Towards Automated Proofs for Asymmetric Encryption Schemes in the Random Oracle Model

Judicaël Courant, Marion Daubignard, Cristian Ene,
Pascal Lafourcade, Yassine Lakhnech*
Université Grenoble 1, CNRS, VERIMAG
firstname.last@imag.fr

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. This procedure has been applied to several examples of encryption schemes among which the construction of Bellare-Rogaway 1993, of Pointcheval at PKC'2000 and REACT.

Categories and Subject Descriptors: E.3 DATA ENCRYPTION: Public key cryptosystems

General Terms: Security, verification.

Keywords: Hoare logics, asymmetric encryption, provable security, automated proofs, random oracle model.

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, 23, 21, 5, 6, 19, 18, 17]. 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 that the value of a variable is indistinguishable from a random value even when one gives the values of a set of variables (or some simple expressions on 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 [5, 18, 17]. 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 3.1, 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

*This work is partially supported by the project AVOTE, SCALP and SFINCS

CCS'08, October 27–31, 2008, Alexandria, Virginia, USA.

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 [6]. 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 [5, 17, 18] 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 [7, 20]. 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 [7, 20, 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 3.1 and 3.12). 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 formalization to prove hardness of the discrete logarithm [1], security of signed ElGamal encryption against interactive attacks [3], and of Schnorr signatures against forgery attacks [22]. They are currently working on formalizing the game-based approach in Coq [2]. D. Nowak provides in [16] 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 4 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, 5]. Using standard notations, we write $H \stackrel{r}{\leftarrow} \Omega$ to denote that H is randomly chosen from the set of functions with appropriate domain. By abuse of notation, for a list $\vec{H} = H_1, \cdots, H_n$ of hash functions, we write $\vec{H} \stackrel{r}{\leftarrow} \Omega$ instead of the sequence $H_1 \stackrel{r}{\leftarrow} \Omega, \ldots, H_n \stackrel{r}{\leftarrow} \Omega$. 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:

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

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 bitstring $bs = b_1 \ldots b_k$ (b_i are bits), $bs[n, m] = b_n \ldots b_m$[1], and

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

$\mathcal{N}$ 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 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).

EXAMPLE 2.1. *The following command encodes the encryption scheme proposed by Bellare and Rogaway in [5] (shortly* $\mathcal{E}(in_e; out_e) = f(r)||in_e \oplus G(r)||H(in_e||r)$*):*

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

Semantics: In addition to the variables in Var, we consider variables $\mathbb{T}_{H_1}, \ldots, \mathbb{T}_{H_n}$. 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 $\mathbb{T}_{H_i}$. A *state* associates a value in $\{0,1\}^*$ to each variable in Var and a list of pairs of values to $\mathbb{T}_H$. 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* $(S, \vec{H}, (f, f^{-1}))$ and yields a distribution on configurations. A configuration is composed of a state S, a vector of hash functions $(H_1, \ldots, H_n)$ 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 $[\![\mathsf{c}]\!]$.

A notational convention: It is easy to prove that commands preserve the values of $\vec{H}$ and (f, f^{-1}). Therefore, we can, without ambiguity, write $S' \xleftarrow{r} [\![c]\!](S, \vec{H}, (f, f^{-1}))$ instead of $(S', \vec{H}, (f, f^{-1})) \xleftarrow{r} [\![c]\!](S, \vec{H}, (f, f^{-1}))$. 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 $\textsc{Dist}(\Gamma, \vec{H}, \mathbb{F})$ and is defined as the set of distributions of the form: $[(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \vec{H} \xleftarrow{r} \Omega; S \xleftarrow{r} A^{\vec{H}, f, f^{-1}}() : (S, \vec{H}, f, f^{-1})]$ where A is an algorithm accessing f, f^{-1} and $\vec{H}$ and which records its queries to hashing oracles into the $\mathbb{T}_H$'s in S.

2.2 Asymmetric Encryption

We study 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.

