Automated Proofs for Asymmetric Encryption

J. Courant, M. Daubignard, C. Ene, P. Lafourcade and Y. Lakhnech *

Université Grenoble 1, CNRS,Verimag

Abstract. Chosen-ciphertext security is by now a standard security property for asymmetric encryption. Many generic constructions for building secure cryptosystems from primitives with lower level of security have been proposed. Providing security proofs has also become standard practice. There is, however, a lack of automated verification procedures that analyze such cryptosystems and provide security proofs. This paper presents an automated procedure for analyzing generic asymmetric encryption schemes in the random oracle model. It has been applied to several examples of encryption schemes among which the construction of Bellare-Rogaway 1993, of Pointcheval at PKC'2000 and REACT.

1 Introduction

Our day-to-day lives increasingly depend upon information and our ability to manipulate it securely. This requires solutions based on cryptographic systems (primitives and protocols). In 1976, Diffie and Hellman invented public-key cryptography, coined the notion of one-way functions and discussed the relationship between cryptography and complexity theory. Shortly after, the first cryptosystem with a reductionist security proof appeared (Rabin 1979). The next breakthrough towards formal proofs of security was the adoption of computational security for the purpose of rigorously defining the security of cryptographic schemes. In this framework, a system is *provably secure* if there is a polynomial-time reduction proof from a hard problem to an attack against the security of the system. The provable security framework has been later refined into *the exact (also called concrete) security framework* where better estimates of the computational complexity of attacks are achieved. While research in the field of provable cryptography has achieved tremendous progress towards rigorously defining the functionalities and requirements of many cryptosystems, little has been done for developing computer-aided proof methods or more generally for investigating a proof theory for cryptosystems as it exists for imperative programs, concurrent systems, reactive systems, etc...

In this paper, we present an automated proof method for analyzing generic asymmetric encryption schemes in the random oracle model (ROM). Generic encryption schemes aim at transforming schemes with weak security properties, such as one-wayness, into schemes with stronger security properties, especially security against chosen ciphertext attacks. Examples of generic encryption schemes are [11, 22, 20, 7, 5, 18, 17, 16]. The paper contains two main contributions. The first one is a compositional Hoare logic for proving IND-CPA-security. That is, we introduce a simple programming language (to specify encryption algorithms that use one-way functions and hash functions) and an assertion language that allows to state invariants and axioms and rules to establish such invariants. Compositionality of the Hoare logic means that the reasoning follows the structure of the program that specifies the encryption oracle. The assertion language consists of three atomic predicates. The first predicate allows us to express

* Grenoble, email:name@imag.fr This work has been partially supported by the ANR projects SCALP, AVOTE and SFINCS

that the value of a variable is indistinguishable from a random value even when given the values of a set of variables. The second predicate allows us to state that it is computationally infeasible to compute the value of a variable given the values of a set of variables. Finally, the third predicate allows us to state that the value of a variable has not been submitted to a hash function.

Transforming the Hoare logic into an (incomplete) automated verification procedure is quite standard. Indeed, we can interpret the logic as a set of rules that tell us how to propagate the invariants backwards. We have done this for our logic resulting in a verification procedure implemented in less than 250 lines of CAML. We have been able to automatically verify IND-CPA security of several schemes among which [7, 17, 16]. Our Hoare logic is incomplete for two main reasons. First, IND-CPA security is an observational equivalence-based property, while with our Hoare logic we establish invariants. Nevertheless, as shown in Proposition 1, we can use our Hoare logic to prove IND-CPA security at the price of completeness. That is, we prove a stronger property than IND-CPA. The second reason, which we think is less important, is that for efficiency reasons some axioms are stronger than needed.

The second contribution of the paper presents a simple criterion for plaintext awareness (PA). Plaintext awareness has been introduced by Bellare and Rogaway in [5]. It has then been refined in [4] such that if an encryption scheme is PA and IND-CPA then it is IND-CCA. Intuitively, PA ensures that an adversary cannot generate a valid cipher without knowing the plaintext, and hence, the decryption oracle is useless for him. The definition of PA is complex and proofs of PA are also often complex. In this paper, we present a simple syntactic criterion that implies plaintext awareness. Roughly speaking the criterion states that the cipher should contain as a sub-string the hash of a bitstring that contains as substrings the plaintext and the random seed. This criterion applies for many schemes such as [7, 16, 17] and easy to check. Although (or maybe because) the criterion is simple, the proof of its correctness is complex.

Putting together these two contributions, we get a proof method for IND-CCA security.

An important feature of our method is that it is not based on a global reasoning and global program transformation as it is the case for the game-based approach [6, 19]. Indeed, both approaches can be considered complementary as the Hoare logic-based one can be considered as aiming at characterizing, by means of predicates, the set of contexts in which the game transformations can be applied safely.

Related work We restrict our discussion to work providing computational proofs for cryptosystems. In particular, this excludes symbolic verification (including ours). We mentioned above the game-based approach [6, 19, 15]. In [8, 9] B. Blanchet and D. Pointcheval developed a dedicated tool, CryptoVerif, that supports security proofs within the game-based approach. CryptoVerif is based on observational equivalence. The equivalence relation induces rewriting rules applicable in contexts that satisfy some properties. Invariants provable in our Hoare logic can be considered as logical representations of these contexts. Moreover, as we work with invariants, that is we follow a state-based approach, we need to prove results that link our invariants to game-based properties such as indistinguishability (cf. Proposition 1 and 3). Our verification method is fully automated. It focusses on asymmetric encryption in the random oracle model, while CryptoVerif is potentially applicable to any cryptosystem.

G. Barthe and S. Tarento were among the first to provide machine-checked proofs of cryptographic schemes without relying on the perfect cryptography hypothesis. They formalized the Generic Model and the Random Oracle Model in the Coq proof assistant, and used this formal-

ization to prove hardness of the discrete logarithm [1], security of signed ElGamal encryption against interactive attacks [3], and of Schnorr signatures against forgery attacks [21]. They are currently working on formalizing the game-based approach in Coq [2]. D. Nowak provides in [**?**] an implementation in Coq of the game-based approach. He illustrates his framework by a proof of the semantic security of the encryption scheme ElGamal and its hashed version. Another interesting work is the Hoare-style proof system proposed by R. Corin and J. Den Hartog for game-based cryptographic proofs [10]. The main difference between our logic and theirs is that our assertion language does not manipulate probabilities explicitly and is at a higher level of abstraction. On the other hand, their logic is more general. In [12], Datta et al. present a computationally sound compositional logic for key exchange protocols. There is, however, no proof assistance provided for this logic neither.

Outline: In Section 2, we introduce notations used for defining our programming language and generic asymmetric encryption schemes. In Section 3, we present our method for proving IND-CPA security. In Section **??** we introduce a criterion to prove plaintext awareness. In Section 5 we explain the automated verification procedure derived from our Hoare logic. Finally, in Section 6 we conclude.

2 Definitions

We are interested in analyzing generic schemes for asymmetric encryption assuming ideal hash functions. That is, we are working in the *random oracle model* [13, 7]. Using standard notations, we write $H \xleftarrow{r} \Omega$ to denote that H is randomly chosen from the set of functions with appropriate domain. By abuse of notation, for a list $\boldsymbol{H} = H_1, \cdots, H_n$ of hash functions, we write $\boldsymbol{H} \xleftarrow{r} \Omega$ instead of the sequence $H_1 \xleftarrow{r} \Omega, \ldots, H_n \xleftarrow{r} \Omega$. We fix a finite set $\mathcal{H} = \{H_1, \ldots, H_n\}$ of hash functions and also a finite set Π of trapdoor permutations and $\mathcal{O} = \Pi \cup \mathcal{H}$. We assume an arbitrary but fixed ordering on Π and $\mathcal{H}$; just to be able to switch between set-based and vector-based notation. A *distribution ensemble* is a countable sequence of distributions $\{X_\eta\}_{\eta \in \mathbb{N}}$. We only consider distribution ensembles that can be constructed in polynomial time by probabilistic algorithms that have oracle access to $\mathcal{O}$. Given two distribution ensembles $X = \{X_\eta\}_{\eta \in \mathbb{N}}$ and $X' = \{X'_\eta\}_{\eta \in \mathbb{N}}$, an algorithm $\mathcal{A}$ and $\eta \in \mathbb{N}$, we define the *advantage* of $\mathcal{A}$ in distinguishing X_η and X'_η as the following quantity:

$$\mathsf{Adv}(\mathcal{A}, \eta, X, X') = \\ \mathsf{Pr}[x \xleftarrow{r} X_\eta : \mathcal{A}^{\mathcal{O}}(x) = 1] - \mathsf{Pr}[x \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathcal{O}}(x) = 1].$$

We insist, above, that for each hash function H, the probabilities are also taken over the set of maps with the appropriate type. Let $\mathsf{Adv}(\eta, X, X') = \sup_{\mathcal{A}}(\mathsf{Adv}(\mathcal{A}, \eta, X, X'))$, the maximal advantage taken over all probabilistic polynomial-time algorithms. Then, two distribution ensembles X and X' are called *indistinguishable* if $\mathsf{Adv}(\eta, X, X')$ is negligible as a function of η and denoted by $X \sim X'$. In other words, for any polynomial-time (in η) probabilistic algorithm $\mathcal{A}$, $\mathsf{Adv}(\mathcal{A}, \eta, X, X')$ is negligible as a function of η. We insist that all security notions we are going to use are in the ROM, where all algorithms, including adversaries, are equipped with oracle access to the hash functions.

2.1 A simple programming language for encryption and decryption oracles

We introduce a simple programming language without loops in which the encryption and decryption oracles are specified. The motivation for fixing a notation is obvious: it is mandatory for developing an automatic verification procedure. Let Var be an arbitrary finite non-empty set of variables. Then, our programming language is built according to the following BNF described in Table 1, where for a bit-string $bs = b_1 \dots b_k$ (b_i are bits), $bs[n, m] = b_n \dots b_m$[1], and $\mathcal{N}$ is the name of the oracle, c its body and x and y are the input and output variable respectively. Note the command $y[n, m]$ is only used in the decryptions, it is why we do not have to consider it in our Hoare logic. With this language we can sample an uniform value to x, apply a one-way function f and its inverse f^{-1}, a hash function, the exclusive-or, the concatenation and substring function, and perform an "if-then-else" (used only in the decryption function).

$$
\begin{array}{rl}
\text{Command} & \mathsf{c} ::= x \xleftarrow{r} \mathcal{U} \mid x := f(y) \mid x := f^{-1}(y) \mid x := H(y) \mid x := y[n, m] \\
& \quad \mid x := y \oplus z \mid x := y||z \mid \mathsf{if}\ x = y\ \mathsf{then\ c1\ else\ c2\ fi} \mid c; c \\
\text{Oracle declaration} & \mathcal{O} ::= \mathcal{N}(x, y) : \mathsf{c}
\end{array}
$$

Table 1. Language grammar.

Example 1. The following command encodes the encryption scheme proposed by Bellare and Rogaway in [7] (shortly $\mathcal{E}(\mathrm{in}_e; \mathrm{out}_e) = f(r)||\mathrm{in}_e \oplus G(r)||H(\mathrm{in}_e||r)$):

$$
\begin{array}{l}
\mathcal{E}(\mathrm{in}_e, \mathrm{out}_e) : \\
r \xleftarrow{r} \{0,1\}^{\eta_0};\ a := f(r);\ g := G(r); \\
b := \mathrm{in}_e \oplus g;\ s := \mathrm{in}_e||r;\ c := H(s); \\
u := a||b||c;\ \mathrm{out}_e := u; \\
\text{where } f \in \Pi \text{ and } G, H \in \mathcal{H}.
\end{array}
$$

Semantics: In addition to the variables in Var, we consider variables $\mathbb{T}_{H_1}, \dots, \mathbb{T}_{H_n}$. Variable $\mathbb{T}_{H_i}$ records the queries to the hash function H_i and *can not be accessed by the adversary.* Thus, we consider states that assign bit-strings to the variables in Var and lists of pairs of bit-strings to $\mathbb{T}_{H_i}$. For simplicity of the presentation, we assume that all variables range over large domains, whose cardinalities are exponential in the security parameter η. $u \xleftarrow{r} \mathcal{U}$ is the uniform sampling of a value u from the appropriate domain. Given a state S, $S(\mathbb{T}_H).\mathsf{dom}$, respectively $S(\mathbb{T}_H).\mathsf{res}$, denotes the list obtained by projecting each pair in $S(\mathbb{T}_H)$ to its first, respectively second, element.

A program takes as input a *configuration* $(S, \boldsymbol{H}, (f, f^{-1}))$ and yields a distribution on configurations. A configuration is composed of a state S, a vector of hash functions $(H_1, \dots, H_n)$ and a pair (f, f^{-1}) of a trapdoor permutation and its inverse. Let Γ denote the set of configurations and $\textsc{Dist}(\Gamma)$ the set of distributions on configurations. The semantics is given in Table 2, where $\delta(x)$ denotes the Dirac measure, i.e. $\mathsf{Pr}(x) = 1$. Notice that the semantic function of commands can be lifted in the usual way to a function from $\textsc{Dist}(\Gamma)$ to $\textsc{Dist}(\Gamma)$. By abuse of notation we also denote the lifted semantics by $[\![\mathsf{c}]\!]$.

[1] Notice that $bs[n, m] = \epsilon$, when $m < n$ and $bs[n, m] = bs[n, k]$, when $m > k$

$$
\begin{array}{l}
[\![x \xleftarrow{r} \mathcal{U}]\!](S, \boldsymbol{H}, (f, f^{-1})) = [u \xleftarrow{r} \mathcal{U} : (S\{x \mapsto u\}, \boldsymbol{H}, (f, f^{-1}))] \\
[\![x := f(y)]\!](S, \boldsymbol{H}, (f, f^{-1})) = \delta(S\{x \mapsto f(S(y))\}, \boldsymbol{H}, (f, f^{-1})) \\
[\![x := f^{-1}(y)]\!](S, \boldsymbol{H}, (f, f^{-1})) = \delta(S\{x \mapsto f^{-1}(S(y))\}, \boldsymbol{H}, (f, f^{-1})) \\
[\![x := y[n, m]]\!](S, \boldsymbol{H}, (f, f^{-1})) = \delta(S\{x \mapsto S(y)[n, m]\}, \boldsymbol{H}, (f, f^{-1})) \\
[\![x := H(y)]\!](S, \boldsymbol{H}, (f, f^{-1})) = \\
\quad \begin{cases} \delta(S\{x \mapsto v\}, \boldsymbol{H}, (f, f^{-1})) & ; \text{if } (S(y), v) \in \mathbb{T}_H \\ \delta(S\{x \mapsto v, \mathbb{T}_H \mapsto S(\mathbb{T}_H) \cdot (S(y), v)\}, \boldsymbol{H}, (f, f^{-1})) \; ; & \\ \qquad \text{if } (S(y), v) \notin \mathbb{T}_H \text{ and } v = \boldsymbol{H}(H)(S(y)) \end{cases} \\
[\![x := y \oplus z]\!](S, \boldsymbol{H}, (f, f^{-1})) = \delta(S\{x \mapsto S(y) \oplus S(z)\}, \boldsymbol{H}, (f, f^{-1})) \\
[\![x := y||z]\!](S, \boldsymbol{H}, (f, f^{-1})) = \delta(S\{x \mapsto S(y)||S(z)\}, \boldsymbol{H}, (f, f^{-1})) \\
[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!] \\
[\![\mathsf{if}\ x\ \mathsf{then}\ \mathsf{c}_1\ \mathsf{else}\ \mathsf{c}_2\ \mathsf{fi}]\!](S, \boldsymbol{H}, (f, f^{-1})) = \begin{cases} [\![\mathsf{c}_1]\!](S, \boldsymbol{H}, (f, f^{-1})) \text{ if } S(x) = 1 \\ [\![\mathsf{c}_2]\!](S, \boldsymbol{H}, (f, f^{-1})) \text{ otherwise} \end{cases} \\
[\![\mathcal{N}(v, y)]\!](S, \boldsymbol{H}, (f, f^{-1})) = [\![c]\!](S\{x \mapsto v\}, \boldsymbol{H}, (f, f^{-1})) \text{ where } c \text{ is the body of } \mathcal{N}.
\end{array}
$$

Table 2. The semantics of the programming language

A notational convention: It is easy to prove that commands preserve the values of $\boldsymbol{H}$ and (f, f^{-1}). Therefore, we can, without ambiguity, write $S' \xleftarrow{r} [\![c]\!](S, \boldsymbol{H}, (f, f^{-1}))$ instead of $(S', \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} [\![c]\!](S, \boldsymbol{H}, (f, f^{-1}))$. According to our semantics, commands denote functions that transform distributions on configurations to distributions on configurations. However, only distributions that are constructible are of interest. Their set is denoted by $\textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ and is defined as the set of distributions of the form:

$[(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \boldsymbol{H} \xleftarrow{r} \Omega; S \xleftarrow{r} A^{\boldsymbol{H}, f, f^{-1}}() : (S, \boldsymbol{H}, f, f^{-1})]$ where A is an algorithm accessing f, f^{-1} and $\boldsymbol{H}$ and which records its queries to hashing oracles into the $\mathbb{T}_H$'s in S.

