\eta \mid r))\}$$

This is not assumed to be a uniform distribution, nor a polynomial distribution.
Notes:

- some relevant variants (considered in the litterature) could be[2]:

$$\begin{aligned}&\forall P \in \mathbf{Pol}_1, \exists N \in \mathbb{N}, \forall x, \forall \eta > N,\\ &\qquad \mathbf{Pr}\{r, r' \stackrel{R}{\longleftarrow} U, (k, k') \stackrel{R}{\longleftarrow} G_\eta : \mathcal{D}(\mathcal{E}(x, k \mid r), k' \mid r') = x\} > 1 - \frac{1}{P(\eta)}\end{aligned}$$

  or

$$\begin{aligned}&\forall P, Q \in \mathbf{Pol}_1, \exists N \in \mathbb{N}, \forall \eta > N, \forall (k, k') \stackrel{R}{\longleftarrow} G_\eta, \forall s \leq Q(\eta).\\ &\qquad \mathbf{Pr}\{r, r' \stackrel{R}{\longleftarrow} U, x \stackrel{R}{\longleftarrow} \{0,1\}^s : \mathcal{D}(\mathcal{E}(x, k \mid r), k' \mid r') = x\} > 1 - \frac{1}{P(\eta)}\end{aligned}$$

[2] I display explicitetly all the quantifiers as well as their ordering, which makes an unusually long formula.

or some mixture of the two.

It seems that moving to such definitions will not change the results. (But this hasn't been checked in detail).

- G can be identified to a key distribution: it is not assumed that the key distribution is uniform, but the security of the encryption scheme will depend on this distribution.
- In [4], G (called $\mathcal{K}$) is a mapping from $\mathbb{N} \times \mathsf{Coins}$ to Keys, which is the same as above since 0^n can be identified with the security parameter in $\mathbb{N}$ and only symmetric keys are considered in [4], for which $k = k'$. The only difference in [4] is that they assumed that the messages are typed. This implies for instance that an intruder can check whether a message is a valid ciphertext, which is not the case of every encryption scheme (consider e.g. AES).

Until section 5, we will only focus on private encryption schemes:

Definition 2.2 *An encryption scheme is* private *or* symmetric *if, for every η, and every r, $G(0^\eta \mid r) = (k, k)$, i.e. decryption key and encryption key are identical.*

2.2.2 Chosen plaintext attacks

Security definitions for encryption are split (e.g in [24]) in so-called *semantic security* and *indistinguishability* on one hand, public-key systems and private keys systems on the other hand Moreover, in each case, several possible levels of security may be considered. In [4], security levels (all related to indistinguishability, PPT adversaries and private keys encryption) are organized in a lattice and all these definitions are stronger than the ones presented in this section. The main result of [4] only concerns a particular security definition. Most of the following soundness results rely on a similar definition, possibly with stronger assumptions in the active case (see section 8 for [4]'s definitions and extensions).

We consider here only indistinguishability (and not semantic security), as, anyway, the two notions are proved to be equivalent in many settings and it seems generally accepted that indistinguishability is the right notion. For similar reasons, we only consider multi-messages indistinguishability.

In the computational world, active attackers are given access to oracles which return the encryption of a a choosen plaintext (*Chosen Plaintext Attacks*, CPA) or they are given access to oracles which return the plaintext of a choosen ciphertext (*Chosen Ciphertext Attacks*, CCA, see section 6.0.1). Within these two categories of security definitions, there are again several variants, which we will try to describe.

In the most classical setting ([16, 24, 25]) the adversary is allowed to query a fixed number of times q an oracle machine which encrypts a message under a given key:

Definition 2.3 *Let $\mathcal{A}$ be a randomized Turing machine with one oracle. The advantage of the adversary $\mathcal{A}$ (under CPA) for a private encryption scheme $(G, \mathcal{E}, \mathcal{D})$ is defined with two parameters η, q:*

$$\begin{aligned}
&\mathsf{ADV}^{IND-CPA}_{\mathcal{A}}(q,\eta) \stackrel{def}{=} \\
&\quad |\boldsymbol{Pr}\{r \stackrel{R}{\longleftarrow} U, r_1, \ldots, r_q \stackrel{R}{\longleftarrow} U, k \stackrel{R}{\longleftarrow} G_\eta : \mathcal{A}^{\mathcal{O}_1, \ldots \mathcal{O}_q}(0^\eta \mid r) = 1\} \\
&\qquad - \boldsymbol{Pr}\{r' \stackrel{R}{\longleftarrow} U, r'_1 \stackrel{R}{\longleftarrow} U, ..., r'_q \stackrel{R}{\longleftarrow} U, k' \stackrel{R}{\longleftarrow} G_\eta : \mathcal{A}^{\mathcal{O}'_1, \ldots, \mathcal{O}'_q}(0^\eta \mid r') = 1\}|
\end{aligned}$$

where $\mathcal{O}_i(x, y) = \mathcal{E}(x, k \mid r_i)$ *if* $|x| = |y|$ *(and* 0 *otherwise) and* $\mathcal{O}'_i(x, y) = \mathcal{E}(y, k' \mid r'_i)$ *if* $|x| = |y|$ *(and* 0 *otherwise)*

In words, the adversary is faced to one of two experiments. In the first experiment, the challenger replies to oracle queries by always encrypting the first component of the submitted pair with a key k. In the second experiment, the challenger replies to oracle queries by always encrypting the second component of the submitted pair with a key k'. The advantage of the adversary measures how much he can guess on which of the two experiments he was faced with.

Notes:

- There are other definitions, which consider only the case $q = 1$ or that are equivalent. The above definition is stronger: it gives the ability to the attacker to *adapt* his request to the oracles, depending on the previous answers. This kind of security notions is often called *adaptative- CPA*, while in ordinary CPA, all requests to the oracle are performed at the same time.

- There are several ambiguities in the corresponding definition of [24]. For instance, in the above definition, the choice of k is bound by the chosen sequence: there are two (probably distinct) keys, one for each sequence, while each element in the sequence is encrypted with the same key. In [24], there is no explicit probability space and keys k are named $G(0^n)$. However, since G is a randomized algorithm, each call to G produces a different output. For instance $(\mathcal{E}(x_1, G(0^\eta)), \mathcal{E}(x_2, G(0^\eta)))$ is distinct from $\mathsf{let}\ k = G(0^\eta)\ \mathsf{in}\ (\mathcal{E}(x_1, k), \mathcal{E}(x_2, k))$. In the first case the two keys are distinct, while they are identical in the second case. Two alternative definitions could then be:

  1. each time $\mathcal{E}$ is called with argument $G(0^n)$, then the encryption is with a fresh key, in which case every element of the sequences has to be encrypted with an independent new key
  2. G is called once at the beginning, in which case all keys for both sequences are identical.

  The latter choice seems to yield a reasonable definition of security, but then we cannot choose to include the key in the sample space and we would have:

$$\mathsf{ADV}^e_{\mathcal{A}}(q, \eta) \stackrel{\text{def}}{=}$$
$$\mathbf{Average}\{k \stackrel{R}{\longleftarrow} G_\eta : \begin{array}{l} |\mathbf{Pr}\{r \stackrel{R}{\longleftarrow} U, r_1 \stackrel{R}{\longleftarrow} U, \ldots, r_q \stackrel{R}{\longleftarrow} U : \mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_q}(0^\eta \mid r) = 1\} \\ \quad -\mathbf{Pr}\{r \stackrel{R}{\longleftarrow} U, r_1 \stackrel{R}{\longleftarrow} U, \ldots, r_q \stackrel{R}{\longleftarrow} U : \mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_q}(0^\eta \mid r) = 1\}| \end{array} \}$$

  Where $\mathcal{O}_i(x) = \mathcal{E}(x, k \mid r_i)$.

- In the above definition, $0^\eta, \overline{x}, \overline{y}$ are useless if the circuit attacker is considered.

Exercise 3

Assume that we modify definition 2.3 and define the oracle as follows: $\mathcal{O}_i(x, 0^{|x|}) = \mathcal{E}(x, k \mid r_i)$ and 0 for other inputs and $\mathcal{O}'_i(x, 0^{|x|}) = \mathcal{E}(0^{|x|}, k' \mid r'_i)$ and 0 for other inputs. We get a new definition of advantage. Show that this new advantage is bounded by twice the advantage, as defined in definition 2.3.

Exercise 4 (2)
Assume that we use the following new definition of advantage:

$$\begin{aligned}&\mathsf{ADV}_{\mathcal{A}}^{IND-CPA}(q,\eta) = |-1+2\times\\ &\quad \mathbf{Pr}\{r \xleftarrow{R} U, \overline{r} \xleftarrow{R} U, k \xleftarrow{R} G_\eta, b \xleftarrow{R} U : \mathcal{A}^{\mathcal{O}_1(\cdot,\cdot,b|r_1),\ldots,\mathcal{O}_q(\cdot,\cdot,b|r_q)}(0^\eta \mid r) = 1-b\}|\end{aligned}$$

where $\mathcal{O}_i(x,y,0) = \mathcal{E}(x,k,\mid r_i)$ if $|x|=|y|$ and $\mathcal{O}_i(x,y,1) = \mathcal{E}(y,k\mid r_i)$ if $|x|=|y|$.

Show that this modifies definition 2.3 by a constant factor only.

In what follows, we will consider a class $\mathcal{C}$ of attackers. Such a class is usually defined using complexity bounds, e.g. $\mathcal{C}$ is the class of PPTs. We assume that $\mathcal{C}$ is closed under composition with PPTs, which implies (when $\mathcal{C}$ is not empty) that $\mathcal{C}$ contains all PPTs. We will also consider extensions to oracle machines: $\mathcal{C}^i$ is the class of machines in $\mathcal{C}$, which are equipped in addition with i oracles. In any case, we assume that the oracles are ranging over machine classes such that $\mathcal{C}^i \subseteq \mathcal{C}$. In other words, replacing the oracles with their definitions, we stay within the class $\mathcal{C}$. Typical examples are the class $\mathcal{C}$ of PPTs and the oracle class of PPTs: since PPTs are closed under composition, $\mathcal{C}^i \subseteq \mathcal{C}$ in this case.

The security under chosen plaintext attacks follows:

Definition 2.4 *A symmetric encryption scheme is IND-CPA w.r.t. $\mathcal{C}$ if*

$$\forall P \in \boldsymbol{Pol}_1,\ \forall q \in \mathbb{N},\ \forall \mathcal{A} \in \mathcal{C}^1,\ \exists N \in \mathbb{N},\ \forall \eta > N,\ \ \mathit{ADV}_{\mathcal{A}}^{IND-CPA}(q,\eta) < \frac{1}{P(\eta)}$$

Note here that $\mathcal{A}$ is any machine (in $\mathcal{C}^1$) that makes exactly q calls to the oracle. The number of calls cannot depend on the input of the machine; q does not depend on η. On the contrary, N may depend on q. In other words, we choose the security parameter depending on the number of possible oracle calls.

Exercise 5 (4)
Show that, if we switch the quantifiers on $\mathcal{A}$ and N in the above definition, there is no longer any secure encryption scheme.

Exercise 6 (2)
Assume that the key generation algorithm G is a PPT. What happens if we pull k out of the sample space in definition 2.3 ?

Open Question 1 *Assume that $\mathcal{C}$ is the class of PPTs. If we replace $\mathit{ADV}^{IND-CPA}$ with ADV^e in definition 2.4, do we get an equivalent definition ? In other words, does there exist an IND-CPA secure encryption scheme, which would not be secure with the other definition of advantage ?*

It is not clear whether anybody ever considered this question.

A stronger notion of security w.r.t. chosen plaintext attacks lets q depend polynomially on η: this corresponds to "type 3" security in [4]:

Definition 2.5 *A symmetric encryption scheme $(G,\mathcal{E},\mathcal{D})$ is type-3 secure (w.r.t. $\mathcal{C}$) if*

$$\forall P \in \boldsymbol{Pol}_1, \forall \mathcal{A} \in \mathcal{C}^1, \exists N, \forall \eta > N, \forall q \in \mathbb{N}, \mathit{ADV}_{\mathcal{A}}^{IND-CPA}(q,\eta) < \frac{1}{P(\eta)}$$

W.r.t. IND-CPA security, quantifiers are inversed: instead of choosing the security parameter after the number of oracle calls, we choose the number of oracle calls after fixing η. It is not clear whether the definitions are equivalent and if there is any known encryption scheme, which is IND-CPA secure and not type-3 secure, even when $\mathcal{C}$ is the class of PPT.

Open Question 2 *Assume $\mathcal{C}$ is the class of PPT. Is there a symmetric encryption scheme, which is IND-CPA secure and not type 3 secure ?*

It is not clear whether anybody ever considered this question. The answer might be straightforward.

Exercise 7
Show that, if $\mathcal{A}$ is allowed to run in time $O(2^\eta)$, then there is no type-3 secure encryption scheme.

Open Question 3 *Assume that $\mathcal{C}$ is the class of machines running in worst case complexity $O(2^{\frac{\eta}{2}})$. Are there type-3 secure encryption schemes w.r.t. $\mathcal{C}$?*

Again, it might be straightforward, but it is beyond the knowledge of the writer of these notes.

Now we can use the following definition:

Definition 2.6 *If f is any mapping from $\mathbb{N}$ to $\mathbb{Q}$. We say that $f(\eta)$* is negligible *if*

$$NEG(f(\eta)) \stackrel{def}{=} \forall P \in \boldsymbol{Pol}_1 \exists N \in \mathbb{N}, \forall \eta > N,\ f(\eta) < \frac{1}{P(\eta)}$$

We must however be cautious when using "negligible" as it hides some quantifiers dependencies.

Exercise 8
Show that $(G, \mathcal{E}, \mathcal{D})$ is type-3 secure iff for all PPT $\mathcal{A}$, $\mathsf{ADV}^3_{\mathcal{A}}(\eta)$ is negligible.

We chose definition 2.5 in order to stress the relationship with IND-CPA. As in many papers following [4], we may simply write:

$$\begin{aligned}\mathsf{ADV}^3_{\mathcal{A}}(\eta) = &\\ &\mathbf{Pr}\{k \stackrel{R}{\longleftarrow} G_\eta, \overline{r} \stackrel{R}{\longleftarrow} U, r \stackrel{R}{\longleftarrow} U : \mathcal{A}^{\mathcal{O}^*_k}(0^\eta \mid r) = 1\}\\ &\quad -\mathbf{Pr}\{k \stackrel{R}{\longleftarrow} G_\eta, \overline{r} \stackrel{R}{\longleftarrow} U, r \stackrel{R}{\longleftarrow} U : \mathcal{A}^{\mathcal{O'}^*_k}(0^\eta \mid r) = 1\}\end{aligned}$$

where the oracles $\mathcal{O}_k$ and $\mathcal{O}'_k$ are defined as above and $\overline{r}$ is a sequence of random numbers of the appropriate length.

Even stronger security notions will include also hiding which key is used (this is discussed in [4]): given two ciphertexts, the adversary cannot guess whether or not they are encrypted using the same key. Now, we let $\mathcal{M}_{k,k'}$ be a machine which has unlimited access to two oracles:

Definition 2.7 *Let $\mathcal{A} \in \mathcal{C}^2$ be a machine with two randomized oracles $\mathcal{O}_1, \mathcal{O}_2$, whose access is unlimited. Let $\mathcal{A}_{k,k'}$ be the machine $\mathcal{A}$ in which the oracles are respectively $\mathcal{O}_1(x \mid r) = \mathcal{E}(x, k \mid r)$ and $\mathcal{O}_2(x \mid r') = \mathcal{E}(x, k' \mid r')$ and $\mathcal{B}_k$ be the machine $\mathcal{A}$ in which the oracles are $\mathcal{O}_1(x \mid r) = \mathcal{O}_2(x \mid r) = \mathcal{E}(0^{|x|}, k \mid r)$.*

The type-1 advantage of the adversary is defined as

$$\begin{array}{rcl} \mathsf{ADV}^1_{\mathcal{A}}(\eta) & \stackrel{def}{=} & \boldsymbol{Average}\{k, k' \stackrel{R}{\longleftarrow} G_\eta : p_{\mathcal{A}_{k,k'}}(\eta)\} - \boldsymbol{Average}\{k \stackrel{R}{\longleftarrow} G_\eta : p_{\mathcal{B}_k}(\eta)\} \\ & = & \begin{array}{c} \boldsymbol{Pr}\{k, k' \stackrel{R}{\longleftarrow} G_\eta, \overline{r} \stackrel{R}{\longleftarrow} U : \mathcal{A}_{k,k'}(0^\eta \mid \overline{r}) = 1\} \\ -\boldsymbol{Pr}\{k \stackrel{R}{\longleftarrow} G_\eta, \overline{r} \stackrel{R}{\longleftarrow} U : \mathcal{B}_k(0^\eta \mid \overline{r}) = 1\} \end{array} \end{array}$$

The encryption scheme is type-1 secure *(w.r.t. $\mathcal{C}$) if*

$$\forall \mathcal{A} \in \mathcal{C}^2,\ |\mathsf{ADV}^1_{\mathcal{A}}(\eta)| \text{ is negligible}$$

Type-0 security and type-2 security are obtained by replacing $0^{|x|}$ with 0 in the oracles defining type 1 and type-3 security respectively; type-0 (resp. type 2) strengthen type-1 (resp. type 3) security by additionally concealing the length of messages. In [4], soundness results are proved for type-0 secure encryption schems.

