J Autom Reasoning (2011) 46:261–291
DOI 10.1007/s10817-010-9186-x

Automated Proofs for Asymmetric Encryption

J. Courant · M. Daubignard · C. Ene ·
P. Lafourcade · Y. Lakhnech

Received: 16 June 2010 / Accepted: 17 June 2010 / Published online: 3 July 2010

Abstract 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. In this paper, we present a sound and automated procedure that allows us to verify that a generic asymmetric encryption scheme is secure against chosen-plaintext attacks 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.

Keywords Provable cryptography · Asymmetric encryption · Automated verification · Hoare logic

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 [15], 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 [20]. The next breakthrough towards formal proofs of security was the adoption of computational theory 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

This work is partially supported by the ANR projects SCALP, AVOTE and SFINCS.

J. Courant · M. Daubignard · C. Ene · P. Lafourcade (✉) · Y. Lakhnech
Université Joseph Fourier (Grenoble 1), CNRS, Verimag, Grenoble, France
e-mail: pascal.lafourcade@imag.fr

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 is 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, specifically security against chosen ciphertext attacks. Examples of generic encryption schemes are [7, 8, 13, 18, 19, 21, 23, 25]. In this paper we propose a compositional Hoare logic for proving IND-CPA security. An important feature of our method is that it is not based on a global reasoning as it is the case for the game-based approach [9, 22]. Instead, it is based on local reasoning. 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. In future work [11], we also present a method for proving plaintext awareness (PA). Plaintext awareness together with IND-CPA security imply IND-CCA security [3]. Combining the results of this paper with plaintext awareness, leads to a proof method for verifying the constructions in [7, 18, 19].

Related work We restrict our discussion to work aiming at providing computational proofs for cryptosystems. In particular, this excludes symbolic verification. We mentioned above the game-based approach [9, 17, 22]. B. Blanchet and D. Pointcheval developed a dedicated tool, CryptoVerif, that supports security proofs within the game-based approach [5, 6]. From the theoretical point of view, the main differences in our approaches are the following. 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 are working 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. Propositions 1 and 3). G. Barthe, J. Cederquist and S. Tarento were among the first to provide machine-checked proofs of cryptographic schemes without relying on the perfect cryptography hypothesis. They have provided formal models of 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 El-Gamal encryption against interactive attacks [10], and of Schnorr signatures against forgery attacks [24]. Recently in [4], the authors have been developing CertiCrypt, which provides support for formalizing game-based proofs in the Coq proof assistant. They have used their formalization to give machine-checked proofs of IND-CPA for OAEP. Another interesting piece of work to mention is the Hoare-style proof system proposed by R. Corin and J. Den Hartog for game-based cryptographic proofs [12]. Yet, there is no computer-assistance for the developed logic. In [14], Datta et al. present a computationally sound compositional logic (PCL) for key exchange protocols. PCL has been applied successfully to the IEEE 802.11i wireless security standard and the IETF GDOI standard. PCL is a computationally sound Hoare-like logic for indistinguishability; one important difference is that, being focused on protocols, PCL does not provide support for standard cryptographic constructions such as one-way functions.

Outline In Section 2, we introduce notions used in the rest of the paper. In Section 3 we define our programming language and generic asymmetric encryption schemes. In Section 4, we present our Hoare logic for proving IND-CPA security. In Section 5, we explain how we can automate our procedure. Finally we conclude in Section 6 .

2 Preliminaries

In this section, we recall some basic definitions as well as poly-time indistinguishability. To do so, let us first introduce the following notations. If d is a distribution over a set E, we use $v \overset{r}{\leftarrow} d$ to denote that an element $v \in E$ is sampled according to distribution d. Moreover, as usual, if $d_1, \ldots, d_n$ are distributions, $[u_1 \overset{r}{\leftarrow} d_1; \ldots; u_n \overset{r}{\leftarrow} d_n : (u_{i_1}, \ldots, u_{i_k})]$ denotes the distribution obtained by performing the samplings in the order they are written and returning the result which is specified after the semicolon.

We are interested in analyzing generic schemes for asymmetric encryption assuming ideal hash functions. That is, we are working in the *random oracle model* [7, 16]. Using standard notations, we write $H \overset{r}{\leftarrow} \Omega$ to denote that H is chosen uniformly at random from the set of functions with appropriate domain. By abuse of notation, for a list $\mathbf{H} = H_1, \cdots, H_n$ of hash functions, we write $\mathbf{H} \overset{r}{\leftarrow} \Omega$ instead of the sequence $H_1 \overset{r}{\leftarrow} \Omega, \ldots, H_n \overset{r}{\leftarrow} \Omega$. We fix a finite set $\{H_1, \ldots, H_n\}$ of hash functions and also a finite set Π of trapdoor permutations. We assume an arbitrary but fixed ordering on Π and $\mathbf{H}$; this allows us to freely switch between set-based and vector-based notation. A *distribution ensemble* is a countable sequence of distributions $\{X_\eta\}_{\eta \in \mathbb{N}}$ over states, $\mathbf{H}$ and Π. We only consider distribution ensembles that can be constructed in polynomial time in η by probabilistic algorithms that have oracle access to $\mathbf{H}$—this set is formally defined in the next section. 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:

$$\mathsf{Adv}(\mathcal{A}, \eta, X, X') = \left|\mathsf{Pr}\left[x \overset{r}{\leftarrow} X_\eta : \mathcal{A}^{\mathbf{H}}(x) = 1\right] - \mathsf{Pr}\left[x \overset{r}{\leftarrow} X'_\eta : \mathcal{A}^{\mathbf{H}}(x) = 1\right]\right|$$

The probabilities are taken over X and the random coins used by the probabilistic adversary $\mathcal{A}$. We insist, above, that for each hash function H, the probabilities are indeed taken over the set of maps with the appropriate type. Two distribution ensembles X and X' are called *indistinguishable*, denoted by $X \sim X'$, if $\mathsf{Adv}(\mathcal{A}, \eta, X, X')$ is negligible as a function of η, for any polynomial-time (in η) probabilistic algorithm $\mathcal{A}$. 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.

3 A Simple Programming Language for Encryption and Decryption Oracles

In this section, we fix a notation for specifying encryption schemes. The notation we choose is a simple imperative language with random assignment. The language does not include loops since loops are usually not used for generic encryption schemes. Moreover, whereas the language allows the application of a trapdoor permutation f, it does not include the application of the inverse of a permutation. This choice is due to the fact that our analysis is carried out on encryption algorithms only, which, in the case of generic encryption schemes, seldom use permutation inverses.

Let Var be a finite set of variables. We assume that Var is large enough to deal with the considered examples. It should be obvious that our results do not depend on the size of Var. Our programming language is built according to the following BNF described in Table 1, where:

- $x \overset{r}{\leftarrow} \mathcal{U}$ samples a value in $\mathcal{U}_\eta$ and assigns it to x. Here, $(\mathcal{U}_\eta)_\eta$ is the family of uniform distributions over the set of bit-strings of length $\tau(x, \eta)$, where $\tau(x, \eta)$ is a polynomial in η.
- $x := f(y)$ applies the trapdoor one-way function f to the value of y and assigns the result to x.
- $x := H(y)$ applies the hash function H to the value of y and assigns the result to x. As a side effect, the pair $(v, H(v))$, where v is the value of y, is added to the variable $\mathbb{T}_H$ which stores the queries to the hash function H.
- $x := y \oplus z$ applies the exclusive or operator to the values of y and z and assigns the result to x.
- $x := y||z$ represents the concatenation of the values of y and z.
- $c_1; c_2$ is the sequential composition of c_1 and c_2.
- $\mathcal{N}(x, y) : \mathbf{var}\ x_1; \cdots; x_n; \mathsf{c}$ is a procedure declaration, where $\mathcal{N}$ is its name (identifier for encryption or decryption procedure), $\mathbf{var}\ x_1; \cdots; x_n;$ declares the local variables $x_1, \cdots, x_n$, c is the body of the procedure and x and y are the input and output variables, respectively.

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

$$\begin{array}{l}
\mathcal{E}(\text{in}_e, \text{out}_e) : \\
\mathbf{var}\ r; a; g; b; s; c; \\
r \overset{r}{\leftarrow} \{0,1\}^\eta;\ \ a := f(r);\ \ g := G(r); \\
b := \text{in}_e \oplus g;\ \ s := \text{in}_e||r;\ \ c := H(s); \\
\text{out}_e := (a||b)||c;\ \ (\text{where } f \in \Pi \text{ and } G, H \in \mathbf{H})
\end{array}$$

Semantics In addition to the variables in Var, we consider variables $\mathbb{T}_{H_1}, \ldots, \mathbb{T}_{H_n}$. Variable $\mathbb{T}_{H_i}$ is used to record the queries to the hash function H_i. Thus, we consider states that assign bit-strings to the variables in Var and lists of pairs of bit-strings to

Table 1 Language grammar

Command	$\mathsf{c} ::= x \overset{r}{\leftarrow} \mathcal{U} \mid x := f(y) \mid x := H(y) \mid x := y \oplus z \mid x := y\|\|z \mid c; c$
Oracle declaration	$\mathcal{O} ::= \mathcal{N}(x, y) : \mathbf{var}\ x_1; \cdots; x_n; \mathsf{c}$

$\mathbb{T}_{H_i}$. Let $\mathbb{T}_{\mathbf{H}} = \{\mathbb{T}_{H_1}, \ldots, \mathbb{T}_{H_n}\}$. The reader should notice that the variables in $\mathbb{T}_{\mathbf{H}}$ are not in Var, and hence, cannot occur in commands.

We assume that all variables range over bit-strings (or pairs of bit-strings) with polynomial size in the security parameter η, so that domains of the variables in Var have a cardinality exponential in η. 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. Also we extend $S(\cdot)$ to expressions in the usual way, for instance $S(y \oplus z) = S(y) \oplus S(z)$, etc.