$bs[n, k]$, when $m > k$

DEFINITION 2.1. *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*-bit string 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 2.2. *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* $X \in \textsc{Dist}(\Gamma, \vec{H}, \mathbb{F})$*. For* $\alpha \in \{cpa, cca\}$ *and* $\eta \in \mathbb{N}$*, let*

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

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

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. As we will show, 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 $\vec{H}$, 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:

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

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

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

Table 1: Language grammar.

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

Table 2: The semantics of the programming language

where $V_1, V_2 \subseteq \mathsf{Var}$ and e is an expression, that is, a variable x or the concatenation of a polynomial number of variables.

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 the value of x or a random value, even when he is additionally 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, even when he is 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 $\mathrm{DIST}(\Gamma, \vec{H}, \mathbb{F})$ 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):

$$
\begin{array}{l}
D(X, V_1, V_2)_\eta = \\
\quad [(S, \vec{H}, (f, f^{-1})) \stackrel{r}{\leftarrow} X : (S(V_1), f(S(V_2)), \vec{H}, f)]
\end{array}
$$

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 3.1. *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: $X_\eta = [\beta; S := S_0\{x \mapsto u, y \mapsto H_1(u)\} : (S, \vec{H}, (f, f^{-1}))]$ and $X'_\eta = [\beta; u' \stackrel{r}{\leftarrow} \{0,1\}^{p(\eta)}; S := S_0\{x \mapsto u, y \mapsto H_1(u')\} : (S, \vec{H}, (f, f^{-1}))]$, where $\beta = \vec{H} \stackrel{r}{\leftarrow} \Omega; (f, f^{-1}) \stackrel{r}{\leftarrow} \mathbb{F}(1^\eta); u \stackrel{r}{\leftarrow} \{0,1\}^{p(\eta)}$, where 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 $X \sim_{V_1;V_2} [u \stackrel{r}{\leftarrow} \mathcal{U};$
  $(S, \vec{H}, (f, f^{-1})) \stackrel{r}{\leftarrow} X : (S\{x \mapsto u\}, \vec{H}, (f, f^{-1}))]$
- $X \models \mathsf{WS}(x; V)$ iff $\Pr[(S, \vec{H}, (f, f^{-1})) \stackrel{r}{\leftarrow} X : A(S(V)) = S(x)]$ is negligible, for any adversary A.
- $X \models \mathsf{H}(H, e)$ iff $\Pr[(S, \vec{H}, (f, f^{-1})) \stackrel{r}{\leftarrow} X : S(e) \in S(\mathbb{T}_H).\mathsf{dom}]$ 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 3.1. *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 $\{\mathit{true}\}c\{\mathit{Indis}(\nu out_e; out_e, in_e)\}$ is valid.*

If $\{\mathsf{true}\}c\{\mathsf{Indis}(\nu\mathrm{out}_e; \mathrm{out}_e, \mathrm{in}_e)\}$ 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 axioms 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 axioms 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 for IND-CPA security.

3.2.1 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 3.2. *The following axioms 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'.*

3.2.2 Random Assignment:

Lemma 3.3. *The following axioms 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 e is x or is of the form $e_1||x||e_2$, $x||e_2$ or $e_1||x$.*

Moreover, the following preservation axioms, 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)* $\{\mathsf{WS}(y; V)\}x \xleftarrow{r} \mathcal{U}\{\mathsf{WS}(y; V, x)\}$

Axiom (R1) is obvious. Axiom (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. Axioms (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.

Henceforth, we write $x \in \mathsf{var}(e)$ to state that e is x or is of the form $e_1||x||e_2$, $x||e_2$ or $e_1||x$.

3.2.3 Hash Function:

Lemma 3.4. *The following basic axioms 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 e is x or is of the form $e_1||x||e_2$, $x||e_2$ or $e_1||x$.*
- *(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 $y \notin V$*

Axiom (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. Axiom (H2) is similar to axiom (R2). Axiom (H3) uses the fact that the value of y can not be queried to the hash oracle.

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

Lemma 3.5. *The following preservation axioms 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 $z \in \mathsf{var}(e) \wedge x \notin \mathsf{var}(e)$*
- *(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 $y \notin V_1$*
- *(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. Axiom (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.