2.3 Computational indistinguishability of ensembles

Two sequences of distributions (called ensembles) indexed by $\eta \in \mathbb{N}$ are indistinguishable w.r.t a class $\mathcal{C}$ of adversaries if any machine in $\mathcal{C}$ cannot tell with significant probability whether a data is drawn from the first distribution or from the second.

Definition 2.8 *Let $D = \{D_\eta\}, D' = \{D'_\eta\}$ be two ensembles. We say that they are* computationally indistinguishable *with respect to $\mathcal{C}$, which is written $D \approx_{\mathcal{C}} D'$ if, for every $\mathcal{A} \in \mathcal{C}$, the advantage*

$$\epsilon_{\mathcal{A}}(\eta) = \boldsymbol{Pr}\{x \overset{R}{\longleftarrow} D_\eta, r \overset{R}{\longleftarrow} U : \mathcal{A}(x, 0^\eta \mid r) = 1\} - \boldsymbol{Pr}\{x \overset{R}{\longleftarrow} D'_\eta, r \overset{R}{\longleftarrow} U : \mathcal{A}(x, 0^\eta \mid r) = 1\}$$

is negligible, that is:

$$\forall \mathcal{A}, \forall P \in \boldsymbol{Pol}_1, \exists N, \forall \eta > N, |\epsilon_{\mathcal{A}}(\eta)| < \frac{1}{P(\eta)}$$

Exercise 9 (1)
Show that the Dirac distributions, defined by $\mathbf{Pr}\{x \overset{R}{\longleftarrow} \delta_\eta : x = 0^\eta\} = 1$ and the uniform distributions U_η are two distinguishable ensembles with respect to PPT.

Exercise 10 (2)
Show that the ensemble defined by $\mathbf{Pr}\{x \overset{R}{\longleftarrow} X_\eta : x = a\} \overset{\text{def}}{=} \mathbf{Pr}\{x, y \overset{R}{\longleftarrow} U_\eta : x \oplus y = a\}$ is indistinguishable from the uniform distributions U_η. (For which classes $\mathcal{C}$?)

Exercise 11 (3)
Fix $k \in \mathbb{N}$ and consider the ensemble U^k_η defined by U_η if $k \geq \eta$ and

$$\mathbf{Pr}\{x \overset{R}{\longleftarrow} U^k_\eta : x = a\} = \begin{cases} 0 & \text{if } a = b0^{\eta-k} \text{ for some } b \\ \frac{1}{2^\eta - 2^k} & \text{otherwise} \end{cases}$$

Show that $\{U_\eta\}_{\eta\in\mathbb{N}} \approx_{PPT} \{U^k_\eta\}_\eta \in \mathbb{N}$. Is it sill the case if $k = \frac{\eta}{2}$? $k = \eta - 2$?

Exercise 12
Show that, if $D \approx_{\mathcal{C}} D'$ and $C \approx_{\mathcal{C}} C'$, then $D \times C \approx_{\mathcal{C}} D' \times C'$.

2.4 Symbolic messages

We first very briefly recall some background on terms and term rewriting, before switching to the important issue: symbolic equivalence.

The section on rewriting can be skipped an serves only as a reference for further definitions.

2.4.1 Rewriting systems

We do not recall here all the basics on terms, term rewriting systems, equational theories (this would require several pages) and refer to [21] for more details. We only focus on the minimal setting.

A *ranked alphabet* is a set of symbols, together with their arity (belonging to $\mathbb{N}$). A *term* over the ranked alphabet $\mathcal{F}$ is a finite tree labeled with $\mathcal{F}$ and such that, if a node is labed with a symbol of arity n, then it has exactly n sons. Terms are also displayed in the algebraic way, as in $f(f(a,b),a)$ for a tree with two internal nodes labeled f (arity 2) and leaves labeled a, b (arity 0). Paths in the terms are denoted by *positions*, which are sequences of positive integers. The set of all positions in a term t is written $\mathrm{Pos}(t)$. For instance, $\mathrm{Pos}(f(f(a,b),a)) = \{\epsilon, 1, 2, 11, 12\}$. The label of a term t at position $p \in \mathrm{Pos}(t)$ is written $t(p)$ (actually t can be seen as a mapping from the set of its positions to $\mathcal{F}$). If $p \in \mathrm{Pos}(t)$, $t|_p$ is the subterm at position p. For instance $f(f(a,b),a)|_1 = f(a,b)$. $t[u]_p$ is the term obtained by replacing $t|_p$ with u. For instance $f(f(a,b),a)[b]_1 = f(b,b)$.

$\mathcal{F}$-albegras are structures in which each function symbol of $\mathcal{F}$ is interpreted by an actual function with the appropriate number of arguments. $T(\mathcal{F})$, the albegra of terms itself, is a $\mathcal{F}$-algebra in which each function symbol is freely interpreted. For instance if $\mathcal{F}_0$ consists of encryption (arity 2), decryption (arity 2), pairing (arity 2), projections (arity 1) and constants, their computational interpretations defines an $\mathcal{F}$-algebra.

$T(\mathcal{F}, X)$ is the free $\mathcal{F}$-algebra generated by X. It is isomorphic to $T(\mathcal{F} \cup X)$, each symbol of X being considered as a new 0-ary symbol. *Substitutions* are endomorphisms of $T(\mathcal{F}, X)$, whose restriction to X has a finite support. They are written $\{x_1 \mapsto t_1; \ldots, x_n \mapsto t_n\}$. For instance, $t = f(f(x,b),x) \in T(\mathcal{F}, X)$ and if $\sigma = \{x \mapsto a\}$, the image of t by σ, written in postfix notation, is $t\sigma = f(f(a,b),a)$.

A *term rewriting system* (over $\mathcal{F}$) is a finite set of pairs of terms in $T(\mathcal{F}, X)$, $l_1 \to r_1, \ldots, l_n \to r_n$. For instance

$$D(\{x\}_k^n, k) \to x$$

is a one-rule rewriting system, on the alphabet $\mathcal{F}_0$ defined above. The *one step reduction relation* $\underset{\mathcal{R}}{\longrightarrow}$ associated with a rewriting system $\mathcal{R}$ is defined by $t \underset{\mathcal{R}}{\longrightarrow} u$ iff there is a rule $l \to r \in \mathcal{R}$, a position $p \in \mathrm{Pos}(t)$ and a substitution σ such that $t|_p = l\sigma$ and $t[r\sigma]_p = u$. Informally, this means that any instance of the left side of the rule can be replaced by the corresponding instance of the right side. For instance, using the above one-rule rewrite system,

$$D(D(\{\{a\}_k^n\}_{k'}^{n'}, k'), k) \underset{\mathcal{R}}{\longrightarrow} D(\{a\}_k^n, k) \underset{\mathcal{R}}{\longrightarrow} a$$

The *reduction relation* associated with $\mathcal{R}$ is the reflexive transitive closure of $\underset{\mathcal{R}}{\longrightarrow}$, written $\underset{\mathcal{R}}{\overset{*}{\longrightarrow}}$. The symmetric, relexive and transitive closure of $\mathcal{R}$ is written $\underset{\mathcal{R}}{\overset{*}{\longleftrightarrow}}$. When $\mathcal{R}$ is symmetric (($l, r \in \mathcal{R}$ iff $r, l \in \mathcal{R}$) then $\mathcal{R}$ is actually called an *equational theory* and the reduction relation is a congruence (w.r.t. the structure of $\mathcal{F}$-algebra) on $T(\mathcal{F}, X)$, written $=_{\mathcal{R}}$.

A term t is *irreducible* w.r.t. rewrite system $\mathcal{R}$ if $t \underset{\mathcal{R}}{\overset{*}{\longrightarrow}} u$ implies $t = u$. A term rewriting system is *terminating* if there is no infinite sequence $t_1 \underset{\mathcal{R}}{\longrightarrow} \ldots \underset{\mathcal{R}}{\longrightarrow} \ldots$. A term rewriting system $\mathcal{R}$ is *convergent*, if it is terminating and, for any term t there exists a unique term $t\downarrow$ (called *the normal form of* t) such that $t \underset{\mathcal{R}}{\overset{*}{\longrightarrow}} t\downarrow$ and $t\downarrow$ is irreducible. We do not define here convergence modulo an equational theory (see [21] instead). Informally, it means also termination and uniqueness of normal forms, up to the equational theory.

2.4.2 Symbolic equivalence

In this section we define a very general setting for symbolic equivalence. This implies sometimes more complicated definitions and one can simply look at the examples to get some simple instances.

In the symbolic world, messages are terms. They are built from names representing random numbers (a set $\mathcal{R}$), names representing keys (a set $\mathcal{N}$), pairing, symmetric encryption, constants, decryption, projection and possibly other primitives: one-way functions, public-key encryption, signatures etc... Such messages form a term algebra $T(\mathcal{F})$. These terms can be simplified according to a rewrite system $\mathcal{S}$, which contains for instance rules $D(\{x\}_k^r, k) \to x$ or $x \oplus x \to 0$ for exclusive or and there can be identities among messages, such as commutativity of exclusive or.

Formally, we assume that $\mathcal{S}$ is a convergent rewrite system (possibly w.r.t. associativity and commutativity of some function symbols) over the set of function symbol $\mathcal{F}$. $\mathcal{M}$ is the quotient algebra $T(\mathcal{F})/=_{\mathcal{S}}$. Since $\mathcal{S}$ is convergent, every term $t \in T(\mathcal{F})$ has a unique normal form $t \downarrow$ (possibly modulo associativity and commutativity) and $\mathcal{M}$ can be identified with the terms in normal form, which constitute a set of representatives.

Sorts, which are often considered in the litterature (in particular [4]) are not included in the definition of messages. We will introduce later predicates, which may (or not) include some sort information.

Example 2.2 *In case of symmetric encryption and pairing (the basic example), $\mathcal{F}_0$ consists in*

- *a set $\mathcal{W}$ of constants: the words over $\{0,1\}$ (bitstrings),*
- *an infinite set of constants $\mathcal{R}$ (the random numbers),*
- *an infinite set of constants $\mathcal{N}$ (for the keys)*
- *a binary symbol $< \cdot, \cdot >$*
- *a ternary symbol $\{\cdot\}^{\cdot}_{\cdot}$. The last argument, displayed as an exponent is the random number used in the encryption algorithm. We found clearer to display it explicitly, as two encryption of the same term with the same key should not yield the same ciphertext, while copying a ciphertext corresponds to the same keys and random numbers.*
- *a binary decryption symbol D*
- *two unary projection symbols π_1, π_2.*

In this case the system $\mathcal{S}_0$ consists of the following rules:

$$\begin{array}{rclcrcl} D(\{x\}_k^n, k) & \to & x & \qquad & \pi_1(< x, y >) & \to & x \\ \pi_2(< x, y >) & \to & y & & & & \end{array}$$

$\mathcal{S}_0$ is convergent. It must be noted that, for instance, $D(k, k')$ is irreducible if k, k' are two nonces. Similarly, $D(\{x\}_{k_1}^r, k_2)$ is irreducible if $k_1 \neq k_2$. Such "wrong use" of decryption is permitted here and does not yield anything but the term itself. Another option consists in adding explicit failure symbols.

Example 2.3 *Other examples of message algebras include the case of public key cryptosystems: add another encryption symbol, another decryption symbol, public keys and private keys and the additional decryption rule:*

$$D_p(\{\{x\}\}^n_{\mathsf{pub}(y)}, \mathsf{priv}(y)) \to x$$

If we add the exclusive or for instance,then $\mathcal{S}$ will contain $x \oplus x \oplus y \to 0, x \oplus 0 \to 0$ and $\oplus$ is associative and commutative.

Example 2.4 *Going further on, there are more complex equational theories and sets of function symbols, here is one taken from [26]: $\mathcal{F} = \{\mathsf{commit}, \mathsf{sign}, \mathsf{pub}(\cdot), \mathsf{blind}\} \cup \mathcal{N} \cup \{0,1\}^*$, open, $\mathsf{checksign}$, $\mathsf{unblind}$ } and $\mathcal{S}$ consists of the rules*

$$\begin{array}{rclcrcl} \mathsf{open}(\mathsf{commit}(x,y),y) & \to & x & \quad & \mathsf{checksign}(\mathsf{sign}(x,\mathsf{priv}(y)),\mathsf{pub}(y)) & \to & x \\ \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) & \to & \mathsf{sign}(x,z) & & & & \end{array}$$

Again, the rewrite system $\mathcal{S}$ is convergent.

Intruder computations consist in building new messages using the public part of $\mathcal{F}$. There are two ways of formalizing this, which are equivalent: either use inference rules or contexts and rewriting. We will mainly use the latter, following the majority of papers in the litterature. First, assume that $\mathcal{F}_P$ is the subset of $\mathcal{F}$ of *public functions*. Typically, the intruder knows the encryption algorithm, pairing, constants, decryption, projections some random numbers and the private keys of compromised agents, as well as a subset of nonces. Nonces generated by honest agents, and $\mathsf{priv}()$ are unknown to the intruder. In the following, we will also assume that the intruder knows at least one key, which corresponds to knowing the length of the keys: $\mathcal{F}_P \cap \mathcal{N} \neq \emptyset$.

A *context with k place holders* is a term in $T(\mathcal{F}_P, \{x_1, \ldots, x_k\})$. The set of such contexts is written $\mathcal{C}_k$. If $C \in \mathcal{C}_k$ and $t_1, \ldots t_k \in \mathcal{M}$, $C[t_1, \ldots, t_k]$ is the term obtained by substituting each x_i by t_i in C. $\mathcal{C} \stackrel{\text{def}}{=} \bigcup_{i=0}^{+\infty} \mathcal{C}_k$.

We need finally views on the terms (for instance, their length, equality of terms,...); we consider a finite set of predicate symbols $\mathcal{P}$ interpreted as a set of relations $\mathcal{P}^I$ on irreducible terms (we sometimes confuse here the predicate symbol and its interpretation). For every $P \in \mathcal{P}$, we write P^I its interpretation, which is viewed both as a relation and as a Boolean function. For decision issues, more properties of the predicate interpretations will be required (see section 7).

Definition 2.9 *Given $\mathcal{F}, \mathcal{F}_P, \mathcal{S}, \mathcal{P}, \mathcal{P}^I$ as above, two sequences of terms $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_n)$ of $\mathcal{M}^n$ are* symbolically equivalent*, which we write $(s_1, \ldots, s_n) \sim (t_1, \ldots, t_n)$, if*

$$\begin{array}{l} \forall P \in \mathcal{P} \text{ of arity } k, \forall C_1, \ldots, C_k \in \mathcal{C}, \\ \quad P^I(C_1[s_1, \ldots, s_n]\downarrow, \ldots, C_k[s_1, \ldots, s_n]\downarrow) \Leftrightarrow P^I(C_1[t_1, \ldots, t_n]\downarrow, \ldots, C_k[t_1, \ldots, t_n]\downarrow) \end{array}$$

This corresponds to the intuitive idea that, after any computation on s or t, and applying any view on the terms, it is impossible for the intruder to distinguish the two sequences of terms. This generalizes both pattern equivalence as defined in [4] and static equivalence as defined in [13], and gives more flexibility on the intruder capabilities that can be modeled.

2.4.3 Examples of useful predicates

Valid messages Let M be a unary predicate symbol and M^I be the subset of $T(\mathcal{F})$ of "valid messages". In case of the symmetric encryption example, M^I is the subset of $T(\mathcal{F} \setminus \{D, \pi_1, \pi_2\})$ such that any encryption symbol is applied to a term in M^I, a constant in $\mathcal{N}$ and a constant in $\mathcal{R}$.

We will include M or not in the set of predicates, depending on whether it is possible for an intruder to detect the success/failure of a decryption attempt.

Equality: EQ^I is a binary relation on terms. In the strict interpretation (which we will follow unless stated otherwise),

$$EQ^I(s,t) \stackrel{\text{def}}{=} M(s) \wedge M(t) \wedge s = t$$

This corresponds to *static equivalence* of [13].

There are looser interpretations, for instance only requiring $s = t$ or even equating all "failures" by adding $\perp$ symbols...

Equal Keys : This relation models the ability to decide whether two ciphertexts are obtained with the same encryption key.

$$EK^I(x,y) \stackrel{\text{def}}{=} \exists t, u, k, r, r'. EQ^I(x, \{t\}_k^r) \wedge EQ^I(y, \{u\}_k^{r'})$$

which implies in particular $M(k), M(x), M(y)$.

Equal Lengths Models the ability to compare the lengths of plaintexts, even when they appear encrypted. Defining EL^I requires to define a length function on irreducible terms t that belong to M^I (e.g. as in [31]), which assumes in particular that the length of an encrypted message only depends on the length of the message and the length of the key. Such a length function depends on the security parameter η, which is used to determine the length of the keys.