A program takes as input a *configuration* $(S, \mathbf{H}, (f, f^{-1}))$ 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, drawn thanks to a generator denoted $\mathbb{F}$. 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. $\mathsf{Pr}[x] = 1$ and $S(\mathbb{T}_H) \cdot (S(y), v)$ denotes the concatenation to $\mathbb{T}_H$ of a query to the hash function and its answer. Notice that the semantic function of commands can be lifted in the usual way to a function from $\textsc{Dist}(\Gamma)$ to $\textsc{Dist}(\Gamma)$. That is, let $F : \Gamma \to \textsc{Dist}(\Gamma)$ be a function. Then, F defines a unique function $F^* : \textsc{Dist}(\Gamma) \to \textsc{Dist}(\Gamma)$ such that $F^*(D) = [\gamma \overset{r}{\leftarrow} D; \gamma' \overset{r}{\leftarrow} F(\gamma) : \gamma']$. By abuse of notation we also denote the lifted semantics by $[\![\mathsf{c}]\!]$.

It is easy to prove that commands preserve the values of $\mathbf{H}$ and (f, f^{-1}). Therefore, we can, without ambiguity, write $S' \overset{r}{\leftarrow} [\![c]\!](S, \mathbf{H}, (f, f^{-1}))$ instead of $(S', \mathbf{H}, (f, f^{-1})) \overset{r}{\leftarrow} [\![c]\!](S, \mathbf{H}, (f, f^{-1}))$. 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.

A family X of distributions is called *constructible*, if there is an algorithm A such that

$$X_\eta = \left[(f, f^{-1}) \overset{r}{\leftarrow} \mathbb{F}(1^\eta); \mathbf{H} \overset{r}{\leftarrow} \Omega; S \overset{r}{\leftarrow} A^{\mathbf{H}}(f, 1^\eta) : (S, \mathbf{H}, f, f^{-1})\right],$$

where $\mathbb{F}$ is a *trapdoor permutation generator* that on input η generates an η-bit-string trapdoor permutation pair (f, f^{-1}), and A is a poly-time probabilistic algorithm with oracle access to the hash function, and such that A's queries to the hash oracles are recorded in the lists $\mathbb{T}_H$'s in S. The latter condition should not be understood as a restriction on the set of considered adversaries. Indeed, it does not mean that the adversaries need to honestly record their queries in the $\mathbb{T}_H$ lists. It rather means that in our reduction proofs, when we simulate such adversaries, we need to record their

Table 2 The semantics of the programming language

$[\![x \overset{r}{\leftarrow} \mathcal{U}]\!](S, \mathbf{H}, (f, f^{-1})) = [u \overset{r}{\leftarrow} \mathcal{U} : (S\{x \mapsto u\}, \mathbf{H}, (f, f^{-1}))]$
$[\![x := f(y)]\!](S, \mathbf{H}, (f, f^{-1})) = \delta(S\{x \mapsto f(S(y))\}, \mathbf{H}, (f, f^{-1}))$
$[\![x := H(y)]\!](S, \mathbf{H}, (f, f^{-1})) =$ $\begin{cases} \delta(S\{x \mapsto v\}, \mathbf{H}, (f, f^{-1})) \text{ if } (S(y), v) \in \mathbb{T}_H \\ \delta(S\{x \mapsto v, \mathbb{T}_H \mapsto S(\mathbb{T}_H) \cdot (S(y), v)\}, \mathbf{H}, (f, f^{-1})) \text{ if } (S(y), v) \notin \mathbb{T}_H \text{ and } v = \mathbf{H}(H)(S(y)) \end{cases}$
$[\![x := y \oplus z]\!](S, \mathbf{H}, (f, f^{-1})) = \delta(S\{x \mapsto S(y) \oplus S(z)\}, \mathbf{H}, (f, f^{-1}))$
$[\![x := y \vert\vert z]\!](S, \mathbf{H}, (f, f^{-1})) = \delta(S\{x \mapsto S(y) \vert\vert S(z)\}, \mathbf{H}, (f, f^{-1}))$
$[\![c_1; c_2]\!] = [\![c_2]\!] \circ [\![c_1]\!]$
$[\![\mathcal{N}(x, y) : \mathbf{var}\ \mathbf{x}; c]\!](S, \mathbf{H}, (f, f^{-1})) = ([\![c]\!](S, \mathbf{H}, (f, f^{-1})))\{\mathbf{x} \mapsto S(\mathbf{x})\}$

queries. We emphasize that the algorithm denoted above by A can, but does not necessarily represent (only) an adversary, e.g. it can be the sequential composition of an adversary and a command. We denote by $\textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ the set of constructible families of distributions. Distributions that are element of a constructible family of distributions are called *constructible* too.

Notice that $[\![c]\!](X) \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, for any command c and $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$.

3.1 Generic Asymmetric Encryption Schemes

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.

Definition 1 A pair $(\mathbb{F}, \mathcal{E}(\text{in}_e, \text{out}_e) : \mathbf{var}\ \mathbf{x}; \mathsf{c})$ defines a *generic encryption scheme*, where:

- $\mathbb{F}$ is a *trapdoor permutation generator* that on input η generates an η-bit-string trapdoor permutation pair (f, f^{-1}),
- $\mathcal{E}(\text{in}_e, \text{out}_e) : \mathbf{var}\ \mathbf{x}; \mathsf{c}$ is a procedure declaration that describes the *encryption algorithm*. The variables in $\mathbf{x}$ are the local variables of the encryption algorithm. We require that the outcome of $\mathcal{E}(\text{in}_e, \text{out}_e)$ only depends on the value of in_e. Formally, for a triple $(S, \mathbf{H}, (f, f^{-1}))$, let $\textsc{Out}(S, \mathbf{H}, (f, f^{-1}))$ denote the distribution:

$$\left[(S', \mathbf{H}, (f, f^{-1})) \xleftarrow{r} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S, \mathbf{H}, (f, f^{-1})) : S'(\text{out}_e)\right].$$

  Then, we require that $\textsc{Out}(S, \mathbf{H}, (f, f^{-1})) = \textsc{Out}(S', \mathbf{H}, (f, f^{-1}))$ holds, for every states S and S' such that $S(\text{in}_e) = S'(\text{in}_e)$.

Let S_0 be the state that associates the bit-string $0^{\tau(x,\eta)}$, for any variable $x \in \mathsf{Var}$ that ranges over $\{0, 1\}^{\tau(x,\eta)}$, and the empty list $[\,]$ with each variable in $\mathbb{T}_H$. Then, the usual definition of the IND-CPA security criterion (e.g. see [3]) can be stated as follows:

Definition 2 Let $GE = (\mathbb{F}, \mathcal{E}(\text{in}_e, \text{out}_e) : \mathbf{var}\ \mathbf{x}; \mathsf{c})$ be a generic encryption scheme and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For $\eta \in \mathbb{N}$, let

$$\begin{aligned}\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta) = |2.\mathsf{Pr}[&(f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); \mathbf{H} \xleftarrow{r} \Omega;\\ &(m_0, m_1, \sigma) \xleftarrow{r} \mathcal{A}_1^{\mathbf{H}}(f);\\ &b \xleftarrow{r} \{0, 1\};\\ &S' \xleftarrow{r} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S_0\{\text{in}_e \mapsto m_b\}, \mathbf{H}, (f, f^{-1})) :\\ &\mathcal{A}_2^{\mathbf{H}}(f, m_0, m_1, \sigma, S'(\text{out}_e)) = b] - 1|\end{aligned}$$

We insist, above, that $\mathcal{A}_1$ outputs bit-strings m_0, m_1 such that $|m_0| = |m_1|$, and an internal state σ to be forwarded to its guess-phase $\mathcal{A}_2$.

We say that GE is IND-CPA secure if $\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Notice that because of the condition put on generic encryption schemes, the choice of the state S_0 is not crucial. In other words, we can replace S_0 by any other state, we then get the same distribution of $S'(\text{out}_e)$ as a result. In fact, an equivalent formulation of Definition 2, consists in considering that the 1st-phase adversary $\mathcal{A}_1$ outputs a state S, such that messages m_0 and m_1 are stored in variables x_0 and x_1 and that σ is stored in variable x_σ:

$$\begin{aligned}\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta) = |2.\mathsf{Pr}[&(f, f^{-1}) \overset{r}{\leftarrow} \mathbb{F}(1^\eta); \mathbf{H} \overset{r}{\leftarrow} \Omega;\\ &S \overset{r}{\leftarrow} \mathcal{A}_1^{\mathbf{H}}(f);\\ &b \overset{r}{\leftarrow} \{0,1\};\\ &S' \overset{r}{\leftarrow} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S\{\text{in}_e \mapsto s(x_b)\}, \mathbf{H}, (f, f^{-1})) :\\ &\mathcal{A}_2^{\mathbf{H}}(f, S(x_0), S(x_1), S(x_\sigma), S'(\text{out}_e)) = b] - 1|\end{aligned}$$

Thus, we arrive at the following more appropriate equivalent definition of IND-CPA:

Definition 3 Let $GE = (\mathbb{F}, \mathcal{E}(\text{in}_e, \text{out}_e) : \textbf{var}\ \mathbf{x}; \mathsf{c})$ be a generic encryption scheme and $\mathcal{A}$ be an adversary and $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$. For $\eta \in \mathbb{N}$, let

$$\begin{aligned}\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta, X_\eta) = 2.\mathsf{Pr}[&(S, \mathbf{H}, (f, f^{-1})) \overset{r}{\leftarrow} X_\eta; b \overset{r}{\leftarrow} \{0,1\};\\ &S' \overset{r}{\leftarrow} [\![\mathcal{E}(\text{in}_e, \text{out}_e)]\!](S\{\text{in}_e \mapsto S(x_b)\}, \mathbf{H}, (f, f^{-1})) :\\ &\mathcal{A}^{\mathbf{H}}(f, S(x_0), S(x_1), S(s_\sigma), S'(\text{out}_e)) = b] - 1\end{aligned}$$

We say that GE is IND-CPA secure, if $\mathsf{Adv}^{ind-cpa}_{\mathcal{A},GE}(\eta, X_\eta)$ is negligible for any constructible distribution ensemble X and polynomial-time adversary $\mathcal{A}$.

4 A Hoare Logic for IND-CPA Security

In this section, we present our Hoare logic for proving IND-CPA security. We prove that the presented logic is sound. In addition to axioms that deal with each basic command and operation, random assignment, concatenation, xor, etc..., our logic includes the usual sequential composition and consequence rules of the Hoare logic. In order to apply the consequence rule, we use entailment (logical implication) between assertions as in Lemma 2.