2.2 Asymmetric Encryption

We are interested in generic constructions that convert any trapdoor permutation into a public-key encryption scheme. More specifically, our aim is to provide an automatic verification method for generic encryption schemes. We also adapt IND-CPA and IND-CCA security notions to our setting.

Definition 1. *A* generic encryption scheme *is defined by a triple* $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$ *such that:*

- $\mathbb{F}$ *is a* trapdoor permutation generator *that on input* η *generates an* η*-bitstring trapdoor permutation* (f, f^{-1})*,*
- $\mathcal{E}(in_e, out_e) : \mathsf{c}$ *and* $\mathcal{D}(in_d, out_d) : \mathsf{c}'$ *are oracle declarations for encryption and decryption.*

Definition 2. *Let GE be a generic encryption scheme defined by* $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$*. Let* $A = (A_1, A_2)$ *be an adversary and* $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$*. For* $\alpha \in \{cpa, cca\}$ *and* $\eta \in \mathbb{N}$*, let*

$$
\begin{array}{rl}
\mathsf{Adv}^{ind-\alpha}_{A,GE}(\eta, X) = 2 * \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; & \\
(x_0, x_1, s) \xleftarrow{r} A_1^{\mathcal{O}_1}(f); b \xleftarrow{r} \{0, 1\}; & \\
S' \xleftarrow{r} [\![\mathcal{E}(x_b, out_e)]\!](S, \boldsymbol{H}, (f, f^{-1})) : & \\
A_2^{\mathcal{O}_2}(f, x_0, x_1, s, S'(out_e)) = b] - 1 &
\end{array}
$$

where if $\alpha = cpa$ then $\mathcal{O}_1 = \mathcal{O}_2 = \boldsymbol{H}$ and if $\alpha = cca$ then $\mathcal{O}_1 = \mathcal{O}_2 = \boldsymbol{H} \cup \{\mathcal{D}\}$.

We insist, above, that A_1 outputs x_0, x_1 such that $|x_0| = |x_1|$ and that in the case of CCA, A_2 does not ask its oracle $\mathcal{D}$ to decrypt $S'(y)$. We say that GE is IND-α secure if $\mathsf{Adv}^{ind-\alpha}_{A,GE}(\eta, X)$ is negligible for any constructible distribution ensemble X and polynomial-time adversary A.

3 IND-CPA security

In this section, we present an effective procedure to verify IND-CPA security. The procedure may fail to prove a secure encryption scheme but never declares correct an insecure one. Thus, we sacrifice completeness for soundness, a situation very frequent in verification[2]. We insist that our procedure does not fail for any of the numerous constructions we tried.

We are aiming at developing a procedure that allows us to prove properties, i.e. invariants, of the encryption oracle. More precisely, the procedure annotates each control point of the encryption command with a set of predicates that hold at that point for any execution except with negligible probability. Given an encryption oracle $\mathcal{E}(\mathrm{in}_e, \mathrm{out}_e) : \mathsf{c}$ we want to prove that at the final control point, we have an invariant that tells us that the value of out_e is indistinguishable from a random value. Classically, this implies IND-CPA security.

A few words now concerning how we present the verification procedure. First, we present in the assertion language the invariant properties we are interested in. Then, we present a set of rules of the form $\{\varphi\}c\{\varphi'\}$, meaning that execution of command c in any distribution that satisfies φ leads to a distribution that satisfies φ'. Using Hoare logic terminology, this means that the triple $\{\varphi\}c\{\varphi'\}$ is valid.

From now on, we suppose that the adversary has access to the hash functions $\boldsymbol{H}$, and he is given the trapdoor permutation f, but not its inverse f^{-1}.

3.1 The Assertion Language

Our assertion language is defined by the following grammar, where ψ defines the set of atomic assertions:

$$\psi ::= \mathsf{Indis}(\nu x; V_1; V_2) \mid \mathsf{WS}(x; V) \mid \mathsf{H}(H, e)$$
$$\varphi ::= \mathsf{true} \mid \psi \mid \varphi \wedge \varphi,$$

where $V_1, V_2 \subseteq \mathsf{Var}$ and e is an expression constructible (by the adversary) out of the variables used in the program, that is to say, possibly using concatenation, xor, hash oracles or f. Moreover, we define the set of the variables used as substring of an expression e and denote it $\mathsf{subvar}(e)$: $x \in \mathsf{subvar}(e)$ iff $e = e_1||x||e_2$, for some expressions e_1 and e_2. For example, we use the predicate $\mathsf{H}(H, R||\mathrm{in}_e||f(R||r)||\mathrm{in}_e \oplus G(R))$, for which, if we denote this latter expression e, we can write $\mathsf{subvar}(e) = \{R, \mathrm{in}_e\}$, since those variables are substrings of e, but $r \notin \mathsf{subvar}(e)$, since it cannot be obtained directly out of e.

Intuitively, $\mathsf{Indis}(\nu x; V_1; V_2)$ is satisfied by a distribution on configurations, if any adversary has negligible probability to distinguish whether he is given results of computations performed using the value of x or a random value, when he is given the values of the variables in V_1 and the image by the one-way permutation of those in V_2. The assertion $\mathsf{WS}(x; V)$ is satisfied by a distribution, if any adversary has negligible probability to compute the value of x, when he is

[2] We conjecture that the IND-CPA verification problem of schemes described in our language is undecidable.

given the values of the variables in V. Finally, $\mathsf{H}(H, e)$ is satisfied when the value of e has not been submitted to the hash oracle H.

Notations: We use $\mathsf{Indis}(\nu x; V)$ instead of $\mathsf{Indis}(\nu x; V; \emptyset)$ and $\mathsf{Indis}(\nu x)$ instead of $\mathsf{Indis}(\nu x; \mathsf{Var})$. We also write V, x instead of $V \cup \{x\}$ and even x, y instead of $\{x, y\}$.

Formally, the meaning of the assertion language is defined by a satisfaction relation $X \models \varphi$, which tells us when a distribution on configurations X satisfies the assertion φ. In order to define the satisfaction relation $X \models \varphi$, we need to generalize indistinguishability as follows. Let X be a family of distributions in $\textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ and V_1 and V_2 be sets of variables in Var. By $D(X, V_1, V_2)$ we denote the following distribution family (on tuples of bit-strings):

$$D(X, V_1, V_2)_\eta = \\ [(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : (S(V_1), f(S(V_2)), \boldsymbol{H}, f)]$$

Here $S(V_1)$ is the point-wise application of S to the elements of V_1 and $f(S(V_2))$ is the point-wise application of f to the elements of $S(V_2)$. We say that X and X' are $V_1; V_2$-*indistinguishable*, denoted by $X \sim_{V_1;V_2} X'$, if $D(X, V_1, V_2) \sim D(X', V_1, V_2)$.

Example 2. Let S_0 be any state and let H_1 be a hash function. Recall that we are working in the ROM. Consider the following distributions: $X_\eta = [\beta; S := S_0\{x \mapsto u, y \mapsto H_1(u)\} : (S, \boldsymbol{H}, (f, f^{-1}))]$ and $X'_\eta = [\beta; u' \xleftarrow{r} \{0,1\}^{p(\eta)}; S := S_0\{x \mapsto u, y \mapsto H_1(u')\} : (S, \boldsymbol{H}, (f, f^{-1}))]$, where $\beta = \boldsymbol{H} \xleftarrow{r} \Omega; (f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); u \xleftarrow{r} \{0,1\}^{p(\eta)}$, and p is a polynomial. Then, we have $X \sim_{\{y\};\{x\}} X'$ but we do not have $X \sim_{\{y,x\};\emptyset} X'$, because then the adversary can query the value of $H_1(x)$ and match it to that of y.

The satisfaction relation $X \models \psi$ is defined as follows:

- $X \models \mathsf{true}$, $X \models \varphi \wedge \varphi'$ iff $X \models \varphi$ and $X \models \varphi'$.
- $X \models \mathsf{Indis}(\nu x; V_1; V_2)$ iff $X \sim_{V_1;V_2} [u \xleftarrow{r} \mathcal{U}; (S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : (S\{x \mapsto u\}, \boldsymbol{H}, (f, f^{-1}))]$
- $X \models \mathsf{WS}(x; V)$ iff $\mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : A(S(V)) = S(x)]$ is negligible, for any adversary A.
- $X \models \mathsf{H}(H, e)$ iff $\mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : S(e) \in S(\mathbb{T}_H).\mathsf{dom}]$ is negligible.

The relation between our Hoare triples and semantic security is established by the following proposition that states that if the value of out_e is indistinguishable from a random value then the scheme considered is IND-CPA.

Proposition 1. *Let $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$ be a generic encryption scheme. It is IND-CPA secure if $\{\mathsf{true}\}\mathsf{c}\{\mathsf{Indis}(\nu\, out_e; out_e, in_e)\}$ is valid.*

Indeed, if $\{\mathsf{true}\}\mathsf{c}\{\mathsf{Indis}(\nu\mathrm{out}_e; \mathrm{out}_e, \mathrm{in}_e)\}$ holds then the encryption scheme is secure with respect to randomness of ciphertext. It is standard that randomness of ciphertext implies IND-CPA security.

3.2 A Hoare Logic for IND-CPA security

In this section we present our Hoare logic for IND-CPA security. We begin with a set of preservation rules that tell us when an invariant established at the control point before a command can be transferred to the next control point. Then, for each command, except $x := f^{-1}(y)$, $x := y[n, m]$ and conditional, we present a set of specific rules that allow us to establish new invariants. The commands that are not considered are usually not used in encryption but only in decryption procedures, and hence, are irrelevant with respect to our way of proving IND-CPA security.

Generic preservation rules: We assume $z \neq x$ and c is either $x \xleftarrow{r} \mathcal{U}$ or $x := y||t$ or $x = y \oplus t$ or $x := f(y)$ or $x := H(y)$ or $x := t \oplus H(y)$.

Lemma 1. *The following rules are sound, when $x \notin V_1 \cup V_2$:*

- *(G1)* $\{\mathsf{Indis}(\nu z; V_1; V_2)\}\ c\ \{\mathsf{Indis}(\nu z; V_1; V_2)\}$
- *(G2)* $\{\mathsf{WS}(z; V_1)\}\ c\ \{\mathsf{WS}(z; V_1)\}$
- *(G3)* $\{\mathsf{H}(H', e[e'/x])\}\ x := e'\ \{\mathsf{H}(H', e)\}$, *provided $H' \neq H$ in case $e' \equiv H(y)$. Here, $e[e'/x]$ is the expression obtained from e by replacing x by e'.*

Random Assignment:

Lemma 2. *The following rules are sound:*

- *(R1)* $\{\mathsf{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathsf{Indis}(\nu x)\}$
- *(R2)* $\{\mathsf{true}\}\ x \xleftarrow{r} \mathcal{U}\ \{\mathsf{H}(H, e)\}$ *if* $x \in \mathsf{subvar}(e)$.

Moreover, the following preservation rules, where we assume $x \neq y$ [3]*, are sound:*

- *(R3)* $\{\mathsf{Indis}(\nu y; V_1; V_2)\} x \xleftarrow{r} \mathcal{U}\{\mathsf{Indis}(\nu y; V_1, x; V_2)\}$
- *(R4)* $\{\mathsf{WS}(y; V)\} x \xleftarrow{r} \mathcal{U}\{\mathsf{WS}(y; V, x)\}$

Rule (R1) is obvious. Rule (R2) takes advantage of the fact that $\mathcal{U}$ is a large set, or more precisely that its cardinality is exponential in the security parameter, and that since e contains the fresh generated x the probability that it has already been submitted to H is small. Rules (R3) and (R4) state that the value of x cannot help an adversary in distinguishing the value of y from a random value in (R3) or computing its value in (R4). This is the case because the value of x is randomly sampled.

Hash Function:

Lemma 3. *The following basic rules are sound, when $x \neq y$, and α is either a constant or a variable:*

- *(H1)* $\{\mathsf{WS}(y; V) \wedge \mathsf{H}(H, y)\} x := \alpha \oplus \mathsf{H}(y) \{\mathsf{Indis}(\nu x; V, x)\}$
- *(H2)* $\{\mathsf{H}(H, y)\}\ x := H(y) \{\mathsf{H}(H', e)\}$, *if* $x \in \mathsf{subvar}(e)$.
- *(H3)* $\{\mathsf{Indis}(\nu y; V; V', y) \wedge \mathsf{H}(H, y)\} x := H(y)$ $\{\mathsf{Indis}(\nu x; V, x; V', y)\}$ *if* $y \notin V$

Rule (H1) captures the main feature of the random oracle model, namely that the hash function is a random function. Hence, if an adversary cannot compute the value of y and this latter has not been hashed yet then he cannot distinguish $H(y)$ from a random value. Rule (H2) is similar to rule (R2). Rule (H3) uses the fact that the value of y can not be queried to the hash oracle.

Lemma 4. *The following preservation rules are sound provided that $x \neq y$ and $z \neq x$:*

- *(H4)* $\{\mathsf{WS}(y; V) \wedge \mathsf{WS}(z; V) \wedge \mathsf{H}(H, y)\} x := H(y)$ $\{\mathsf{WS}(z; V, x)\}$
- *(H5)* $\{\mathsf{H}(H, e) \wedge \mathsf{WS}(z; y)\} x := H(y) \{\mathsf{H}(H, e)\}$, *if* $z \in \mathsf{subvar}(e) \wedge x \notin \mathsf{subvar}(e)$
- *(H6)* $\{\mathsf{Indis}(\nu y; V_1; V_2, y) \wedge \mathsf{H}(H, y)\} x := H(y)$ $\{\mathsf{Indis}(\nu y; V_1, x; V_2, y)\}$, *if* $y \notin V_1$

[3] By $x = y$ we mean syntactic equality.

– *(H7)* $\{\mathsf{Indis}(\nu z; V_1, z; V_2) \wedge \mathsf{WS}(y; V_1 \cup V_2, z) \wedge \mathsf{H}(H, y)\} x := H(y) \{\mathsf{Indis}(\nu z; V_1, z, x; V_2)\}$

The idea behind (H4) is that to the adversary the value of x is seemingly random so that it can not help to compute z. Rule (H5) states that the value of e not having been hashed yet reminds true as long as e contains a variable z whose value is not computable out of y. (H6) and (H7) give necessary conditions to the preservation of indistinguishability that is based on the seemingly randomness of a hash value.

One-way Function:

Lemma 5. *The following rule is sound, when $y \notin V \cup \{x\}$:*

– *(O1)* $\{\mathsf{Indis}(\nu y; V; y)\}\ x := f(y)\ \{\mathsf{WS}(y; V, x)\}$.

Rule (O1) captures the one-wayness of f.

Lemma 6. *The following rules are sound when $z \neq x$:*

– *(O2)* $\{\mathsf{Indis}(\nu z; V_1, z; V_2, y)\}\ x := f(y) \{\mathsf{Indis}(\nu z; V_1, z, x; V_2, y)\}$, *if* $z \neq y$
– *(O3)* $\{\mathsf{WS}(z; V) \wedge \mathsf{Indis}(\nu y; V; y, z)\}\ x := f(y)\ \{\mathsf{WS}(z; V, x)\}$

For one-way permutations, we also have the following rule:

– *(P1)* $\{\mathsf{Indis}(\nu y; V_1; V_2, y)\}\ x := f(y)$
$\{\mathsf{Indis}(\nu x; V_1, x; V_2)\}$, *if* $y \notin V_1 \cup V_2$

Rule (O2) is obvious since $f(y)$ is given to the adversary in the precondition and rule (O3) follows from the fact that y and z are independent. Rule (P1) simply ensues from the fact that f is a permutation.

The Xor operator In the following rules, we assume $y \neq z$.

Lemma 7. *The following rule is sound when $y \notin V_1 \cup V_2$:*

– *(X1)* $\{\mathsf{Indis}(\nu y; V_1, y, z; V_2)\} x := y \oplus z \{\mathsf{Indis}(\nu x; V_1, x, z; V_2)\}$,

Moreover, we have the following rules that are sound provided that $t \neq x, y, z$.