3.2.4 One-way Function:

Lemma 3.6. *The following axiom is sound, when $y \notin V \cup \{x\}$:*

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

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

Lemma 3.7. *The following axioms are sound when $z \neq x$:*

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

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

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

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

3.2.5 The Xor operator

In the following axioms, we assume $y \neq z$.

Lemma 3.8. *The following axiom 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 axioms 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 axiom (X1) one should consider y as a key and think about x as the one-time pad encryption of z with the key y. Axioms (X2) and (X3) take advantage of the fact that is easy to compute x given y and z.

3.2.6 *Concatenation:*

LEMMA 3.9. *The following axioms are sound:*

- *(C1)* $\{\mathit{WS}(y;V)\}\ x := y||z\ \{\mathit{WS}(x;V)\}$*, if* $x \notin V$*. A dual axiom applies for* z*.*
- *(C2)* $\{\mathit{Indis}(\nu y;V_1,y,z;V_2) \wedge \mathit{Indis}(\nu z;V_1,y,z;V_2)\}\ x := y||z\ \{\mathit{Indis}(\nu x;V_1;V_2)\}$*, if* $y,z \notin V_1 \cup V_2$
- *(C3)* $\{\mathit{Indis}(\nu t;V_1,y,z;V_2)\} x := y||z$ $\{\mathit{Indis}(\nu t;V_1,x,y,z;V_2)\}$*, if* $t \neq x,y,z$
- *(C4)* $\{\mathit{WS}(t;V,y,z)\}\ x := y||z\ \{\mathit{WS}(t;V,y,z,x)\}$*, 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 axioms 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 3.10.

LEMMA 3.10. *Let* $X \in \mathrm{DIST}(\Gamma, \vec{H}, \mathbb{F})$ *be a distribution ensemble:*

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

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

PROPOSITION 3.11. *The Hoare triples of Section 3.2 are valid.*

EXAMPLE 3.2. *We illustrate our proposition with Bellare & Rogaway's generic construction [5].*

1) $r \xleftarrow{r} \{0,1\}^{n_0}$
$\mathit{Indis}(\nu r;\mathit{Var}) \wedge \mathit{H}(G,r) \wedge \mathit{H}(H, in_e||r)$
2) $a := f(r)$
$\mathit{Indis}(\nu a;\mathit{Var}-r) \wedge \mathit{WS}(r;\mathit{Var}-r) \wedge \mathit{H}(G,r) \wedge$
$\mathit{H}(H, in_e||r)$
3) $g := G(r)$
$\mathit{Indis}(\nu a;\mathit{Var}-r) \wedge \mathit{Indis}(\nu g;\mathit{Var}-r) \wedge$
$\mathit{WS}(r;\mathit{Var}-r) \wedge \mathit{H}(H, in_e||r)$
4) $b := in_e \oplus g$
$\mathit{Indis}(\nu a;\mathit{Var}-r) \wedge \mathit{Indis}(\nu b;\mathit{Var}-g-r) \wedge$
$\mathit{WS}(r;\mathit{Var}-r) \wedge \mathit{H}(H, in_e||r)$
5) $s := in_e||r$
$\mathit{Indis}(\nu a;\mathit{Var}-r-s) \wedge \mathit{Indis}(\nu b;\mathit{Var}-g-r-s) \wedge$
$\mathit{WS}(s;\mathit{Var}-r-s) \wedge \mathit{H}(H,s)$
6) $c := H(s)$
$\mathit{Indis}(\nu a;\mathit{Var}-r-s) \wedge \mathit{Indis}(\nu b;\mathit{Var}-r-g-s) \wedge$
$\mathit{Indis}(\nu c;\mathit{Var}-r-s)$
7) $out_e := a||b||c$
$\mathit{Indis}(\nu out_e;\mathit{Var}-a-b-c-r-g-s)$

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 [18] and the second one by the Rapid Enhanced-security Asymmetric Cryptosystem Transform (REACT) [17]. For obvious reasons, we cannot recall the definitions of the security of these functions; we explain them informally.