Our Hoare logic can be easily transformed into a procedure that allows us to prove properties by computing 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. Given an encryption oracle $\mathcal{E}(\text{in}_e, \text{out}_e) : \textbf{var}\ \mathbf{x}; \mathsf{c}$ 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.

First, we present the assertion language we use to express the invariant properties we are interested in. Then, we present a set of rules of the form $\{\varphi\}\mathsf{c}\{\varphi'\}$, 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 $\{\varphi\}\mathsf{c}\{\varphi'\}$ is valid. From now on, we suppose that the adversary has access to the hash functions $\mathbf{H}$, and he is given the trapdoor permutation f, but not its inverse f^{-1}.

4.1 The Assertion Language

Before specifying the assertion language, we give a few definitions and notations that we use to define the predicates of the language.

Definition 4 The set of variables used as substring of an expression e is denoted $\mathsf{subvar}(e)$: $x \in \mathsf{subvar}(e)$ iff $e = x$ or $e = e_1||e_2$ and $x \in \mathsf{subvar}(e_1) \cup \mathsf{subvar}(e_2)$, for some expressions e_1 and e_2.

For example, consider the following expression: $e = (R||(\text{in}_e||f(R||r)))||g \oplus G(R)$. Here, $\mathsf{subvar}(e) = \{R, \text{in}_e\}$, but $r, g \notin \mathsf{subvar}(e)$.

Definition 5 Let X be a family of distributions in $\textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ and V_1 and V_2 be sets of local variables or variables in Var. By $D(X, V_1, V_2)$ we denote the following distribution family (on tuples of bit-strings):

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

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 $D(X, V_1, V_2) \sim D(X', V_1, V_2)$.

We emphasize that in the above definition, we have that $V_1, V_2 \subseteq \mathsf{Var}$, and since for any $i \in \{1, \ldots, n\}$, $\mathbb{T}_{H_i} \notin \mathsf{Var}$, we get $\mathbb{T}_{H_i} \notin V_1 \cup V_2$ for any $i \in \{1, \ldots, n\}$. Hence, every time we use the equivalence $\sim_{V_1;V_2}$, the variables H_i *are not given to the adversary*.

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: $X_\eta = [\beta; S := S_0\{x \mapsto u, y \mapsto H_1(u)\} : (S, \mathbf{H}, (f, f^{-1}))]$ and $X'_\eta = [\beta; u' \xleftarrow{r} \{0,1\}^\eta; S := S_0\{x \mapsto u, y \mapsto H_1(u')\} : (S, \mathbf{H}, (f, f^{-1}))]$, where $\beta = \mathbf{H} \xleftarrow{r} \Omega; (f, f^{-1}) \xleftarrow{r} \mathbb{F}(1^\eta); u \xleftarrow{r} \{0,1\}^\eta$. 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.

Definition 6 We write $\nu x \cdot X_\eta$ to denote the following distribution:

$$\left[v \xleftarrow{r} \mathcal{U}_\eta; (S, \mathbf{H}, (f, f^{-1})) \xleftarrow{r} X_\eta : (S\{x \mapsto v\}, \mathbf{H}, (f, f^{-1}))\right],$$

We recall that $\mathcal{U}$ is the family of uniform distributions on the values on which x ranges. We lift this notation to families of distributions in usual way: $\nu x.X = (\nu x.X_\eta)_{\eta\in\mathbb{N}}$.

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

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

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.

Intuitively, $\mathsf{Indis}(\nu x; V_1; V_2)$ is satisfied by a distribution on configurations, if given values of variables in V_1 and images by f of values of variables in V_2, any polynomial adversary in η has negligible probability to distinguish between the following two distributions: first, the distribution resulting of computations performed using the original value of x as is in X, secondly, the distribution resulting from computations performed replacing everywhere the value of x by a random value of the same length as x. In Section 4.4, in order to analyze schemes using one-way functions f that are not permutations, we generalize the predicate Indis into Indis_f. The predicate Indis_f models the fact that the adversary cannot distinguish between the value of a variable and the image by f of a random value sampled uniformly. The assertion $\mathsf{WS}(x; V_1; V_2)$ stands for Weak Secret and is satisfied by a distribution, if any adversary has negligible probability to compute the value of x, 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. Lastly, $\mathsf{H}(H, e)$ is satisfied when the probability that the value of e has been submitted to the hash oracle H is negligible.

Notations We use $\mathsf{Indis}(\nu x; V)$ instead of $\mathsf{Indis}(\nu x; V; \emptyset)$ and $\mathsf{Indis}(\nu x)$ instead of $\mathsf{Indis}(\nu x; \mathsf{Var})$. Similarly $\mathsf{WS}(x; V)$ stands for $\mathsf{WS}(x; V; \emptyset)$.

Formally, the meaning of the assertion language is defined by a satisfaction relation $X \models \varphi$, which tells us when a family of distributions on configurations X satisfies the assertion φ.

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 $X \sim_{V_1;V_2} \nu x \cdot X$
- $X \models \mathsf{WS}(x; V_1; V_2)$ iff $\Pr[(S, \mathbf{H}, (f, f^{-1})) \overset{r}{\leftarrow} X_\eta : \mathcal{A}^{\mathbf{H}}(S(V_1), f(S(V_2))) = S(x)]$ is negligible, for any adversary $\mathcal{A}^{\mathbf{H}}$.
- $X \models \mathsf{H}(H, e)$ iff $\Pr[(S, \mathbf{H}, (f, f^{-1})) \overset{r}{\leftarrow} X_\eta : S(e) \in S(\mathbb{T}_H).\mathsf{dom}]$ is negligible.

Moreover, we write $\varphi \Rightarrow \varphi'$ iff for any family of distributions X such that $X \models \varphi$, then $X \models \varphi'$.

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 (see [2] for details).

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

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

In the rest of the paper, for simplicity, we omit to write the draw of f and its inverse, and we do not mention them in the description of the configurations either.

4.2 Some Properties of the Assertion Language

In this section, we prove properties of our assertions that are useful for proving IND-CPA security of encryption schemes and soundness of our Hoare Logic.

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

Lemma 1 *For any* $X, X' \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$*, any sets of variables* V_1 *and* V_2*, and any variable* x*:*

1. *if* $X \sim_{V_1;V_2} X'$ *then* $X \models \mathsf{Indis}(\nu x; V_1; V_2) \iff X' \models \mathsf{Indis}(\nu x; V_1; V_2)$.
2. *if* $X \sim_{V_1;V_2\cup\{x\}} X'$ *then* $X \models \mathsf{WS}(x; V_1; V_2) \iff X' \models \mathsf{WS}(x; V_1; V_2)$.
3. *if* $X \sim X'$ *then* $X \models \mathsf{H}(H, e) \iff X' \models \mathsf{H}(H, e)$.

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

Let us start with the proof of the first item. We assume $X \sim_{V_1;V_2} X'$ which is equivalent to $X' \sim_{V_1;V_2} X$. Hence, $\nu x.X \sim_{V_1;V_2} \nu x.X'$; this 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 now consider the second item. We prove by reduction that $X' \models \mathsf{WS}(x; V_1; V_2)$ must hold, provided that $X \sim_{V_1;V_2\cup\{x\}} X'$ and $X \models \mathsf{WS}(x; V_1; V_2)$. Given an adversary $\mathcal{A}$ that falsifies $X' \models \mathsf{WS}(x; V_1; V_2)$, we construct an adversary $\mathcal{B}$ that falsifies $X \sim_{V_1;V_2\cup\{x\}} X'$. Informally, on input values for variables in V_1 and V_2, denoted $(\mathbf{v}, \mathbf{v}')$, and a value u for $f(x)$, $\mathcal{B}$ runs A on $(\mathbf{v}, \mathbf{v}')$, which outputs v_x, its guess for x. Then $\mathcal{B}$ compares $f(v_x)$ to u. If the equality holds, $\mathcal{B}$ answers 1 (that is, the values $(\mathbf{v}, \mathbf{v}')$ were sampled according to distribution X'); otherwise $\mathcal{B}$ answers a bit picked at random uniformly.

Let A denote the event $\mathcal{A}^{\mathbf{H}}(S(V_1), f(S(V_2))) = f(S(x))$, $\neg A$ denote the event $\mathcal{A}^{\mathbf{H}}(S(V_1), f(S(V_2))) \neq f(S(x))$ and B denote the event $\mathcal{B}^{\mathbf{H}}(S(V_1), f(S(V_2)),$

$f(S(x))) = 1$. Moreover, let $\mathsf{Adv}_{\mathcal{B}}$ abbreviate $\mathsf{Adv}(\mathcal{B}^{\mathbf{H}}, \eta, D(X, V_1, V_2 \cup \{x\})_\eta, D(X', V_1, V_2 \cup \{x\})_\eta)$. Then, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}} &= \left|\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : B\right] - \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : B\right]\right| \\
&= \left|\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : B|A\right].\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : A\right]\right. \\
&\quad +\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : B|\neg A\right].\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : \neg A\right] \\
&\quad -\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : B|A\right].\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : A\right] \\
&\quad \left.-\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : B|\neg A\right].\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : \neg A\right]\right| \\
&= \left|\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : A\right] + \frac{1}{2}.\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : \neg A\right]\right. \\
&\quad \left.-\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : A\right] - \frac{1}{2}.\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : \neg A\right]\right| \\
&\qquad \text{using } \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : B|A\right] = 1 = \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : B|A\right] \\
&\qquad \text{and } \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : B|\neg A\right] = \frac{1}{2} = \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : B|\neg A\right] \\
&= \frac{1}{2}.\left|\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : A\right] - \frac{1}{2}.\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X : A\right]\right| \\
&\qquad \text{using } \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : \neg A\right] = 1 \quad - \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : A\right] \\
&\qquad \text{and } \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : \neg A\right] = 1 - \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : A\right]
\end{aligned}$$

Therefore,

$$\left|\Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : A\right] - \Pr\left[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : A\right]\right| = 2\mathsf{Adv}_{\mathcal{B}}.$$

Since $X \models \mathsf{WS}(x; V_1; V_2)$, we obtain that $\left|\Pr[(S,\mathbf{H}) \overset{r}{\leftarrow} X_\eta : A]\right|$ is negligible. Hence $\left|\Pr[(S,\mathbf{H}) \overset{r}{\leftarrow} X'_\eta : A]\right|$ is negligible if and only if $\mathsf{Adv}_{\mathcal{B}}$ is negligible.