– *(X2)* $\{\mathsf{Indis}(\nu t; V_1, y, z; V_2)\} x := y \oplus z \{\mathsf{Indis}(\nu t; V_1, x, y, z; V_2)\}$
– *(X3)* $\{\mathsf{WS}(t; V, y, z)\} x := y \oplus z \{\mathsf{WS}(t; V, y, z, x)\}$

To understand rule (X1) one should consider y as a key and think about x as the one-time pad encryption of z with the key y. Rules (X2) and (X3) take advantage of the fact that is easy to compute x given y and z.

Concatenation:

Lemma 8. *The following rules are sound:*

– *(C1)* $\{\mathsf{WS}(y; V)\}\ x := y||z\ \{\mathsf{WS}(x; V)\}$, *if* $x \notin V$. *A dual rule applies for z.*
– *(C2)* $\{\mathsf{Indis}(\nu y; V_1, y, z; V_2) \wedge \mathsf{Indis}(\nu z; V_1, y, z; V_2)\}\ x := y||z\ \{\mathsf{Indis}(\nu x; V_1, x; V_2)\}$, *if* $y, z \notin V_1 \cup V_2$
– *(C3)* $\{\mathsf{Indis}(\nu t; V_1, y, z; V_2)\} x := y||z \{\mathsf{Indis}(\nu t; V_1, x, y, z; V_2)\}$, *if* $t \neq x, y, z$

– *(C4)* $\{\mathsf{WS}(t; V, y, z)\}\ x := y||z\ \{\mathsf{WS}(t; V, y, z, x)\}$, *if* $t \neq x, y, z$

(C1) states that if computing a substring of x out of the elements of V is hard, then so is computing x itself. The idea behind (C2) is that y and z being random implies randomness of x, with respect to V_1 and V_2. Eventually, x being easily computable from y and z accounts for rules (C3) and (C4).

In addition to the rules above, we have the usual sequential composition and consequence rules of the Hoare logic. In order to apply the consequence rule, we use entailment (logic implication) between assertions as in Lemma 9.

Lemma 9. *Let* $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ *be a distribution ensemble:*

1. *If* $X \models \mathsf{Indis}(\nu x; V_1; V_2)$, $V_1' \subseteq V_1$ *and* $V_2' \subseteq V_1 \cup V_2$ *then* $X \models \mathsf{Indis}(\nu x; V_1'; V_2')$.
2. *If* $X \models \mathsf{WS}(x; V')$ *and* $V \subseteq V'$ *then* $X \models \mathsf{WS}(x; V)$.
3. *If* $X \models \mathsf{Indis}(\nu x; V_1; V_2 \cup \{x\})$ *and* $V \subseteq V_1 \setminus \{x\}$ *then* $X \models \mathsf{WS}(x; V)$.

The soundness of the Hoare Logic follows by induction from the soundness of each rule and soundness of the Consequence and Sequential composition rules.

Proposition 2. *The Hoare triples given in Section 3.2 are valid.*

Example 3. We illustrate our proposition with Bellare & Rogaway's generic construction [7].

$$
\begin{array}{l}
1)\ r \xleftarrow{r} \{0,1\}^{n_0} \\
\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e||r) \\
2)\ a := f(r) \\
\mathsf{Indis}(\nu a; \mathsf{Var} - r) \wedge \mathsf{WS}(r; \mathsf{Var} - r) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e||r) \\
3)\ g := G(r) \\
\mathsf{Indis}(\nu a; \mathsf{Var} - r) \wedge \mathsf{Indis}(\nu g; \mathsf{Var} - r) \wedge \\
\mathsf{WS}(r; \mathsf{Var} - r) \wedge \mathsf{H}(H, \mathrm{in}_e||r) \\
4)\ b := \mathrm{in}_e \oplus g \\
\mathsf{Indis}(\nu a; \mathsf{Var} - r) \wedge \mathsf{Indis}(\nu b; \mathsf{Var} - g - r) \wedge \\
\mathsf{WS}(r; \mathsf{Var} - r) \wedge \mathsf{H}(H, \mathrm{in}_e||r) \\
5)\ s := \mathrm{in}_e||r \\
\mathsf{Indis}(\nu a; \mathsf{Var} - r - s) \wedge \mathsf{Indis}(\nu b; \mathsf{Var} - g - r - s) \wedge \\
\mathsf{WS}(s; \mathsf{Var} - r - s) \wedge \mathsf{H}(H, s) \\
6)\ c := H(s) \\
\mathsf{Indis}(\nu a; \mathsf{Var} - r - s) \wedge \mathsf{Indis}(\nu b; \mathsf{Var} - r - g - s) \wedge \\
\mathsf{Indis}(\nu c; \mathsf{Var} - r - s) \\
7)\ \mathrm{out}_e := a||b||c \\
\mathsf{Indis}(\nu \mathrm{out}_e; \mathsf{Var} - a - b - c - r - g - s)
\end{array}
$$

1) $(R1)$, $(R2)$, and $(R2)$.
2) $(P1)$, $(O1)$, $(G3)$, and $(G3)$.
3) $(H7)$, $(H1)$, $(H4)$, and $(G3)$.
4) $(X2)$, $(X1)$, $(X3)$, and $(G3)$.
5) $(G1)$, $(G1)$, $(C1)$, and $(G3)$.
6) $(H7)$, $(H7)$, and $(H1)$.
7) $(C2)$ twice.

3.3 Extensions

In this section, we show how our Hoare logic, and hence our verification procedure, can be adapted to deal with on one hand injective partially trapdoor one-way functions and on the other hand OW-PCA (probabilistic) functions. The first extension is motivated by Pointcheval's construction in [17] and the second one by the Rapid Enhanced-security Asymmetric Cryptosystem Transform (REACT) [16].

The first observation we have to make is that Proposition 1 is too demanding in case f is not a permutation. Therefore, we introduce a new predicate $\mathsf{Indis}_f(\nu x; V_1; V_2)$ whose meaning is as follows:

$X \models \mathsf{Indis}_f(\nu x; V_1; V_2)$ if and only if $X \sim_{V_1;V_2} [u \xleftarrow{r} \mathcal{U}; (S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : (S\{x \mapsto f(u)\}, \boldsymbol{H}, (f, f^{-1}))]$.

Notice that, when f is a bijection, $\mathsf{Indis}_f(\nu x; V_1; V_2)$ is equivalent to $\mathsf{Indis}(\nu x; V_1; V_2)$ (f can be the identity function as in the last step of Example 4 and 5). Now, let out_e, the output of the encryption oracle, have the form $a_1||\cdots||a_n$ with $a_i = f_i(x_i)$. Then, we can prove the following:

Proposition 3. *Let GE be a generic encryption scheme of the form* $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$.

If $\{\mathsf{true}\}\mathsf{c}\{\bigwedge_{i=1}^{n} \mathsf{Indis}_{f_i}(\nu a_i; a_1, \ldots, a_n, in_e)\}$ *is valid then GE is IND-CPA.*

Now, we introduce a new rule for $\mathsf{Indis}_f(\nu x; V_1; V_2)$ that replaces rule (P1) in case the one-way function f is not a permutation:

$$
\begin{array}{ll}
(P1') & \{\mathsf{Indis}(\nu y; V_1; V_2, y)\} \\
& x := f(y) \\
& \{\mathsf{Indis}_f(\nu x; V_1, x; V_2)\} \text{ if } y \notin V_1 \cup V_2
\end{array}
$$

Clearly all preservation rules can be generalized for Indis_f.

Injective partially trapdoor one-way functions: In contrast to the previous section, we do not assume f to be a permutation. On the other hand, we demand a stronger property than one-wayness. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function and let $f^{-1} : \mathcal{Z} \to \mathcal{X}$ be such that $\forall z \in \mathsf{dom}(f^{-1}) \exists y \in \mathcal{Y},\ z = f(f^{-1}(z), y)$. Here f^{-1} is a partial function. The function f is said *partially one-way*, if for any given $z = f(x, y)$, it is computationally impossible to compute a corresponding x. In order to deal with the fact that f is now partially one-way, we add the following rules, where we assume $x, y \notin V \cup \{z\}$ and where we identify f and $(x, y) \mapsto f(x||y)$:

$$
\begin{array}{ll}
\text{(PO1)} & \{\mathsf{Indis}(\nu x; V, x, y) \wedge \mathsf{Indis}(\nu y; V, x, y)\} \\
& z := f(x||y) \\
& \{\ \mathsf{WS}(x; V, z) \wedge \mathsf{Indis}_f(\nu z; V, z)\ \}
\end{array}
$$

The intuition behind the first part of (PO1) is that f guarantees one-way secrecy of the x-part of $x||y$. The second part follows the same idea that (P1').

Example 4. We verify Pointcheval's transformer [17].

$$
\begin{array}{l}
1)\ r \xleftarrow{r} \{0,1\}^{n_0} \\
\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r) \\
2)\ s \xleftarrow{r} \{0,1\}^{n_0} \\
\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{Indis}(\nu s; \mathsf{Var}) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e || s) \\
3)\ w := \mathrm{in}_e || s \\
\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{WS}(w; \mathsf{Var} - s - w) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, w) \\
4)\ h := H(w) \\
\mathsf{Indis}(\nu r; \mathsf{Var} - w - s) \wedge \mathsf{Indis}(\nu h; \mathsf{Var} - w - s) \wedge \mathsf{H}(G, r) \\
5)\ a := f(r || h) \\
\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - s - w - h) \\
\wedge \mathsf{WS}(r; \mathsf{Var} - r - s - w - h) \wedge \mathsf{H}(G, r) \\
6)\ b := w \oplus G(r) \\
\mathsf{Indis}_f(\nu a; a, \mathrm{in}_e) \wedge \mathsf{Indis}(\nu b; a, b, \mathrm{in}_e) \\
7)\ \mathrm{out}_e := a || b \\
\mathsf{Indis}_f(\nu a; a, \mathrm{in}_e) \wedge \mathsf{Indis}(\nu b; a, b, \mathrm{in}_e)
\end{array}
$$

1) $(R1)$ and $(R2)$; 2) $(R3)$, $(R1)$, $(G3)$ and $(R2)$; 3) $(C3)$, $(C1)$, $(G3)$, and $(G3)$; 4) $(H7)$, $(H1)$, and $(G3)$; 5) New rule $(PO1)$ and $(G3)$; 6) Extension of $(G1)$ to Indis_f, and $(H1)$; 7) Extension of $(G1)$ to Indis_f, and $(G1)$.

To conclude, we use the fact that $\mathsf{Indis}_f(\nu a; a, \mathrm{in}_e)$ and $\mathsf{Indis}(\nu b; a, b, \mathrm{in}_e)$ implies $\mathsf{Indis}_f(\nu a; a, b, \mathrm{in}_e)$

OW-PCA: Some constructions such as REACT are based on probabilistic one-way functions that are difficult to invert even when the adversary has access to a plaintext checking oracle (PC), which on input a pair (m, c), answers whether c encrypts m. In order to deal with OW-PCA functions, we need to strengthen the meaning of our predicates allowing the adversary to access to the additional plaintext checking oracle. For instance, the definition of $\mathsf{WS}(x; V)$ becomes: $X \models \mathsf{WS}(x; V)$ iff $\mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : A^{PCA}(S(V)) = S(x)]$ is negligible, for any adversary A. Now, we have to revisit Lemma 9 and the rules that introduce $\mathsf{WS}(x; V)$ in the postcondition. It is, however, easy to check that they are valid.

Example 5. REACT [16]

$$
\begin{array}{l}
1)\ r \xleftarrow{r} \{0,1\}^{n_0} \\
\mathsf{Indis}(\nu r; \mathsf{Var}) \\
2)\ R \xleftarrow{r} \{0,1\}^{n_0} \\
\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{Indis}(\nu R; \mathsf{Var}) \wedge \mathsf{H}(G, R) \wedge \\
\mathsf{H}(H, R||\mathrm{in}_e||f(R||r)||\mathrm{in}_e \oplus G(R)) \\
3)\ a := f(R||r) \\
\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - R) \wedge \mathsf{WS}(R; \mathsf{Var} - r - R) \wedge \\
\mathsf{H}(G, R) \wedge \mathsf{H}(H, R||\mathrm{in}_e||a||\mathrm{in}_e \oplus G(R)) \\
4)\ g := G(R) \\
\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - R) \wedge \mathsf{Indis}(\nu g; \mathsf{Var} - r - R) \wedge \\
\mathsf{WS}(R; \mathsf{Var} - r - R) \wedge \mathsf{H}(H, R||\mathrm{in}_e||a||\mathrm{in}_e \oplus g) \\
5)\ b := \mathrm{in}_e \oplus g \\
\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - R) \wedge \mathsf{Indis}(\nu b; \mathsf{Var} - g - r - R) \wedge \\
\mathsf{WS}(R; \mathsf{Var} - r - R) \wedge \mathsf{H}(H, R||\mathrm{in}_e||a||b) \\
6)\ w := R||\mathrm{in}_e||a||b \\
\mathsf{Indis}_f(\nu a; \mathsf{Var} - r - w - R) \\
\wedge \mathsf{Indis}(\nu b; \mathsf{Var} - g - r - w - R) \\
\wedge \mathsf{WS}(w; \mathsf{Var} - r - w - R) \wedge \mathsf{H}(H, w) \\
7)\ c := H(w) \\
\mathsf{Indis}_f(\nu a; a, b, c, \mathrm{in}_e) \wedge \mathsf{Indis}(\nu b; a, b, c, \mathrm{in}_e) \\
\wedge \mathsf{Indis}(\nu c; a, b, c, \mathrm{in}_e) \\
8)\ \mathrm{out}_e := a||b||c; \\
\mathsf{Indis}_f(\nu a; a, b, c, \mathrm{in}_e) \wedge \mathsf{Indis}(\nu b; a, b, c, \mathrm{in}_e) \\
\wedge \mathsf{Indis}(\nu c; a, b, c, \mathrm{in}_e)
\end{array}
$$

1) $(R1)$
2) $(R3)$, $(R1)$, $(R2)$ and $(R2)$
3) $(PO1)$, $(G3)$ and $(G3)$.
4) Extension of $(H7)$ to Indis_f, $(H1)$, $(H4)$, and $(G3)$.
5) Extension of $(X2)$ to Indis_f, $(X1)$, $(X3)$, and $(G3)$.
6) Extension of $(G1)$ to Indis_f, $(G1)$, $(C1)$, and $(G3)$.
7) Extension of $(H7)$ to Indis_f, $(H7)$, and $(H1)$.
8) Extension of $(G1)$ to Indis_f, $(G1)$ and $(G1)$.

4 Achieving A Stronger Criterion: IND-CCA Security Of A Scheme

Up to now, we have been interested in demonstrating the most basic notion of security, namely IND-CPA. Nevertheless, most of the schemes achieve a stronger level of security, since they are IND-CCA secure. The great difference between IND-CPA and IND-CCA is that adversaries attacking IND-CCA security are granted access to the decryption oracle all along their game against the scheme, on condition that they do not ask for the decryption of their challenge. More powerful adversaries mean stronger security criteria, and it is easy to figure out that an IND-CCA scheme is IND-CPA.
However, the whole logic that has been developed was meant to deal with IND-CPA. The notion of Weak Secrecy for example is biaised in the new IND-CCA context, since the ability to

decipher messages often allows to bypass the computation of the inverse of the one-way function used in the scheme. Indistinguishability itself is a far more difficult property to achieve, since giving $f(V_2)$ may permit the adversary to create ciphertexts he can then submit to the decryption oracle. This is why a change is required in the way to carry out our proofs.
In their article [4], Bellare et al. list and compare the most classical security criteria. They show that IND-CCA security is implied by IND-CPA security of a scheme plus its plaintext awareness. This is the way we choose to deal with IND-CCA.

4.1 Introduction Of Plaintext Awareness

Plaintext awareness (PA) was first introduced in [5], but its original definition was slightly weaker. It was then refined in [4] to be the following notion. The idea is that the adversary should not be able to obtain ciphertexts without knowing the corresponding plaintext. If it is the case, we can consider that he asks the encryption oracle to cipher them, so that we do not need to care much about this capacity since it is yet taken into account by the IND-CPA criterion. Queries to the decryption oracle are adding a new element to the knowledge of the adversary if he can ask the decryption of interesting ciphertexts, that is, some that he hasn't obtained from the encryption oracle. Otherwise functional correctness of the scheme imposes the result and the query is useless.
In practice, an extra algorithm is introduced to define PA. This is the plaintext extractor K. As its name allows to suppose, it is meant to simulate the decryption algorithm. The idea is to say that if to any ciphertext the adversary manages to output, the plaintext extractor can associate the corresponding plaintext *without* asking anything to $\mathcal{D}$, but only looking at the adversary's queries to hash oracles and the encryption algorithm, then the scheme is plaintext aware. That is to say, no poly-time adversary can output a ciphertext he couldn't decipher on his own.