Then EL^I is the set of pairs of ciphertexts encrypting messages of equal lengths:

$$\begin{array}{l} EL^I(s,t) \stackrel{\text{def}}{=} M^I(s) \wedge M^I(t) \wedge \exists s_1, s_2, t_1, t_2, r_1, r_2. \\ \qquad s = \{s_1\}_{s_2}^{r_1}, t = \{t_1\}_{t_2}^{r_2} \wedge \forall \eta \in \mathbb{N}.\, l(s_2, \eta) = l(t_2, \eta) \wedge l(s_1, \eta) = l(t_1, \eta) \end{array}$$

Exercise 13
Consider $\mathcal{S}_0, \mathcal{F}_0$ as in example 2.2, and $\mathcal{P}_3 = \{M, EQ, EK, EL\}$. In the following cases, is the sequence (t_1, t_2, t_3, t_4) symbolically equivalent to

$$(s_1, s_2, s_3, s_4) \quad = \quad (\{< k_1, k_2 >\}_{k_3}^{r_1}, \ < \{k_1\}_{k_3}^{r_2}, \{k_3\}_{k_4}^{r_3} >, \ \{k_5\}_{k_2}^{r_4}, \ k_5)$$

1. $t_1 = k_1'$, $t_2 =< \{k_1'\}_{k_3'}^{r_5}, \{k_3'\}_{k_5'}^{r_6} >$, $t_3 = k_4'$, $t_4 = k_5'$
2. $t_1 = k_1'$, $t_2 =< \{k_1'\}_{k_3'}^{r_5}, \{k_3'\}_{k_2'}^{r_6} >$, $t_3 = k_4'$, $t_4 = k_5'$
3. $t_1 = \{m\}_{k_1'}^{r_5}$, $t_2 =< \{k_1'\}_{k_3'}^{r_6}, \{k_3'\}_{k_2'}^{r_7} >$, $t_3 = \{m'\}_{k_4'}^{r_8}$, $t_4 = k_5'$

The following exercise compares our definition of symbolic equivalence with the static equivalence of [13]: assuming that keys and random numbers are defined (e.g. through an infinite rewrite system, or assuming a representation of these names), then the two notions coincide, as long as only positive information is needed. Slight differences will show up in section 4 and there are also situations in which the equational formalisation is not possible.

Exercise 14
In [13], static equivalence is defined without a set of predicates. Show that, if we assume that predicates N and R are defined, which are true only on $\mathcal{N}$ and $\mathcal{R}$ respectively, it is possible to design a finite convergent rewrite system $\mathcal{S}$ which, for every t, u, rewrites $EQ(t, u)$, $M(t)$, $EK(t, u)$, $EL(t, u)$ to `true` exactly when these predicates are satisfied on t, u.

Exercise 15 (5)
We define the *patterns* and *known-keys* of a sequence $s_1, \ldots, s_n$ as follows:

- $k \in$ known-keys$(s_1, \ldots, s_n)$ iff $\exists C \in \mathcal{C}_n$, $C[s_1, \ldots, s_n] \downarrow = k$
- pattern$(s_1, \ldots, s_n) \stackrel{\text{def}}{=}$ (pattern$(s_1), \ldots,$ pattern(s_n))
- pattern$(< s_1, s_2 >) =<$ pattern(s_1), pattern$(s_2) >$
- pattern$(\{s_1\}_k^n) = \begin{cases} \{\text{pattern}(s_1)\}_k n & \text{if } k \in \text{known-keys}(s_1, \ldots, s_n) \\ \Box & \text{otherwise} \end{cases}$ where $\Box$ is a new constant
- pattern$(s) = s$ in all other cases.

Now, two sequences $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_n)$ are pattern-equivalent if there is a bijection σ on keys such that pattern$(s_1, \ldots, s_n) = \sigma$(pattern$(t_1, \ldots, t_n)$).

Define the rewrite system $\mathcal{S}$ and the predicates $\mathcal{P}$ such that the symbolic equivalence is pattern-equivalence.

Other examples of predicates may include decision of key cycles; this is somehow the point of view of [27], since they define the symbolic equivalence, taking key cycles into account. However, in their setting, the symbolic equivalence is much stronger since two expressions which contain key cycles are equivalent iff they are identical (up to renaming). This can be modeled using a function, which, in case of key cycles returns the term itself and, otherwise, returns 0 and defining a relation stating that the images by this function of the two arguments are identical. We may also refine the equivalence in such cases, allowing for instance $< \{K_1\}_{K_2}^n, \{K\}_K^{n'} >$ and $< \{0^\eta\}_{K_2} n, \{K\}_K n' >$ to be equivalent.

Actually, as in [12], families of symbolic equivalences could be considered in an abstract way. We prefer however the above framework, in which we can also discuss computability and complexity issues of the symbolic equivalence.

3 Soundness of symbolic abstraction: the simplest case

In the following, we will show that computational indistinguishability implies symbolic equivalence. We focus on symmetric encryption only and assume encryption with atomic keys. This follows the original paper of [4], with slight differences (for instance we consider type-3 security instead of type-0 security) and we try also to construct a detailed, stepwise proof.

3.1 Computational interpretation of terms

Let us recall that $\mathcal{F}_0 = \mathcal{N} \cup \mathcal{R} \cup \mathcal{W} \cup \{\{\cdot\}_\cdot^\cdot, D(\cdot,\cdot), <\cdot,\cdot>, \pi_1(\cdot), \pi_2(\cdot)\}$. $\mathcal{N}$ is a set of constants called *names* (or *nonces*, used for keys), $\mathcal{R}$ is a set of constants called *random numbers* (used for random inputs of machines), $\mathcal{W}$ is a finite set of constants (interpreted as particular words). We consider a particular subset of *valid terms* $\mathcal{M}_0$, which is the least set of messages containing $\mathcal{N} \cup \mathcal{R} \cup A$ and such that:

$$\begin{array}{rcl} x \in \mathcal{M}_0,\ k \in \mathcal{N},\ r \in \mathcal{R} & \Rightarrow & \{x\}_k^r \in \mathcal{M}_0 \\ x \in \mathcal{M}_0,\ y \in \mathcal{M}_0 & \Rightarrow & < x, y > \in \mathcal{M}_0 \end{array}$$

In particular, messages in $\mathcal{M}_0$ do not contain any projection or decryption symbol.

If $t_1, \ldots, t_n \in \mathcal{M}_0$, $\mathcal{N}(t_1, \ldots, t_n)$ is the set of names occurring in some t_i. Similary, $\mathsf{Coins}(t_1, \ldots, t_n)$ is the set of random numbers occurring as arguments of encryption in $t_1, \ldots, t_n$.

Every constant in $\mathcal{W}$ gets a fixed interpretation, for instance $[\![0001]\!] = 0001$.

If t is a term and $\mathcal{N}(t) = \{k_1, \ldots, k_n\}$, $\mathsf{Coins}(t) = \{r_1, \ldots, r_m\}$, the computational semantics of t is given by an ensemble $\{[\![t]\!]_\eta\}_{\eta \in \mathbb{N}}$ defined as follows:

- for each key $k_i \in \mathcal{N}(t)$, draw a key $\tau(k_i)$ from G_η and for each $r \in \mathsf{Coins}(t)$, draw $\tau(r)$ according to the uniform distribution. This gives a function τ from $\mathrm{Dom}\tau = \mathcal{N}(t) \cup \mathsf{Coins}(t)$ to bitstrings.
- compute $[\![t]\!]_\eta^\tau$ recursively as follows:
  - If $t \in \mathcal{W}$, then $[\![t]\!]_\eta^\tau = [\![t]\!]$
  - If t is a key k_i, then $[\![k_i]\!]_\eta^\tau = \tau(k_i)$
  - If $t =< t_1, t_2 >$ then $[\![t]\!]_\eta^\tau = [\![t_1]\!]_\eta^\tau \bullet [\![t_2]\!]_\eta^\tau$
  - If $t = \{t_1\}_k^r$ then
  $$[\![t]\!]_\eta^\tau = \mathcal{E}([\![t_1]\!]_\eta^\tau, \tau(k) \mid \tau(r))$$

where $\bullet$ is the (some kind of) string concatenation: we need at least a marker separating the two components of the pair, in order to know how to project on each component. In addition, if we want to avoid the associativity of $\bullet$, we need (other) markers also at the beginning and end of the pair. The superscript here is intended to record the context of the interpretation. We will also use $[\![u]\!]_\eta^\tau$ when u contains keys or random numbers, which are not in the domain of τ. In such a case, the ensemble is obtained by first drawing the keys and nonces whose interpretation is not fixed and then use the above defined interpretation.

For each term u, $\{[\![u]\!]_\eta\}_{\eta \in \mathbb{N}}$ is a polynomially computable ensemble, according to section 2.1.1. For tuples of terms, $[\![t_1, \ldots, t_m]\!]_\eta$ is defined in a similar way: first draw all keys and random numbers and then return the tuple $([\![t_1]\!]_\eta^\tau, \ldots [\![t_m]\!]_\eta^\tau)$. This is again a (multiple-outputs) PPT for each η, hence a polynomially computable ensemble.

The only important thing in this definition is that keys (or any name of the appropriate sort) is drawn before the computation, so that multiple occurrences of the same key are interpreted the same way. The above is not tailored to any specific set of security primitives and can be easily generalized: see [13] for a parametric definition of computational interpretations.

Exercise 16
Show that, if $(\mathcal{N}(s_1,\ldots,s_n)\cup\mathsf{Coins}(s_1,\ldots,s_n))\cap(\mathcal{N}(t_1,\ldots,t_m)\cup\mathsf{Coins}(t_1,\ldots,t_n))=\emptyset$, then $[\![s_1,\ldots,s_n,t_1,\ldots,t_m]\!]_\eta = [\![s_1,\ldots,s_n]\!]_\eta \times [\![t_1,\ldots,t_m]\!]_\eta$.

3.2 Security of encryption and indistinguishability

We consider now the case of type-3 security. We assume that all keys of G_η have the same length and that encrypting two messages of equal lengths yields two messages of equal length. In that case, we can define the *length* of a message term t (w.r.t. η) as the length of any sample of $[\![t]\!]_\eta$; $l(t,\eta)=|w|$ for any $w\in[\![t]\!]_\eta$. [3] $0^{l(u,\eta)}$ is a (constant) term in $\mathcal{W}$

Unless specified otherwise, the security is always considered w.r.t. a class $\mathcal{C}$ of adversaries and the indistinguishability relation $\approx$ is a shorthand for $\approx_\mathcal{C}$, for the same $\mathcal{C}$. We only assume that $\mathcal{C}$ is closed by composition with PPTs.

Lemma 3.1 *Let $u\in\mathcal{M}_0$ be such that $k,n\notin\mathcal{N}(u)\cup\mathsf{Coins}(u)\cup Dom(\tau)$. If $(G,\mathcal{E})$ is type-3 secure, then*

$$[\![\{u\}_k^n]\!]_\eta^\tau \approx [\![\{0^{l(u,\eta)}\}_k^n]\!]_\eta^\tau$$

In other words, given the actual keys $k_1,\ldots,k_n$, an encryption of u cannot be distinguished with an encryption of 0.

Proof:
Assume, by contradiction, that there is a machine $\mathcal{A}_1\in\mathcal{C}$ such that

$$\begin{aligned}\epsilon_{\mathcal{A}_1}(\eta) = &\,\mathbf{Pr}\{x \xleftarrow{R} [\![\{u\}_k^n]\!]_\eta^\tau, r \xleftarrow{R} U : \mathcal{A}_1(x,0^\eta \mid r)=1\}\\ &-\mathbf{Pr}\{x \xleftarrow{R} [\![\{0^{l(u,\eta)}\}_k^n]\!]_\eta^\tau, r \xleftarrow{R} U : \mathcal{A}_1(x,0^\eta \mid r)=1\}\end{aligned}$$

is not negligible. Then we define the machine $\mathcal{A}_2$, equipped with a randomized oracle f as follows:

1. $\mathcal{A}_2$ draws x from $[\![u]\!]_\eta^\tau$
2. draws a random input n for the oracle
3. calls the oracle f, returning a value $y=f(x\mid n)$
4. simulates $\mathcal{A}_1$ with input $(y,0^\eta)$.

$\mathcal{A}_2\in\mathcal{C}$. Let $\mathcal{M}_1^k$ be the machine obtained from $\mathcal{A}_2$ when the oracle $f(x\mid n)$ is the encryption of x with k and $\mathcal{M}_2^k$ be the machine obtained when $f(x\mid n)$ is the encryption of $0^{|x|}$ with k.

[3]Note that we state here a correspondance between the length of bitstring and the abstract length function l used in defining EL^I.

The probability of 0^η being accepted by $\mathcal{M}_1$ is

$$\begin{aligned} p_{\mathcal{M}_1^k}(\eta, k) &= \mathbf{Pr}\{n \stackrel{R}{\longleftarrow} U, x \stackrel{R}{\longleftarrow} [\![u]\!]_\eta^\tau, r \stackrel{R}{\longleftarrow} U : \mathcal{A}_1(\mathcal{E}(x, k \mid n), 0^\eta \mid r) = 1\} \\ &= \mathbf{Pr}\{x \stackrel{R}{\longleftarrow} \mathcal{E}([\![u]\!]_\eta^\tau, k \mid n), r \stackrel{R}{\longleftarrow} U : \mathcal{A}_1(x, 0^\eta \mid r) = 1\} \end{aligned}$$

since k, n are not occurring in u, $[\![\{u\}_k^n]\!]_\eta^\tau = \mathcal{E}([\![u]\!]_\eta^\tau, k \mid n)$, the polynomially computable distribution defined by the randomized encryption algorithm. Hence

$$\begin{aligned} &\mathbf{Average}\{k \stackrel{R}{\longleftarrow} G_\eta : p_{\mathcal{M}_1^k}(\eta, k)\} \\ &\quad = \mathbf{Pr}\{k \stackrel{R}{\longleftarrow} G_\eta, n \stackrel{R}{\longleftarrow} U, x \stackrel{R}{\longleftarrow} \mathcal{E}([\![u]\!]_\eta^\tau, k \mid n), r \stackrel{R}{\longleftarrow} U : \mathcal{A}_1(x, 0^\eta \mid r) = 1\} \\ &\quad = \mathbf{Pr}\{x \stackrel{R}{\longleftarrow} [\![\{u\}_k^n]\!]_\eta^\tau, r \stackrel{R}{\longleftarrow} U : \mathcal{A}_1(x, 0^\eta \mid r) = 1\} \end{aligned}$$

Similarly

$$\begin{aligned} &\mathbf{Average}\{k \stackrel{R}{\longleftarrow} G_\eta : p_{\mathcal{M}_2^k}(\eta, k)\} \\ &\quad = \mathbf{Pr}\{k \stackrel{R}{\longleftarrow} G_\eta, n \stackrel{R}{\longleftarrow} U, x \stackrel{R}{\longleftarrow} [\![u]\!]_\eta^\tau, r \stackrel{R}{\longleftarrow} U : \mathcal{A}_1(\mathcal{E}(0^{|x|}, k \mid n), 0^\eta \mid r) = 1\} \\ &\quad = \mathbf{Pr}\{k \stackrel{R}{\longleftarrow} G_\eta, n \stackrel{R}{\longleftarrow} U, x \stackrel{R}{\longleftarrow} [\![u]\!]_\eta^\tau, r \stackrel{R}{\longleftarrow} U : \mathcal{A}_1(\mathcal{E}(0^{l(u,\eta)}, k \mid n), \eta \mid r) = 1\} \\ &\quad = \mathbf{Pr}\{x \stackrel{R}{\longleftarrow} [\![\{0^{l(u,\eta)}\}_k^n]\!]_\eta^\tau, r \stackrel{R}{\longleftarrow} U : \mathcal{A}_1(x, 0^\eta \mid r) = 1\} \end{aligned}$$

It follows that

$$\mathsf{ADV}^3_{\mathcal{A}_2}(\eta) = \epsilon_{\mathcal{A}_1}(\eta)$$

which was assumed not to be negligible. Hence the adversary $\mathcal{A}_2$ gets a non-negligible advantage: the encryption scheme is not type-3 secure.

□

This can be easily extended to tuples:

Lemma 3.2 *Let $\{u_1\}_k^{n_1}, \ldots, \{u_m\}_k^{n_m} \in \mathcal{M}_0$ be such that $k, n_1, \ldots, n_m \notin Dom(\tau) \cup \mathcal{N}(u_1, \ldots, u_m) \cup$ $\mathsf{Coins}(u_1, \ldots, u_m)$. Assume also that $n_1, \ldots, n_m$ are distinct. If $(G, \mathcal{E})$ is type-3 secure, then*

$$[\![\{u_1\}_k^{n_1}, \ldots, \{u_m\}_k^{n_m}]\!]_\eta^\tau \approx [\![\{0^{l(u_1,\eta)}\}_k^{n_1}, \ldots, \{0^{l(u_m,\eta)}\}_k^{n_m}]\!]_\eta^\tau$$

Proof:
(sketch) The proof is very similar to the proof of lemma 3.1. For completeness, we give for instance the definition of the machine $\mathcal{A}_2$:

1. $\mathcal{A}_2$ draws $x_1, \ldots, x_m$ respectively from $[\![u_i]\!]_\eta^\tau$
2. Draw the m random inputs $n_1, \ldots, n_m$ for the oracle
3. calls m times the oracle f and store the results $y_1 = f(x_1 \mid n_1), \ldots y_m = f(x_m \mid n_m)$
4. Simulates $\mathcal{A}_1$ on input $(y_1, \ldots, y_m, \eta)$

□

Exercise 17
Complete the proof of the previous lemma. Is the distinctness of $n_1, \ldots, n_m$ necessary ?

We can also remove the exponents:

Lemma 3.3 *Assume that* $(G, \mathcal{E})$ *is a type-3 secure encryption scheme. Let* $\{u_1\}_k^{n_1}, \ldots, \{u_m\}_k^{n_m} \in \mathcal{M}_0$*,* $k, n_1, \ldots, n_m \notin \mathcal{N}(u_1, \ldots, u_m) \cup \mathsf{Coins}(u_1, \ldots, u_m)$ *and assume that* $n_1, \ldots, n_m$ *are distinct. Then* $[\![\{u_1\}_k^{n_1}, \ldots, \{u_m\}_k^{n_m}]\!]_\eta \approx [\![\{0^{l(u_1,\eta)}\}_k^{n_1}, \ldots, \{0^{l(u_m,\eta)}\}_k^{n_m}]\!]_\eta$.