The first observation we have to make is that Proposition 3.1 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 $X \sim_{V_1;V_2} [u \xleftarrow{r} \mathcal{U}; (S,\vec{H},(f,f^{-1})) \xleftarrow{r} X : (S\{x \mapsto f(u)\}, \vec{H}, (f,f^{-1}))]$.

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_i can be the identity function as in the last step of Example 3.3 and 3.4). 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 3.12. *We consider GE 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 $\{\mathsf{true}\}\mathsf{c}\{\bigwedge_{i=1}^{n} \mathit{Indis}_{f_i}(\nu a_i; a_1,\ldots,a_n, in_e)\}$ *is valid then GE is IND-CPA.*

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

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

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 axioms, where we assume $x, y \notin V \cup \{z\}$ and where we identify f and $(x,y) \mapsto f(x||y)$:

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

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 3.3. *We verify Pointcheval's transformer [18].*

1) $r \xleftarrow{r} \{0,1\}^{n_0}$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r)$
2) $s \xleftarrow{r} \{0,1\}^{n_0}$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{Indis}(\nu s; \mathsf{Var}) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, in_e||s)$
3) $w := in_e||s$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{WS}(w; \mathsf{Var} - s - w) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, w)$
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) $b := w \oplus G(r)$
$\mathsf{Indis}_f(\nu a; a, in_e) \wedge \mathsf{Indis}(\nu b; a, b, in_e)$
7) $out_e := a||b$
$\mathsf{Indis}_f(\nu a; a, in_e) \wedge \mathsf{Indis}(\nu b; a, b, in_e)$

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 $\mathsf{Indis}_f(\nu a; a, in_e)$ and $\mathsf{Indis}(\nu b; a, b, in_e)$ implies $\mathsf{Indis}_f(\nu a; a, b, in_e)$

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 $\Pr[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X : A^{PCA}(S(V)) = S(x)]$ is negligible, for any adversary A. Now, we have to revisit Lemma 3.10 and the axioms that introduce $\mathsf{WS}(x; V)$ in the postcondition. It is, however, easy to check that they are valid.

EXAMPLE 3.4. *REACT [17]*

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

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. PLAINTEXT AWARENESS

Bellare and Rogaway introduced *plaintext awareness (PA)* in [6][4]. The motivation is to decompose IND-CCA security of an encryption scheme into IND-CPA and PA security. Indeed, a public-key encryption scheme that satisfies IND-CPA (in the ROM) and the original definition of PA is IND-CCA1 (in the ROM). PA has been refined in [4] such that if an encryption scheme is PA and IND-CPA then it is IND-CCA. Intuitively, plaintext awareness means that the decryption oracle can be simulated by a *plaintext extractor* that does not have access to the inverse permutation f^{-1}. Now we introduce a simple analysis that allows us to automatically verify that an encryption scheme is PA in the strong sense [4]. Hence, combined with the results of the previous sections we obtain an analysis that allows to verify IND-CCA security.

We recall the definition of PA-security following the notations and conventions of [4]. Let $GE = (\mathbb{F}, \mathcal{E}(\mathrm{in}_e, \mathrm{out}_e) : \mathsf{c}, \mathcal{D}(\mathrm{in}_d, \mathrm{out}_d) : \mathsf{c}')$ be a generic encryption scheme. An adversary B for plaintext awareness is given the public permutation f, oracle access to the encryption algorithm $\mathcal{E}$ and to the ideal hash functions $\vec{H} = H_1, \cdots, H_n$. His goal is to output a cipher-text that cannot be correctly decrypted by the plaintext extractor. Hence, the success of plaintext extractor K against B in the distribution $X \in \textsc{Dist}(\Gamma, \vec{H}, \mathbb{F})$ is defined by:

$$\begin{aligned}&\mathsf{Succ}^{\mathsf{pa}}_{K,B,GE}(\eta, X) = \\ &\quad \mathsf{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); \\ &\quad S'' \xleftarrow{r} [\![\mathcal{D}(y, \mathrm{out}_d)]\!](S', \vec{H}, (f, f^{-1})) : \\ &\quad y \in C \vee (y \notin C \wedge K(hH, C, y, f) = S''(\mathrm{out}_d))]\end{aligned}$$