Formally, an adversary $\mathcal{B}$ against PA security of a scheme outputs a list hH of his hash queries and their results, a list C of his queries to $\mathcal{E}$, and a ciphertext y that he challenges the plaintext extractor to decipher. $\mathcal{B}$ wins if the plaintext extractor does not output the same thing as the decryption oracle. Otherwise, if $y \in C$ (the adversary has cheated and output a ciphertext he obtained from $\mathcal{E}$) or if $K(y)$ is the same as $\mathcal{D}(y)$, K is the winner of the experiment. We thus define:

Definition 3 (Success Probability of the Plaintext Extractor). *Let X be a distribution on configurations and GE be a generic scheme. Then the probability that the plaintext extractor K succeeds against adversary $\mathcal{B}$ is worth:*
$\mathsf{Succ}^{\mathsf{pa}}_{K,\mathcal{B},GE}(\eta, X) = \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} \mathcal{B}^{\mathcal{E}(),\boldsymbol{H}}(f);$
$S'' \xleftarrow{r} [\![\mathcal{D}(y)]\!](S', \boldsymbol{H}, (f, f^{-1})) : y \in C \vee (y \notin C \wedge K(hH, C, y, f) = S''(out_d))]$

Now the formal definition of plaintext awareness is easy to state:

Definition 4 (Plaintext Awareness). *An encryption scheme $GE = (\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$ is* PA-secure*, if there is a polynomial time probabilistic algorithm K such that for every distribution $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ and adversary $\mathcal{B}$, $1 - \mathsf{Succ}^{\mathsf{pa}}_{K,\mathcal{B},GE}(\eta, X)$ is a negligible function in η.*

4.2 Intuition on Means to Ensure Plaintext Awareness

Getting used to working with plaintext awareness, and trying to acquire an intuition about it on usual schemes, one can notice that a certain form of decryption algorithms are particularly well-suited for the verification of this criterion.
The thing is, some decryption algorithms can be split into two parts: a first part that actually computes the plaintext out of the ciphertext, and a second one that checks whether the ciphertext was 'legally' obtained. This last verification, that we call the '*sanity check*', is the one ensuring PA (and hence IND-CCA) security. It allows to discriminate random bitstrings or ciphertexts that have been tampered with from valid ciphertexts output by the encryption algorithm.
More precisely, if we consider a scheme $\mathcal{GE} = (\mathbb{F}, \mathcal{E}, \mathcal{D}(\text{in}_d, \text{out}_d) : \mathsf{c}')$ using $\boldsymbol{H} = (H_1, \ldots, H_n)$, let us suppose that c' has the following pattern:

- some command c_1 computes the plaintext,
- $h^* := H_1(t^*)$ is computed,
- then comes the branching $\mathsf{if}\ \mathcal{V}(\boldsymbol{x}, h^*) = v\ \mathsf{then}\ \text{out}_d := m^*\ \mathsf{else}\ \text{out}_d := \textit{"error"}\ \mathsf{fi}$, where $\boldsymbol{x}$ is a vector of variables (possibly empty) and $\mathcal{V}$ is a function such that for given $\boldsymbol{x}$ and v, $Pr[r \xleftarrow{r} \mathcal{U} : \mathcal{V}(\boldsymbol{x}, r) = v]$ is negligible. The condition $\mathcal{V}(\boldsymbol{x}, h^*) = v$ is called the sanity check.

The idea behind such a test is simple: as a hash value cannot be forged (in the ROM), the hash oracle *has* to be queried on some t^*, whose right value is meant to be computable by a normal execution of the encryption algorithm only. The challenge of a PA-adversary thus becomes to compute the right hash value, hence we can easily prove he has negligible probability of success. Nevertheless, soundness of such an argument requires the weak injectivity property we impose on $\mathcal{V}$. Indeed, if a great number of values verified the sanity check, we would not be able to deduce from its validity that the adversary can compute the right hash value. The uniform distribution of argument r in the hypothesis on $\mathcal{V}$ is meant to simulate the distribution of hash values.

In practice, we need two more assumptions on the form of the program and the use of the variables. First, the encryption algorithm is supposed to make an unique call to H_1 on a variable t. Secondly, the value t^* that $\mathcal{D}$ computes matches the value of t computed by $\mathcal{E}$ during a sound execution of the pair of algorithms. Figure 4.2 illustrates this assumption of t and t^* playing the same role in respectively the encryption and the decryption algorithm.

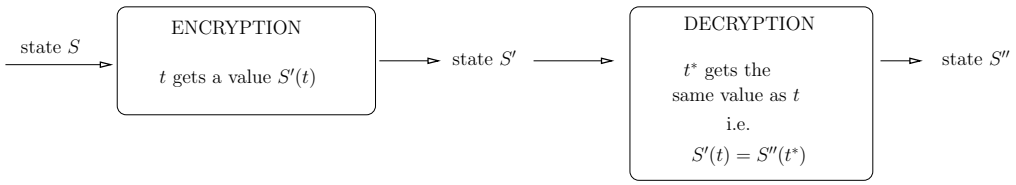

Fig. 1. The hypothesis about t and t^*

Hereafter, the reader can find the example of the scheme designed in 1993 by Bellare and Rogaway [7], that illustrates the discussion above very well. To lighten the notations, we use

directly a match command, that is not in the language, instead of doing it by hand by cutting the bitstring in three. The sanity check is highlighted in the code of the decryption oracle. Notice that this code indeed falls into two parts: first, the computation of the plaintext m^*, that does not involve c^*. This latter only serves the last test purpose, which is to ensure that the right value of t^*, and thus the right values of m^* and r^*, have been computed. This conditional branching somewhat forces whoever attacks plaintext awareness of the scheme to invert f and query G on the result of the inversion itself; it creates an extremely strong link between the random seed and the plaintext by placing it under a hash function.

Encryption $\mathcal{E}(\text{in}_e, \text{out}_e) =$
$r \xleftarrow{r} \mathcal{U}$;
$a = f(r)$;
$g := G(r)$;
$b := \text{in}_e \oplus g$;
$t := x||r$
$c := H(t)$;
$\text{out}_e := a||b||c$

Decryption $\mathcal{D}(\text{in}_d, \text{out}_d)$
match in_d with $a^*||b^*||c^*$;
$r^* := f^{-1}(a^*)$;
$g^* := G(r^*)$;
$m^* := b^* \oplus g^*$;
$t^* := m^*||r^*$;
$h^* := H(t^*)$;
if $h^* = c^*$ **then** $\textbf{out}_d := m^*$
else $\textbf{out}_d := error$

4.3 Formal Semantic Criterion For Plaintext Awareness

We recall that we suppose the decryption oracle to be of the following form:
$\mathsf{c}_1; h^* := H_1(t^*); \mathsf{if}\ \mathcal{V}(\boldsymbol{x}, h^*) = v\ \mathsf{then}\ \text{out}_d := m^*\ \mathsf{else}\ \text{out}_d := \text{"}error\text{"}\ \mathsf{fi}$,
and that H_1 is called once in $\mathcal{E}$ on a variable t. On top of that, we require that if $S \xleftarrow{r} [\![\mathcal{E}]\!](\text{in}_e)$ and $S' := \mathcal{D}(S(\text{out}_e))$, then $S(t) = S'(t^*)$. This last condition simply states that t and t^* play the same role in both algorithms.

The intuition behind the semantic criterion is quite easily understandable. We are going to impose three conditions to ensure the ability to construct a plaintext extractor enjoying an overwhelming probability of success. That is to say, we design conditions to enable an efficient simulation of the decryption algorithm. We know that K, the plaintext extractor, is granted access to the list hH_1 of oracle queries of the adversary $\mathcal{B}$ to H_1 and their results.
The idea is that, if K is able to select among $hH_1.\mathsf{dom}$ the right value of t^* the decryption algorithm would compute, then (looking at the example above where $t^* = m^*||r^*$), the extraction of the plaintext is pretty likely to succeed (in the example selecting the prefix suffices!). Such a selection could be done by testing candidates to the sanity check one by one. Since there is only a polynomial number of queries ($\mathcal{B}$ queried the oracle and is poly-time), this takes a polynomial time. Therefore, showing the existence of K amounts to constructing an efficient tester.
The first condition thus consists in assuming the existence of a poly-time algorithm called the tester, able to discriminate valid candidates to the sanity check from unsatisfactory ones. Then, we impose that the extraction of the plaintext be easily achievable from a good candidate. Eventually, to get rid of possible ambiguity, we add that to a value cd (for candidate) of t^* corresponds at most one possible ciphertext, so that the extracted plaintext is indeed the one the decryption oracle outputs when verifying the sanity check on cd.

Here is the formal statement of the semantic criterion:

Definition 5 (PA Semantic Criterion). *We say that GE satisfies the* PA-semantic criterion*, if there exist efficient algorithms $\mathcal{T}$ and Ext that satisfy the following conditions:*

1. *The* tester $\mathcal{T}$ *takes as input (hH, C, y, cd, f) and returns a value in $\{0,1\}$. We require that for any adversary $\mathcal{B}$ and any distribution $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$,*

$$1 - Pr[\ (S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} \mathcal{B}^{\mathcal{E}(),\boldsymbol{H}}(f); \\ S'' \xleftarrow{r} [\![\mathcal{D}(y)]\!](S', \boldsymbol{H}, (f, f^{-1})); cd \xleftarrow{r} hH_1.\mathsf{dom}; b \xleftarrow{r} \mathcal{T}(hH, C, y, cd, f) : \\ \big(b = 1 \Rightarrow (H_1(cd) = H_1(S''(t^*)) \wedge\ \mathcal{V}(S''(x), H_1(cd)) = S''(v)\big) \wedge \\ \big(b = 0 \Rightarrow \mathcal{V}(S''(x), H_1(cd)) \neq S''(v)\big)] \text{ is negligible.}$$

2. *For Ext, we require that for any adversary $\mathcal{B}$ and any distribution $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$,*

$$1 - Pr[\ (S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} \mathcal{B}^{\mathcal{E}(),\boldsymbol{H}}(f); \\ S'' \xleftarrow{r} [\![\mathcal{D}(y)]\!](S', \boldsymbol{H}, (f, f^{-1})) :\ \mathsf{Ext}(hH, C, y, S''(t^*), f) = S''(out_d)] \text{ is negligible.}$$

3. *Finally, we require that for any adversary $\mathcal{B}$ and any distribution $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$,*

$$Pr[\ (S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, y', S') \xleftarrow{r} \mathcal{B}^{\mathcal{E}(),\boldsymbol{H}}(f); \\ S_1 \xleftarrow{r} [\![\mathcal{D}(y)]\!](S', \boldsymbol{H}, (f, f^{-1})); S_2 \xleftarrow{r} [\![\mathcal{D}(y')]\!](S', \boldsymbol{H}, (f, f^{-1})) : \\ y \neq y' \wedge S_1(t^*) = S_2(t^*) \wedge S_1(out_d) \neq \text{"error"} \wedge S_2(out_d) \neq \text{"error"}] \text{ is negligible.}$$

If a scheme satisfies definition 5, given such a tester $\mathcal{T}$ and such an extraction algorithm Ext, the plaintext extractor can be constructed as follows:
$K^{\mathcal{T},\mathsf{Ext}}(hH, C, y, f)$: Let $L = (cd \mid cd \in \mathsf{dom}(hH_1)$ such that $\mathcal{T}(hH, C, y, cd, f) = 1)$
if $L = \epsilon$ then return "error" else $cd \xleftarrow{r} L$; return $\mathsf{Ext}(hH, C, y, cd, f)$

We can then demonstrate that our semantic criterion indeed implies plaintext awareness.

Theorem 1. *Let GE be a generic encryption scheme that satisfies the PA-semantic criterion. Then, GE is PA-secure.*

Of course there are generic encryption schemes for which the conditions above are satisfied under the assumption that $\mathcal{T}$ has access to an extra oracle such as a plaintext checking oracle (PC), or a ciphertext validity-checking oracle (CV), which on input c answers whether c is a valid ciphertext. In this case, the semantic security of the scheme has to be established under the assumption that f is OW-PCA, respectively OW-CVA. Furthermore, our definition of the PA-semantic criterion makes perfect sense for constructions that apply to IND-CPA schemes such as Fujisaki and Okamoto's converter [14]. In this case, f has to be considered as the IND-CPA encryption oracle.

4.4 A Syntactic Criterion for Plaintext Awareness

An easy syntactic check that implies the PA-semantic criterion is as follows.

Definition 6. *A generic encryption scheme GE satisfies the* PA-syntactic criterion*, if the sanity check has the form $\mathcal{V}(t, h) = v$, where $\mathcal{D}$ is such that h is assigned $H_1(t)$, t is assigned $in_e||r$, in_e is the plaintext and $\mathcal{E}(in_e; r)$ is the ciphertext (i.e., r is the random seed of $\mathcal{E}$).*

It is not difficult to see that if *GE* satisfies the PA-syntactic criterion then it also satisfies the PA-semantic one with a tester $\mathcal{T}$ as follows (Ext is obvious):

Look in hH_1 for a bit-string s such that $\mathcal{E}(x^*; r^*) = y$, where y is the challenge and $x^*||r^* = s$.

Here are some examples that satisfy the syntactic criterion (we use $\cdot^*$ to denote the values computed by the decryption oracle):

Example 6. – Bellare and Rogaway [7]: $\mathcal{E}(\text{in}_e; r) = a||b||c = f(r)||\text{in}_e \oplus G(r)||H(\text{in}_e||r)$. The "sanity check" of the decryption algorithm is $H(m^*||r^*) = c^*$.

– OAEP+ [18]: $\mathcal{E}(\text{in}_e; r) = f(a||b||c)$, where $a = \text{in}_e \oplus G(r)$, $b = H'(\text{in}_e||r)$, $c = H(s) \oplus r$ and $s = \text{in}_e \oplus G(r)||H'(\text{in}_e||r)$. The "sanity check" of the decryption algorithm has the form $H'(m^*||r^*) = b^*$.

– Fujisaki and Okamoto [14]: if $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$ is a public encryption scheme (that is CPA) then $\mathcal{E}(\text{in}_e; r) = \mathcal{E}'((\text{in}_e||r); H(\text{in}_e||r))$. The "sanity check" of the decryption algorithm is: $\mathcal{E}'(m^*||r^*; H(m^*||r^*)) = \text{in}_d$.

The PA-semantic criterion applies to the following constructions but not the syntactic one:

Example 7.

– Pointcheval [17]:
$\mathcal{E}(\text{in}_e; r; s) = f(r||H(\text{in}_e||s))||((\text{in}_e||s) \oplus G(r))$, where f is a partially trapdoor one-way injective function. The "sanity check" of the decryption oracle $\mathcal{D}(a||b)$ has the form $f(r^*||H(m^*||s^*)) = a^*$. The tester looks in hG and hH for r^* and $m^*||s^*$ such that $\mathcal{E}(m^*; r^*; s^*) = y$.

– REACT [16]: This construction applies to any trapdoor one-way function (possibly probabilistic). It is quite similar to the construction in [7]: $\mathcal{E}(\text{in}_e; R; r) = a||b||c = f(R; r)||\text{in}_e \oplus G(r)||H(R||\text{in}_e||a||b)$, where $a = f(R; r)$ and $b = \text{in}_e \oplus G(R)$. The "sanity check" of the decryption algorithm is $H(R^*||m^*||a^*||b^*) = c$. For this construction, one can provide a tester $\mathcal{T}$ that uses a PCA oracle to check whether a is the encryption of R by f. Hence, the PA security of the construction under the assumption of the OW-PCA security of f. The tester looks in hH for $R^*||m^*||a^*||b^*$ such that $c^* = H(R^*||m^*||a^*||b^*)$ and $a^* = f(R^*)$, which can be checked using the CPA-oracle.

And now some examples of constructions that do not satisfy the PA-semantic criterion (and hence, not the syntactic one):

Example 8. – Zheng-Seberry Scheme [22]:
$\mathcal{E}(x; r) = a||b = f(r)||(G(r) \oplus (x||H(x))$. The third condition of the PA-semantic criterion is not satisfied by this construction. Actually, there is an attack [20] on the IND-CCA security of this scheme that exploits this fact.

– OAEP [5]: $\mathcal{E}(\text{in}_e; r) = a = f(\text{in}_e||0^k \oplus G(r)||r \oplus H(s))$, where $s = \text{in}_e||0^k \oplus G(r)$. Here the third condition is not satisfied.