Concerning the third item, it easy to see that a polynomial adversary dealing with $X \sim X'$, has access to all variables in $\mathsf{Var} \cup \mathbb{T}_{\mathbf{H}}$, and hence he can evaluate the expression e and check whether the value he gets, is or not among the bit-strings obtained by projecting the list given by $\mathbb{T}_H \in \mathbb{T}_{\mathbf{H}}$ to the first element. □

We now present a lemma that relates atomic assertions and states some monotonicity properties:

Lemma 2 *Let $X \in \text{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ be a distribution:*

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 $X \models \mathsf{WS}(x; V_1; V_2)$ and $V'_1 \subseteq V_1$ and $V'_2 \subseteq V_1 \cup V_2$ then $X \models \mathsf{WS}(x; V'_1; V'_2)$.*
3. *If $X \models \mathsf{Indis}(\nu x; V_1; V_2 \cup \{x\})$ and $x \notin V_1 \cup V_2$ then $X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$.*

Proof The first two properties are straightforward.
To prove the last assertion, we let $X \in \text{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ such that $X \models \mathsf{Indis}(\nu x; V_1; V_2 \cup \{x\})$. Thus, $X \sim_{V_1; V_2 \cup \{x\}} \nu x.X$. Lemma 1 allows us to say that it is sufficient to prove that $\nu x.X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$ in order to conclude that $X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$. Let us consider an adversary $\mathcal{A}$ against $\nu x.X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$. It takes as input, among others, a value $f(u)$ for $f(x)$,

with $u \overset{r}{\leftarrow} \mathcal{U}$. If $\mathcal{A}$ successfully computes a value for x, it obviously equals u, so that $\mathcal{A}$ in fact computes the pre-image of the one-way function f on the random value $f(u)$. This latter event has a negligible probability to happen. Hence, we can conclude that $X \models \mathsf{WS}(x; V_1; V_2 \cup \{x\})$. □

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 computations.

Lemma 3 *For any* $X, X' \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$*, 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* $[\![x := e]\!](X) \sim_{V_1 \cup \{x\};V_2} [\![x := e]\!](X')$.

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

Corollary 1 *For any* $X, X' \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$*, 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* $z \notin \{x\} \cup \mathit{Var}(e)$ *if* $X \models \mathit{Indis}(\nu z; V_1; V_2)$ *then* $[\![x := e]\!](X) \models \mathit{Indis}(\nu z; V_1 \cup \{x\}; V_2)$*. We emphasize that here we use the notation* $\mathit{Var}(e)$ *(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 3 we get $[\![x := e]\!](X) \sim_{V_1 \cup \{x\};V_2} [\![x := e]\!](\nu z.X)$. Since $z \notin \{x\} \cup \mathsf{Var}(e)$ we have that $[\![x := e]\!](\nu z.X) = \nu z.[\![x := e]\!](X)$ and hence $[\![x := e]\!](X) \sim_{V_1 \cup \{x\};V_2} \nu z.[\![x := e]\!](X)$, that is $[\![x := e]\!](X) \models \mathsf{Indis}(\nu z; V_1 \cup \{x\}; V_2)$. □

Lemma 4 *For any* $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$*, 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* $[\![x := e]\!](X) \models \mathit{WS}(z; V_1 \cup \{x\}; V_2)$.

Proof If we suppose that $[\![x := e]\!](X) \not\models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$, then there exists $\mathcal{A}$ 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 $\mathcal{B}$ be the following adversary against $X \models \mathsf{WS}(z; V_1; V_2)$:
$\mathcal{B}(V_1, f(V_2))$:= let $x := e$ in $\mathcal{A}(V_1 \cup \{x\}, f(V_2))$.

Since $\mathcal{A}$ and $\mathcal{B}$ have the same advantage, we obtain a contradiction, so that $[\![x := e]\!](X) \not\models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$ cannot be true. □

4.3 The Hoare Logic

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 control point following the command. Then, for each command, we present a set of specific rules that allow us to establish new invariants. The commands that are not considered are usually irrelevant for IND-CPA security. We summarize all our Hoare logic rules in Table 3 (given at the end of the paper).

4.3.1 Generic Preservation Rules

We assume $z \neq x$[1] and c is $x \overset{r}{\leftarrow} \mathcal{U}$ or of the form $x := e'$ with e' being either $t||y$ or $t \oplus y$ or $f(y)$ or $H(y)$ or $t \oplus H(y)$.

Before getting started, let us notice that for any of these commands c, $[\![\mathsf{c}]\!]$ affects at most x and $\mathbb{T}_H$. The next lemma directly follows from this remark.

Lemma 5 *For every $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, all sets V_1 and V_2 such that $x \notin V_1 \cup V_2$, and all commands c of the form $x \overset{r}{\leftarrow} \mathcal{U}$ or $x := e$, we have $[\![\mathsf{c}]\!](X) \sim_{V_1;V_2} X$.*

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

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

□

We now give generic preservation rules for our predicates. We comment them right below.

Lemma 6 *The following rules are sound, when $z \neq x$, and c is $x \overset{r}{\leftarrow} \mathcal{U}$ or of the form $x := e'$ with e' being either $t||y$ or $t \oplus y$ or $f(y)$ or $H(y)$ or $t \oplus H(y)$:*

- (G1) $\{\mathit{Indis}(\nu z; V_1; V_2)\}\ \mathsf{c}\ \{\mathit{Indis}(\nu z; V_1; V_2)\}$, *provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$.*
- (G2) $\{\mathit{WS}(z; V_1; V_2)\}\ \mathsf{c}\ \{\mathit{WS}(z; V_1; V_2)\}$, *provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$.*

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

Table 3 Summary of our Hoare logic rules

Generic preservation rules: when $z \neq x^1$ and c is $x \xleftarrow{r} \mathcal{U}$ or of the form $x := e'$ with e' being either $t||y$ or $t \oplus y$ or $f(y)$ or $H(y)$ or $t \oplus H(y)$

(G1) $\{\mathsf{Indis}(\nu z; V_1; V_2)\}$ c $\{\mathsf{Indis}(\nu z; V_1; V_2)\}$, provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$

(G2) $\{\mathsf{WS}(z; V_1; V_2)\}$ c $\{\mathsf{WS}(z; V_1; V_2)\}$, provided $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$

(G3) $\{\mathsf{H}(H', e[e'/x])\}$ c $\{\mathsf{H}(H', e)\}$, provided $H' \neq H$ in case c is either $x := H(y)$ or $x := t \oplus H(y)$ Here, by convention, $e[e'/x]$ is either e if c is $x \xleftarrow{r} \mathcal{U}$ or the expression obtained from e by replacing x by e' in case $c \equiv x := e'$

Random assignment rules

(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 $x \in \mathsf{subvar}(e)$

We assume $x \neq y$, for the next two rules

(R3) $\{\mathsf{Indis}(\nu y; V_1; V_2)\} x \xleftarrow{r} \mathcal{U} \{\mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2)\}$

(R4) $\{\mathsf{WS}(y; V_1; V_2)\} x \xleftarrow{r} \mathcal{U} \{\mathsf{WS}(y; V_1 \cup \{x\}; V_2)\}$

Hash functions rules: when $x \neq y$, and α is either a constant or a variable

(H1) $\{\mathsf{WS}(y; V_1; V_2) \wedge \mathsf{H}(H, y)\} x := \alpha \oplus \mathsf{H}(y) \{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$

(H2) $\{\mathsf{H}(H, y)\}$ $x := H(y)$ $\{\mathsf{H}(H', e)\}$, if $x \in \mathsf{subvar}(e)$

(H3) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)\}$ $x := H(y)$ $\{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2 \cup \{y\})\}$ if $y \notin V_1$

We assume $x \neq y$ and $z \neq x$ for the next rules

(H4) $\{\mathsf{WS}(y; V_1; V_2) \wedge \mathsf{WS}(z; V_1; V_2) \wedge \mathsf{H}(H, y)\} x := H(y)$ $\{\mathsf{WS}(z; V_1 \cup \{x\}; V_2)\}$

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

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

One-way function rules: when $y \notin V \cup \{x\}$

(O1) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})\}$ $x := f(y)$ $\{\mathsf{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})\}$

(O2) $\{\mathsf{Indis}(\nu z; V_1 \cup \{z\}; V_2 \cup \{y\})\}$ $x := f(y)$ $\{\mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2 \cup \{y\})\}$, if $z \neq y$

(O3) $\{\mathsf{WS}(z; V_1; V_2) \wedge \mathsf{Indis}(\nu y; V_1; \{y, z\} \cup V_2)\}$ $x := f(y)$ $\{\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})\}$

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

(PO1) $\{\mathsf{Indis}(\nu x; V \cup \{x, y\}) \wedge \mathsf{Indis}(\nu y; V \cup \{x, y\})\} z := f(x||y) \{\mathsf{WS}(x; V \cup \{z\}) \wedge \mathsf{Indis}_f(\nu z; V \cup \{z\})\}$

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

Exclusive or rules: when $y \notin V_1 \cup V_2$, and $y \neq x$

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

(X2) $\{\mathsf{Indis}(\nu t; V_1 \cup \{y, z\}; V_2)\} x := y \oplus z \{\mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, provided that $t \neq x, y, z$

(X3) $\{\mathsf{WS}(t; V_1 \cup \{y, z\}; V_2)\} x := y \oplus z \{\mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)\}$, if $t \neq x$

Concatenation rules

(C1) $\{\mathsf{WS}(y; V_1; V_2)\}$ $x := y||z$ $\{\mathsf{WS}(x; V_1; V_2)\}$, if $x \notin V_1 \cup V_2$. A dual rule applies for z

(C2) $\{\mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2) \wedge \mathsf{Indis}(\nu z; V_1 \cup \{y, z\}; V_2)\}$ $x := y||z$ $\{\mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$, if $y, z \notin V_1 \cup V_2$

(C3) $\{\mathsf{Indis}(\nu t; V_1 \cup \{y, z\}; V_2)\}$ $x := y||z$ $\{\mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, if $t \neq x, y, z$