Here by $(hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f)$ we mean the following: run B on input f with oracle access to H_i, $i = 1, \cdots, n$ and $\mathcal{E}$ (which calls f and H_i), recording B's interaction with the hash functions in hH and his interaction with $\mathcal{E}$ in C. Thus, hH is a list $(hH_1, \cdots, hH_n)$ of lists. Each list $hH_i = ((h_1, v_1), \cdots, (h_{q_i}, v_{q_i}))$ records all of B's H_i-oracle queries $h_1, \cdots, h_{q_i}$ and the corresponding answers $v_1, \cdots, v_{q_i}$. The modified state S' is due to calls of the hash functions either by B or the encryption oracle. The list C records the cipher-texts received in reply to $\mathcal{E}$-queries [5]. Finally, y is B's challenge to the plaintext extractor K. Please notice that K wins whenever B outputs a value $y \in C$.

DEFINITION 4.1. *An encryption scheme given by $GE = (\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c}, \mathcal{D}(in_d, out_d) : \mathsf{c}')$ is* PA-secure, *if there*

[4]While in the original work by Bellare and Rogaway and in subsequent ones, plaintext awareness includes semantic security, we prefer to separate plaintext extraction and semantic security.
[5]This list was not included in the original definition by Bellare and Rogaway. Without it only IND-CCA1 can be proved but not IND-CCA.

is a polynomial-time probabilistic algorithm K such that for every distribution $X \in \mathrm{Dist}(\Gamma, \vec{H}, \mathbb{F})$ and adversary B, we have $1 - \mathsf{Succ}^{pa}_{K,B,GE}(\eta, X)$ is a negligible function in η.

The rest of the section is organized as follows. We first introduce a semantic condition on $\mathcal{D}$ that implies the existence of a plaintext extractor. Then, we provide a syntactic criterion that implies the semantic criterion.

In the remainder of this section, we consider an encryption scheme GE that uses the hash functions $\vec{H} = H_1, \cdots, H_n$. We assume that c' has the following form

$\mathsf{c}_1; h := H_1(t);$

$\mathsf{if}\ \mathcal{V}(\vec{x}, h) = v\ \mathsf{then}\ \mathrm{out}_d := m\ \mathsf{else}\ \mathrm{out}_d :=$ *"error"* fi,

where $\vec{x}$ is a vector of variables (possibly empty) and $\mathcal{V}$ is a function (possibly the identity in which case we do not write it) such that for given $\vec{x}$ and v, $Pr[r \xleftarrow{r} \mathcal{U} : \mathcal{V}(\vec{x}, r) = v]$ is negligible. Furthermore, we require that the hash function H_1 is not called in c_1 and that the encryption algorithm c makes exactly one call to the oracle H_1. Consider, for instance, the scheme in [5], $f(r)||\mathrm{in}_e \oplus G(r)||H(\mathrm{in}_e||r)$. Here, t gets assigned the value $\mathrm{in}_e||r$. We call the condition $\mathcal{V}(\vec{x}, h) = v$ (or equivalently $\mathcal{V}(\vec{x}, H_1(t)) = v$) the "*sanity check*".

It allows us to discriminate valid cipher-text from arbitrary bit-string. We also assume that decryption behaves correctly with respect to encryption: if y is generated using the algorithm of encryption, then the value of t as computed by the decryption oracle coincides with the value used as argument in the call to H_1 by the encryption algorithm.

Example 4.1. *Bellare and Rogaway [5]:*

$\mathcal{D}(in_d = a^*||b^*||v, out_d)$:

$r^* := f^{-1}(a^*); g^* := G(r^*); m^* := b^* \oplus g^*; t := m^*||r^*;$

$h := H(t);$

$\mathit{if}\ h = v\ \mathit{then}\ out_d := m^*\ \mathit{else}\ out_d :=$ "error" *fi*

A semantic criterion for PA Our semantic criterion for PA-security is composed of three conditions. We begin with an informal presentation of these conditions and how they will enable us to construct a plaintext extractor.