5 Automation

We can now fully automate our verification procedure of IND-CCA for the encryption schemes we consider as follows:

1. Automatically establish invariants

2. Check the syntactic criterion for PA.

Point 2 can be done by a simple syntactic analyzer taking as input the decryption program, but has not been implemented yet.

Point 1 is more challenging. The idea is, for a given program, to compute invariants backwards, starting with the invariant $\mathsf{Indis}(\nu\mathrm{out}_e; \mathrm{out}_e, \mathrm{in}_e)$ at the end of the program.

As several rules can lead to a same postcondition, we in fact compute a set of sufficient conditions at all points of the program: for each set $\{\phi_1, \ldots, \phi_n\}$ and each instruction c, we can compute a set of assertions $\{\phi'_1, \ldots, \phi'_m\}$ such that

1. for $i = 1, \ldots, m$, there exists j such that $\{\phi'_i\}c\{\phi_j\}$ can be derived using the rules given section 3.2,
2. and for all j and all ϕ' such that $\{\phi'\}c\{\phi_j\}$, there exists i such that ϕ' entails ϕ'_i and that this entailment relation can be derived using lemma 9.

Of course, this verification is potentially exponential in the number of instructions of the encryption program as each postcondition may potentially have several preconditions. However this is mitigated as

- the considered encryption scheme are generally implemented in a few instructions (around 10)
- we implement a simplification procedure on the computed set of invariants: if ϕ_i entails ϕ_j (for $i \neq j$), then we can safely delete ϕ_i from the set of assertions $\{\phi_1, \ldots, \phi_n\}$. In other words, we keep only the minimal preconditions with respect to strength in our computed set of invariants (the usual Hoare logic corresponds to the degenerated case where this set has a minimum element, called the weakest precondition).

In practice, checking Bellare & Rogaway generic construction is instantaneous.

We implemented that procedure as an Objective Caml program, taking as input a representation of the encryption program. This program is only 230 lines long and is available on the web page of the authors.

6 Conclusion

In this paper we proposed an automatic method to prove IND-CCA security of generic encryption schemes in the random oracle model. IND-CPA is proved using a Hoare logic and plaintext awareness using a syntactic criterion. It does not seem difficult to adapt our Hoare logic to allow a security proof in the concrete framework of provable security. Another extension of our Hoare logic could concern OAEP. Here, we need to express that the value of a given variable is indistinguishable from a random value as long as a value r has not been submitted to a hash oracle G. This can be done by extending the predicate $\mathsf{Indis}(\nu x; V_1; V_2)$. The details are future work.

References

1. G. Barthe, J. Cederquist, and S. Tarento. A Machine-Checked Formalization of the Generic Model and the Random Oracle Model. In *IJCAR*, pages 385–399, 2004.
2. G. Barthe, B. Grégoire, R. Janvier, and S. Zanella Béguelin. A framework for language-based cryptographic proofs. In *ACM SIGPLAN Workshop on Mechanizing Metatheory*, 2007.

3. G. Barthe and S. Tarento. A machine-checked formalization of the random oracle model. In *Proceedings of TYPES'04*, volume 3839, pages 33–49. Springer, 2004.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO'98*, pages 26–45, 1998.
5. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT'04*, volume 950 of *LNCS*, pages 92–111, 1994.
6. M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004.
7. Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS'93*, pages 62–73, 1993.
8. B. Blanchet. A computationally sound mechanized prover for security protocols. In *S&P'06*, pages 140–154, 2006.
9. B. Blanchet and D. Pointcheval. Automated security proofs with sequences of games. In *CRYPTO'06*, volume 4117, pages 537–554, 2006.
10. R. Corin and J. den Hartog. A probabilistic hoare-style logic for game-based cryptographic proofs. In *ICALP'06*, pages 252–263, 2006.
11. I. Damgard. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO'91*, pages 445–456, 1992.
12. A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *CSFW'06*, pages 321–334, 2006.
13. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptol.*, 1(2):77–94, 1988.
14. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *PKC'99*, pages 53–68, 1999.
15. S. Halevi. A plausible approach to computer-aided cryptographic proofs. ePrint archive report 2005, 2005.
16. T. Okamoto and D. Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA'01*, pages 159–175, 2001.
17. D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *PKC'00*, pages 129–146, 2000.
18. V. Shoup. Oaep reconsidered. *J. Cryptology*, 15(4):223–249, 2002.
19. V. Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004. URL: `http://eprint.iacr.org/2004/332`.
20. D. Soldera, J. Seberry, and C. Qu. The analysis of zheng-seberry scheme. In *ACISP*, volume 2384 of *LNCS*, pages 159–168, 2002.
21. S. Tarento. Machine-checked security proofs of cryptographic signature schemes. In *ESORICS'05*, volume 3679, pages 140–158, 2005.
22. Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *J. on Selected Areas in Communications*, 11(5):715–724, 1993.

Notations used in the appendix.

- Let $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$. We define the distribution $\nu x.X = [(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U} : (S\{x \mapsto u\}, \boldsymbol{H}, (f, f^{-1})]$. Notice that consequently, $X \models \mathsf{Indis}(\nu x; V_1; V_2)$ iff $X \sim_{V_1,V_2} \nu x.X$.
- For a hash function H, we let $H\{x \mapsto y\}$ be the function mapping x to y (or $S(x)$ to $S(y)$) and any other value v to $H(v)$.
- $\mathsf{WS}(x; V_1; V_2)$ is an extension of the WS predicate such that $X \models \mathsf{WS}(x; V_1; V_2)$ iff $X \models \mathsf{WS}(x; V_1, f(V_2))$.

A Soundness of the Hoare Logic

A.1 Preliminaries

The properties Indis and WS are compatible with indistinguishability in the following sense:

Lemma 10. *For any $X, X' \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, any sets of variables V_1 and V_2, and any variable x:*

- *if $X \sim_{V_1;V_2} X'$ then $X \models \mathsf{Indis}(\nu x; V_1; V_2) \iff X' \models \mathsf{Indis}(\nu x; V_1; V_2)$.*
- *if $X \sim_{V_1;V_2,x} X'$ then $X \models \mathsf{WS}(x; V_1; V_2) \iff X' \models \mathsf{WS}(x; V_1; V_2)$.*

Proof. By symmetry of indistinguishability and equivalence, for each proposition, the conclusion follows from a single implication.

- We assume $X \sim_{V_1;V_2} X'$. Hence, $\nu x.X \sim_{V_1;V_2} \nu x.X'$, that can be justified by an immediate reduction. Moreover, the hypothesis $X \models \mathsf{Indis}(\nu x; V_1; V_2)$ implies $X \sim_{V_1;V_2} \nu x.X$. By transitivity of the indistinguishability relation, we get $X' \sim_{V_1;V_2} \nu x.X'$. Thus, $X' \models \mathsf{Indis}(\nu x; V_1; V_2)$.
- We prove that $X \models \mathsf{WS}(x; V_1; V_2) \Rightarrow X' \models \mathsf{WS}(x; V_1; V_2)$ by transposition. Indeed, if $X \models \mathsf{WS}(x; V_1; V_2)$ and $X' \not\models \mathsf{WS}(x; V_1; V_2)$, there exists A a poly-time adversary that, on input V_1 and $f(V_2)$ drawn from X', computes the right value for x with non-negligible probability. We let B be the following adversary against $X \sim_{V_1;V_2,x} X'$:
  $B(V_1, f(V_2), x)$:= let $v := A(V_1, f(V_2))$ in $v \neq x$;;
  The idea is to consider that A computes the right value of x whenever the values were taken from X', otherwise B answers he was provided with values from X. Thus, the advantage of B is greater than that of A, which contradicts $X \sim_{V_1;V_2} X'$.

An expression e is called *constructible from* $(V_1; V_2)$, if it can be constructed from variables in V_1 and images by f of variables in V_2, calling oracles if necessary. Obviously, we can give an inductive definition: if $x \in V_1$ then x is constructible from $(V_1; V_2)$; if $x \in V_2$ then $f(x)$ is constructible from $(V_1; V_2)$; if e_1, e_2 are constructible from $(V_1; V_2)$ then $H(e_1)$, $f(e_1)$, $e_1||e_2$ and $e_1 \oplus e_2$ are constructible from $(V_1; V_2)$. Then Indis is preserved by constructible computings.

Lemma 11. *For any $X, X' \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, any sets of variables V_1 and V_2, any expression e constructible from $(V_1; V_2)$, and any variable x, if $X \sim_{V_1;V_2} X'$ then $[\![x := e]\!](X) \sim_{V_1,x;V_2} [\![x := e]\!](X')$.*

Proof. We assume $X \sim_{V_1;V_2} X'$. If we suppose that $[\![x := e]\!](X) \not\sim_{V_1,x;V_2} [\![x := e]\!](X')$, then there exists A a poly-time adversary that, on input V_1, x and $f(V_2)$ drawn either from $[\![x := e]\!](X)$ or $[\![x := e]\!](X')$, guesses the right initial distribution with non-negligible probability.

We let B be the following adversary against $X \sim_{V_1;V_2} X'$:
$B(V_1, f(V_2))$:= let $x := e$ in $A(V_1, x, f(V_2))$.
The idea is that B can evaluate in polynomial time the expression e using its own inputs. Hence it can provide the appropriate inputs to A. It is clear that the advantage of B is exactly that of A, which would imply that it is not negligible, although we assumed $X \sim_{V_1;V_2} X'$.

Corollary 1. *For any $X, X' \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, any sets of variables V_1 and V_2, any expression e constructible from $(V_1; V_2)$, and any variable x, z such that $z \notin \{x\} \cup \mathsf{Var}(e)$ if $X \models \mathit{Indis}(\nu z; V_1; V_2)$ then $[\![x := e]\!](X) \models \mathit{Indis}(\nu z; V_1, x; V_2)$. We emphasize here that we suppose that here we use the notation $\mathsf{Var}(e)$ (in its usual sense), that is to say, the variable z does not appear* at all *in e.*

Proof. $X \models \mathsf{Indis}(\nu z; V_1; V_2)$ is equivalent to $X \sim_{V_1;V_2} \nu z.X$. Using Lemma 11 we get $[\![x := e]\!](X) \sim_{V_1,x;V_2} [\![x := e]\!](\nu z.X)$. Since $z \notin \{x\} \cup \mathsf{Var}(e)$ we have that $[\![x := e]\!](\nu z.X) = \nu z.[\![x := e]\!](X)$ and hence $[\![x := e]\!](X) \sim_{V_1,x;V_2} \nu z.[\![x := e]\!](X)$, that is $[\![x := e]\!](X) \models \mathsf{Indis}(\nu z; V_1, x; V_2)$.

Lemma 12. *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, any sets of variables V_1 and V_2, any expression e constructible from $(V_1; V_2)$, and any variable $x \neq z$, if $X \models \mathit{WS}(z; V_1; V_2)$ then $[\![x := e]\!](X) \models \mathit{WS}(z; V_1, x; V_2)$.*

Proof. If we suppose that $[\![x := e]\!](X) \not\models \mathsf{WS}(z; V_1, x; V_2)$, then there exists A a poly-time adversary that, on input V_1, x and $f(V_2)$ drawn from $[\![x := e]\!](X)$, computes the right value for z with non-negligible probability.
We let B be the following adversary against $X \models \mathsf{WS}(z; V_1; V_2)$:
$B(V_1, f(V_2))$:= let $x := e$ in $A(V_1, x, f(V_2))$.
Since A and B have the same advantage, we obtain a contradiction, so that $[\![x := e]\!](X) \not\models \mathsf{WS}(z; V_1, x; V_2)$ cannot be true.

A.2 Weakening Lemmas

Lemma 13. *Let $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ be a distribution:*

1. *If $X \models \mathit{Indis}(\nu x; V_1; V_2)$, $V_1' \subseteq V_1$ and $V_2' \subseteq V_1 \cup V_2$ then $X \models \mathit{Indis}(\nu x; V_1'; V_2')$.*
2. *If $X \models \mathit{WS}(x; V_1; V_2)$ and $V_1' \subseteq V_1$ and $V_2' \subseteq V_1 \cup V_2$ then $X \models \mathit{WS}(x; V_1'; V_2')$.*
3. *If $X \models \mathit{Indis}(\nu x; V_1; V_2 \cup \{x\})$ and $x \notin V_1 \cup V_2$ then $X \models \mathit{WS}(x; V_1; V_2 \cup \{x\})$.*

Proof. The first pair of properties are quite straightforward: they are based on the fact that an adversary provided with less values of variables obviously turns out less powerful.
To prove the last assertion, we let $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ such that $X \models \mathsf{Indis}(\nu x; V_1; V_2, x)$. Thus, $X \sim_{V_1;V_2 \cup \{x\}} \nu x.X$, so that with lemma 10, proving $\nu x.X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$ allows to conclude. Besides, considering an adversary A against WS possibly querying f or the hashing functions, $\mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} \nu x.X; A(S(V_1), S(f(V_2 \cup \{x\})), \boldsymbol{H}, f) = S(x)] =$
$\mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; v \xleftarrow{r} \mathcal{U}; A(S(V_1), S(f(V_2)), f(v), \boldsymbol{H}, f) = v]$, and the last probability is the advantage of an adversary trying to invert the one-way function f. This latter being negligible, we conclude that $X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$.

A.3 Generic preservation rules

In this part, we assume that $z \neq x$ and c is either $x \overset{r}{\leftarrow} \mathcal{U}$ or $x := y||t$ or $x := y \oplus t$ or $x := f(y)$ or $x := H(y)$ or $x := t \oplus H(y)$. Moreover, $x \notin V_1 \cup V_2$.

Before getting started, let us notice that for any command c stated above, $[\![\mathsf{c}]\!]$ affects at most x and $\mathbb{T}_H$, in following sense: for any $(S, \boldsymbol{H}, (f, f^{-1}))$, for any $(S', \boldsymbol{H'}, (f', f'^{-1}))$ drawn in $[\![\mathsf{c}]\!](S, \boldsymbol{H}, (f, f^{-1}))$, $\boldsymbol{H'} = \boldsymbol{H}$, $f' = f$, $f'^{-1} = f^{-1}$, and S' disagree with S at most on x and $\mathbb{T}_H$. The next lemma directly follows from this remark.

Lemma 14. *For all distribution $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ and all sets V_1 and V_2 such that $x \notin V_1 \cup V_2$, we have $[\![\mathsf{c}]\!](X) \sim_{V_1;V_2} X$.*

Proof. Let $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$.

$$\begin{aligned}
D(\mathsf{c}(X), V_1, V_2) &= D([(S, \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} [\![\mathsf{c}]\!](X) : (S, \boldsymbol{H}, (f, f^{-1}))], V_1, V_2) \\
&= [(S, \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} X; (S', \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} \mathsf{c}((S, \boldsymbol{H}, (f, f^{-1}))) : \\
&\quad (S'(V_1), f(S'(V_2)), \boldsymbol{H}, f)] \\
&= [(S, \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} X : (S(V_1), f(S(V_2)), \boldsymbol{H}, f)] \\
&\quad \text{since } x \notin V_1 \cup V_2 \\
&= D(X, V_1, V_2)
\end{aligned}$$

Proposition 4 (Rule G1). $\{\mathit{Indis}(\nu z; V_1; V_2)\}\ c\ \{\mathit{Indis}(\nu z; V_1; V_2)\}$

Proof. As $x \notin V_1 \cup V_2$, the previous lemma entails $[\![\mathsf{c}]\!](X) \sim_{V_1;V_2} X$. Then, according to the preservation of properties through indistinguishability proved in lemma 10, $X \models \mathsf{Indis}(\nu z; V_1; V_2)$ implies $[\![\mathsf{c}]\!](X) \models \mathsf{Indis}(\nu z; V_1; V_2)$.

Proposition 5 (Rule G2). $\{\mathit{WS}(z; V_1)\}\ c\ \{\mathit{WS}(z; V_1)\}$

Proof. This rule follows from the very same reasoning as the previous one.

Proposition 6 (Rule G3). $\{H(H', e[e'/x])\}\ x := e'\ \{H(H', e)\}$, *provided $H' \neq H$ in case $e' \equiv H(y)$. Here, $e[e'/x]$ is the expression obtained from e by replacing x by e'.*

Proof. Consider any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ and any $[\![\mathsf{c}]\!]$ affecting at most x and $\mathbb{T}_H$ such that $X \models \mathsf{H}(H', e[e'/x])$. Let $p = \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} [\![\mathsf{c}]\!](X) : S(e) \in S(\mathbb{T}_{H'}.\mathsf{dom}]$. Then

$$\begin{aligned}
p &= \mathsf{Pr}[(S', \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} X; (S, \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} [\![\mathsf{c}]\!](S', \boldsymbol{H}, (f, f^{-1})) : S(e) \in S(\mathbb{T}_{H'}.\mathsf{dom}] \\
&= \mathsf{Pr}[(S', \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} X; (S, \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} [\![\mathsf{c}]\!](S', \boldsymbol{H}, (f, f^{-1})) : S'(e[e'/x]) \in S(\mathbb{T}_{H'}.\mathsf{dom}] \\
&\quad \text{since } S'(e[e'/x]) \text{ is equal to } S(e) \\
&= \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \overset{r}{\leftarrow} X : S'(e[e'/x]) \in S(\mathbb{T}_{H'}.\mathsf{dom}]
\end{aligned}$$

Since $X \models \mathsf{H}(H', e[e'/x])$, the last probability is negligible. Therefore p is negligible and $[\![\mathsf{c}]\!](X) \models \mathsf{H}(H', e)$.