Proof:
We simply apply lemma 3.2, which holds *for any* key assignments, besides k. □

Exercise 18 (4)
Assuming that $(G, \mathcal{E})$ is a type-3 secure encryption scheme, construct a type-3 secure encryption scheme $(G, \mathcal{E}')$ such that $[\![\{k\}_k]\!]_\eta \not\approx [\![\{0^{l(k,\eta)}\}_k]\!]_\eta$.

In the proof, we used two hypotheses:

1. The key k does not occur in u: actually we used a weaker hypothesis, namely that k *is hidden in* u. More precisely. $[\![\{u\}_k^r]\!]_\eta \approx \mathcal{E}([\![u]\!]_\eta, \cdot \mid \cdot)$ is sufficient where $\mathcal{E}([\![u]\!]_\eta, \cdot \mid \cdot)$ is the distribution defined by

$$\mathbf{Pr}\{x \xleftarrow{R} \mathcal{E}([\![u]\!]_\eta, \cdot \mid \cdot) : x = a\} = \mathbf{Pr}\{y \xleftarrow{R} [\![u]\!]_\eta, k \xleftarrow{R} \mathcal{K}_G, r \xleftarrow{R} U : \mathcal{E}(y, k \mid r) = a\}$$

   Equivalently $[\![< u, k >]\!]_\eta \approx [\![< u, k' >]\!]_\eta$ where k' is a key not occurring in u. Examples include $u = k_1$ or $u = \{k\}_{k_1}^r$. Counter examples are $u = k$ or $u =< k, k_1 >$.

2. The length of all encryptions of equal length messages are equal length messages. This is necessary for replacing $|x|$ with $l(u, \eta)$ when x is drawn from $[\![u]\!]_\eta$. This hypothesis is necessary for the lemma, unless we assume an encryption scheme which is length-concealing. In the latter case indeed, we could replace $0^{l(u,k)}$ in the statement with 0 and don't have to pay attention to the possible different lengths of samples of $[\![\{u\}_k^r]\!]_\eta$.

Lemma 3.3 can be adapted to other security levels, as sketched above for length-concealing encryption.

Exercise 19 (4)
If we want to take key cycles into account, show the following:

> If $(G, \mathcal{E})$ is type-3 secure, $\eta \in \mathbb{N}$ and u, v are terms such that $l(u, \eta) = l(v, \eta)$, for any key k' not occurring in u, v, $[\![< u, k >]\!]_\eta \approx [\![< u, k' >]\!]_\eta$ and $[\![< v, k >]\!]_\eta \approx [\![< v, k' >]\!]_\eta$, then, for any $r \notin \mathsf{Coins}(u, v)$, $[\![\{u\}_k^r]\!]_\eta \approx [\![\{v\}_k^r]\!]_\eta$.

Exercise 20
Show the following extension of lemma 3.3. If all occurrences of k in u are inside an encryption by a key, which cannot be recovered from u and $r \notin \mathsf{Coins}(u)$, then $[\![\{u\}_k^r]\!]_\eta \approx [\![\{0^{l(u,\eta)}\}_k^r]\!]_\eta$.

More precisely, we assume that, if $u|_p = k$, then there is a $p' < p$ such that $u|_{p'} = \{v\}_{k'}^{r'}$ such that, for all contexts $C \in \mathcal{C}$, $C[u]$ cannot be rewritten to k'.

Lemma 3.4 *Assume that the encryption scheme is type-3 secure. Let* $(\{u_1\}_k^{n_1},\ldots,\{u_m\}_k^{n_m},s_{m+1},\ldots,s_n)$ *be a sequence of terms in* $\mathcal{M}_0$ *such that* k *does not occur in* $s_{m+1},\ldots,s_n,u_1,\ldots,u_m$. *Assume moreover that* $n_1,\ldots,n_m$ *are distinct and do not occur in* $s_{m+1},\ldots,s_n$, *then*

$$[\![\{u_1\}_k^{n_1},\ldots,\{u_m\}_k^{n_m},s_{m+1},\ldots,s_n]\!]_\eta \approx [\![\{0^{l(u_1,\eta)}\}_k^{n_1},\ldots,\{0^{l(u_m,\eta)}\}_k^{n_m},s_{m+1},\ldots,s_n]\!]_\eta$$

Proof:
Let $s_i=\{u_i\}_k^{n_i}$ for $i=1,...,m$. Let $\{k_1,\ldots,k_r\}=\mathcal{N}(s_1,\ldots,s_n)\setminus\{k\}$ and $\{n_{m+1},\ldots,n_p\}=\mathsf{Coins}(s_1,\ldots,s_n)\setminus\{n_1,\ldots,n_m\}$ Since $k,n_1,\ldots,n_m$ do not occur in $s_{m+1},\ldots,s_n$, for any interpretation τ whose domain is $\{k_1,\ldots,k_r\}\cup\{n_{m+1},\ldots,n_p\}$,

$$[\![s_1,\ldots,s_n]\!]_\eta^\tau = [\![s_1,\ldots,s_m]\!]_\eta^\tau \times [\![s_{m+1},\ldots,s_n]\!]_\eta^\tau$$

since fixing the interpretation of $k_1,\ldots,k_p,n_{m+1},\ldots,n_p$, the interpretation of $s_{m+1},\ldots,s_n$ is fixed. Now, by lemma 3.2

$$[\![s_1,\ldots,s_m]\!]_\eta^\tau \approx [\![\{0^{l(u_1,\eta)}\}_k^{p_1},\ldots,\{0^{l(u_m,\eta)}\}_k^{p_m}]\!]_\eta^\tau$$

It follows, for the same reasons as above, that

$$[\![s_1,\ldots,s_n]\!]_\eta^\tau \approx [\![\{0^{l(u_1,\eta)}\}_k^{p_1},\ldots,\{0^{l(u_m,\eta)}\}_k^{p_m,s_{m+1},\ldots,s_n}]\!]_\eta^\tau$$

And since this holds for any τ:

$$[\![s_1,\ldots,s_n]\!]_\eta \approx [\![\{0^{l(u_1,\eta)}\}_k^{p_1},\ldots,\{0^{l(u_m,\eta)}\}_k^{p_m,s_{m+1},\ldots,s_n}]\!]_\eta$$

□

3.3 The soundness theorem

We need first two additional assumptions on term sequences. The first is not a restriction, as it states that random numbers are random numbers indeed: once they are used they can only be copied[4]. The second hypothesis is a true restriction on key occurrences.

The use of random number in a sequence $s_1,\ldots,s_n$ is *consistent* if

> $\forall r\in\mathsf{Coins}(s_1,\ldots,s_n)$, if r has two occurrences in the sequence: $s_i|_{p_1}=r$ and $s_j|_{p_2}=r$, then there are q_1,q_2,u,k such that $p_1=q_1\cdot 3, p_2=q_2\cdot 3$ and $s_i|_{q_1}=s_j|_{q_2}=\{u\}_k^r$

A sequence $(s_1,\ldots,s_n)$ induces a relation on $\mathcal{N}(s_1,\ldots,s_n)$ defined as the smallest reflexive and transitive relation such that:

$$k\le_{(s_1,\ldots,s_n)} k' \quad \text{if} \quad \exists i.\exists s'_i.\exists p.\exists r_i.s_i|_p=\{s'_i\}_{k'}^{r_i} \text{ and } k\in\mathcal{N}(s'_i)$$

In what follows, we will only consider valid message sequences:

Definition 3.5 *A message sequence* $s_1,\ldots,s_n\in\mathcal{M}_0$ *is* valid *if*

[4]Note however that [4] assumes that they cannot even be copied, which is an unnecessary assumption.

1. *The use of random number is consistent in the sequence* $s_1, \ldots, s_n$
2. $\leq_{(s_1,\ldots,s_n)}$ *is an ordering.*

Example 3.1 *The following are not valid message sequences:*

$$(\{\{k_1\}_{k_2}^{r_1}\}_{k_3}^{r_2}, \{\{k_2\}_{k_4}^{r_3}\}_{k_1}^{r_4})\ ,\ (\{\{k_1\}_{k_2}^{r_1}\}_{k_3}^{r_2}, \{k_1\}_{k_3}^{r_1})$$

While the following are valid

$$(\{\{k_1\}_{k_2}^{r_1}\}_{k_3}^{r_2}, \{\{k_1\}_{k_4}^{r_3}\}_{k_3}^{r_4})\ ,\ (\{\{k_1\}_{k_2}^{r_1}\}_{k_3}^{r_2}, \{k_1\}_{k_2}^{r_1})$$

Why this new assumption ? The first observation is that we cannot encrypt a key with itself, according to exercise 18. One may wonder whether it is sufficient to disallow this type of cycles. This is referred to as an open question in [4, 5].

Open Question 4 *Is there a type-3 secure encryption scheme such that* $[\![\{k\}_{k'}^r, \{k'\}_k^{r'}]\!]_\eta \not\approx [\![\{0^\eta\}_k^r, \{0^\eta\}_{k'}^{r'}]\!]_\eta$ *?*

This is a true open question. There are variants asking for stronger distinguishability properties.

Lemma 3.6 *If* $[\![s_1, \ldots, s_n, u_1, \ldots, u_m]\!]_\eta \approx [\![t_1, \ldots, t_n, v_1, \ldots, v_m]\!]_\eta$*,* f *is interpreted as a PPT constructible function, and* $r \notin \mathsf{Coins}(s_1, \ldots, s_n, u_1, \ldots, u_m)$*, then* $[\![f(s_1, \ldots, s_n \mid r), u_1, \ldots, u_m]\!]_\eta \approx [\![f(t_1, \ldots, t_n \mid r), v_1, \ldots, v_m]\!]$*.*

Proof:
For, by contradiction: if $\mathcal{A}$ distinguishes between $[\![f(s_1, \ldots, s_n \mid r), u_1, \ldots, u_m]\!]_\eta$ and $[\![f(t_1, \ldots, t_n \mid r), v_1, \ldots, v_m]\!]$, we construct a machine $\mathcal{A}'$, which, on input $x_1, \ldots, x_n, y_1, \ldots, y_m$, first applies (the interpretation of) f to $x_1, \ldots, x_n, r$ and then simulates $\mathcal{A}$. $\mathcal{A}'$ distinguishes between the two distributions $[\![s_1, \ldots, s_n, u_1, \ldots, u_m]\!]$ and $[\![t_1, \ldots, t_n, v_1, \ldots, v_m]\!]$. □

Exercise 21
Complete the proof of the previous lemma.

Exercise 22
Show that the converse of the previous lemma is false.

In the rest of the section, we assume $\mathcal{F} = \mathcal{F}_0$, $\mathcal{P} = \mathcal{P}_3 = \{M, EQ, EL, EK\}$, and $\mathcal{S} = \mathcal{S}_0$. $\mathcal{F}_P = \mathcal{F}_0 \setminus (\mathcal{N}_S \cup \mathcal{R}_S)$ where $\mathcal{N}_S$ is a subset of $\mathcal{N}$ and $\mathcal{R}_S$ is a strict subset of $\mathcal{R}$. In particular $\mathcal{W} \subset \mathcal{F}_P$.

We will also use the notion of *size* of a term t, written $|t|$ and defined inductively as follows: $|c| \stackrel{\text{def}}{=} 1$ if c is a constant, $| < u, v > | \stackrel{\text{def}}{=} 1 + |u| + |v|$ and $|encpukr| \stackrel{\text{def}}{=} 3 + |u|$. Let us recall that $u[t \mapsto t']$ is the replacement of term t with t' in u (all occurrences of t are replaced with t').

Because random numbers can be repeated in term sequences (a case which is not considered in [4]), we need the following lemma, which is similar to the equivalence with transparent frames in [13]:

Lemma 3.7 *Let $(s_1, \ldots, s_n)$ be a valid sequence of terms in $\mathcal{M}_0$ and $t = \{u\}_k^r \in \mathcal{M}_0$. If, for every context C, $C[s_1, \ldots, s_n] \downarrow\neq k$, then*

$$[\![s_1, \ldots, s_n]\!]_\eta \approx [\![s_1[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta$$

where $\hat{t} = \{0^{l(u,\eta)}\}_k^r$

Proof:
We prove the lemma by induction on $|s_1| + \ldots + |s_n|$. In the base case, there is nothing to prove since t cannot occur as a subterm of some s_i.

Now, if some s_i is a pair $s_i =< s_i', s_i'' >$, then there are contexts C_1, C_2 such that $C_1[s_1, \ldots, s_n] \downarrow= s_i'$ and $C_2[s_1, \ldots, s_n] \downarrow= s_i''$. Hence there is no context C such that $C[s_1, \ldots, s_{i-1}, s_i', s_i'', s_{i+1}, \ldots, s_n] \downarrow= k$. By induction hypothesis,

$$[\![s_1, \ldots, s_{i-1}, s_i', s_i'', s_{i+1}, \ldots, s_n]\!]_\eta \approx \\ [\![s_1[t \mapsto \hat{t}], \ldots, s_{i-1}[t \mapsto \hat{t}], s_i'[t \mapsto \hat{t}], s_i''[t \mapsto \hat{t}], s_{i+1}[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta$$

and, by lemma 3.6,

$$[\![s_1, \ldots, s_n]\!]_\eta \approx [\![s_1[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta$$

If $s_i = s_j$ for a pair of distinct indices, we can also simply apply the induction hypothesis: $[\![s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n]\!]_\eta \approx [\![s_1[t \mapsto \hat{t}], \ldots, s_{i-1}[t \mapsto \hat{t}], s_{i+1}[t \mapsto \hat{t}], \ldots s_n[t \mapsto \hat{t}]]\!]_\eta$ and we get the result by repeating the compotent j at the place i.

Now, assume that there is no repeated term and that all members of the sequence are keys, constants or ciphertexts. Let k_0 be a maximal key in the sequence. We distinguish again two cases.

If k_0 is not a member of the sequence then we may assume (up to a re-numbering) that $s_1 = \{u_1\}_{k_0}^{r_1}, \ldots, s_m = \{u_m\}_{k_0}^{r_m}$ and k_0 does not occur in $s_{m+1}, \ldots, s_n$. In addition, since $s_1, \ldots, s_m$ are distinct and $s_1, \ldots, s_n$ is a valid sequence, $r_1, \ldots, r_m$ are distinct and do not occur in $s_{m+1}, \ldots, s_n$. Then, by lemma 3.4,

$$\begin{aligned} [\![s_1, \ldots, s_n]\!]_\eta &\approx [\![\{0^{l(u_1,\eta)}\}_{k_0}^{r_1}, \ldots, \{0^{l(u_m,\eta)}\}_{k_0}^{r_m}, s_{m+1}, \ldots, s_n]\!]_\eta \\ &\approx [\![\{0^{l(u_1,\eta)}\}_{k_0}^{r_1}, \ldots, \{0^{l(u_m,\eta)}\}_{k_0}^{r_m}]\!]_\eta \times [\![s_{m+1}, \ldots, s_n]\!]_\eta \end{aligned}$$

Moreover, for every context C, $C[s_{m+1}, \ldots, s_n] \downarrow\neq t$. By induction hypothesis,

$$[\![s_{m+1}, \ldots, s_n]\!]_\eta \approx [\![s_{m+1}[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta$$

It follows that

$$\begin{aligned} [\![s_1, \ldots, s_n]\!]_\eta &\approx [\![\{0^{l(u_1,\eta)}\}_{k_0}^{r_1}, \ldots, \{0^{l(u_m,\eta)}\}_{k_0}^{r_m}]\!]_\eta \times [\![s_{m+1}[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta \\ &\approx [\![\{0^{l(u_1,\eta)}\}_{k_0}^{r_1}, \ldots, \{0^{l(u_m,\eta)}\}_{k_0}^{r_m}, s_{m+1}[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta \end{aligned}$$

And, by lemma 3.4,

$$[\![s_1, \ldots, s_n]\!]_\eta \approx [\![\{u_1[t \mapsto \hat{t}]\}_{k_0}^{r_1}, \ldots, \{u_m[t \mapsto \hat{t}]\}_{k_0}^{r_m}, s_{m+1}[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta$$

which yields the desired result since $t \notin \{s_1, \ldots, s_m\}$ by hypothesis.

If k_0 is a member of the sequence , possibly after a re-numbering, we may assume w.l.o.g. that $s_1 = k_0, s_2 = \{u_2\}_{k_0}^{r_2}, \ldots, s_m = \{u_m\}_{k_0}^{r_m}$ and k_0 does not occur in $s_{m+1}, \ldots, s_n$.