(C4) $\{\mathsf{WS}(t; V_1 \cup \{y, z\}; V_2)\}$ $x := y||z$ $\{\mathsf{WS}(t; V_1 \cup \{y, z, x\}; V_2)\}$, if $t \neq x$

Consequence and sequential composition rules

(Csq) if $\varphi_0 \Rightarrow \varphi_1$, $\{\varphi_1\}$ c $\{\varphi_2\}$ and $\varphi_2 \Rightarrow \varphi_3$ then $\{\varphi_0\}$ c $\{\varphi_3\}$

(Seq) if $\{\varphi_0\}$ c_1 $\{\varphi_1\}$ and $\{\varphi_1\}$ c_2 $\{\varphi_2\}$, then $\{\varphi_0\}$ $\mathsf{c}_1; \mathsf{c}_2$ $\{\varphi_2\}$

(Conj) if $\{\varphi_0\}$ c $\{\varphi_1\}$ and $\{\varphi_2\}$ c $\{\varphi_3\}$, then $\{\varphi_0 \wedge \varphi_2\}$ c $\{\varphi_1 \wedge \varphi_3\}$

– (G3) $\{H(H', e[e'/x])\}$ c $\{H(H', e)\}$, *provided* $H' \neq H$ *in case* c *is either* $x := H(y)$ *or* $x := t \oplus H(y)$. *Here, by convention,* $e[e'/x]$ *is either* e *if* c *is* $x \xleftarrow{r} \mathcal{U}$ *or the expression obtained from* e *by replacing* x *by* e' *in case* $c \equiv x := e'$.

The rules deal with predicates on a variable z different from x, y, t appearing in the command c that is applied. Thus, the predicates Indis and WS are quite intuitively preserved as soon as either x does not appear in sets V_1, V_2 the adversary is provided, or x does appear, but the value of e' was already deducible from values given for V_1, V_2. As for rule (G3), it is meant to express preservation of predicate $\mathsf{H}(H', .)$. Intuitively, if for states drawn in distribution X, $e[e'/x]$ has negligible probability to belong to $\mathbb{T}_H$ before performing $x := e'$, then e has negligible probability to belong to $\mathbb{T}_H$ for states drawn in $[\![x := e']\!](X)$, that is, once the command is performed.

Proof

(G1) The case when e' is constructible from $(V_1 \setminus \{z\}; V_2 \setminus \{z\})$ follows from Corollary 1. Let us suppose that $x \notin V_1 \cup V_2$. 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 1, $X \models \mathsf{Indis}(\nu z; V_1; V_2)$ implies $[\![\mathsf{c}]\!](X) \models \mathsf{Indis}(\nu z; V_1; V_2)$. The case of $x \xleftarrow{r} \mathcal{U}$ is obvious.

(G2) As for (G1) using Lemma 4 instead of Corollary 1.

(G3) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ and any $[\![\mathsf{c}]\!]$ affecting at most x and $\mathbb{T}_H$ such that $X \models \mathsf{H}(H', e[e'/x])$. Let $p_\eta = \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} [\![\mathsf{c}]\!](X_\eta) : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom}]$. Then $p_\eta = \mathsf{Pr}[S' \xleftarrow{r} X_\eta; S \xleftarrow{r} [\![\mathsf{c}]\!](S', \mathbf{H}) : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom}]$. Now, $S'(e[e'/x])^2$ is equal to $S(e)$ and $S'(\mathbb{T}_{H'}).\mathsf{dom} = S(\mathbb{T}_{H'}).\mathsf{dom}$. Hence,

$$\begin{aligned} p_\eta &= \mathsf{Pr}\big[S' \xleftarrow{r} X_\eta; S \xleftarrow{r} [\![\mathsf{c}]\!]S' : S'(e[e'/x]) \in S'(\mathbb{T}_{H'}).\mathsf{dom}\big] \\ &= \mathsf{Pr}\big[(S', \mathbf{H}) \xleftarrow{r} X_\eta : S'(e[e'/x]) \in S'(\mathbb{T}_{H'}).\mathsf{dom}\big] \end{aligned}$$

Since $X \models \mathsf{H}(H', e[e'/x])$, the last probability is a negligible function of η. Therefore p_η is also negligible in η and $[\![\mathsf{c}]\!](X) \models \mathsf{H}(H', e)$. □

4.3.2 Random Assignment

Rule (R1) below states that $\mathsf{Indis}(\nu x)$ is satisfied after assigning a randomly sampled value to the variable x. Rule (R2) takes advantage of the fact that the cardinality of $\mathcal{U}$ is exponential in the security parameter, and that since e contains the freshly 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.

Lemma 7 *The following rules are sound:*

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

²We recall that in the case of $x \xleftarrow{r} \mathcal{U}$, $e[e'/x]$ is actually e.

Additionally, we have the following preservation rules, where we assume $x \neq y$, are sound:

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

Proof

(R1) Immediate.

(R2) The fact that $x \in \mathsf{subvar}(e)$ implies 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

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

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

(R3) The intuition is that x being completely random, providing its value to the adversary does not help him in any way. We show the result by reduction. Assume that there exists an adversary B against $[\![x \xleftarrow{r} \mathcal{U}]\!](X) \models \mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2)$ that can distinguish with non-negligible advantage between y and a random value given the values of $V_1 \cup \{x\}$ 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.

(R4) The previous reduction can be adapted in a straightforward way to prove (R4). □

4.3.3 Hash Functions

In this section, we present a set of proof rules that deal with hash functions in the random oracle model. We first state properties of hash functions that are used to prove soundness of our proof 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, then one can redraw it without this changing anything from the adversary's point of view. We introduce the notation $H\{v \mapsto u\}$ to denote the function behaving like H at any point except v, with which it associates u.

Lemma 8 (Dynamic draw) *For any* $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ *and any* $y \in \mathsf{Var}$ *such that* $X \models \mathsf{H}(H, y)$, $X \sim ([(S,H) \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; \mathbf{H} \leftarrow \{H\{S(y) \mapsto u\}\} \cup (\mathbf{H} \setminus \{H\}) : (S, \mathbf{H})])_\eta$.

Proof Let $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ and $y \in \mathsf{Var}$ such that $X \models \mathsf{H}(H, y)$. We denote X'_η the distribution $[(S, \mathbf{H}) \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; \mathbf{H} \leftarrow \{H\{S(y) \mapsto u\}\} \cup (\mathbf{H} \setminus \{H\}) : (S, \mathbf{H})]$. Let us we recall that for $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, all queries that have been made to the hash oracles are recorded in the lists $\mathbb{T}_H$.

First, we begin with some remarks which help us bounding the advantage of an adversary $\mathcal{A}$ trying to distinguish between X and X'. Since $X \models \mathsf{H}(H, y)$, and using the definition of X', it follows that the probability $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}]$ is equal to $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}] = n(\eta)$ where $n(\cdot)$ is a negligible function. Indeed, the state output by distribution X_η is not modified in the computation of X'_η. Moreover $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}] = \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}]$, since under the condition $S(y) \notin S(\mathbb{T}_H).\mathsf{dom}$, drawing $\mathbf{H}$ in Ω and redrawing the value of H on $S(y)$ yields the same distribution as just drawing $\mathbf{H}$ in Ω.

$$\begin{aligned}
&\mathsf{Adv}(\mathcal{A}^{\mathbf{H}}, \eta, D(X, \mathsf{Var}, \emptyset)_\eta, D(X', \mathsf{Var}, \emptyset)_\eta)\\
&\quad = \big|\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1]\\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1]\big|
\end{aligned}$$

We then distinguish according to whether $S(y) \in \mathbb{T}_H.\mathsf{dom}$

$$\begin{aligned}
&\quad = \big|\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}].\\
&\qquad\ \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : S(y) \notin S(\mathbb{T}_H).\mathsf{dom}]\\
&\qquad + \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}].\\
&\qquad\ \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}]\\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \notin S(\mathbb{T}_H).\mathsf{dom}].\\
&\qquad\ \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \notin S(\mathbb{T}_H).\mathsf{dom}]\\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}].\\
&\qquad\ \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}]\big|
\end{aligned}$$

We then take into account the equalities between terms justified above.

$$\begin{aligned}
&\quad = \big|\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}]\\
&\qquad - \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : \mathcal{A}^{\mathbf{H}}(S(\mathsf{Var})) = 1 | S(y) \in S(\mathbb{T}_H).\mathsf{dom}]\big|.\\
&\qquad\ \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(y) \in S(\mathbb{T}_H).\mathsf{dom}]\\
&\quad \leq 2.n(\eta)
\end{aligned}$$

□

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: $S(\mathbb{T}_H) \bullet (v, u)$ means that if v belongs to $S(\mathbb{T}_H).\mathsf{dom}$, then its associated value in $S(\mathbb{T}_H).\mathsf{res}$ is replaced by u, otherwise, it is the concatenation of (v, u) to $S(\mathbb{T}_H)$.

Definition 7 We define $\mathsf{rebind}_H^{y \mapsto e}(S, \mathbf{H})$ by

$$(S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), S(e))\}, \mathbf{H} \setminus \{H\} \cup \{H\{S(y) \mapsto S(e)\}\}).$$

We extend this definition canonically to any family of distributions $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$:

$\mathsf{rebind}_H^{y \mapsto e}(X) = ([(S, \mathbf{H}) \overset{r}{\leftarrow} X_\eta : \mathsf{rebind}_H^{y \mapsto e}(S, \mathbf{H})])_\eta$. It simply denotes the family of distributions where $H(S(y))$ is defined to be equal to $S(e)$.