A.4 Random Assignment Rules

Proposition 7 (Rule R1). *Let $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$. For any variable x, $[\![x \xleftarrow{r} \mathcal{U}]\!](X) \models \{\mathsf{Indis}(\nu x)\}$.*

Proof. This rule is just a way to state that $\nu x.X \models \mathsf{Indis}(\nu x)$, for the distributions coincide.

Proposition 8 (Rule R2). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, expression e and variable x such that $x \in \mathsf{subvar}(e)$, $[\![x \xleftarrow{r} \mathcal{U}]\!](X) \models \mathsf{H}(H, e)$.*

Proof. The fact that $x \in \mathsf{subvar}(e)$ means that there exists a poly-time function g such that $g(S(e)) = S(x)$ for any state S (namely g consists in extracting the right substring corresponding to x from the expression e). We are interested in bounding

$$\begin{aligned}
&\mathsf{Pr}[S \xleftarrow{r} [\![x \xleftarrow{r} \mathcal{U}]\!](X) : S(e) \in S(\mathbb{T}_H).\mathsf{dom}] \\
&= \mathsf{Pr}[S \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U}; S' := S\{x \mapsto u\} : S'(e) \in S'(\mathbb{T}_H).\mathsf{dom}] \\
&= \mathsf{Pr}[S \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U}; S' := S\{x \mapsto u\} : S'(e) \in S(\mathbb{T}_H).\mathsf{dom}] \\
&= \mathsf{Pr}[S \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U} : u \in g(S(\mathbb{T}_H).\mathsf{dom})]
\end{aligned}$$

which is negligible for the cardinal of $\mathbb{T}_H$ is bounded by a polynomial.

Proposition 9 (Rule R3). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, V_1, V_2 sets of variables, and y and x, such that $x \neq y$, if $X \models \mathsf{Indis}(\nu y; V_1; V_2)$, then $[\![x \xleftarrow{r} \mathcal{U}]\!](X) \models \mathsf{Indis}(\nu y; V_1, x; V_2)$.*

Proof. The intuition is that x being completely random, providing its value to the adversary does not help this latter in any way. We show the result by reduction. Assume that there exists an adversary B against $[\![x \xleftarrow{r} \mathcal{U}]\!](X) \models \mathsf{Indis}(\nu y; V_1, x; V_2)$ that can distinguish with non-negligible advantage between y and a random value given the values of V_1, x and $f(V_2)$. Then, we can construct an adversary $A(V_1, f(V_2))$ playing against $X \models \mathsf{Indis}(\nu y; V_1; V_2)$ that has the same advantage as B: $A(V_1, f(V_2))$ draws a value u at random and runs $B(V_1, u, f(V_2))$, and then returns B's answer. If B has non-negligible advantage, then so does A, which contradicts our hypothesis.

Proposition 10 (Rule R4). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, V set of variables, and y and x, such that $x \neq y$, if $X \models \mathsf{WS}(y; V)$, then $[\![x \xleftarrow{r} \mathcal{U}]\!](X) \models \mathsf{WS}(y; V, x)$.*

Proof. The previous reduction can be adapted in a straightforward way to prove this fourth rule.

A.5 Hash Rules

Preliminary Results In the random oracle model, hash functions are drawn uniformly at random from the space of functions of suitable type at the beginning of the execution of a program. Thus, the images that the hash function associates to different inputs are completely independent. Therefore, one can delay the draw of each hash value until needed. This is the very idea that the first lemma formalizes. It states that while a hash value has not been queried by the adversary, i.e. while $\mathsf{H}(H, y)$ remains true, then one can redraw it without this changing anything from the adversary's point of view.

Lemma 15 (Dynamic draw). *Let $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$. For any y such that $X \models \mathsf{H}(H, y)$, $X \sim [(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \boldsymbol{H} \xleftarrow{r} \Omega; S \xleftarrow{r} A^{\boldsymbol{H}, (f, f^{-1})}(); u \xleftarrow{r} \mathcal{U} : (S, H\{y \mapsto u\} \cup (\boldsymbol{H} - H), (f, f^{-1}))]$.*

Proof. To simplify, we prove the lemma considering programs using only one hash function. By definition of $\textsc{Dist}(\Gamma, H, \mathbb{F})$, $X = [(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \boldsymbol{H} \xleftarrow{r} \Omega; S \xleftarrow{r} A^{H,(f,f^{-1})}() : (S, H, (f, f^{-1}))]$. We can decompose the adversary A possibly querying the hash oracle in a polynomial number of adversaries that do not access the oracle. Indeed, it only requires to externalize the queries of A to the oracle to deal with them 'manually'. Thus, we can rewrite the distribution the following way:
$[(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); H \xleftarrow{r} \Omega; S \xleftarrow{r} A^{\boldsymbol{H},(f,f^{-1})}() : (S, H, (f, f^{-1}))] =$
$[(S_0, q_0) \xleftarrow{r} A_0^{(f,f^{-1})}(); r_0 := H(q_0); \mathbb{T}_H := \mathbb{T}_H.(q_0, r_0);$
$(S_1, q_1) \xleftarrow{r} A_1^{(f,f^{-1})}(S_0, q_0, r_0); \ldots; (S_n, q_n) \xleftarrow{r} A_n^{(f,f^{-1})}(S_{n-1}, q_{n-1,r_{n-1}});$
$r_n := H(q_n); \mathbb{T}_H := \mathbb{T}_H.(q_n, r_n); S \xleftarrow{r} A_{n+1}^{(f,f^{-1})}(S_n, q_n, r_n) : (S, H, (f, f^{-1}))]$
where n is polynomial in the security parameter. Now, instead of drawing H at the beginning of the execution, we draw its outputs as the program goes along. The hash values not queried for by the adversary are all drawn at the end.
$X = [(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); (S_0, q_0) \xleftarrow{r} A^{(f,f^{-1})}(); \nu r_0; \mathbb{T}_H := \mathbb{T}_H.(q_0, r_0);$
$(S_1, q_1) \xleftarrow{r} A^{(f,f^{-1})}(S_0, q_0, r_0); \ldots; (S_n, q_n) \xleftarrow{r} A^{(f,f^{-1})}(S_{n-1}, q_{n-1,r_{n-1}});$
$\nu r_n; \mathbb{T}_H := \mathbb{T}_H.(q_n, r_n); S \xleftarrow{r} A^{(f,f^{-1})}(S_n, q_n, r_n);$
$\forall u \in^c (\mathbb{T}_H.dom), \nu r_u$ and $H := H\{u \mapsto r_u\}$[4]: $(S, H, (f, f^{-1}))]$.
To conclude, let us notice that our hypothesis $X \models \mathsf{H}(H, y)$ implies that with overwhelming probability, $y \in^c (\mathbb{T}_H.dom)$, so that its hash value can be drawn last (or redrawn, which is strictly equivalent since the results of the draws do not appear in $\mathbb{T}_H$).

We now want to prove something a little stronger, involving the variable $\mathbb{T}_H$. Indeed, to execute the command $x := \alpha \oplus H(y)$, we can either draw a value for $H(y)$ at random and bind it by storing it in $\mathbb{T}_H$, or draw x at random and bind $H(y)$ to be worth $x \oplus \alpha$. This uses the same idea as before, but this time we have to carefully take into account the side effects of the command on $\mathbb{T}_H$. To deal with rebinding matters, we introduce a new notation.

Definition 7. *We define* $\mathsf{rebind}_H^{y \mapsto x}(S, \boldsymbol{H}, (f, f^{-1})) = (S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \cdot (S(y), S(x))\}, H\{S(y) \mapsto S(x)\}, (f, f^{-1}))$ *and extend this definition canonically to any distribution* $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$*:* $\mathsf{rebind}_H^{y \mapsto x}(X) = [(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : \mathsf{rebind}_H^{y \mapsto x}(S, \boldsymbol{H}, (f, f^{-1}))]$*. It simply denotes the distribution where* $H(S(y))$ *is defined to be worth* $S(x)$*.*

Lemma 16 (Rebinding Lemma). *For any* $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$*, any hash function symbol* H*, any variables* x *and* y*, if* $X \models \mathsf{H}(H, y)$*, then*

$$[\![x := \alpha \oplus H(y)]\!](X) \sim \mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X),$$

where α *is either a constant or a variable.*

Proof. To lighten the proof, we assume without loss of generality that there is only one hash function H. Moreover, we omit to write the draw of f and its inverse, and we do not mention them in the description of the states either. First, since $X \models \mathsf{H}(H, y)$, thanks to the dynamic draw lemma, we know that $X \sim [H \xleftarrow{r} \Omega; S \xleftarrow{r} A^{H,(f,f^{-1})}(); u \xleftarrow{r} \mathcal{U} : (S, H\{y \mapsto u\})]$.

[4] written this way, this might suggest an exponential time computation. In fact, the exact thing we do is that we draw another hash function H' on the domain $^c(\mathbb{T}_H.dom)$ and extend H to be worth H' on that domain. The real time of computation thus remains polynomial, but we chose the 'simplest' way to denote our probability distribution.

Then, we apply lemma 10:
$[\![x := \alpha \oplus H(y)]\!](X) \sim [\![x := \alpha \oplus H(y)]\!]([H \xleftarrow{r} \Omega; S \xleftarrow{r} A^{H,(f,f^{-1})}(); u \xleftarrow{r} \mathcal{U} : (S, H\{y \mapsto u\})])$.
Executing the hash command, the second distribution is in turn equal to
$[H \xleftarrow{r} \Omega; S \xleftarrow{r} A^{H,(f,f^{-1})}(); u \xleftarrow{r} \mathcal{U} : (S\{x \mapsto \alpha \oplus u\}, H\{y \mapsto u\})]$.
Then, we eventually replace the draw of y by that of x, and propagate the side effects of that change, to obtain another way to denote the same distribution:
$[H \xleftarrow{r} \Omega; S \xleftarrow{r} A^{H,(f,f^{-1})}(); v \xleftarrow{r} \mathcal{U} : (S\{x \mapsto v\}, H\{y \mapsto v \oplus \alpha\})]$.
Now, this last distribution is exactly $\mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X)$, and we conclude.

Now we are interested in formally proving the useful and intuitive following lemma, which states that to distinguish between a distribution and its 'rebound' version, an adversary must be able to compute the argument y whose hash value has been rebound. More precisely,

Lemma 17 (Hash vs rebind). *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, any two variables x and y, any two finite sets of variables V_1 and V_2, and any hash function H, if $X \models \mathsf{WS}(y; V_1; V_2)$, then*

$$X \sim_{V_1;V_2} \mathsf{rebind}_H^{y \mapsto x}(X)$$

.

Proof. Consider finite sets V_1 and V_2 and a distribution X such that $X \models \mathsf{WS}(y; V_1; V_2)$. Once more, we omit to mention (f, f^{-1}) in the state descriptions. The sole difference between the distributions is the value of $H(y)$. Namely,

$$\begin{aligned}
D(\mathsf{rebind}_H^{y \mapsto x}(X), V_1, V_2) = [&(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; \\
&S' \leftarrow S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \cdot (S(y), S(x))\} : \\
&(S'(V_1), f(S'(V_2)), H\{S(y) \mapsto S(x)\})] \\
= [&(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; \\
&S' \leftarrow S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \cdot (S(y), S(x))\} : \\
&(S(V_1), f(S(V_2)), H\{S(y) \mapsto S(x)\})] \\
&\text{and since } S' \text{ is not used anywhere,} \\
= [&(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : \\
&(S(V_1), f(S(V_2)), H\{S(y) \mapsto S(x)\})]
\end{aligned}$$

An adversary trying to distinguish $D(X, V_1, V_2)$ from this last distribution can only succeed if it calls H on $S(y)$. Nevertheless, the probability of an adversary computing $S(y)$ is negligible since $X \models \mathsf{WS}(y; V_1; V_2)$. Therefore, $D(\mathsf{rebind}_H^{y \mapsto x}(X), V_1, V_2) \sim D(X, V_1, V_2)$.

Proofs of the rules Thanks to the three lemmas of the previous part, that capture the important features of hash functions, we can know design fairly simple proofs of the soundness of our rules.

Proposition 11 (Rule H1). *Let $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$. When $x \neq y$, and α is either a constant or a variable:*

$$\{\mathsf{WS}(y; V) \wedge \mathsf{H}(H, y)\}x := \alpha \oplus \mathsf{H}(y)\{\mathsf{Indis}(\nu x; V, x)\}$$

Proof. First, we use that $X \models \mathsf{WS}(y; V)$, that provides thanks to the standard preservation rule (G1) $\nu x.X \models \mathsf{WS}(y; V, x)$. Hence, y being practically secret to the adversary, the 'hash-vs-rebind' lemma applies, so that $\nu x.X \sim_{V,x} \mathsf{rebind}_H^{y \mapsto x}(\nu x.X)$. Of course, we can replace x by $x \oplus \alpha$ as a value for $H(y)$. Indeed, we would like to have $\nu x.X \sim_{V,x} \mathsf{rebind}_H^{y \mapsto x \oplus \alpha}(\nu x.X)$. Formally, as the weak secrecy property of y is inherited by $\mathsf{rebind}_H^{y \mapsto x}(X)$, we can see that as an additional application of lemma 17 (with this time $x \oplus \alpha$ in the role of x). Then, as we assumed that $X \models \mathsf{H}(H, y)$, we can use the rebinding lemma, according to which we have $\mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X) \sim_{V,x} [\![x := \alpha \oplus H(y)]\!](X)$. By transitivity of the indistinguishability relation, we thus have $\nu x.X \sim_{V,x} [\![x := \alpha \oplus H(y)]\!](X)$, which is equivalent to $[\![x := \alpha \oplus H(y)]\!](X) \models \mathsf{Indis}(x; V, x)$.

Proposition 12 (Rule H2). *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, if $x \neq y$, then*

$$\{\mathsf{H}(H, y)\}\ x := H(y)\ \{\mathsf{H}(H', e)\} \text{ provided } x \in \mathsf{subvar}(e)$$

Proof. Consider any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ such that $X \models \mathsf{H}(H, y)$, and let $X' = \mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X)$. Since $X \models \mathsf{H}(H, y)$, the rebinding lemma implies $[\![x := H(y)]\!]X \sim X'$. Consider an expression e such that $x \in \mathsf{subvar}(e)$. It suffices to show $X' \models \mathsf{H}(H', e)$, that is, that $p = \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X' : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom}]$ is negligible.

$$\begin{aligned}
p &= \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} \nu x \cdot X; (S', \boldsymbol{H'}, (f, f^{-1})) \leftarrow \mathsf{rebind}_H^{y \mapsto x}(S, \boldsymbol{H}, (f, f^{-1})) : \\
&\quad S'(e) \in S'(\mathbb{T}_{H'}).\mathsf{dom}] \\
&\quad \text{since } S'(\mathbb{T}_{H'}).\mathsf{dom} = S(\mathbb{T}_{H'}).\mathsf{dom} \cup \{S(y)\}, \text{ we have} \\
&= \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} \nu x \cdot X; (S', \boldsymbol{H'}, (f, f^{-1})) \leftarrow \mathsf{rebind}_H^{y \mapsto x}(S, \boldsymbol{H}, (f, f^{-1})) : \\
&\quad S'(e) \in S(\mathbb{T}_{H'}).\mathsf{dom} \text{ or } S'(e) = S(y)] \\
&\quad \text{now with } S(e) = S'(e) \text{ by definition of the rebinding:} \\
&= \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} \nu x \cdot X; (S', \boldsymbol{H'}, (f, f^{-1})) \leftarrow \mathsf{rebind}_H^{y \mapsto x}(S, \boldsymbol{H}, (f, f^{-1})) : \\
&\quad S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom} \text{ or } S(e) = S(y)] \\
&\quad \text{we can remove the rebinding, since it does not change the event:} \\
&= \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} \nu x \cdot X : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom} \text{ or } S(e) = S(y)] \\
&\quad \text{which is by definition worth} \\
&= \mathsf{Pr}[S_1 \xleftarrow{r} X; v \xleftarrow{r} \mathcal{U}; S \leftarrow S_1\{x \mapsto v\} : S(e) \in S_1(\mathbb{T}_{H'}).\mathsf{dom} \text{ or } S(e) = S_1(y)] \\
&\quad \text{and as } x \in \mathsf{subvar}(e), \text{ i.e. } x \text{ is some substring of } e \\
&\leq \frac{Card(S_1(\mathbb{T}_{H'}).\mathsf{dom}) + 1}{2^{|x|}}
\end{aligned}$$

Moreover, for all state S_1, $Card(S_1(\mathbb{T}_{H'}).\mathsf{dom})$ is bounded by a polynomial, so that p is negligible.