If $m = 1$, then k_0 has only one occurrence, as a member of the sequence and $[\![s_1, \ldots, s_n]\!]_\eta \approx [\![k_0]\!]_\eta \times [\![s_2, \ldots, s_n]\!]_\eta$. In that case, it suffices to apply the induction hypothesis ($k \neq k_0$):

$$\begin{array}{rcl} [\![s_1, \ldots, s_n]\!]_\eta & \approx & [\![k_0]\!]_\eta \times [\![s_2[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!] \\ & \approx & [\![s_1[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!] \end{array}$$

Otherwise, as before, by maximality of k_0, and since the sequence is valid, $r_2, \ldots, r_m$ are distinct and do not occur in $s_{m+1}, \ldots, s_n$. Moreover, since $u_2 = D(s_2, s_1) \downarrow$, there is no context C such that $C[k_0, u_2, s_3, \ldots, s_n] \downarrow = k$. Hence, by induction hypothesis,

$$[\![k_0, u_2, s_3, \ldots, s_n]\!]_\eta \approx [\![k_0, u_2[t \mapsto \hat{t}], s_3[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta$$

Now, $t \neq s_2$ by hypothesis, hence, $s_2[t \mapsto \hat{t}] = \{u_2[t \mapsto \hat{t}]\}_{k_0}^{r_2}$. Then, by lemma 3.6,

$$[\![s_1, \ldots, s_n]\!]_\eta \approx [\![s_1[t \mapsto \hat{t}], \ldots, s_n[t \mapsto \hat{t}]]\!]_\eta$$

□

Lemma 3.8 *Assume that $(s_1, \ldots, s_n), (t_1, \ldots, t_n)$ are valid term sequences such that*

- $(s_1, \ldots, s_n) \sim_3 (t_1, \ldots, t_n)$
- *for every $t = \{u\}_k^r \in \mathcal{M}_0$, if $u \notin \mathcal{W}$ and t occurs in some s_i (resp. some t_i), then there is a context C such that $C[s_1, \ldots, s_n] \downarrow = k$ (resp. $C[t_1, \ldots, t_n] \downarrow = k$).*

Then, $s_i \in \mathcal{N}(s_j), i \neq j$ implies $t_i \in \mathcal{N}(t_j)$.

Proof:
We prove the lemma by induction on $|s_j|$.

In the base case, $|s_j| = 1$, then $s_i = s_j$. Using EQ, it follows that $t_i = t_j$. Moreover, using M (and e.g. the context $\{0\}_{x_i}^r$ for some $r \in \mathcal{R} \cap \mathcal{F}_P$), t_i must be a key.

Now, for the induction step. If $s_j =< s'_j, s''_j >$, using M, t_j must also be a pair $t_j =< t'_j, t''_j >$ and

$$(s_1, \ldots, s_{j-1}, s'_i, s''_i, s_{j+1}, \ldots, s_n) \sim_3 (t_1, \ldots, t_{i-1}, t'_i, t''_i, t_{i+1}, \ldots, t_n)$$

Moreover, either s_i occurs in s'_j or else s_i occurs in s''_j. By induction hypothesis, t_i is a key occurring either in t'_j or t''_j, hence occurs in $< t'_j, t''_j >$.

If $s_j = \{s'_j\}_{k_j}^{r_j}$, then we consider the context $D(x_j, x_i)$. Using M, if $s_i = k_j$, we must have $t_j = \{t'_j\}_{t_i}^{r'_j}$ for some t'_j, r'_j, hence t_i occurs in t_j. Otherwise, s_i occurs in s'_j and therefore $s'_j \notin \mathcal{W}$. By hypothesis, there is a context C such that $C[s_1, \ldots, s_n] \downarrow = k_j$. Using M again, $C[t_1, \ldots, t_n] \downarrow = k'_j$ is a key and, considering the context $D(x_j, C)$, t_j must be $\{t'_j\}_{k'_j}^{r'_j}$ for some t'_j. Now, $(s_1, \ldots, s_{j-1}, s'_j, s_{j+1}, \ldots, s_n) \sim_3 (t_1, \ldots, t_{j-1}, t'_j, t_{j+1}, \ldots, t_n)$ and, by induction hypothesis, t_i occurs in t'_j, hence in t_j.

□

Lemma 3.9 *Assume that $(G, \mathcal{E})$ is type-3 secure and $\eta \in \mathbb{N}$. Let $(s_1, \ldots, s_n), (t_1, \ldots, t_n)$ be two valid term sequences such that, $\forall C \in \mathcal{C}_n$, if C itself is irreducible, then $C[s_1, \ldots, s_n]$ and $C[t_1, \ldots, t_n]$ are irreducible. If, moreover, for all predicates $P \in \mathcal{P}$ of arity k and all index sequences $i_1, \ldots, i_k$, $P^I(s_{i_1}, \ldots, s_{i_k}) \Leftrightarrow P^I(t_{i_1}, \ldots, t_{i_k})$, then $[\![s_1, \ldots, s_n]\!]_\eta \approx [\![t_1, \ldots, t_n]\!]_\eta$.*

Proof:
The irreducibility of $C[s_1, \ldots, s_n]$ and $C[t_1, \ldots, t_n]$ implies that none of the terms $t_1, \ldots, t_n$ is a pair: otherwise, take $C = \pi_1(x_i)$ and there is a redex. Similarly, there cannot be two indices i, j such that $s_i = \{s'_i\}^{r_i}_{s_j}$, for some term s'_i: in such a case, there is a redex with the irreducible context $D(x_i, x_j)$. A similar property holds on the sequence $t_1, \ldots, t_n$. For similar reasons, there cannot be any $\{u\}^r_k$ in any of the sequences, if $k \in \mathcal{F}_P$.

We prove the lemma by induction on n. The case $n = 0$ is trivial. If one of the sequences contains two identical terms, for instance $s_i = s_j$ then the corresponding terms in the other sequence must be equal: $t_i = t_j$ (because they satisfy the same EQ predicates). Hence we can remove s_i, t_i and apply the induction hypothesis. Similarly, if $s_i \in T(\mathcal{F}_P)$ for some i, then $t_i = s_i$ and we can remove the two terms from the sequences and apply the induction hypothesis.

Now, the two sequences contain only distinct keys and distinct ciphertexts. Moreover, for every $k \in \mathcal{N}$ either k has only one occurrence, as a member of the sequence, or else there is no context C such that $C[s_1, \ldots, s_n] \downarrow = k$ (resp. $C[t_1, \ldots, t_n] \downarrow = k$). By lemma 3.7, we may assume w.l.o.g. that every s_i and every t_i is either a key or a term in $\mathcal{W}$ encrypted with a key. (Formally, this follows by induction on the number of key occurrences and by transitivity of $\approx$).

If a key k occurs as a member of the sequence, say $s_1 = k$, then k does not occur in any other s_j. Moreover, t_1 must also be a key (thanks to EK or its derived predicate Cipher), which occurs nowhere else in $t_1, \ldots, t_n$. By induction hypothesis $[\![s_2, \ldots, s_n]\!]_\eta \approx [\![t_2, \ldots, t_n]\!]_\eta$, hence

$$[\![s_1, \ldots, s_n]\!]_\eta = [\![s_1]\!]_\eta \times [\![s_2, \ldots, s_n]\!]_\eta = [\![t_1]\!]_\eta \times [\![s_2, \ldots, s_n]\!]_\eta \approx [\![t_1]\!]_\eta \times [\![t_2, \ldots, t_n]\!]_\eta = [\![t_1, \ldots, t_n]\!]_\eta$$

Now, assume that sequences are ciphertexts only. Let $s_1 = \{u_1\}^{p_1}_{k_1}$ and consider all terms in the sequence whose encryption key is k_1. For simplicity we will assume that they are indexed $s_1, \ldots s_r$. Since at most one key occurs in each term, $\mathcal{N}(s_1, \ldots, s_r) \cap \mathcal{N}(s_{r+1}, \ldots, s_n) = \emptyset$. Then t_1 is also a ciphertext $t_1 = \{v_1\}^{p'_1}_{k_2}$ and (because of EK), $t_2 = \{v_2\}^{p'_2}_{k_2}, \ldots t_r = \{v_r\}^{p'_r}_{k_2}$. Moreover, if we let $k_0 \in \mathcal{N} \cap \mathcal{F}_P$ (which is assumed to be non empty), for every $i \in \{1, \ldots, r\}$, $(s_i, \{0^{l(u_i,\eta)}\}^{p_0}_{k_0}) \in EL^I$, hence $(t_i, \{0^{l(v_i,\eta)}\}^{p_0}_{k_0}) \in EL^I$, which implies $l(u_i, \eta) = l(v_i, \eta)$ since $\{0^{l(v_i,\eta)}\}^{p_0}_{k_0} \in T(\mathcal{F}_P)$. Finally, $p_1, \ldots, p_r$ (resp. $p'_1, \ldots, p'_r$) are distinct since $s_1, \ldots, s_r$ (resp. $t_1, \ldots, t_r$) are distinct and by consistency of the random number occurrences. For the same reason, and because $s_1, \ldots, s_r$ (resp. $t_1, \ldots, t_r$) are ciphertexts using the same encryption key, $p_1, \ldots, p_r$ (resp. $p'_1, \ldots, p'_r$) cannot occur in $u_1, \ldots, u_r$ (resp. $v_1, \ldots, v_r$).

On one hand, by induction hypothesis, we have $[\![s_{r+1}, \ldots, s_n]\!]_\eta \approx [\![t_{r+1}, \ldots, t_n]\!]_\eta$ and, on the other hand, by lemma 3.3, we have

$$\begin{array}{rcl}
[\![s_1, \ldots, s_r]\!]_\eta & \approx & [\![\{0^{l(u_1,\eta)}\}^{p_1}_{k_1}, \ldots, \{0^{l(u_r,\eta)}\}^{p_r}_{k_1}]\!] \\
& = & [\![\{0^{l(v_1,\eta)}\}^{p_1}_{k_1}, \ldots, \{0^{l(v_r,\eta)}\}^{p_r}_{k_1}]\!] \\
& = & [\![\{0^{l(u_1,\eta)}\}^{p_1}_{k_2}, \ldots, \{0^{l(u_r,\eta)}\}^{p_r}_{k_2}]\!] \\
& \approx & [\![t_1, \ldots, t_r]\!]_\eta
\end{array}$$

It follows, (using exercises 12 and 16) and since $\mathcal{N}(s_1,\ldots,s_r)\cap\mathcal{N}(s_{r+1},\ldots,s_n) = \emptyset = \mathcal{N}(t_1,\ldots,t_r)\cap\mathcal{N}(t_{r+1},\ldots,t_n)$, that

$$[\![s_1,\ldots,s_n]\!]_\eta = [\![s_1,\ldots,s_r]\!]_\eta \times [\![s_{r+1},\ldots,s_n]\!]_\eta \approx [\![t_1,\ldots,t_r]\!]_\eta \times [\![t_{r+1},\ldots,t_n]\!]_\eta = [\![t_1,\ldots,t_n]\!]_\eta$$

$\square$

We fix $\sim_3$ as the symbolic equivalence corresponding to $\mathcal{F}_0$, $\mathcal{S}_0$, $\mathcal{P}_3$.

Theorem 3.10 (Soundness) *Assume $(s_1,\ldots,s_n)$ and $(t_1,\ldots,t_n)$ are two valid sequences of terms and that the encryption scheme is type-3 secure, then*

$$(s_1,\ldots,s_n) \sim_3 (t_1,\ldots,t_n) \quad \Rightarrow \quad [\![s_1,\ldots,s_n]\!]_\eta \approx [\![t_1,\ldots,t_n]\!]_\eta$$

Proof:

We prove, by induction on the total size of $s_1,\ldots,s_n,t_1,\ldots,t_n$ that $(s_1,\ldots,s_n) \sim_3 (t_1,\ldots,t_n)$ implies $[\![s_1,\ldots,s_n]\!]_\eta \approx [\![t_1,\ldots,t_n]\!]_\eta$. The base case ($n = 0$) is trivial.

If there are repetitions in the sequence: $s_i = s_j$, then, thanks to the EQ predicate, we must have also $t_i = t_j$, then we can simply remobe s_i, t_i from the sequences and apply the induction hypothesis.

If some s_i is a pair, then the corresponding t_i must also be a pair (due to predicate M, using a context $\pi_j(x_i)$) Let $s_i = < s'_i, s''_i >$ and $t_i = < t'_i, t''_i >$, we replace s_i with s'_i, s''_i and t_i with t'_i, t''_i in the sequences. We still get equivalent sequences since s'_i, s''_i are obtained as $\pi_1(x_i)$ and $\pi_2(x_i)$ respectively. Applying the induction hypothesis, we get $[\![s_1,\ldots,s_{i-1},s'_i,s''_i,s_{i+1},\ldots,s_n]\!]_\eta \approx [\![t_1,\ldots,t_{i-1},t'_i,t''_i,t_{i+1},\ldots,t_n]\!]_\eta$ and we conclude by lemma 3.6.

Now, if none of the s_i's are pairs, all terms in the sequences are either constants, keys or ciphertexts. According to lemma 3.7, we can also assume w.l.o.g. that, for every $t = \{u\}_k^r \in \mathcal{M}_0$, either there is a context C such that $C[s_1,\ldots,s_n]\downarrow= k$ (resp. $C[t_1,\ldots,t_n]\downarrow= k$) or else $u \in \mathcal{W}$. By lemma 3.8, we have in addition that, if $s_i \in \mathcal{N}$ occurs in $s_j, j \neq i$, then $t_i \in \mathcal{N}$ occurs in t_j. In particular, if s_i is a maximal key in the sequence $(s_1,\ldots,s_n)$, then t_i is a maximal key in the sequence $t_1,\ldots,t_n$.

Now, either there is no irreducible context C such that $C[s_1,\ldots,s_n]$ or $C[t_1,\ldots,t_n]$ is reducible, in which case we conclude by lemma 3.9, or else there is an irreducible context such that $C[s_1,\ldots,s_n]$ contains a redex. Such a redex must be an instance of the rule $D(\{x\}_y^r, y) \to x$: there are terms s_i, s_j, s'_j such that $s_j = \{s'_j\}_k^{r_j}$ and $s_i = k$. Moreover, we can always choose k to be maximal,among the keys occurring as some s_j. Then k is maximal among all keys occuring in $s_1,\ldots,s_n$: the only possible occurrences of a larger key k' would be as $s_{i'} = \{u'\}_{k'}^{r'}$ and k' itself is not a member of the sequence. But, in that case, we assumed u' constant, hence k' cannot be larger than k. Now, as noticed above, t_i must also be a maximal key in the sequence $(t_1,\ldots,t_n)$. Applying the context $D(x_j,x_i)$ to both sequences, and thanks to the M predicate, $t_j = \{t'_j\}_{t_i}^{r'_j}$. Next, we replace s_j with s'_j and t_j with t'_j; we get again equivalent sequences and, by induction hypothesis, $[\![s_1,\ldots,s_{j-1},s'_j,s_{j+1},\ldots,s_n]\!]_\eta \approx [\![t_1,\ldots,t_{j-1},t'_j,t_{j+1},\ldots,t_n]\!]_\eta$. Hence

$$[\![s_1,\ldots,s_{j-1},s'_j,s_{j+1},\ldots,s_n]\!]_\eta \approx [\![t_1,\ldots,t_{j-1},t'_j,t_{j+1},\ldots,t_n]\!]_\eta.$$

Finally, apply lemma 3.6 with encryption for f and since, by maximality of the keys t_i and s_i, the random numbers r_j, r'_j occur only once in the sequence. $\square$

There are various ways of relaxing the conditions on key cycles, as shown below.

Exercise 23
Instead of defining an ordering on keys, we define a relation on pairs (k, r) as the smallest reflexive and transitive relation such that:

$$(k, r) \leq_{(s_1,\ldots,s_n)} (k', r') \qquad \text{if} \qquad \exists i. \exists s'_i. s_i = \{s'_i\}_{k'}^{r'} \;\text{ and }\; (k \in \mathcal{N}(s'_i) \quad \text{or} \quad r \in \mathsf{Coins}(s'_i))$$

1. Show that, if $(s_1, \ldots, s_n)$ is valid, then the above relation is an ordering
2. Show that the distinctness of $n_1, \ldots, n_m$ is an unnecessary hypothesis of lemma 3.2.
3. Show that the previous results hold if we only assume that the above relation is an ordering (instead of the hypothesis that sequences are valid).

Exercise 24
Now we replace the definition of $k \leq_{(s_1,\ldots,s_n)} k'$ with the smallest reflexive and transitive relation such that:

$$k \preceq_{(s_1,\ldots,s_n)} k' \qquad \text{if} \qquad \exists i. \exists s'_i. \exists r_i. s_i \vdash \{s'_i\}_{k'}^{r_i} \;\text{ and }\; s'_i \vdash k$$

where $\vdash$ is the deduction relation defined by: $t_1, \ldots, t_n \vdash u$ iff there is a context C such that $C[t_1, \ldots, t_n] \xrightarrow[\mathcal{S}]{*} u$.

Show that $\preceq_{(s_1,\ldots,s_n)} \subseteq \leq_{(s_1,\ldots,s_n)}$ and yet that replacing $\leq$ with $\preceq$ in definition 3.5, the theorem 3.10 still holds.

3.4 Other security levels

Theorem 3.10 is established for a particular symbolic equivalence and a type-3 secure symmetric encryption scheme. If we switch to type-0,1,2 security, we have similar results, using also similar proofs. We indicate below, in each case, what changes in the symbolic equivalence and in the proofs.

The results and proofs can also be easily extended to other primitives under various assumptions: see section 6.

3.4.1 Type-2

This level of security assumes some padding of messages, so that the encryption scheme is length-concealing. More precisely, all definitions are identical to type-3 security, except that the oracles in the definition of advantage (definition 2.3) now return the encryption (of x or y) even when x, y do not have the same length.

We define then a slightly different symbolic equivalence: we simply remove EL from $\mathcal{P}_3$. Let $\sim_2$ be the resulting equivalence.