1. The first condition says that there is an algorithm that checks whether a given bit-string t^*, that has been submitted to H_1 by B, corresponds to the challenge y. That is, if the tester answers "yes" (1), then t^* matches with the value of t as computed by the decryption oracle and additionally satisfies the sanity check; and if it answers "no" (0), then t^* does not satisfy the sanity check.

2. The second condition states that it is easy to compute the plaintext from t^*.

3. The third condition states that for each value of t there is at most one corresponding ciphertext y.

Assume now that these conditions are satisfied. Then, we can construct a plaintext extractor K as follows. Using the algorithm of the first condition, that we call the tester, scan the list hH_1 to find a suitable t^*. If none is found, answer "error". Otherwise, apply the algorithm of the second condition on the value found for t^* to extract the plaintext. The third condition ensures that each value of t^* corresponds to at most one ciphertext, which is necessary to ensure that the extracted plaintext is the correct one. Let us now tackle the formal treatment of these ideas.

Definition 4.2. *We say that GE a generic encryption scheme 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, t^*, f) and returns a value in $\{0,1\}$. We require that for any adversary B and any distribution $X \in \mathrm{Dist}(\Gamma, \vec{H}, \mathbb{F})$,*

$$\begin{array}{l}1- \\ \mathit{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); \\ S'' \xleftarrow{r} [\![\mathcal{D}(y, out_d)]\!](S', \vec{H}, (f, f^{-1})); t^* \xleftarrow{r} hH_1.\mathit{dom}; \\ b \xleftarrow{r} \mathcal{T}(hH, C, y, t^*, f) : (b = 1 \Rightarrow H_1(t^*) = H_1(S''(t)) \\ \wedge \mathcal{V}(S''(x), H_1(t^*)) = S''(v)) \wedge \\ (b = 0 \Rightarrow \mathcal{V}(S''(x), H_1(t^*)) \neq S''(v))]\end{array}$$

is negligible.

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

$$\begin{array}{l}1- \\ \Pr[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); \\ S'' \xleftarrow{r} [\![\mathcal{D}(y, out_d)]\!](S', \vec{H}, (f, f^{-1})) : \\ \mathsf{Ext}(hH, C, y, S''(t), f) = S''(out_d)]\end{array}$$

is negligible.

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

$$\begin{array}{l}\mathit{Pr}[(S, \vec{H}, (f, f^{-1})) \xleftarrow{r} X; (hH, C, y, y', S') \xleftarrow{r} B^{\mathcal{E}, \vec{H}}(f); \\ S_1 \xleftarrow{r} [\![\mathcal{D}(y, out_d)]\!](S', \vec{H}, (f, f^{-1})); \\ S_2 \xleftarrow{r} [\![\mathcal{D}(y', out_d)]\!](S', \vec{H}, (f, f^{-1})) : \\ y \neq y' \wedge S_1(t) = S_2(t) \wedge S_1(out_d) \neq \text{"error"} \wedge \\ S_2(out_d) \neq \text{"error"}]\end{array}$$

is negligible.

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.

Given a tester $\mathcal{T}$ and an algorithm Ext as in Definition 4.2, we construct a plaintext extractor as follows:

$K^{\mathcal{T}, \mathsf{Ext}}(hH, C, y, f)$:

Let $L = \{t^* \mid t^* \in \mathsf{dom}(hH_1), \mathcal{T}(hH, C, y, t^*, f) = 1\}$

if $L = \epsilon$ then return "error" else $t^* \xleftarrow{r} L$;

return $\mathsf{Ext}(hH, C, y, t^*, f)$

Theorem 4.1. *Let GE be a generic encryption scheme that satisfies the PA-semantic criterion. Then, GE is PA-secure.*

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

Definition 4.3. *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 4.2.

- *Bellare and Rogaway [5]:* $\mathcal{E}(in_e; r) = a||b||c = f(r)||in_e \oplus G(r)||H(in_e||r)$. *The "sanity check" of the decryption algorithm is* $H(m^*||r^*) = c^*$.