Lemma 9 (Rebinding Lemma) *For any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, 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. First, since $X \models \mathsf{H}(H, y)$, thanks to the dynamic draw lemma, we know that $X_\eta \sim [(S, H) \overset{r}{\leftarrow} X_\eta; u \overset{r}{\leftarrow} \mathcal{U}_\eta : (S, H\{S(y) \mapsto u\})]$. Then, using a similar reasoning to that used in the proof of Lemma 3 (but this time the adversary $\mathcal{B}$ has to update also the variable T_H and to pass it to the adversary $\mathcal{A}$), we get:

$$[\![x{:=}\alpha \oplus H(y)]\!](X_\eta) \sim [\![x{:=}\alpha \oplus H(y)]\!] \left(\left[(S, H) \overset{r}{\leftarrow} X_\eta; u \overset{r}{\leftarrow} \mathcal{U}_\eta : (S, H\{S(y) \mapsto u\})\right]\right).$$

Executing the hash command, the second distribution is in turn equal to

$$\left[(S, H) \overset{r}{\leftarrow} X_\eta; u \overset{r}{\leftarrow} \mathcal{U}_\eta : (S\{x \mapsto S(\alpha) \oplus u; \mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), u)\}, H\{S(y) \mapsto u\})\right].$$

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:

$$\Big[(S, H) \overset{r}{\leftarrow} X_\eta; v \overset{r}{\leftarrow} \mathcal{U}_\eta : (S\{x \mapsto v, \mathbb{T}_H \mapsto S(\mathbb{T}_H)$$

$$\bullet (S(y), v \oplus S(\alpha))\}, H\{S(y) \mapsto v \oplus S(\alpha)\})\Big].$$

Now, this last distribution is exactly $(\mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X))_\eta$, 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 10 (Hash vs. rebind) *For any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, 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*

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

where α is either a constant or a variable.

Proof Consider finite sets V_1 and V_2 and X such that $X \models \mathsf{WS}(y; V_1; V_2)$. The sole difference between the distributions is the value of $H(y)$. Namely,

$$\begin{aligned}
D(\mathsf{rebind}_H^{y \mapsto x \oplus \alpha}(X), V_1, V_2)_\eta = {} & [(S, \mathbf{H}) \xleftarrow{r} X_\eta; \\
& S' \leftarrow S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), S(\alpha \oplus x))\} : \\
& (S'(V_1), f(S'(V_2)), H\{S(y) \mapsto S(\alpha \oplus x)\} \cup (\mathbf{H} \setminus \{H\}))] \\
& \text{since } \mathbb{T}_H \notin V_1 \cup V_2 \text{ by definition,} \\
& \text{and it is the only difference between } S \text{ and } S' \\
= {} & [(S, \mathbf{H}) \xleftarrow{r} X_\eta; \\
& S' \leftarrow S\{\mathbb{T}_H \mapsto S(\mathbb{T}_H) \bullet (S(y), S(\alpha \oplus x))\} : \\
& (S(V_1), f(S(V_2)), H\{S(y) \mapsto S(\alpha \oplus x)\} \cup (\mathbf{H} \setminus \{H\}))] \\
& \text{and since } S' \text{ is not used anywhere,} \\
= {} & [(S, \mathbf{H}) \xleftarrow{r} X_\eta : \\
& (S(V_1), f(S(V_2)), H\{S(y) \mapsto S(\alpha \oplus x)\} \cup (\mathbf{H} \setminus \{H\}))]
\end{aligned}$$

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

Proof rules for hash functions

We are now prepared to present and prove our proof rules for hash functions. 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): as the hash of a fresh value is seemingly random, it has negligible probability to have already been queried to another hash oracle.

Rule (H3) deserves a more elaborate comment. It concludes to a predicate still involving variable y, which is why it is different from rule (H1). It states that the value of variable x is random given first values of V_1, x and $f(V_2)$ as in rule (H1), but also the value of $f(y)$. Indeed, as the value of y is seemingly random, we can use the definition of one-wayness to state that an adversary cannot efficiently compute a satisfactory value for $f^{-1}(f(y))$. Hence, the value of y is unlikely to be queried to H, and the predicate holds.

Lemma 11 *The following basic rules are sound, when $x \neq y$, and α is either a constant or a variable:*

- (H1) $\{\mathit{WS}(y; V_1; V_2) \wedge \mathit{H}(H, y)\} x := \alpha \oplus \mathit{H}(y) \{\mathit{Indis}(\nu x; V_1 \cup \{x\}; V_2)\}$
- (H2) $\{\mathit{H}(H, y)\}\ x := H(y)\ \{\mathit{H}(H', e)\}$, *if* $x \in \mathit{subvar}(e)$.
- (H3) $\{\mathit{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathit{H}(H, y)\}\quad x := H(y)\quad \{\mathit{Indis}(\nu x; V_1 \cup \{x\}; V_2 \cup \{y\})\}$ *if* $y \notin V_1$

Proof

(H1) First, we use that $X \models \mathsf{WS}(y; V_1; V_2)$, that provides thanks to rule (R4) $\nu x.X \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2)$. Hence, the 'hash-vs-rebind' Lemma 10 applies, we obtain the following $\nu x.X \sim_{V_1 \cup \{x\}; V_2} \mathsf{rebind}_H^{y \mapsto x \oplus \alpha}(\nu x.X)$. Then, as we assumed that $X \models \mathsf{H}(H, y)$, we can use the rebinding lemma, according to which we have the following $\mathsf{rebind}_H^{y \mapsto \alpha \oplus x}(\nu x \cdot X) \sim_{V_1 \cup \{x\}; V_2} [\![x := \alpha \oplus H(y)]\!](X)$. By transitivity of the indistinguishability relation, we thus have $\nu x.X \sim_{V_1 \cup \{x\}; V_2} [\![x := \alpha \oplus H(y)]\!](X)$. Finally, noticing that $\nu x.X \sim_{V_1 \cup \{x\}; V_2} \nu x.[\![x := \alpha \oplus H(y)]\!](X)$ (since carrying out the command only impacts on the values of x and $\mathbb{T}_H$ these family of distributions are in fact equal), we have by transitivity $\nu x.[\![x := \alpha \oplus H(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2} [\![x := \alpha \oplus H(y)]\!](X)$. This last statement is equivalent to $[\![x := \alpha \oplus H(y)]\!](X) \models \mathsf{Indis}(x; V_1 \cup \{x\}; V_2)$.

(H2) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ 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 $x \in \mathsf{subvar}(e)$. Using Lemma 1(3), it suffices to show $X' \models \mathsf{H}(H', e)$, that is, that $p_\eta = \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X'_\eta : S(e) \in S(\mathbb{T}_{H'}).\mathsf{dom}]$ is negligible.

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

Moreover, for every state S_1, $Card(S_1(\mathbb{T}_H).\mathsf{dom})$ is bounded by a polynomial in η, and variables in Var have a size polynomial in η too, so that p_η is indeed a negligible function in η.

(H3) Consider any $X \in \mathrm{DIST}(\Gamma, \mathbf{H}, \mathbb{F})$. Assume $y \notin V_1$ and $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)$. Then, $X \models \mathsf{WS}(y; V_1; V_2 \cup \{y\})$ follows from the third weakening lemma (see Lemma 2). Consequently, rule H1 provides $[\![x := H(y)]\!](X) \models \mathsf{Indis}(\nu x; V_1 \cup \{x\}; V_2 \cup \{y\})$. □

We now comment on four other rules to deal with hash commands which are stated below. The idea behind (H4) is the following one: an adversary that is not able to compute the value of y, can not ask this value to H; hence, the value of x (computed as $H(y)$) seems completely random; so if z was not efficiently computable by the adversary given V_1, $f(V_2)$, it remains so when it is additionally provided x. Rule (H5) states that the fact that the value of e has probably not been hashed yet remains true after a hash command, as long as e contains a variable z whose value is not computable out from y. (H6) and (H7) give necessary conditions to the preservation of indistinguishability that is based on the apparent randomness of a hash value. The intuition behind rule (H6) is very similar to that of rule (H3). As for rule (H7), as we want more than just preserving the seemingly randomness of z with respect to V_1, $f(V_2)$, the conditions under which x doesn't help an adversary are that y is not easily deductible from V_1, V_2 and that x is a fresh hash value.

Lemma 12 *The following preservation rules are sound provided that $x \neq y$ and $z \neq x$:*

- (H4) $\{\mathsf{WS}(y; V_1; V_2) \wedge \mathsf{WS}(z; V_1; V_2) \wedge \mathsf{H}(H, y)\}x := H(y) \quad \{\mathsf{WS}(z; V_1 \cup \{x\}; V_2)\}$
- (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) $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)\} \quad x := H(y) \quad \{\mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2 \cup \{y\})\}$, *if* $y \notin V_1$
- (H7) $\{\mathsf{Indis}(\nu z; V_1 \cup \{z\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z\}; V_2) \wedge \mathsf{H}(H, y)\}x := H(y)\{\mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2)\}$

Proof

(H4) First, we use rule (R4), to state that since $X \models \mathsf{WS}(y; V_1; V_2)$, $\nu x.X \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2)$. Then, from the hash-vs-rebind Lemma 10 applied on $\nu x.X$, we obtain that $\nu x.X \sim_{V_1 \cup \{x\}; V_2} \mathsf{rebind}_H^{y \mapsto x}(\nu x.X)$. Now, using the assumption $X \models \mathsf{H}(H, y)$ and the rebinding lemma, $\mathsf{rebind}_H^{y \mapsto x}(\nu x.X) \sim_{V_1 \cup \{x\}; V_2} [\![x := H(y)]\!](X)$. Hence, $\nu x.X \sim_{V_1 \cup \{x\}; V_2} [\![x := H(y)]\!](X)$. Besides, as $X \models \mathsf{WS}(z; V_1; V_2)$, rule (R4) provides the conclusion $\nu x.X \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$. With Lemma 1, we can conclude that $[\![x := H(y)]\!](X) \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2)$ 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 $\mathsf{WS}(z; V_1; V_2)$ before the execution of the command could simulate an adversary against $\mathsf{WS}(z; V_1 \cup \{x\}; V_2)$ by providing this latter with a randomly sampled value in place of x. Both those adversaries would therefore have the same advantage.

(H5) Since $z \in \mathsf{subvar}(e)$, there is a polynomial function g such that for every S, $g(S(e)) = S(z)$ (namely g consists in extracting the right substring corresponding to z from the expression e). Given $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$, let p_η be equal to:

$$\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta; (S', \mathbf{H}') \xleftarrow{r} [\![x := H(y)]\!](S) : S'(e) \in S'(\mathbb{T}_H).\mathsf{dom}]$$

Then, since the command only has an effect on x and $\mathbb{T}_H$,

$$\begin{aligned} p_\eta &= \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta; (S', \mathbf{H}') \xleftarrow{r} [\![x := H(y)]\!](S) : \\ &\qquad S(e) \in S(\mathbb{T}_H).\mathsf{dom} \cup \{S(y)\}] \\ &\leq \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) \in S(\mathbb{T}_H)] + \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) = S(y)] \end{aligned}$$

Now, we can bound the second term as follows:

$$\begin{aligned} \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) = S(y)] &\leq \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : g(S(e)) = g(S(y))] \\ &= \mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(z) = g(S(y))] \\ &\qquad \text{for this is how } g \text{ was defined} \end{aligned}$$

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 $\mathsf{Pr}[(S, \mathbf{H}) \xleftarrow{r} X_\eta : S(e) \in S(\mathbb{T}_H)]$ is negligible.