Proposition 13 (Rule H3). *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, if $x \neq y$ and $y \notin V$, then*

$$\{\mathsf{Indis}(\nu y; V; V', y) \wedge \mathsf{H}(H, y)\}\ x := H(y)\ \{\mathsf{Indis}(\nu x; V, x; V', y)\}$$

Proof. Consider any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$. Assume $y \notin V$ and $X \models \mathsf{Indis}(\nu y; V; V', y) \wedge \mathsf{H}(H, y)$. Then, $X \models \mathsf{WS}(y; V; V', y)$ follows from the third weakening lemma (see lemma 9). Consequently, rule H1 provides $[\![x := H(y)]\!](X) \models \mathsf{Indis}(\nu x; V, x; V', y)$.

Proposition 14 (Rule H4). *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, if $x \neq y$ and $x \neq z$ then*

$$\{\mathsf{WS}(z; V) \wedge \mathsf{WS}(y; V) \wedge \mathsf{H}(H, y)\}\ x := H(y)\ \{\mathsf{WS}(z; V, x)\}$$

Proof. First, we apply rule R4, to state that since $X \models \mathsf{WS}(y; V)$, $\nu x.X \models \mathsf{WS}(y; V, x)$. Then, from the hash-vs-rebind lemma applied on $\nu x.X$, we obtain that $\nu x.X \sim_{V,x} \mathsf{rebind}_H^{y \mapsto x}(\nu x.X)$. Now, using the assumption $X \models \mathsf{H}(H, y)$ and the rebinding lemma, $\mathsf{rebind}_H^{y \mapsto x}(\nu x.X) \sim_{V,x} [\![x := H(y)]\!](X)$. Hence, $\nu x.X \sim_{V,x} [\![x := H(y)]\!](X)$. Besides, as $X \models \mathsf{WS}(z; V)$, rule R4 provides the conclusion $\nu x.X \models \mathsf{WS}(z; V, x)$. With lemma 10, we can conclude that $[\![x := H(y)]\!](X) \models \mathsf{WS}(z; V, x)$ too.
We could do this proof by reduction too, the main idea being that as the value of x is random to an adversary, any adversary against $\mathsf{WS}(z; V)$ before the execution of the command could simulate an adversary against $\mathsf{WS}(z; V, x)$ by providing this latter with a randomly sampled value in place of x. Both those adversaries would therefore have the same advantage.

Proposition 15 (Rule H5). *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, if $x \neq y$ and $x \neq z$ then*

$$\{\mathsf{H}(H, e) \wedge \mathsf{WS}(z; y)\}\ x := H(y)\ \{\mathsf{H}(H, e)\} \text{ provided } z \in \mathsf{subvar}(e) \text{ and } x \notin \mathsf{subvar}(e)$$

is sound.

Proof. Since $z \in \mathsf{subvar}(e)$, there is a polynomial function g such that for all S, $g(S(e)) = S(z)$ (namely g consists in extracting the right substring corresponding to x from the expression e). Given a distribution $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, let

$$p = \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (S', \boldsymbol{H'}, (f, f^{-1})) \xleftarrow{r} [\![x := H(y)]\!](S) : S'(e) \in S'(\mathbb{T}_H).\mathsf{dom}]$$

Then, since the command only affects x and $\mathbb{T}_H$,

$$\begin{aligned} p &= \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (S', \boldsymbol{H'}, (f, f^{-1})) \xleftarrow{r} [\![x := H(y)]\!](S) : \\ &\qquad S(e) \in S(\mathbb{T}_H).\mathsf{dom} \cup \{S(y)\}] \\ &\leq \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : S(e) \in S(\mathbb{T}_H)] + \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : S(e) = S(y)] \end{aligned}$$

Now, we can bound the second term as follows:

$$\begin{aligned} \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : S(e) = S(y)] &\leq \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : g(S(e)) = g(S(y))] \\ &\qquad \text{since } g \text{ consists in taking a substring} \\ &= \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : S(z) = g(S(y))] \\ &\qquad \text{for this is how } g \text{ was defined} \end{aligned}$$

Besides, $X \models \mathsf{WS}(z; y)$, so that the probability one can extract the value of z from that of y is negligible. Moreover if $X \models \mathsf{H}(H, e)$ then $\mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X : S(e) \in S(\mathbb{T}_H)]$ is negligible.

Proposition 16 (Rule H6). *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, if $y \notin V_1$ then*

$$\{\mathit{Indis}(\nu y; V_1; V_2, y) \wedge \mathsf{H}(H, y)\}\ x := H(y)\ \{\mathit{Indis}(\nu y; V_1, x; V_2, y)\}$$

Proof. Consider any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$. Assume $y \notin V_1$, and $X \models \mathsf{Indis}(\nu y; V_1, y; V_2) \wedge \mathsf{H}(H, y)$. By rule R3 for random assignment, $\nu x \cdot X \models \mathsf{Indis}(\nu y; V_1, x; V_2, y)$. Therefore, by the third weakening lemma $\nu x \cdot X \models \mathsf{WS}(\nu y; V_1, x; V_2, y)$, so that the hash-vs-rebind lemma provides us with $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \sim_{V_1, x; V_2, y} \nu x \cdot X$. Thus, $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \models \mathsf{Indis}(\nu y; V_1, x; V_2, y)$ by lemma 10. Since $X \models \mathsf{H}(H, y)$, by the rebinding lemma, we have $[\![x := H(y)]\!] \sim \mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X)$, so that the result follows from applying once more lemma 10.

Proposition 17 (Rule H7). *For any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, if $z \neq x$ then*

$$\{\mathit{Indis}(\nu z; V_1, z; V_2) \wedge \mathit{WS}(y; V_1 \cup V_2, z) \wedge \mathit{H}(H, y)\} x := H(y) \{\mathit{Indis}(\nu z; V_1, z, x; V_2)\}$$

Proof. Consider any $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$ such that $X \models \mathsf{Indis}(\nu z; V_1, z; V_2) \wedge \mathsf{WS}(y; V_1 \cup V_2, z) \wedge \mathsf{H}(H, y)$.

By rule R3 and R4 for random assignment, $\nu x \cdot X \models \mathsf{Indis}(\nu z; V_1, z, x; V_2) \wedge \mathsf{WS}(y; V_1 \cup V_2, z, x)$. Then $\nu x \cdot X \models \mathsf{WS}(y; V_1, z, x; V_2)$ by the weakening lemma on WS. Therefore, the hash-vs-rebind lemma allows to conclude that $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \sim_{V_1, z, x; V_2} \nu x \cdot X$. Thus, by the preservation of properties on indistinguishable distributions, $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \models \mathsf{Indis}(\nu z; V_1, z, x; V_2)$. Eventually, the rebinding lemma entails $[\![x := H(y)]\!](X) \sim \mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X)$. Therefore $[\![x := H(y)]\!](X) \models \mathsf{Indis}(\nu z; V_1, z, x; V_2)$, once more by the preservation lemma 10.

A.6 One-way rules

Proposition 18 (Rule O1). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V, variables x, and y, such that $y \notin V \cup \{x\}$, if $X \models \mathit{Indis}(\nu y; V; y)$, then $[\![x := f(y)]\!](X) \models \mathit{WS}(y; V, x)$.*

Proof. Let X be a distribution such that $X \models \mathsf{Indis}(\nu y; V; y)$. It follows from lemma 13 that $X \models \mathsf{WS}(y; V; y)$. Since $f(y)$ is obviously constructible from $(V; y)$, we apply lemma 12, to obtain $[\![x := f(y)]\!](X) \models \mathsf{WS}(y; V, x)$. Notice that the one-wayness of f is not used apparently here. Indeed, the proof of the weakening lemma uses it, and once we apply it, there is only a simple rewriting step left to be able to conclude.

Proposition 19 (Rule O2). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V_1 and V_2, variables x, y and z, such that $z \neq y$ and $z \neq x$, if $X \models \mathit{Indis}(\nu z; V_1, z; V_2, y)$, then $[\![x := f(y)]\!](X) \models \mathit{Indis}(\nu z; V_1, z, x; V_2, y)$.*

Proof. Since $f(y)$ is constructible from $(V_1, z; V_2, y)$, we apply corollary 1 to obtain $[\![x := f(y)]\!](X) \models \mathsf{Indis}(\nu z; V_1, z, x; V_2, y)$.

Proposition 20 (Rule O3). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V, variables x, y and z, such that $z \neq x$, if $X \models \mathit{WS}(z; V) \wedge \mathit{Indis}(\nu y; V; y, z)$, then $[\![x := f(y)]\!](X) \models \mathit{WS}(z; V, x)$.*

Proof. If $z = y$, then the assertion is a consequence of proposition 18. Hence we assume $z \neq y$. From $X \models \mathsf{Indis}(\nu y; V; z, y)$ it follows by definition that $X \sim_{V; z, y} \nu y.X$. Using lemma 11 we get $[\![x := f(y)]\!](X) \sim_{V, x; z, y} [\![x := f(y)]\!](\nu y.X)$. Now using lemma 10, it suffices to prove $[\![x := f(y)]\!](\nu y.X) \models \mathsf{WS}(z; V, x)$. Intuitively, this comes from the randomness of x, that allows us to think it is useless to any adversary trying to compute z. Formally, we want to show that: $\mathsf{Pr}[S \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U}; S_1 = S\{y \mapsto u; x \mapsto f(u)\} : \mathcal{A}(S_1(V), S_1(x)) = S_1(z)]$ is negligible. We have

that $\max_{\mathcal{A}} \mathsf{Pr}[u \xleftarrow{r} \mathcal{U}; S \xleftarrow{r} X; S_1 = S\{y \mapsto u\} : \mathcal{A}(S_1(V), f(S_1(y))) = S_1(z)] \leq \max_{\mathcal{A}} \mathsf{Pr}[S \xleftarrow{r} X : \mathcal{A}(S(V)) = S(z)]$. Indeed, the set of adversaries of the first type is included in the set of adversaries of the second type, since out of any adversary of the second type, one can construct an adversary of the first type (simply simulating the draw of y) that has the same advantage. Eventually, the last advantage is that of adversaries attacking $X \models \mathsf{WS}(z; V)$, which is negligible by hypothesis.

Proposition 21 (Rule P1). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V_1,V_2, variables x and y, such that $y \notin V_1 \cup V_2$, if f is a permutation and $X \models \mathsf{Indis}(\nu y; V_1; V_2, y)$, then $[\![x := f(y)]\!](X) \models \mathsf{Indis}(\nu x; V_1, x; V_2)$.*

Proof. Since $X \models \mathsf{Indis}(\nu y; V_1; V_2, y)$ and $f(y)$ is constructible from $(V_1; V_2, y)$, we apply lemma 11 to obtain $[\![x := f(y)]\!](X) \sim_{V_1,x;V_2,y} [\![x := f(y)]\!](\nu y.X)$, and by weakening we get $[\![x := f(y)]\!](X) \sim_{V_1,x;V_2} [\![x := f(y)]\!](\nu y.X)$. Using that f is a permutation and that $y \notin V_1 \cup V_2$, we have $D([\![x := f(y)]\!](\nu y.X), V_1 \cup \{x\}, V_2) = D(\nu x.X, V_1 \cup \{x\}, V_2)$, and hence by transitivity of indistinguishability, $[\![x := f(y)]\!](X) \sim_{V_1,x;V_2} \nu x.X$. Now we use $\nu x.X = \nu x.[\![x := f(y)]\!](X)$ to conclude.

A.7 The Xor operator

Proposition 22 (Rule X1). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V_1 and V_2, variables x, y and z, such that $y \notin V_1 \cup V_2 \cup \{z\}$, if $X \models \mathsf{Indis}(\nu y; V_1, y, z; V_2)$, then $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu x; V_1, x, z; V_2)$.*

Proof. Let X be a distribution such that $X \models \mathsf{Indis}(\nu y; V_1, y, z; V_2)$, which we can rewrite $X \sim_{V_1,y,z;V_2)} \nu y.X$. Moreover, $y \oplus z$ is constructible from $(V_1, y, z; V_2)$. We apply lemma 11 to obtain $[\![x := y \oplus z]\!](X) \sim_{V_1,x,y,z;V_2} [\![x := y \oplus z]\!](\nu y.X)$, and by weakening it we get $[\![x := y \oplus z]\!](X) \sim_{V_1,x,z;V_2} [\![x := y \oplus z]\!](\nu y.X)$.

$$
\begin{aligned}
D([\![x := y \oplus z]\!](\nu y.X), V_1 \cup \{x, z\}, V_2) &= [S \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U}; S' := S\{y \mapsto u\}; \\
&\quad S'' \xleftarrow{r} [\![x := y \oplus z]\!](S') : S''(V_1 \cup \{x, z\}), f(S''(V_2))] \\
&= [S \xleftarrow{r} X; u \xleftarrow{r} \mathcal{U}; S' := S\{y \mapsto u\}; \\
&\quad S'' := S'\{x \mapsto u \oplus S(z)\} : S''(V_1 \cup \{x, z\}), f(S''(V_2))] \\
&\quad \text{and since xor is idempotent we can write:} \\
&= [S \xleftarrow{r} X; v \xleftarrow{r} \mathcal{U}; S'' := S\{x \mapsto v; y \mapsto v \oplus S(z)\} : \\
&\quad S''(V_1 \cup \{x, z\}), f(S''(V_2))] \\
&\quad \text{but changing } y \text{ is useless since } y \notin V_1 \cup V_2 \cup \{z\} \\
&= [S \xleftarrow{r} X; v \xleftarrow{r} \mathcal{U}; S'' := S\{x \mapsto v\} : \\
&\quad S''(V_1 \cup \{x, z\}), f(S''(V_2))] \\
&= D(\nu x.X, V_1 \cup \{x, z\}, V_2)
\end{aligned}
$$

Another way to write this equality of distributions is $[\![x := y \oplus z]\!](\nu y.X) \sim_{V_1,x,z;V_2} \nu x.X$. Then, by transitivity of indistinguishability, we can conclude that $[\![x := y \oplus z]\!](X) \sim_{V_1,x,z;V_2} \nu x.X$, which we can rewrite $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu x; V_1, x, z; V_2)$.

Proposition 23 (Rule X2). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V_1 and V_2, variables x, y, z and t, such that $t \notin \{x, y, z\}$, if $X \models \mathsf{Indis}(\nu t; V_1, y, z; V_2)$, then $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu t; V_1, x, y, z; V_2)$.*

Proof. Since $y \oplus z$ is constructible from $(V_1, y, z; V_2)$, we apply corollary 1 to obtain $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu t; V_1, x, y, z; V_2)$.

Proposition 24 (Rule X3). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V, variables x, y, z and t, such that $t \notin \{x, y, z\}$, if $X \models \mathsf{WS}(t; V, y, z)$, then $[\![x := y \oplus z]\!](X) \models \mathsf{WS}(t; V, x, y, z)$.*

Proof. Since $y \oplus z$ is constructible from $(V_1, y, z; V_2)$, we apply lemma 12 to obtain $[\![x := y \oplus z]\!](X) \models \mathsf{WS}(t; V, x, y, z)$.

A.8 The Concatenation operator

Proposition 25 (Rule C1). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V, variables x, y and z, such that $x \notin V$, if $X \models \mathsf{WS}(y; V)$, then $[\![x := y||z]\!](X) \models \mathsf{WS}(x; V)$.*

Proof. From $X \models \mathsf{WS}(y; V)$ we have that for any adversary $\mathcal{A}$, $\mathsf{Pr}[S \xleftarrow{r} X : \mathcal{A}(S(V)) = S(y)]$ is negligible. This implies that for any adversary $\mathcal{B}$, $\mathsf{Pr}[S \xleftarrow{r} X : \mathcal{B}(S(V)) = S(y)||S(z)]$ is negligible; if not we can build an adversary $\mathcal{A}$ that uses $\mathcal{B}$ as a subroutine and whose advantage is the same as $\mathcal{B}$'s advantage. $\mathcal{A}$ calls $\mathcal{B}$ and then uses the answer of $\mathcal{B}$ to extract the value of $S(y)$ from $S(y)||S(z)$. Using that $x \notin V$, we get that for any adversary $\mathcal{B}$, $\mathsf{Pr}[S \xleftarrow{r} [\![x := y||z]\!](X) : \mathcal{B}(S(V)) = S(x)]$ is negligible.