- *OAEP+ [19]:* $\mathcal{E}(in_e; r) = f(a||b||c)$, *where* $a = in_e \oplus G(r)$, $b = H'(in_e||r)$, $c = H(s) \oplus r$ *and* $s = in_e \oplus G(r)||H'(in_e||r)$. *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* $\mathcal{E}(in_e; r) = \mathcal{E}'((in_e||r); H(in_e||r))$. *The "sanity check" of the decryption algorithm is:*
$\mathcal{E}'(m^*||r^*; H(m^*||r^*)) = in_d$.

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

EXAMPLE 4.3.

- *Pointcheval [18]:*
$\mathcal{E}(in_e; r; s) = f(r||H(in_e||s))||((in_e||s) \oplus G(r))$, *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 [17]: This construction applies to any trapdoor one-way function (possibly probabilistic). It is quite similar to the construction in [5]:* $\mathcal{E}(in_e; R; r) = a||b||c = f(R; r)||in_e \oplus G(r)||H(R||in_e||a||b)$, *where* $a = f(R; r)$ *and* $b = in_e \oplus G(R)$. *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 4.4.

- *Zheng-Seberry Scheme [23]:*

  $\mathcal{E}(x; r) = a||b = f(r)||(G(r) \oplus (x||H(x))$. *The third condition of the PA-semantic criterion is not satisfied by this construction. Actually, there is an attack [21] on the IND-CCA security of this scheme that exploits this fact.*

- *OAEP [6]:* $\mathcal{E}(in_e; r) = a = f(in_e||0^k \oplus G(r)||r \oplus H(s))$, *where* $s = in_e||0^k \oplus G(r)$. *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 3.10.

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.

7. REFERENCES

[1] G. Barthe, J. Cederquist, and S. Tarento. A Machine-Checked Formalization of the Generic Model and the Random Oracle Model. In D. Basin and M. Rusinowitch, editors, *Proceedings of IJCAR'04*, volume 3097 of *LNCS*, pages 385–399, 2004.

[2] Gilles Barthe, Bejamin Grégoire, Romain Janvier, and Santiago Zanella Béguelin. A framework for language-based cryptographic proofs. In *2nd Informal ACM SIGPLAN Workshop on Mechanizing Metatheory*, Oct 2007.

[3] Gilles Barthe and Sabrina Tarento. A machine-checked formalization of the random oracle model. In Jean-Christophe Filliâtre, Christine Paulin-Mohring, and Benjamin Werner, editors, *Proceedings of TYPES'04*, volume 3839 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2004.

[4] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 26–45, London, UK, 1998. Springer-Verlag.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, New York, USA, November 1993. ACM, ACM.

[6] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.

[7] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/`.

[8] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *S&P*, pages 140–154. IEEE Computer Society, 2006.

[9] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 2006.

[10] Ricardo Corin and Jerry den Hartog. A probabilistic hoare-style logic for game-based cryptographic proofs. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2006.

[11] Ivan Damgard. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 445–456, London, UK, 1992. Springer-Verlag.

[12] Anupam Datta, Ante Derek, John C. Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *CSFW*, 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] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In *PKC '99: Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, pages 53–68, London, UK, 1999. Springer-Verlag.

[15] Shai Halevi. A plausible approach to computer-aided cryptographic proofs. `http://theory.lcs.mit.edu/~shaih/pubs.html`, 2005.

[16] David Nowak. A framework for game-based security proofs. In *ICICS*, pages 319–333, 2007.

[17] Tatsuaki Okamoto and David Pointcheval. React: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, pages 159–175, London, UK, 2001. Springer-Verlag.

[18] David Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In *PKC '00: Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography*, pages 129–146, London, UK, 2000. Springer-Verlag.

[19] Victor Shoup. Oaep reconsidered. *J. Cryptology*, 15(4):223–249, 2002.

[20] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004. http://eprint.iacr.org/2004/332.

[21] David Soldera, Jennifer Seberry, and Chengxin Qu. The analysis of zheng-seberry scheme. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 159–168. Springer, 2002.

[22] Sabrina Tarento. Machine-checked security proofs of cryptographic signature schemes. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *Computer Security - ESORICS 2005*, volume 3679 of *Lecture Notes in Computer Science*, pages 140–158. Springer, 2005.

[23] Yuliang Zheng and Jennifer Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, 1993.