(H6) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$. Assume $y \notin V_1$, and $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\}) \wedge \mathsf{H}(H, y)$. By rule (R3) for random assignment, since $y \neq x$, $\nu x \cdot X \models \mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2 \cup \{y\})$. Therefore, using that $y \notin V_1$ and $y \neq x$ and applying Lemma 2.3 we get $\nu x \cdot X \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})$. Now, the hash-vs-rebind Lemma 10 provides us with $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \sim_{V_1 \cup \{x\}; V_2 \cup \{y\}} \nu x \cdot X$. Thus, $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \models \mathsf{Indis}(\nu y; V_1 \cup \{x\}; V_2 \cup \{y\})$ by Lemma 1. 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 1.

(H7) Consider any $X \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ such that $X \models \mathsf{Indis}(\nu z; V_1 \cup \{z\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z\}; V_2) \wedge \mathsf{H}(H, y)$. By rule (R3) and (R4) for random assignment, and because $x \neq z, y$, $\nu x \cdot X \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2) \wedge \mathsf{WS}(y; V_1 \cup \{z, x\}; V_2)$. Therefore, the hash-vs-rebind Lemma 10 allows to conclude that $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \sim_{V_1 \cup \{z,x\}; V_2} \nu x \cdot X$. Thus, by the preservation Lemma 1, $\mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X) \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2)$. Finally, the rebinding lemma entails $[\![x := H(y)]\!](X) \sim \mathsf{rebind}_H^{y \mapsto x}(\nu x \cdot X)$. Therefore $[\![x := H(y)]\!](X) \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2)$, once more by Lemma 1.

□

4.3.4 One-Way Functions

Rules to deal with one-way functions are given below. The first rule captures one-wayness of f. Indeed, it states that an adversary can not efficiently compute a pre-image to an apparently random challenge. Rule (O2) and (O3) are meant to provide a little more than mere preservation of the properties of z. (O2) is quite obvious since $f(y)$ is given to the adversary in the precondition. As for rule (O3), it follows from the fact that since y is apparently random with respect to values V_1, z, $f(V_2)$, hence computing x boils down to computing the image by f of a random value.

Consequently, providing an adversary with the values of x and $f(y)$ does not help it. Rule (P1) simply ensues from the fact that f is a permutation and is thus surjective. However, y has to be removed from the sets in the conclusion, otherwise an adversary could compare the value of $f(y)$ with the value given for x and trivially tell if x is real or random.

Lemma 13 *The following rules are sound when $z \neq x$:*

- (O1) $\{\mathit{Indis}(\nu y; V_1; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathit{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})\}$ *if* $y \notin V_1 \cup \{x\}$.
- (O2) $\{\mathit{Indis}(\nu z; V_1 \cup \{z\}; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathit{Indis}(\nu z; V_1 \cup \{z, x\}; V_2 \cup \{y\})\}$, *if* $z \neq y$
- (O3) $\{\mathit{WS}(z; V_1; V_2) \wedge \mathit{Indis}(\nu y; V_1; \{y, z\} \cup V_2)\}\ x := f(y)\ \{\mathit{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})\}$

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

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

Proof

(O1) Let X be such that $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})$. It follows from Lemma 2 that $X \models \mathsf{WS}(y; V_1; V_2 \cup \{y\})$. Since $f(y)$ is obviously constructible from $(V_1; V_2 \cup \{y\})$, we apply Lemma 4, to obtain $[\![x := f(y)]\!](X) \models \mathsf{WS}(y; V_1 \cup \{x\}; V_2 \cup \{y\})$. Notice that the one-wayness of f is not used apparently here. Indeed, the proof of the weakening lemma (see Lemma 2) uses it, and once we apply it, there is only a simple rewriting step left to be able to conclude.

(O2) Since $f(y)$ is constructible from $(V_1 \cup \{z\}; V_2 \cup \{y\})$, we apply Corollary 1 to obtain $[\![x := f(y)]\!](X) \models \mathsf{Indis}(\nu z; V_1 \cup \{z, x\}; V_2 \cup \{y\})$.

(O3) If $z = y$, then the assertion is a consequence of Rule (O1). Hence, we assume $z \neq y$. From $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{z, y\})$ it follows by definition that $X \sim_{V_1; V_2 \cup \{z,y\}} \nu y.X$. Using Lemma 3 we get $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2 \cup \{z,y\}} [\![x := f(y)]\!](\nu y.X)$. Now using Lemma 1, to be able to conclude to $[\![x := f(y)]\!](X) \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$, it suffices to prove $[\![x := f(y)]\!](\nu y.X) \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$. Intuitively, this comes from the randomness of x and y, which allows us to think it is useless to any adversary trying to compute z. Formally, we show that: $\Pr[S \xleftarrow{r} X_\eta; u \xleftarrow{r} \mathcal{U}_\eta; S_1 = S\{y \mapsto u; x \mapsto f(u)\} : \mathcal{A}(S_1(V_1), S_1(x), f(S_1(V_2)), f(S_1(y))) = S_1(z)]$ is negligible. Now let $\mathcal{A}$ be an efficient adversary against $\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$. Let $\mathcal{B}(\mathbf{v})$ be the adversary against $\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$ that proceeds as follows: it samples a value $u \xleftarrow{r} \mathcal{U}_\eta$ and replaces every occurrence of y by u, and every occurrence of x by $f(u)$, in the values $\mathbf{v}$ it got as an input. This provides a tuple of values $\mathbf{v}'$. Adversary $\mathcal{B}$ runs $\mathcal{A}$ on $\mathbf{v}'$, before outputting $\mathcal{A}$'s guess for the value of z. This adversary $\mathcal{B}$ has the same advantage as $\mathcal{A}$ in falsifying $\mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$. As we assumed this latter was an efficient adversary, $\mathcal{B}$ is efficient as well, which contradicts $X \models \mathsf{WS}(z; V_1 \cup \{x\}; V_2 \cup \{y\})$.

(P1) Since $X \models \mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})$ and $f(y)$ is constructible from $(V_1; V_2 \cup \{y\})$, we apply Lemma 3 to obtain $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2 \cup \{y\}} [\![x := f(y)]\!](\nu y.X)$, and by weakening (see Lemma 2) we get $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2}$

$[\![x := f(y)]\!](\nu y.X)$. Using that f is a permutation and $y \notin V_1 \cup V_2$, 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, $[\![x := f(y)]\!](X) \sim_{V_1 \cup \{x\}; V_2} \nu x.X$. Now we use $\nu x.X = \nu x.[\![x := f(y)]\!](X)$ to conclude.

□

4.3.5 The Exclusive or Operator

In the following rules, we assume $y \neq z$. 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. Of course, y has to be random given y *and* z and not just only y; otherwise, there may exist a some relation between both subterms of x that may allow an adversary to distinguish this latter from a random value. Rules (X2) and (X3) take advantage of the fact that is easy to compute x given y and z.

Lemma 14 *The following rule is sound when $y \notin V_1 \cup V_2$, and $y \neq x$:*

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

Moreover, we have the following rules that are sound:

– (X2) $\{\mathsf{Indis}(\nu t; V_1 \cup \{y, z\}; V_2)\} x := y \oplus z \{\mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)\}$, *provided that $t \neq x, y, z$.*
– (X3) $\{\mathsf{WS}(t; V_1 \cup \{y, z\}; V_2)\} x := y \oplus z \{\mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)\}$, *if $t \neq x$.*

Proof

(X1) Let X be such that $X \models \mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2)$, which means $X \sim_{(V_1 \cup \{y,z\}; V_2)} \nu y.X$. Moreover, $y \oplus z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$. We apply Lemma 3 to obtain $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,y,z\}; V_2} [\![x := y \oplus z]\!](\nu y.X)$, and by weakening (see Lemma 2) it we get $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,z\}; V_2} [\![x := y \oplus z]\!](\nu y.X)$.

$$D([\![x := y \oplus z]\!](\nu y.X), V_1 \cup \{x, z\}, V_2)_\eta$$
$$= [S \overset{r}{\leftarrow} X_\eta; u \overset{r}{\leftarrow} \mathcal{U}_\eta; S' := S\{y \mapsto u\}; S'' \overset{r}{\leftarrow} [\![x := y \oplus z]\!](S') :$$
$$S''(V_1 \cup \{x, z\}), f(S''(V_2))]$$
$$= [S \overset{r}{\leftarrow} X_\eta; u \overset{r}{\leftarrow} \mathcal{U}_\eta; S' := S\{y \mapsto u\}; S'' := S'\{x \mapsto u \oplus S(z)\} :$$
$$S''(V_1 \cup \{x, z\}), f(S''(V_2))]$$

and since xor is a permutation we can write:

$$= [S \overset{r}{\leftarrow} X_\eta; v \overset{r}{\leftarrow} \mathcal{U}_\eta; S'' := S\{x \mapsto v; y \mapsto v \oplus S(z)\} :$$
$$S''(V_1 \cup \{x, z\}), f(S''(V_2))]$$

but changing y is useless since $y \notin V_1 \cup V_2 \cup \{z\}$

$$= [S \overset{r}{\leftarrow} X_\eta; v \overset{r}{\leftarrow} \mathcal{U}_\eta; S'' := S\{x \mapsto v\} : S''(V_1 \cup \{x, z\}), f(S''(V_2))]$$
$$= D(\nu x.X, V_1 \cup \{x, z\}, V_2)_\eta$$

From this equality of distributions, we get $[\![x := y \oplus z]\!](\nu y.X) \sim_{V_1 \cup \{x,z\};V_2} \nu x.X$. Then, by transitivity of indistinguishability, we can conclude that $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,z\};V_2} \nu x.X$. Then, as $\nu x.X = \nu x.[\![x := y \oplus z]\!](X)$, and applying transitivity once more, we can conclude to $[\![x := y \oplus z]\!](X) \sim_{V_1 \cup \{x,z\};V_2} \nu x.[\![x := y \oplus z]\!](X)$, which is exactly the definition of $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu x; V_1 \cup \{x, z\}; V_2)$.