Theorem 3.11 *Assume that $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_n)$ are two valid sequences of terms and that the encryption scheme is type-2 secure, then*

$$(s_1, \ldots, s_n) \sim_2 (t_1, \ldots, t_n) \Rightarrow [\![s_1, \ldots, s_n]\!]_\eta \approx [\![t_1, \ldots, t_n]\!]_\eta$$

Proof:
(sketch) we only indicate what changes in the soundness proof for type-3 secure encryption schemes. First, we prove an analog of lemma 3.1: if $u \in \mathcal{M}_0$ and $\mathcal{N}(u) \cup \mathsf{Coins}(u) \subseteq \mathrm{Dom}(\tau)$ and $k, n \notin \mathrm{Dom}(\tau)$ and $(G, \mathcal{E})$ is type-2 secure, then $[\![\{u\}_k^n]\!]_\eta^\tau \approx [\![\{0\}_k^n]\!]_\eta^\tau$. Only the exponent $l(u, \eta)$ has been removed.

The proof of this lemma works the same way as the proof of lemma 3.1: we reason by contradiction and build an adversary, who will break the level-2 security of the encryption scheme.

Analogs of lemmas 3.2, 3.3, 3.4 follow. Lemma 3.6 is independent of the security level. Lemma 3.7 only depend on the rewrite systems and lemma 3.4. Lemma 3.8 and its proof only use the predicates EQ and M. Only the proof of lemma 3.9 has to be modified and, actually, simplified: we replace in the proof all occurrences of symbols $0^{l(t,\eta)}$ with 0. We cannot use the predicate EL, but it was used in that proof only for deriving identities $l(u, \eta) = l(v, \eta)$, which are no longer relevant in the absence of boxes.

Finally, the proof of theorem 3.10 is unchanged: it only relies on the form of rules in $\mathcal{S}$, the predicate M and the predicate EQ. and previous lemmas.

□

3.4.2 Type 1

We rely here on the definition 2.7. We keep the same definition of equivalence, removing however EK from $\mathcal{P}_3$ and adding a unary predicate Cipher, which is interpreted as the subset of M^I of messages which are ciphertexts[5]. We let $\sim_1$ be the resulting equivalence.

Lemmas 3.1, 3.2, 3.3 also hold in this setting. We need however a stronger property:

Lemma 3.12 *Let $u_1, \ldots, u_m \in \mathcal{M}_0$ be such that $\mathcal{N}(u_1, \ldots, u_m) \cup \mathit{Coins}(u_1, \ldots, u_m) \subseteq \mathit{Dom}(\tau)$ and $k, k_1, \ldots, k_m, n_1, \ldots, n_m \notin \mathit{Dom}(\tau)$. If $(G, \mathcal{E})$ is type-1 secure, then*

$$[\![\{u_1\}_{k_1}^{n_1}, \ldots, \{u_m\}_{k_m}^{n_m}]\!]_\eta^\tau \approx [\![\{0^{l(u_1,\eta)}\}_k^{n_1}, \ldots, \{0^{l(u_m,\eta)}\}_k^{n_m}]\!]_\eta^\tau$$

Proof:
(sketch) We prove first the lemma when $k_1 = \ldots = k_s$ and $k_{s+1} = \ldots = k_m$.

By contradiction, we assume that the two distributions can be distinguished by a machine $\mathcal{A}_1$, which gets a significant advantage $\epsilon_{\mathcal{A}_1}(\eta)$. Let $\mathcal{A}_2$ be the two-oracles (f_1, f_2) machine with input η defined by:

1. for $i = 1, \ldots, m$ draw x_i from $[\![u_i]\!]_\eta^\tau$
2. draw $n_1, \ldots, n_m$, the random inputs for the oracles.
3. for $i = 1, ..., s$ call f_1 on input x_i and store the result in y_i
4. for $i = s + 1, ..., m$, call f_2 on input x_i and store the result in y_i

[5]We actually only need to distinguish between ciphers and keys. If the length of a cipher is longer than the length of the corresponding plaintext, we do not need this additional predicate

5. simulate the machine $\mathcal{A}_1$ with input $(y_1, \ldots, y_m, \eta)$.

We claim that $\mathsf{ADV}^1_{A_2}(\eta) = \epsilon_{\mathcal{A}_1}(\eta)$. Hence a contradiction with type-1 security. It follows that the lemma is proved when there are exactly two distinct elements in the sequence $k_1, \ldots, k_n$.

Now, let us prove the lemma by induction on m. When $m = 1$, it is lemma 3.1. Assume $m > 1$. If $k_1 = \ldots = k_m$, then it is lemma 3.2. If there are at least two distinct keys, assume $k_1 = \ldots = k_s$ and $k_1 \notin \{k_{s+1}, \ldots, k_m\}$ $(m > s)$. Since $\mathcal{N}(u_1, \ldots, u_m) \cup \mathsf{Coins}(u_1, \ldots, u_m) \subseteq \mathrm{Dom}(\tau)$,

$$[\![u_1, \ldots, u_m]\!]^\tau_\eta = [\![u_1, \ldots, u_s]\!]^\tau_\eta \times [\![u_{s+1}, \ldots, u_m]\!]^\tau_\eta$$

We now apply the induction hypothesis and the independence of distributions backwards:

$$\begin{array}{rcl} [\![u_1, \ldots, u_m]\!]^\tau_\eta & \approx & [\![u_1, \ldots, u_s]\!]^\tau_\eta \times [\![\{0^{l(u_{s+1},\eta)}\}^{n_{s+1}}_k, \ldots, \{0^{l(u_m,\eta)}\}^{n_m}_k]\!]^\tau_\eta \\ & = & [\![u_1, \ldots, u_s, \{0^{l(u_{s+1},\eta)}\}^{n_{s+1}}_k, \ldots, \{0^{l(u_m,\eta)}\}^{n_m}_k]\!]^\tau_\eta \end{array}$$

and use the case of two distinct keys to conclude. □

As before, lemmas 3.6 and 3.7 hold for security levels 3 and upwards. Lemma 3.8 only rely on M, EQ. While lemma 3.12 is slightly more complex than for the type-3 case, the analog of lemma 3.9 for type-1 security is much easier to prove, relying on lemma 3.12. We still start by reducing the general case to the case of ciphertexts sequences (this requires Cipher, EQ only). Then, we can simply apply the above lemma and assume that all maximal keys occur only once. Then split the distribution into products on both sides; we use EL in the base case only, applying lemma 3.1. Then we get the following theorem (using the same proof as in theorem 3.10):

Theorem 3.13 *Assume that $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_n)$ are two valid sequences of terms and that the encryption scheme is type-1 secure, then*

$$(s_1, \ldots, s_n) \sim_1 (t_1, \ldots, t_n) \Rightarrow [\![s_1, \ldots, s_n]\!]_\eta \approx [\![t_1, \ldots, t_n]\!]_\eta$$

3.4.3 Type 0

Now we assume type-1 security and length-concealing. We consider again the same equivalence relation, in which predicates are now $\mathcal{P} = \{M, EQ, \mathsf{Cipher}\}$: we do not have EL nor EK. This defines a symbolic relation $\sim_0$.

Theorem 3.14 *Assume that $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_n)$ are two valid sequences of terms and that the encryption scheme is type-0 secure, then*

$$(s_1, \ldots, s_n) \sim_0 (t_1, \ldots, t_n) \Rightarrow [\![s_1, \ldots, s_n]\!]_\eta \approx [\![t_1, \ldots, t_n]\!]_\eta$$

In this case, the proof is much simpler: many parts of previous proofs can be removed.

Proof:
(sketch) We follow the proof of theorem 3.13, in particular the proof of lemma 3.12, replacing the expressions 0^α, $0^{l(\alpha,\eta)}$ with 0. EL was used only in the simplified version of lemma 3.9, and only in deriving identities on lengths, which are useless now. □

3.4.4 Decryption-confusing encryption schemes

In all above security assumptions, we kept M in the predicate symbols since nothing guarantees that an attacker cannot decide whether a decryption succeeds or not. For the completeness results of next sections, we will require the implementability of M, which might not be the case for some encryption schemes: it might be the case that an adversary cannot decide whether its decryption attempt succeeded or not. Typical examples are AES or DES schemes.

We define $[\![D(s,t)]\!]_\eta^\tau$ as:

1. draw all keys in $\mathcal{N}(s,t) \setminus \mathrm{Dom}(\tau)$, according to the distribution G_η
2. draw all random numbers in $\mathsf{Coins}(s,t) \setminus \mathrm{Dom}(\tau)$ according to the uniform distribution.
3. The two previous steps yield, together with τ, an assignment τ' whose domain contains $\mathcal{N}(s,t) \cup \mathsf{Coins}(s,t)$.
4. Apply the function D to $[\![s]\!]_\eta^{\tau'}$ and $[\![t]\!]_\eta^{\tau'}$. This function returns a special bitstring $\perp$ in case it is not defined on the sample input.

Intuitively, an encryption scheme is *decryption confusing* if $[\![D(\{k_1\}_{k_2}^r, k_3)]\!]_\eta \approx [\![k_1]\!]_\eta$ for any distinct names k_1, k_2, k_3. I.e. if the plaintext is chosen randomly, then we cannot distinguish between a successful decryption and a failure. More generally, we must have $[\![D(\{x\}_{k_1}^r, k_2)]\!]$ indistinguishable from a random bit string of the same length as x. Formally, we need actually a slightly stronger definition: for instance we would like $[\![\{k_1\}_{k_3}^r, k_2]\!]_\eta \approx [\![\{k_1\}_{k_2}^r, k_2]\!]_\eta$ (Note that this is exactly the example of [31].) Since we cannot assume that encryption is the inverse of decryption (this holds only for deterministic encryption), we must state the definition on ciphertexts instead. Finally, since encryption of random number should not be distinguishable from random numbers, this yields the following (minimal) definition:

Definition 3.15 *An encryption scheme is* decryption-confusing *if:*

1. *for every name k, $[\![\{0^\eta\}_k^r]\!]_\eta \approx [\![k]\!]_\eta$*
2. *for every $x \in \mathcal{M}_0$ and every $k_1, k_2 \in \mathcal{N} \setminus \mathcal{N}(x)$ such that $k_1 \neq k_2$,*
$$[\![\{\{x\}_{k_1}^{r_1}\}_{k_2}^{r_2}, k_2]\!]_\eta \approx [\![\{x\}_{k_1}^{r_1}, k_2]\!]_\eta$$

As shown below, this implies what we want:

Lemma 3.16 *If the encryption scheme is decryption confusing, then*

1. *for every distinct names k_1, k_2,*
$$[\![D(k_1, k_2), k_2]\!]_\eta \approx [\![k_1, k_2]\!]_\eta \approx [\![\{k_1\}_{k_2}^r, k_2]\!]$$
2. *For any term x not containing projection symbols, and for any names $k_1, k_2 \in \mathcal{N}$ such that $k_1 \neq k_2$, and $k_1, k_2 \notin \mathcal{N}(x)$,*
$$[\![D(\{x\}_{k_1}^r, k_2), k_2]\!]_\eta \approx [\![\{x\}_{k_1}^r, k_2]\!]_\eta \approx [\![\{\{x\}_{k_1}^{r_1}\}_{k_2}^{r_2}, k_2]\!]_\eta$$

Exercise 25 (3)
Prove the above lemma.

Comments :

- There are possible weaker assumptions about decryption confusion, yielding however much weaker soundness results.

  For instance, in [12] there is a soundness proof for one-time pad, which only works if each key appears at most once in the term sequences.

- The definition implies that encryption and decryption are length preserving, as shown by lemma 3.16.

- In the context of decryption-confusing schemes, the hierarchy type-0,...,type-3 is irrelevant: encryption is length preserving, hence not hiding the length and it is always which-key concealing (otherwise we could implement the success of a decryption using EK). Hence, we consider here only type-1 encryption schemes.

- If the encryption is deterministic, then we must add the identity $\{D(x,k)\}_k = x$ since applying the context $\{D(x_1,x_2)\}_{x_2}$ to $\{k_0\}_k, k$ and $\{k_0\}_{k_1}, k_2$ should give the same result.

- In [13] consider, as an application of general results, the case of encryption schemes, which are decryption confusing. They assume moreover that the encryption scheme is deterministic and that $\{D(x,k)\}_k = x$. Messages are lists of blocks, each block having the same length. The number of blocks can be observed as well as equality of each block. They show a soundness result, assuming type-1 security and also IND-CCA1. This last assumption is due to the presence (which they allow) of decryption commands in the messages.

- In the second part, we essentially replace a name with any ciphertext. This extends the indistinguishability to messages of length larger than η. Typically, this is satisfied by block-ciphers (CBC or ECB).

  Also, the definition does not seem to be vacuous, as AES and DES seem to satisfy the decryption-confusing assumption.

In what follows here, we assume a type-1 decryption-confusing encryption scheme.

The symbolic equivalence includes three predicates El and Eq, m which are interpreted in a slightly different way as before since, now, decryption can not fail. m is interpreted as the set of terms, which do not contain projection symbols. El is interpreted as the set of pairs of terms whose lengths are identical, where the length L is defined by:

$$\begin{array}{rcll rcl} L(k) & = & S & \text{For } k \in \mathcal{N} & L(D(x,k)) & = & L(x) \\ L(c) & = & S & \text{for any constant c} & L(\{x\}_k^r) & = & L(x) \\ & & & & L(< x,y >) & = & L(x) + L(y) \end{array}$$

S is an integer constant, used only for the purpose of this definition. Also, we assume $L(c) = S$ for every constant, for simplicity, which means that $\mathcal{W}$ is now restricted to words whose length is a multiple of S. We can think of S as the length of a block in a block cipher encryption scheme.

Eq is interpreted as equality on the terms that are in m. The rewrite system is unchanged. This defines a symbolic equivalence $\sim_c$.

Exercise 26
Show that the following sequences are (symbolically as well as computationally) distinguishable:

1. $(\{<\mathbf{0}, k_0>\}_{k_1}^{r}, k_2, k_1)$ and $(\{<\mathbf{1}, k_0>\}_{k_1'}^{r'}, k_1', k_2')$

2. $(\{\{<\mathbf{0}, k_0>\}_{k_2}^{r_1}\}_{k_1}^{r_2}, k_2, k_1)$ and $(\{\{<\mathbf{0}, k_0>\}_{k_2'}^{r_1'}\}_{k_1'}^{r_2'}, k_1', k_2')$

3. $(\{k_0\}_{k_1}^{r_1}, k_1, \{<k_2, k_3>\}_{k_1}^{r_2}, k_2)$ and $(\{k_0'\}_{k_1'}^{r_1'}, k_2', \{<k_2', k_3'>\}_{k_1'}^{r_2'}, k_1')$

Where $\mathbf{0}$ and $\mathbf{1}$ are sequences of 0's and 1's respectively, of the same appropriate length.

Exercise 27
Show that the following are symbolically equivalent:

1. $(\{\{<\mathbf{0}, k_0>\}_{k_3}^{r_1}\}_{k_1}^{r_2}, k_2, k_1)$ and $(\{\{<\mathbf{1}, k_0>\}_{k_3'}^{r_1'}\}_{k_1'}^{r_2'}, k_1', k_2')$

2. $(\{k_0\}_{k_1}^{r_1}, k_1, k_2, \{\{k_3\}_{k_2}^{r_2}\}_{k_1}^{r_3})$ and $(\{k_0'\}_{k_1'}^{r_1'}, k_2', k_1', \{\{k_3'\}_{k_2'}^{r_2'}\}_{k_1'}^{r_3'})$

Valid sequences are defined as before: sequences of messages in $\mathcal{M}_0$ (in particular not containing decryption symbols) with a consistent use of random numbers and no key-cycle.

Let $s_1, \ldots, s_n$ be a valid sequence of terms in $\mathcal{M}_0$. t is *deducible* from $s_1, \ldots, s_n$ if there is a context C such that $C[s_1, \ldots, s_n] \downarrow = t$. We let $ST(s_1, \ldots, s_n)$ be the set of subterms of $s_1, \ldots, s_n$ and $RT(s_1, \ldots, s_n)$ be the set of subterms of $s_1, \ldots, s_n$ of the form $\{w\}_k^r$ with $w \in \mathcal{W}$ and k not deducible from $s_1, \ldots, s_n$. Also, following the terminology of [13], a term sequence $(s_1, \ldots, s_n)$ is *transparent* if, for any $k \in \mathcal{N}(s_1, \ldots, s_n)$, either there is a context C such that $C[s_1, \ldots, s_n] \downarrow = k$ or else k only occurs in $RT(s_1, \ldots, s_n)$.

If $(s_1, \ldots, s_n)$ is a valid sequence, let $\sqsubseteq_{(s_1,\ldots,s_n)}$ be defined by:

$$\begin{array}{lll} s \sqsubseteq_{(s_1,\ldots,s_n)} t \text{ iff} & s = t & \\ & \text{Or } t = <t_1, t_2> & \text{and either } s \sqsubseteq_{(s_1,\ldots,s_n)} t_1 \text{ or } s \sqsubseteq_{(s_1,\ldots,s_n)} t_2 \\ & \text{Or } t = \{t_1\}_{k_1}^{r_1} & \text{and } s \sqsubseteq_{(s_1,\ldots,s_n)} t_1 \text{ and } t \notin RT(s_1, \ldots, s_n) \end{array}$$

Lemma 3.17 *Assume a decryption-confusing type 1 encryption scheme. Let $(s_1, \ldots, s_n)$ be a transparent sequence such that*

1. *$s_1, \ldots, s_n$ are names or ciphertexts*

2. *$\mathcal{W} \cap ST(s_1, \ldots, s_n) = \mathcal{W} \cap ST(RT(s_1, \ldots, s_n))$: the only occurrences of $w \in \mathcal{W}$ in the sequence are in expressions $\{w\}_k^r$ where k is not deducible.*

3. *for any u, v, $<u, v> \notin ST(s_1, \ldots, s_n)$*

4. *for every $i \neq j$, $s_i \not\sqsubseteq_{(s_1,\ldots,s_n)} s_j$*

Then

$$[\![s_1, \ldots, s_n]\!]_\eta \approx [\![\{0^{l(s_1,\eta)}\}_{k_1}^{r_1}, \ldots, \{0^{l(s_n,\eta)}\}_{k_n}^{r_n}]\!]_\eta$$

where $k_1, \ldots, k_n \in \mathcal{N}$ are distinct and $r_1, \ldots, r_n$ are distinct.