Proposition 26 (Rule C2). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V_1 and V_2, variables x, y and z, such that $y, z \notin (V_1 \cup V_2)$, if $X \models \mathsf{Indis}(\nu y; V_1, y, z; V_2) \wedge \mathsf{Indis}(\nu z; V_1, y, z; V_2)$, then $[\![x := y||z]\!](X) \models \mathsf{Indis}(\nu x; V_1, x; V_2)$.*

Proof. $X \models \mathsf{Indis}(\nu z; V_1, y, z; V_2)$ implies $X \sim_{V_1,y,z;V_2} \nu z.X$, so that in turn $\nu y.X \sim_{V_1,y,z;V_2} \nu y.\nu z.X$. But $X \models \mathsf{Indis}(\nu y; V_1, y, z; V_2)$ can be written as $X \sim_{V_1,y,z;V_2} \nu y.X$. Hence, by transitivity we get $X \sim_{V_1,y,z;V_2} \nu y.\nu z.X$. Since $y||z$ is constructible from $(V_1, y, z; V_2)$, we apply lemma 11 to obtain $[\![x := y||z]\!](X) \sim_{V_1,x,y,z;V_2} [\![x := y||z]\!](\nu y.\nu z.X)$, and by weakening we get $[\![x := y||z]\!](X) \sim_{V_1,x;V_2} [\![x := y||z]\!](\nu y.\nu z.X)$. Using the properties of $||$ and that $\{y, z\} \cap (V_1 \cup V_2) = \emptyset$, we have $D([\![x := y||z]\!](\nu y.\nu z.X), V_1 \cup \{x\}, V_2) = D(\nu x.X, V_1 \cup \{x\}, V_2)$, and hence by transitivity of indistinguishability, $[\![x := y||z]\!](X) \sim_{V_1,x;V_2} \nu x.X$.

Proposition 27 (Rule C3). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V_1 and V_2, variables x, y, z and t, such that $t \notin \{x, y, z\}$, if $X \models \mathsf{Indis}(\nu t; V_1, y, z; V_2)$, then $[\![x := y||z]\!](X) \models \mathsf{Indis}(\nu t; V_1, x, y, z; V_2)$.*

Proof. Since $y||z$ is constructible from $(V_1, y, z; V_2)$, we apply corollary 1 to obtain $[\![x := y||z]\!](X) \models \mathsf{Indis}(\nu t; V_1, x, y, z; V_2)$.

Proposition 28 (Rule C4). *For all $X \in \textsc{Dist}(\Gamma, \boldsymbol{H}, \mathbb{F})$, set of variables V, variables x, y, z and t, such that $t \notin \{x, y, z\}$, if $X \models \mathsf{WS}(t; V, y, z)$, then $[\![x := y||z]\!](X) \models \mathsf{WS}(t; V, x, y, z)$.*

Proof. Since $y||z$ is constructible from $(V_1, y, z; V_2)$, we apply lemma 12 to obtain $[\![x := y||z]\!](X) \models \mathsf{WS}(t; V, x, y, z)$.

B Plaintext Awareness

Theorem 2. *Let GE be a generic encryption scheme that satisfies the PA-semantic criterion. Then, GE is PA-secure.*

Proof. We prove that the probability of K *failing* to simulate the decryption oracle is negligible rather than working with its probability of success: we show that

$$\mathsf{Fail}^{\mathsf{pa}}_{K,B,GE}(\eta, X) = \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}(),\boldsymbol{H}}(f); \\ S'' \xleftarrow{r} [\![\mathcal{D}(y)]\!](S', \boldsymbol{H}, (f, f^{-1})) : y \notin C \wedge K(hH, C, y, f) \neq S''(\mathrm{out}_d)]$$

is negligible. We proceed by decomposing our event in disjoint possible cases and bounding each probability. Indeed, several things can happen for K and $\mathcal{D}$ outputs to be different:

- either $y \notin C$ and $out_K \neq error$ and $S''(\mathrm{out}_d) \neq error$ but $out_K \neq S''(\mathrm{out}_d)$, which we call Event I,
- either $y \notin C$ and $out_K \neq error$ and $S''(\mathrm{out}_d) = error$, which we call Event II,
- or $y \notin C$ and $out_K = error$ and $S''(\mathrm{out}_d) \neq error$, which we call Event III.

To lighten notations, we omit to write $(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}(),\boldsymbol{H}}(f); S'' \xleftarrow{r} [\![\mathcal{D}(y)]\!](S', \boldsymbol{H}, (f, f^{-1}))$ at the beginning of each probability, even though all probabilities are taken on these draws. Moreover, for a set E provided with the uniform distribution, we write $\mathsf{Pr}[\mathcal{U}(E) = t_0]$ instead of $\mathsf{Pr}[t \xleftarrow{r} E : t = t_0]$. In the same way, we denote $\mathsf{Pr}[\mathcal{T}(hH, C, y, cd, f) = 1]$, or even $\mathsf{Pr}[\mathcal{T}(.) = 1]$ when the arguments are clear, to mean $\mathsf{Pr}[b \xleftarrow{r} \mathcal{T}(hH, C, y, cd, f) : b = 1]$. Finally, we bound any negligible probability by a generic function $\epsilon(\eta)$, that tends to 0 when $\eta \to \infty$ even if multiplied by any polynomial in η.

Let us first take care of Event I, that is, when $y \notin C$ and $out_K \neq error$, $S''(\mathrm{out}_d) \neq error$ but $out_K \neq S''(\mathrm{out}_d)$. We consider all possible events that can result in those two different outputs.

- First, the tester $\mathcal{T}$ may answer 1 for a candidate cd whereas $H_1(cd) \neq H_1(S''(t^*))$ or $\mathcal{V}(S''(x), H_1(cd)) \neq S''(v)$. Then, the first condition of our semantic criterion implies that $\mathsf{Pr}[cd \xleftarrow{r} hH_1.\mathsf{dom}; b \xleftarrow{r} \mathcal{T}(hH, C, y, cd, f) : b = 1 \wedge \{H_1(cd) \neq H_1(S''(t^*)) \vee \mathcal{V}(S''(x), H_1(cd)) \neq S''(v)\}]$ is negligible.
- If the tester answers properly for cd, namely so that $H_1(cd) = H_1(S''(t^*))$ and $\mathcal{V}(S''(x), H_1(cd)) = S''(v)$, we can safely assume that $cd = S''(t^*)$, for the probability of collisions is negligible in the random oracle model. Then, as we know that $out_K \neq S''(\mathrm{out}_d)$, the computation of the plaintext corresponding to cd achieved by algorithm Ext is not accurate. But according to condition 2, $\mathsf{Pr}[\mathsf{Ext}(hH, C, y, S''(t^*), f) \neq S''(\mathrm{out}_d)]$ is negligible.

In terms of probabilities, this can be written as follows:

$$\mathsf{Pr}[\text{Event I}] = \sum_{cd \in hH_1.\mathsf{dom}} \mathsf{Pr}[\mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(hH, C, y, cd, f) = 1 \wedge \text{ Event I }]$$

since the probability of Event I and $T(.) = 0$ is obviously null. We now split the probability in two, following whether the sanity check holds:

$$\mathsf{Pr}[\text{Event I}] = \sum_{cd \in hH_1.\mathsf{dom}} \mathsf{Pr}[\ \mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(.) = 1 \wedge \text{ Event I} \wedge \\ \{H_1(cd) \neq H_1(S''(t^*)) \vee \mathcal{V}(S''(x), H_1(cd)) \neq S''(v)\}] \\ + \mathsf{Pr}[\ \mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(.) = 1 \wedge \text{ Event I} \wedge \\ \{H_1(cd) = H_1(S''(t^*)) \wedge \mathcal{V}(S''(x), H_1(cd)) = S''(v)\}]$$

We can bound the first probability by $\epsilon(\eta)$ for condition 1 provides its negligibility. The second term is split once more according to whether $cd = S''(t^*)$.

$$\begin{aligned}\mathsf{Pr}[\text{Event I}] = \sum_{cd \in hH_1.\mathsf{dom}} \big(\epsilon(\eta) + \mathsf{Pr}[\mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(.) = 1 \wedge \text{ Event I} \wedge \\ \{H_1(cd) = H_1(S''(t^*)) \wedge \mathcal{V}(S''(x), H_1(cd)) = S''(v)\} \wedge cd = S''(t^*)] \\ + \mathsf{Pr}[\mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(.) = 1 \wedge \text{ Event I} \wedge \\ \{H_1(cd) = H_1(S''(t^*)) \wedge \mathcal{V}(S''(x), H_1(cd)) = S''(v)\} \wedge cd \neq S''(t^*)]\big)\end{aligned}$$

The last term is negligible because it would mean a collision for the hash oracle and we work in the random oracle model. Besides, to bound the second term, we simply notice as stated above that under all these conditions, we have $\mathsf{Ext}(hH, C, y, S''(t^*), f) \neq S''(\text{out}_d)$, thus the term is less than $\mathsf{Pr}[\mathsf{Ext}(hH, C, y, S''(t^*), f) \neq S''(\text{out}_d)]$, which is negligible according to condition 2. We thus have

$$\mathsf{Pr}[\text{Event I}] = \sum_{cd \in hH_1.\mathsf{dom}} (\epsilon(\eta) + \epsilon'(\eta) + \epsilon''(\eta)) \leq \mathbf{Card}(hH_1.\mathsf{dom})(\epsilon(\eta) + \epsilon'(\eta) + \epsilon''(\eta)) \leq \tilde{\epsilon}(\eta)$$

since the cardinality of $hH_1.\mathsf{dom}$ is bounded by the time of execution of the adversary, that we assumed ran in polynomial time. Thus we have proved that $\mathsf{Pr}[\text{Event I}]$ is negligible.

Let us study Event II, when the plaintext extractor outputs a plaintext whereas the decryption oracle returns *error*. The fact that the plaintext extractor outputs something implies that there exists at least a value of cd for which $\mathcal{T}(hH, C, y, cd, f) = 1$. However, the decryption oracle answers *error*. The only possibility for this to happen is that the tester is mistaken about the value cd: otherwise, $\mathcal{V}(S''(x), H_1(cd)) = S''(v)$ and $H_1(cd) = H_1(S''(t^*))$ would hold, so that $\mathcal{V}(S''(x), H_1(S''(t^*))) = S''(v)$ would be true and the decryption oracle cannot possibly output *error*. Consequently, we have

$$\begin{aligned}\mathsf{Pr}[\text{Event II}] &= \sum_{cd \in hH_1.\mathsf{dom}} \mathsf{Pr}[\mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(hH, C, y, cd, f) = 1 \wedge \text{Event II}] \\ &= \sum_{cd \in hH_1.\mathsf{dom}} \mathsf{Pr}[\mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(hH, C, y, cd, f) = 1 \wedge \text{Event II} \\ &\qquad \wedge \{\mathcal{V}(S''(x), H_1(cd)) \neq S''(v) \vee H_1(cd) \neq H_1(S''(t^*))\}]\end{aligned}$$

But condition 1 provides the negligibility of this latter probability. Thus $\mathsf{Pr}[\text{Event II}] \leq \mathbf{Card}(hH_1.\mathsf{dom})\epsilon(\eta)$, which is negligible too.

Finally, let us analyze what can happen for Event III to occur. The fact that $S''(\text{out}_d) \neq$ *error* implies that $\mathcal{V}(S''(x), H_1(S''(t^*))) = S''(v)$. We know that $S''(t^*) \in \mathbb{T}_{H_1}$, since at least the decryption oracle queried H_1. According to who asked the hash oracle on this value, we are able to bound the probability. We thus write $\mathbb{T}_{H_1}$ as the disjoint union $\mathbb{T}_{H_1} = hH_1 \uplus (\mathcal{E}_q - hH_1) \uplus (\mathcal{D}_q - \mathcal{E}_q - hH_1)$, where $\mathcal{E}_q$ denotes queries of the encryption oracle and $\mathcal{D}_q$ denotes those of the decryption oracle.

- If $S''(t^*) \in hH_1$, $\mathcal{B}$ queried H_1 on this value. But $out_K = error$ implies the tester $\mathcal{T}$ answered 0 when asked about this value of the candidate, whereas the sanity check held.

According to condition 1, $\mathsf{Pr}[\mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(hH, C, y, cd, f) = 0 \wedge \mathcal{V}(S''(x), H_1(cd)) = S''(v)]$ is negligible. Once more, summing on all possible values for cd,

$$\begin{aligned}\mathsf{Pr}[\text{Event III} \wedge S''(t^*) \in hH_1] = \sum_{cd \in hH_1.\mathsf{dom}} \mathsf{Pr}[\mathcal{U}(hH_1.\mathsf{dom}) = cd \wedge \mathcal{T}(hH, C, y, cd, f) = 0 \\ \wedge \mathcal{V}(S''(x), H_1(cd)) = S''(v)] \\ \leq \mathbf{Card}(hH_1.\mathsf{dom})\epsilon(\eta)\end{aligned}$$

– If $S''(t^*) \in \mathcal{E}_q - hH_1$, let y' be the output of the execution of the encryption oracle that queried $H_1(S''(t^*))$. As $y' \in C$ and $y \notin C$, we definitely have $y \neq y'$. Let us recall the few hypothesis we make on the form of our encryption and decryption codes. First, we require that the hash function H_1 is not called during the execution of the decryption oracle, except for the sanity check. Secondly, we assume that the encryption algorithm makes exactly one call to the oracle H_1, namely on t, receives h, and that the value of the variable t^* after running the decryption oracle is that of t. Therefore, the value $S'''(t^*)$ of the variable t^* at the end of a query of $\mathcal{D}(y')$ is the same as $S''(t^*)$: the corresponding execution of the encryption algorithm queried $H_1(S''(t^*))$.
We can thus write:

$$\begin{aligned}\mathsf{Pr}[\text{Event III} \wedge S''(t^*) \in \mathcal{E}_q - hH_1] \leq \mathsf{Pr}[S''' \xleftarrow{r} [\![\mathcal{D}(y')]\!](S', \boldsymbol{H}, (f, f^{-1})) : y \neq y' \wedge S''(t^*) = S'''(t^*) \\ \wedge S''(\mathrm{out}_d) \neq \textit{"error"} \wedge S'''(\mathrm{out}_d) \neq \textit{"error"}] \\ \leq \epsilon'(\eta),\end{aligned}$$

the last quantity being negligible according to condition 3.

– If $S''(t^*) \in \mathbb{T}_{H_1} - \mathcal{E}_q - hH_1$, then $S''(t^*) \in \mathcal{D}_q$. The fact that the decryption oracle outputs something means that the sanity check $\mathcal{V}(S''(x), H_1(S(t^*))) = S''(v)$ holds. As we assume weak injectivity of $\mathcal{V}$, namely that for given values of x and v, $Pr[r \xleftarrow{r} \mathcal{U} : \mathcal{V}(S''(x), r) = S''(v)]$ is negligible, we can bound this probability by $\epsilon''(\eta)$.

In the end, we have:

$$\begin{aligned}\mathsf{Pr}[\text{Event III}] = \mathsf{Pr}[\text{Event III} \wedge S''(t^*) \in hH_1.\mathsf{dom}] + \mathsf{Pr}[\text{Event III} \wedge S''(t^*) \in \mathcal{E}_q - hH_1] + \\ \mathsf{Pr}[\text{Event III} \wedge S''(t^*) \in \mathbb{T}_{H_1} - \mathcal{E}_q - hH_1] \\ \leq \epsilon(\eta) + \epsilon'(\eta) + \epsilon''(\eta)\end{aligned}$$

We have thus proved that $\mathsf{Pr}[\text{Event III}]$ is negligible. We can thus conclude that

$$\begin{aligned}\mathsf{Fail}^{\mathsf{pa}}_{K,B,GE}(\eta, X) = \mathsf{Pr}[(S, \boldsymbol{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}(), \boldsymbol{H}}(f); \\ S'' \xleftarrow{r} [\![\mathcal{D}(y)]\!](S', \boldsymbol{H}, (f, f^{-1})) : y \notin C \wedge K(hH, C, y, f) \neq S''(\mathrm{out}_d)] \\ = \mathsf{Pr}[\text{Event I}] + \mathsf{Pr}[\text{Event II}] + \mathsf{Pr}[\text{Event III}] \\ \leq \epsilon(\eta),\end{aligned}$$

which is negligible.