Proof:
We proceed by induction on $|s_1| + \ldots + |s_n|$. In the base case, $n = 1$ and $s_1 \in \mathcal{N}$ and this is a consequence of definition 3.15.

If there is no key in the sequence, this is a consequence of lemma 3.12 (with $u_i = 0^{l_i}$). If there is a key in the sequence, which occurs nowhere else, then we can simply break the distribution as a product distribution and apply the induction hypothesis.

Now, assume that $s_1 \in \mathcal{N}$ (hence $L(s_1, \eta) = S$) and $s_2 = \{s_2'\}_{s_1}^{r_1}$. From the last hypothesis of the lemma, r_1 has only one occurrence in the sequence. Moreover, $s_1, s_2', s_3, \ldots, s_n$ satisfy the same hypotheses as $s_1, \ldots, s_n$: we can apply the induction hypothesis, using also $L(s_2, \eta) = L(s_2', \eta)$:

$$[\![s_1, s_2', s_3, \ldots, s_n]\!]_\eta \approx [\![\{0^{l(s_1,\eta)}\}_{k_1}^{r_1}, \ldots, \{0^{l(s_n,\eta)}\}_{k_n}^{r_n}]\!]_\eta$$

By definition 3.15 and since the keys $k_1, \ldots, k_n$ are distinct and $r_1, \ldots, r_n$ are distinct,

$$[\![s_1, s_1, s_2', s_3, \ldots, s_n]\!]_\eta \approx [\![k_1, k_1, \{0^{l(s_2,\eta)}\}_{k_2}^{r_2}, \ldots, \{0^{l(s_n,\eta)}\}_{k_n}^{r_n}]\!]_\eta$$

By lemma 3.6,

$$[\![s_1, s_2, \ldots, s_n]\!]_\eta \approx [\![k_1, \{\{0^{l(s_2,\eta)}\}_{k_2}^{r_2}\}_{k_1}^{r_1}, \ldots, \{0^{l(s_n,\eta)}\}_{k_n}^{r_n}]\!]_\eta$$

Now, using the second property in the definition 3.15: $[\![k_1, \{\{0^{l(s_2,\eta)}\}_{k_2}^{r_2}\}_{k_1}^{r_1}]\!]_\eta \approx [\![k_1, \{0^{l(s_2,\eta)}\}_{k_2}^{r_2}]\!]_\eta$ and since k_1, k_2 occur nowhere else in the right sequence, we get

$$[\![s_1, s_2, \ldots, s_n]\!]_\eta \approx [\![k_1, \{0^{l(s_2,\eta)}\}_{k_2}^{r_2}, \ldots, \{0^{l(s_n,\eta)}\}_{k_n}^{r_n}]\!]_\eta$$

Hence the result by yet another use of the first part of the definition and independence of distributions. $\square$

When $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s_j$ and $s_1, \ldots, s_n$ is a transparent sequence, there are special contexts allowing to access the appropriate subterm of s_j: let the *direct-access contexts* w.r.t. a variable x be the smallest set such that x is a direct access context and if C is a direct-access context w.r.t. x and C' is any context, then $D(C[x, x_1, \ldots, x_n], C'[x, x_1, \ldots, x_n])$, $\pi_1(C[x, x_1, \ldots, x_n])$, $\pi_2(C[x, x_1, \ldots, x_n])$ are direct access contexts w.r.t. x. In other words, direct access contexts decompose their first argument (initially x) at every evaluation step.

Lemma 3.18 *If $s_1, \ldots, s_n$ is a transparent sequence, $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s_j$, then there is a direct access context $C[x_1, \ldots, x_n]$ w.r.t. x_j such that $C[s_1, \ldots, s_n] \downarrow = s_i$.*

Proof:
(Sketch) Let $s_i = s_j|_p$. We prove the lemma by induction on the size of p. $\square$

Lemma 3.19 *If $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_n)$ are valid transparent sequences of terms and $(s_1, \ldots, s_n) \sim_c (t_1, \ldots, t_n)$, then $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s_j$ implies $t_i \sqsubseteq_{(t_1,\ldots,t_n)} t_j$.*

Proof:
By induction on $|s_j|$: if $|s_j| = 1$ then $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s_j$ implies $s_i = s_j$, then, using Eq, $t_i = t_j$, hence $t_i \sqsubseteq_{(s_1,\ldots,s_n)} t_j$.

If $s_j = < s'_j, s''_j >$, then (by m) t_j is a pair $< t'_j, t''_j >$ and

$$(s_1, \ldots, s_{j-1}, s'_j, s''_j, s_{j+1}, \ldots, s_n) \sim_c (t_1, \ldots, t_{j-1}, t'_j, t''_j, t_{j+1}, \ldots, t_n)$$

Either $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s'_j$ or $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s''_j$ and, in both cases, we conclude by induction hypothesis.

If $s_j = \{s'_j\}_{k_j}^{r_j}$. In that case, $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s'_j$. Since the sequence is transparent, according to lemma 3.18, there is a direct access context C_0 w.r.t. x_j such that $C_0[s_1, \ldots, s_n] \downarrow = s_i$. In addition, according to the definition of direct-access-contexts,

$$C_0 = C_1[D(x_j, C_2[x_1, \ldots, x_n]), x_1, \ldots, x_n]$$

where $C_1[x, x_1, \ldots, x_n]$ is a direct-access context w.r.t. x; $C_1[s'_j, s_1, \ldots, s_n] \downarrow = s_i$. Now, by Eq, we must have

$$C_0[t_1, \ldots, t_n] \downarrow = t_i$$

In particular, there is no occurrence of decryption symbols in $C_0[t_1, \ldots, t_n] \downarrow$. This implies that $D(t_j, C_2[t_1, \ldots, t_n]) \downarrow = t'_j \in \mathcal{M}_0$, hence $t_j = \{t'_j\}_{k_j}^{r_j}$. We then apply the induction hypothesis to the sequences $(s'_j, s_1, \ldots, s_n)$ and $(t'_j, t_1, \ldots, t_n)$. Since $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s'_j$, we have $t_i \sqsubseteq_{(t_1,\ldots,t_n)} t'_j \sqsubseteq_{(t_1,\ldots,t_n)} t_j$.

□

Theorem 3.20 *Assuming the encryption scheme is type-1 secure and decryption confusing, if $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$ are valid sequences, then*

$$(s_1, \ldots, s_n) \sim_c (t_1, \ldots, t_n) \Rightarrow [\![s_1, \ldots, s_n]\!]_\eta \approx [\![t_1, \ldots, t_n]\!]_\eta$$

Proof:
(sketch) Note that lemmas 3.12, 3.7 also hold in this framework (they do not use the predicate symbols).

We proceed by induction on $|s_1| + \ldots + |s_n| + |t_1| + \ldots + |t_n|$. As in previous results, using Eq and m, we reduce the general case to the case where $s_1, \ldots, s_n, t_1, \ldots, t_n$ are distinct keys or ciphertexts. Using lemma 3.7, we can also assume that the sequences are transparent.

In summary, we want to eventually applu lemma 3.17. In all situations in which this lemma does not apply, we have a way to distinguish the success of description by comparing with other members of the sequence. Then we may decompose the sequences, apply the induction hypothesis and reconstruct the sequences in that cases, as we did in the proof of theorem 3.10.

If $s_i \sqsubseteq_{(s_1,\ldots,s_n)} s_j$ for some $i \neq j$ (and $s_i \notin \mathcal{W}$), by lemma 3.19, $t_i \sqsubseteq_{(t_1,\ldots,t_n)} t_j$. Let us assume for instance $s_1|_p = s_2$ and therefore $s_1 = \{s'_1\}_{k_1}^{r_1}$ with $s_2 \sqsubseteq_{(s_1,\ldots,s_n)} s'_1$. We also assume w.l.o.g. that s_1 is maximal w.r.t. $\sqsubseteq_{(s_1,\ldots,s_n)}$, in which case t_1 is also maximal w.r.t. $\sqsubseteq_{(t_1,\ldots,t_n)}$, thanks to lemma 3.19. Moreover, thanks to lemma 3.18 (and using Eq), $t_1 = \{t'_1\}_{k'_1}^{r'_1}$ and

$t_2 \sqsubseteq_{(t_1,\ldots,t_n)} t'_1$. We then apply the induction hypothesis replacing s_1 with s'_1, k_1 and t_1 with t'_1, k'_1: k_1, k'_1 are deducible from their respective sequences since the sequences are transparent, hence we get equivalent sequences. Then, since t_1 and s_1 are maximal in their respective sequences and since the terms are distinct, r_1 and r'_1 can only occur once in the sequences. We use now lemma 3.6 to reconstruct the original sequences and get the result.

The other cases: there is a $w \in \mathcal{W} \cap ST(s_1, \ldots, s_n)$ and $w \notin ST(RT(s_1, \ldots, s_n))$, and for some $< u, v >$ and some i, $< u, v > \sqsubseteq_{(s_1,\ldots,s_n)} s_i$ are similar: we decompose the appropriate s_i, making sure that the corresponding decryption of t_i is successful, since it is along the path of a testable term.

Finally, if we are not in any of the above cases, we conclude by lemma 3.17.

□

3.4.5 Summary

Let us summarize the soundness results we got so-far:

security level	type 3	type 2	type 1	type 0	type 1 decryption-confusing
$\mathcal{P}$ containts *at most*	M, EQ, EK, EL	M, EQ, EK	M, EQ, EL, Cipher	M, EQ, Cipher	m, Eq, El

3.5 Discussion

3.5.1 Key cycles

The question of key-cycles is a relevant question, as they may naturally appear, for instance when group keys are used. There are several questions, which are not yet solved (or at least unclear):

1. The example of exercise 18 shows that self encryption is still forbidden. But what about longer key-cycles ?

2. We could take key-cycles into account at the symbolic level, simply adding some power to the symbolic attacker. This is probably the best approach since, when we will come to active attacks, or even when the attacker builds contexts, we cannot simply assume that the adversary refrains from building key cycles.

   But it is unclear what we should put here. For instance, can we infer k from a key cycle $\{k\}_k$? As the example suggests, this should be possible, at least for some encryption schemes. What about other key cycles ?

3. We have seen that the condition that k should not occur in u when u is encrypted with k can be relaxed to $[\![< u, k >]\!]_\eta \approx [\![\{u\}_{k'}]\!]_\eta$ for $k' \notin \mathcal{N}u$. This is further expanded in exercise 30, which gives a generalisation of theorem 3.10 including key cycles. Can we do better ?

3.5.2 Hypotheses on $\mathcal{R}$

In the proof of theorem 3.10, we rely on the particular definition of $\mathcal{R}$. However, this can be abstracted a bit:

- We must show that any reduction on the first sequence can be simulated by a reduction on the second sequence. This is actually only implied by the orthogonality of $\mathcal{R}$: Assume that there is an irreducible context C such that $C[s_1, \ldots, s_n]$ contains a redex, and assume such a minimal context (w.r.t. size). Then $C[s_1, \ldots, s_n] = l\sigma$ for some lhs l of $\mathcal{R}$. And, since C is irreducible, there is a non variable subterm of l, say l_1 such that $l_1\sigma = s_i$. Moreover, by orthogonality, either $C[t_1, \ldots, t_n]$ is irreducible (but in that case $(s_1, \ldots, s_n) \not\sim (t_1, \ldots, t_n)$ since $C[s_1, \ldots, s_n] \downarrow \in \mathcal{M}$ and $C[t_1, \ldots, t_n] \downarrow \notin \mathcal{M}$) or else $C[t_1, \ldots, t_n] = l\theta$.
- Second, we used an induction, whose base case is lemma 3.9 and induction step is derived from lemma 3.6. We need only to consider any well-founded ordering (and its multiset extension) such that, if $C[s_1, \ldots, s_n] = l\sigma$ for some irreducible context and irreducible $s_1, \ldots, s_n$, then there are contexts $C_1, \ldots, C_m$ such that, for every j, $C_j[s_1, \ldots, s_m] = l_j\sigma_j$ and there is an index i such that
  1. $\forall j = 1..m, s_i > r_j\sigma_j$
  2. $\exists C' \in \mathcal{C}, s_i = C'[s_1, \ldots, s_{i-1}, r_1\sigma_1, \ldots, r_m\sigma_m, s_{i+1}, \ldots, s_n] \downarrow$
- For the base case (lemma 3.9), we rely more heavily on the particular instance of our problem: that is the only place where we actually use the set $\mathcal{P}_3$ (except that, in the theorem, we assumed EQ in the set of predicates). On the other hand, this lemma holds for a much more general family of rewrite systems $\mathcal{R}$: we used only the fact that, for η sufficienty large, there is at least one bitstring of length $l(u, \eta)$ which does not occur as a member of $\mathcal{R}$.

3.5.3 Relevance of other hypotheses

Is it relevant to assume that the length of $\{u\}_k$ only depend on the length of u and the length of k ? If not, how should we modify the proof ?

Is the "same-key" predicate relevant ? It implies for instance that ciphertexts can be distinguished from random sequences of the appropriate length. An alternative is to put such a typing predicate instead.

3.5.4 Composed keys

In [27], the case of composed keys is considered and a result extending [4] is proposed. The computational interpretation of composed keys must however use a random oracle: the composed keys are simply redistributed according to a uniform distribution, hiding their possible relationships.

It seems difficult to avoid the random oracle in case of compound keys, as shown by the following

Exercise 28
Assume that $(G, \mathcal{E})$ is a type-0 secure encryption scheme Construct another type-0 secure encryption scheme and a term u not containing k_1 such that, for names k_1, k_2, k_3, $[\![\{k_1\}_u, k_2, k_1]\!] \not\approx [\![\{k_1\}_u, k_2, k_3]\!]$

Either the encyption scheme has to hide the structure of the key (this is the approach of [27]) or the symbolic equivalence has to distinguish between keys that have a shared structure.

The latter has not been investigated so far. Is it worth to investigate ?

4 Completeness

Completeness states that computational indistinguishability implies symbolic equivalence. This means that, in the set of predicates and in the rewrite system, we only put predicates and computations that can be performed by an adversary.

If we put "same key" as a predicate in the symbolic equivalence, then the encryption scheme must not conceal the keys: it should be possible for an adversary, given two cypher-texts, to decide if they are encrypted with the same key.

To get both soundness and completeness, must know the exact level of security of the encryption scheme. Typically, we prove here that, if the encryption scheme is *not* type-1 secure and *not* type 2 secure, then the completeness result holds. We need also to be able to implement the partial functions in $\mathcal{F}_P$: retrieve the two components of a pair for instance.

For completeness, we do not need any hypotheses on key-cycles or level of security... The sequences $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$ may be non-valid sequences. In that case, we need of course to have a computational interpretation of decryption of any bit-string with any bit-string (possibly yielding a special failure notice) and, more generally, an interpretation of any function symbol in $\mathcal{F}_P$ and any predicate symbol in $\mathcal{P}$.

4.1 Computational algebras and computational structures

Definition 4.1 *A $\mathcal{F}, \mathcal{F}_P, \mathcal{P}$-*computational algebra *consists of*

- *for each $f \in \mathcal{F}_P$ a Polynomial Time deterministic Turing machine $[\![f]\!]^{\mathcal{A}}$ with as many entries as the arity of f,*
- *for each $f \in \mathcal{F} \setminus (\mathcal{F}_P \cup \mathcal{N} \cup \mathcal{R})$, a function $[\![f]\!]^{\mathcal{A}}$ from bitstrings to bitstring, with as many entries as the arity of f.*
- *for each $P \in \mathcal{P}$, a Polynomial Time deterministic Turing machine $[\![P]\!]^{\mathcal{A}}$ with as many entries as the arity of P and output 0/1*

In this definition, $[\![f]\!]^{\mathcal{A}}$, $[\![P]\!]^{\mathcal{A}}$,... are not randomized since they get an explicit argument for the possible random input. But we can also allow them to be PPT and possibly hide some random input.

Given a computational algebra $\mathcal{A}$, for every $s \in \mathcal{M}$ and $\eta \in \mathbb{N}$ and τ an assignment of $\mathcal{N}(s) \cup \mathsf{Coins}(s)$, $[\![s]\!]_\eta^{\mathcal{A},\tau}$ is defined by induction on the structure of s: a computational algebra is a $\mathcal{F}$-algebra. This generalizes the definition of computational intepretation of terms of section 3.1: $[\![s]\!]_\eta^{\mathcal{A}}$ is the distribution obtained by first drawing keys and coins (according to the appropriate distributions depending on η), yielding an interpretation τ, then computing $[\![s]\!]_\eta^{\mathcal{A},\tau}$.

Exercise 29
What would happen if we let $[\![f]\!]$ depend on η ?

Definition 4.2 *Given a computational algebra $\mathcal{A}$, a predicate $P \in \mathcal{P}$ of arity k is* implementable *if*

$$\forall s_1, \ldots, s_k \in \mathcal{M}, \exists Q \in \boldsymbol{Pol}_1, \exists N \in \mathbb{N}, \forall \eta > N.$$
$$\boldsymbol{Pr}\{(x_1, \ldots, x_k) \xleftarrow{R} [\![s_1, \ldots, s_k]\!]_\eta : [\![P]\!]^{\mathcal{A}}(x_1, \ldots, x_k) = P^I(s_1, \ldots, s_k)\} > \frac{1}{2} + \frac{1}{Q(\eta)}$$

Definition 4.3 *Given a convergent rewrite system $\mathcal{S}$ on $T(\mathcal{F})$, and an interpretation of the predicate symbols, a $\mathcal{S}, \mathcal{P}^I$-*computational structure *is a computational algebra such that, moreover,*

$$\forall s, t \in \mathcal{M}, \forall \eta \in \mathbb{N}, \forall \tau. \; (s =_{\mathcal{S}} t \;\; \Rightarrow \;\; [\![s]\!]_\eta^{\mathcal{A},\tau} = [\![t]\!]_\eta^{\mathcal{A},\tau})$$

and, for every $P \in \mathcal{P}$, P is implementable.

Note here that we require not only indistinguishability, but true equality: τ is universally quantified over the assignments of appropriate lengths.

4.2 Completeness

Theorem 4.4 (completeness) *If $\mathcal{A}$ is a $\mathcal{S}, \mathcal{F}, \mathcal{F}_P, \mathcal{P}$-computational algebra, then*

$$[\![s_1, \ldots, s_n]\!]_\eta^{\mathcal{A}} \approx [\![t_1, \ldots, t_n]\!]_\eta^{\mathcal{A}} \Rightarrow (s_1, \ldots, s_n) \sim (t_1, \ldots, t_n).$$

Proof:
By definition, if $(s_1, \ldots, s_n) \not\sim (t_1, \ldots, t_n)$, then there are contexts $C_1, \ldots, C_k \in \mathcal{C}$ and a predicate $P \in \mathcal{P}$ such that $P^I(C_1[s_1, \ldots, s_n] \downarrow, \ldots, C_k[s_1, \ldots, s_n] \downarrow) \not\Leftrightarrow P^I(C_1[t_1, \ldots, t_n] \downarrow, \ldots, C_k[t_1, \ldots, t_n] \downarrow)$. For instance, assume w.l.o.g. that $P^I(C_1[s_1, \ldots, s_n] \downarrow, \ldots, C_k[s_1, \ldots, s_n] \downarrow) = 1$ and $P^I(C_1[t_1, \ldots, t_n] \downarrow, \ldots, C_k[t_1, \ldots, t_n] \downarrow) = 0$.

Moreover, $C_1, \ldots, C_k$ only contain symbols in $\mathcal{F}_P$. Therefore, (using a simple induction on the length of the rewrite sequence and the first part of definition **??**), there is a deterministic polynomial time Turing machine $\mathcal{B}$ (whose size maybe non-polynomial), which, on input $x_1, \ldots, x_n$, computes $[\![P(C_1[x_1, \ldots, x_n] \downarrow, \ldots, C_k[x_1, \ldots, x_n] \downarrow)]\!]^{\mathcal{A}}$. By definition, for every $\eta \in \mathbb{N}$, for every assignment τ of keys and random numbers occcurring in $u_1, \ldots, u_n$,

$$\mathcal{B}([\![u_1]\!]_\eta^{\mathcal{A},\tau}, \ldots, [\![u_n]\!]_\eta^{\mathcal{A},\tau}) = [\![P]\!]^{\mathcal{A}}([\![C_1[u_1, \ldots, u_n] \downarrow]\!]_\eta^{\mathcal{A},\tau}, \ldots [\![C_k[u_1, \ldots, u_n] \downarrow]\!]_\eta^{\mathcal{A},\tau})$$

So,

$$\begin{aligned} &\mathbf{Pr}\{(x_1, \ldots, x_n) \xleftarrow{R} [\![u_1, \ldots, u_n]\!]_\eta^{\mathcal{A}} : \mathcal{B}(x_1, \ldots, x_n) = 1\} = \\ &\qquad \mathbf{Pr}\{(y_1, \ldots, y_k) \xleftarrow{R} [\![C_1[u_1, \ldots, u_n] \downarrow, \ldots, C_k[u_1, \ldots, u_n] \downarrow]\!]_\eta^{\mathcal{A}} : [\![P]\!]^{\mathcal{A}}(y_1, \ldots, y_k) = 1\} \end{aligned}$$

On the other hand, according to the definition, there are Q_1 and N_1 such that, for all $\eta > N_1$,

$$\mathbf{Pr}\{(y_1, \ldots, y_k) \xleftarrow{R} [\![C_1[s_1, \ldots, s_n] \downarrow, \ldots, C_k[s_1, \ldots, s_n] \downarrow]\!]_\eta^{\mathcal{A}} : [\![P]\!]^{\mathcal{A}}(y_1, \ldots, y_k) = 1\} > \frac{1}{2} + \frac{1}{Q_1(\eta)}$$

and there are Q_2 and N_2 such that, for all $\eta > N_2$,

$$\mathbf{Pr}\{(y_1, \ldots, y_k) \xleftarrow{R} [\![C_1[t_1, \ldots, t_n] \downarrow, \ldots, C_k[t_1, \ldots, t_n] \downarrow]\!]_\eta^{\mathcal{A}} : [\![P]\!]^{\mathcal{A}}(y_1, \ldots, y_k) = 0\} > \frac{1}{2} + \frac{1}{Q_2(\eta)}$$

Altogether, if we let

$$\begin{aligned} &\epsilon(\eta) = \\ &\qquad \mathbf{Pr}\{(x_1, \ldots, x_n) \xleftarrow{R} [\![s_1, \ldots, s_n]\!]_\eta^{\mathcal{A}} : \mathcal{B}(x_1, \ldots, x_n) = 1\} - \\ &\qquad\quad \mathbf{Pr}\{(x_1, \ldots, x_n) \xleftarrow{R} [\![t_1, \ldots, t_n]\!]_\eta^{\mathcal{A}} : \mathcal{B}(x_1, \ldots, x_n) = 1\} \end{aligned},$$

For $\eta > \max(N_1, N_2)$,

$$\epsilon(\eta) > (\frac{1}{2} + \frac{1}{Q_1(\eta)}) - (\frac{1}{2} - \frac{1}{Q_2(\eta)}) = \frac{1}{Q_1(\eta)} + \frac{1}{Q_2(\eta)}$$

It follows that $(\llbracket s_1, \ldots, s_n \rrbracket_\eta^{\mathcal{A}})_{\eta \in \mathbb{N}} \not\approx (\llbracket t_1, \ldots, t_n \rrbracket_\eta^{\mathcal{A}})_{\eta \in \mathbb{N}}$.

□

4.3 Examples of computational structures

In the previous theorem, we actually gave the definitions, which exactly entail the desired conclusions; we got the converse of the table of section 3.4.5 for completeness results.

security level	type 3	type 2	type 1	type 0
$\mathcal{A}$ implements *at least*	M, EQ, EK, EL	M, EQ, EK	$M, EQ, EL, \mathsf{Cipher}$	M, EQ, Cipher

So, we pushed the difficulty in proving that a given interpretation is a computational algebra indeed.

4.3.1 Confusion freeness

M, the predicate defining "valid messages" in section 2.4.3 can be implemented if there is an algorithm, which can recognize when a message is a pair (which we always assume), an algorithm which can recognize when a message is a key, and also an algorithm, which decides when the decryption algorithm is used with a valid input.

More precisely, following [31], we assume a particular bitstring $\perp$ (an error message), which is returned when π_i is applied on a bitstring, which is not a pair, or when there is an attempt to encrypt a message, which is not a key or when trying to decrypt something, which is not a ciphertext. Following a strict interpretation, we also assume that any function applied to the error message $\perp$ returns $\perp$. This corresponds actually to a typing predicate, which we assume to be implemented deterministically, for instance using tags (it would also work with a probabilistic implementation of such predicates). Finally, we assume *confusion freeness* as defined below:

Definition 4.5 ([31]) *An encryption scheme is* confusion free *if, for every bitstring x,*

$$\boldsymbol{Pr}\{k_1, k_2 \xleftarrow{R} G_\eta, r \xleftarrow{R} U : \mathcal{D}(\mathcal{E}(x, k_1 \mid r), k_2) \neq \perp\} = \nu(\eta)$$

is negligible.

In other words: it is very likely to get an error message when trying to decrypt with a wrong key.

Proposition 4.6 *If the encryption scheme is confusion-free, then the predicates M, EQ are implementable.*

Proof:
Let us start with M. $[\![M]\!]^{\mathcal{A}}(x) = 1$ iff $x \neq \perp$.

Let $s \in \mathcal{M}$. We have to compute

$$\epsilon(s,\eta) \stackrel{\text{def}}{=} \mathbf{Pr}\{x \stackrel{R}{\longleftarrow} [\![s]\!]_\eta : M^I(s) \neq [\![M]\!]^{\mathcal{A}}(x)\}$$

If $s \in \mathcal{M}_0$, then $M^I(s) = 1$ and $[\![M]\!]^{\mathcal{A}}(x) = 1$ and therefore $\epsilon(s,\eta) = 0$. So, we only have to consider the case $M^I(s) = 0$. We proceed by induction on s. If s is a constant, then the only possibility for not being accepted by M^I is $s = \perp$, in which case $[\![M]\!]^{\mathcal{A}}(x) = 0$.

Otherwise, if s is a pair or an encryption with a valid key, then, since M^I has a strict interpretation, there must be a direct subterm of s which does not belong to $\mathcal{M}_0$. Then, we use the induction hypothesis and the strictness of $[\![M]\!]^{\mathcal{A}}$: if s_1, s_2 are the direct subterms of s,

$$\begin{aligned}
\epsilon(s,\eta) &= \mathbf{Pr}\{(x_1,x_2) \stackrel{R}{\longleftarrow} [\![s_1,s_2]\!]_e ta : M^{\mathcal{A}}(x_1) = 1 \wedge M^{\mathcal{A}}(x_2) = 1\} \\
&\leq \min(\mathbf{Pr}\{(x_1,x_2) \stackrel{R}{\longleftarrow} [\![s_1,s_2]\!]_e ta : M^{\mathcal{A}}(x_1) = 1\}, \mathbf{Pr}\{(x_1,x_2) \stackrel{R}{\longleftarrow} [\![s_1,s_2]\!]_e ta : M^{\mathcal{A}}(x_2) = 1\}) \\
&= \min(\mathbf{Pr}\{x_1 \stackrel{R}{\longleftarrow} [\![s_1]\!]_e ta : M^{\mathcal{A}}(x_1) = 1\}, \mathbf{Pr}\{x_2 \stackrel{R}{\longleftarrow} [\![s_2]\!]_e ta : M^{\mathcal{A}}(x_2) = 1\}) \\
&\leq \min(\epsilon(s_1,\eta), \epsilon(s_2,\eta))
\end{aligned}$$

If $s = \{u\}_v^r$ and v is not a valid key or if s is a projection of a term which is not a pair, or if s is a decryptoin of a term which is not a ciphertext, then $[\![s]\!]_\eta$ is always $\perp$, by definition, hence $[\![M]\!]^{\mathcal{A}}(x) = 0$ (for all $x \in [\![s]\!]_\eta$).

Finally, if $s = \mathcal{D}(\{t\}_{k_1}^r, k_2)$, if $\{t\}_{k_1}^r$ is not in M^I, we are back to the previous computation. Assume now that $\{t\}_{k_1}^r \in M^I$ and therefore that $k_2 \neq k_1$. Then, by definition of confusion freeness, $\epsilon(s,\eta) = \nu(\eta)$

Next, $[\![EQ]\!]^{\mathcal{A}}(x,y)$ is defined as $[\![M]\!]^{\mathcal{A}}(x) \wedge [\![M]\!]^{\mathcal{A}}(y) \wedge x = y$, which is implementable, as $M^{\mathcal{A}}$ is implementable. □

Corollary 4.7 (completeness for type-0) *If the encryption scheme is confusion-free, then* $[\![s_1,\ldots,s_n]\!]_\eta \approx [\![t_1,\ldots,t_n]\!]_\eta \Rightarrow (s_1,\ldots,s_n) \sim_0 (t_1,\ldots,t_n)$.

4.3.2 Equal keys and Equal lengths

In order to get completeness results for type 1,2,3 security, it suffices to have implementations of EL and/or EK.

4.3.3 Decryption confusing

For such schemes, M is no longer implementable. However m, Eq, El are implementable in a straightforward way

4.3.4 Summary

We get the following completeness results:

Security level	0	1	2	3	DC-1
Implementable predicates	M, EQ, Cipher	M,EQ, EL, Cipher	M, EQ, EK	M, EQ, EK, EL	m, Eq, El

Acknowledgments

I would thank Steve Kremer, David Nowak, Nicolas Perrin and all researchers attending the lectures for their comments, which helped improving these notes.

References

[1] M. Abadi, M. Baudet, and B. Warinschi. Guessing attacks and the computational soundness of static equivalence. In L. Aceto and A. Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 2006.

[2] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

[3] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148(1), 1999.

[4] M. Abadi and P. Rogaway. Reconciling two views of cryptography: the computational soundness of formal encryption. In *Proc. 1rst IFIP International Conference on Theoretical Computer Science*, volume 1872 of *Lecture Notes in Computer Science*, Sendai, Japan, 2000.

[5] P. Adao, G. Bana, J. Herzog, and A. Scedrov. Soundness and completeness of formal encryption: the cases of key-cycles and partial information leakage. *Journal of Computer Security*, 2007. to appear.

[6] M. Backes, M. Drmuth, and R. Ksters. On simulatability soundness and mapping soundness of symbolic cryptography. In *Proceedings of 27th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, December 2007.

[7] M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable dolev-yao style cryptographic library. In *Proc. IEEE Computer Security Foundations workshop*, 2004.

[8] M. Backes and B. Pfitzmann. Limits of the cryptographic realization of Dolev-Yao style XOR. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS)*, 2005.

[9] M. Backes, B. Pfitzmann, and A. Scedrov. Key-dependent message security under active attacks. brsim/uc-soundness of symbolic encryption with key cycles. In *Proc. IEEE Computer Security Foundations*, 2007.

[10] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Concerence on Computer and Communications Security (CCS'03)*, 2003.

[11] M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (rsim) framework for asynchronous systems. To appear in *Information and Computation*, 2007.

[12] G. Bana. Soundness and completeness of formal logics of symmetric encryption. Cryptology ePrint Archive, Report 2005/101, 2005. `http://eprint.iacr.org/`.

[13] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 652–663, Lisboa, Portugal, July 2005. Springer.

[14] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Proc. ASIACRYPT'01*, volume 2248 of *Lecture Notes in Computer Science*, 2001.

[15] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: security proofs and improvements. In *Proc. Advances in Cryptology– EUROCRYPT'00*, volume 1807 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[16] M. Bellare and P. Rogaway. Introduction to modern cryptography. lecture notes, Univ. Calif. San Diego, 2005.

[17] R. Canetti. Universal composable security: a new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[18] Y. Chevalier and M. Rusinowitch. Hierarchical combination of intruder theories. In *Proc. Rewriting Techniques and Application*, volume 4098 of *Lecture Notes in Computer Science*, pages 108–122, 2006.

[19] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171, 2005.

[20] A. Datta, A. Derek, J. Mitchell, A. Ramanathan, and A. Scedrov. Games and the impossibility of realizable ideal functionality. In *Proc. 3rd Theory of Cryptography conference (TCC)*, 2006.

[21] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.

[22] D. Dolev and A. Yao. On the security of public key protocols. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 350–357, 1981.

[23] F. T. Fabrega, J. Herzog, and J. Guttman. Strand spaces: Proving security protocol correct. *Journal of Computer Security*, 7:191–230, 1999.

[24] O. Goldreich. *The foundations of cryptography - Volume 2*. Cambridge University Press, 2004.

[25] S. Goldwasser and M. Bellare. Lecture notes on cryptography, MIT, 2001.

[26] S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In M. Sagiv, editor, *Programming Languages and Systems — Proceedings of the 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, Edinburgh, U.K., Apr. 2005. Springer.

[27] P. Laud and R. Corin. Sound computational interpretation of formal encryption with composed keys. In *Proc. 6th International Conference on Information Security and Cryptology*, volume 2971 of *Lecture Notes in Computer Science*, 2003.

[28] Y. Lindell. General composition and universal composition in secure multiparty computation. In *Proc. 44th IEEE Symp. Foundations of Computer Science (FOCS)*, 2003.

[29] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In Margaria and Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166, 1996.

[30] C. Meadows. The nrl protocol analyzer: an overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[31] D. Micciancio and B. Warinschi. Completeness theorems for the abadi-rogaway language of encrypted expressions. *Journal of Computer Security*, 2004.

[32] D. Micciancio and B. Warinschi. Soundness of formal encryption in presence of an active attacker. In *Proc. Theory of Cryptography Conference (TCC'04)*, volume 2951 of *LNCS*, 2004.

[33] J. Mitchell, A. Ramanathan, and V. Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theoretical Comput. Sci.*, 353:118–164, 2006.

[34] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.