(X2) Since $y \oplus z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Corollary 1 to obtain $[\![x := y \oplus z]\!](X) \models \mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)$.

(X3) Since $y \oplus z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Lemma 4 to obtain $[\![x := y \oplus z]\!](X) \models \mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)$.

□

4.3.6 Concatenation

We have four rules to deal with concatenation command $x := y||z$. Rule (C1) states that if computing a substring of x out of the elements of V_1 and V_2 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. Of course, y has to be random given *y and z* and not just only y; otherwise, there might exist a dependency between both substrings of x that allows an adversary to distinguish this latter from a random value. A similar comment can be made concerning z. Eventually, rules (C3) and (C4) are more than the simple preservation of the properties of a variable t different from x, y, z that the preservation rules would provide. The value of x being easily computable from those of y and z accounts for soundness of these rules.

Lemma 15 *The following rules are sound:*

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

Proof

(C1) From $X \models \mathsf{WS}(y; V_1; V_2)$, for any adversary $\mathcal{A}$, we have that the probability $\mathsf{Pr}[S \overset{r}{\leftarrow} X_\eta : \mathcal{A}(S(V_1), f(S(V_2))) = S(y)]$ is negligible. This implies that for any adversary $\mathcal{B}$, $\mathsf{Pr}[S \overset{r}{\leftarrow} X_\eta : \mathcal{B}(S(V_1), f(S(V_2))) = S(y)||S(z)]$ is negligible; otherwise 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 as follows. $\mathcal{A}$ calls $\mathcal{B}$ and then uses the answer of $\mathcal{B}$ to extract the value of $S(y)$ from $S(y)||S(z)$. Since $x \notin V_1 \cup V_2$, we get that for any adversary $\mathcal{B}$, $\mathsf{Pr}[S \overset{r}{\leftarrow} [\![x := y||z]\!](X_\eta) : \mathcal{B}(S(V_1), f(S(V_2))) = S(x)] = \mathsf{Pr}[S \overset{r}{\leftarrow} X_\eta : \mathcal{B}(S(V_1), f(S(V_2))) = S(y)||S(z)]$, which is negligible.

(C2) We have that $X \models \mathsf{Indis}(\nu z; V_1 \cup \{y, z\}; V_2) \Rightarrow X \sim_{V_1 \cup \{y,z\};V_2} \nu z.X$, so that in turn $\nu y.X \sim_{V_1 \cup \{y,z\};V_2} \nu y.\nu z.X$. But $X \models \mathsf{Indis}(\nu y; V_1 \cup \{y, z\}; V_2)$ can be written as $X \sim_{V_1 \cup \{y,z\};V_2} \nu y.X$. Hence, by transitivity we get $X \sim_{V_1 \cup \{y,z\};V_2} \nu y.\nu z.X$. Since $y||z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Lemma 3 to obtain $[\![x := y||z]\!](X) \sim_{V_1\cup\{x,y,z\};V_2} [\![x := y||z]\!](\nu y.\nu z.X)$, and by weakening (see Lemma 2) we get $[\![x := y||z]\!](X) \sim_{V_1\cup\{x\};V_2} [\![x := y||z]\!](\nu y.\nu z.X)$. 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, $[\![x := y||z]\!](X) \sim_{V_1\cup\{x\};V_2} \nu x.X$.

(C3) Since $y||z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Corollary 1 to obtain $[\![x := y||z]\!](X) \models \mathsf{Indis}(\nu t; V_1 \cup \{x, y, z\}; V_2)$.

(C4) Since $y||z$ is constructible from $(V_1 \cup \{y, z\}; V_2)$, we apply Lemma 4 to obtain $[\![x := y||z]\!](X) \models \mathsf{WS}(t; V_1 \cup \{x, y, z\}; V_2)$.

□

4.3.7 Additional General Rules

Classically, to reason on programs built according to the language grammar described in Table 1, we additionally need the following couple of rules.

Lemma 16 *Let $\varphi_0, \varphi_1, \varphi_2, \varphi_3$ be assertions from our language, and c, c_1, c_2 be any commands. The following rules are sound:*

– (Csq) *if $\varphi_0 \Rightarrow \varphi_1$ and $\{\varphi_1\}$ c $\{\varphi_2\}$ and $\varphi_2 \Rightarrow \varphi_3$ then $\{\varphi_0\}$ c $\{\varphi_3\}$.*
– (Seq) *if $\{\varphi_0\}$ c_1 $\{\varphi_1\}$ and $\{\varphi_1\}$ c_2 $\{\varphi_2\}$ then $\{\varphi_0\}$ $c_1; c_2$ $\{\varphi_2\}$.*
– (Conj) *if $\{\varphi_0\}$ c $\{\varphi_1\}$ and $\{\varphi_0\}$ c $\{\varphi_2\}$, then $\{\varphi_0\}$ c $\{\varphi_1 \wedge \varphi_2\}$.*

We omit the proofs of these classical rules. The soundness of the Hoare logic follows by induction from the soundness of each rule.

Proposition 2 *The Hoare triples given in Section 4.3 are valid.*

Example 3 We illustrate our proposition with Bellare & Rogaway's generic construction [7], which can be written shortly as $f(r)||(\mathrm{in}_e \oplus G(r))||H(\mathrm{in}_e||r)$. Where we note $\mathsf{Var} = \{\mathrm{in}_e, \mathrm{out}_e, x_\sigma, r, a, g, b, s, c\}$.

```
var $r; a; g; b; s; c;$
$true$
1. $r \overset{r}{\leftarrow} \{0,1\}^{n_0}$                using $(R1)$, $(R2)$, and $(R2)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
2. $a := f(r)$                using $(P1)$, $(O1)$, $(G3)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{WS}(r; \mathsf{Var} \setminus \{r\}) \wedge$
$\mathsf{H}(G, r) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
3. $g := G(r)$                using $(H7)$, $(H1)$, $(H4)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{Indis}(\nu g; \mathsf{Var} \setminus \{r\}) \wedge$
$\mathsf{WS}(r; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
4. $b := \mathrm{in}_e \oplus g$                using $(X2)$, $(X1)$, $(X3)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{Indis}(\nu b; \mathsf{Var} \setminus \{g, r\}) \wedge$
$\mathsf{WS}(r; \mathsf{Var} \setminus \{r\}) \wedge \mathsf{H}(H, \mathrm{in}_e||r)$
5. $s := \mathrm{in}_e||r$                using $(G1)$, $(G1)$, $(C1)$, and $(G3)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \setminus \{r, s\}) \wedge$
$\mathsf{Indis}(\nu b; \mathsf{Var} \setminus \{g, r, s\}) \wedge$
$\mathsf{WS}(s; \mathsf{Var} \setminus \{r, s\}) \wedge \mathsf{H}(H, s)$
```

6. $c := H(s)$ using $(H7)$, $(H7)$, and $(H1)$
$\mathsf{Indis}(\nu a; \mathsf{Var} \smallsetminus \{r, s\}) \wedge$
$\mathsf{Indis}(\nu b; \mathsf{Var} \smallsetminus \{r, g, s\}) \wedge$
$\mathsf{Indis}(\nu c; \mathsf{Var} \smallsetminus \{r, s\})$
7. $\mathrm{out}_e := a||b||c$ using $(C2)$ twice
$\mathsf{Indis}(\nu \mathrm{out}_e; \mathsf{Var} \smallsetminus \{a, b, c, r, g, s\})$

4.4 Extensions of the Logic

In this section, we show how our Hoare logic, and hence our verification procedure, can be adapted to deal with injective partially trapdoor one-way functions. This extension is motivated by Pointcheval's construction in [19].

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 $X \sim_{V_1;V_2} [u \overset{r}{\leftarrow} \mathcal{U}; (S, \mathbf{H}) \overset{r}{\leftarrow} X : (S\{x \mapsto f(u)\}, \mathbf{H})]$.

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. Now, let out_e, the output of the encryption oracle, have the form $a_1|| \cdots ||a_n$ with $a_i = f_i(x_i)$, where f_i are arbitrary functions. Then, we can prove the following:

Proposition 3 *Let GE be a generic encryption scheme of the form* $(\mathbb{F}, \mathcal{E}(in_e, out_e) : \mathsf{c})$, *and let f_i be any functions. Let assume that out_e, the output of the encryption oracle, has the form $a_1|| \cdots ||a_n$ with $a_i = f_i(x_i)$.*

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

The proof of this proposition follows from the transitivity of the relation $\sim_{V_1;V_2}$. 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:

(P1') $\{\mathsf{Indis}(\nu y; V_1; V_2 \cup \{y\})\}\ x := f(y)\ \{\mathsf{Indis}_f(\nu x; V_1 \cup \{x\}; V_2)\}$ if $y \notin V_1 \cup V_2$.

Many of rules that hold for *Indis* could be generalized to Indis_f. For simplicity we consider only the rules that are needed in the examples. Clearly all preservation rules can be generalized for Indis_f. Concretely, we have the following lemma, and the proof is similar as for the case Indis.

Lemma 17 (Generalization to Indis_f) *Let $X, X' \in \textsc{Dist}(\Gamma, \mathbf{H}, \mathbb{F})$ be arbitrary distributions, let V_1 and V_2 be arbitrary sets of variables, and let x, y, z, t be arbitrary variables. Then, the following assertions hold.*

1. *If $X \sim_{V_1;V_2} X'$ then $X \models \mathsf{Indis}_f(\nu x; V_1; V_2) \iff X' \models \mathsf{Indis}_f(\nu x; V_1; V_2)$.*
2. *For any expression e constructible from $(V_1; V_2)$ such that $z \notin \{x\} \cup \mathsf{Var}(e)$, if $X \models \mathsf{Indis}_f(\nu z; V_1; V_2)$ then $[\![x := e]\!](X) \models \mathsf{Indis}_f(\nu z; V_1 \cup \{x\}; V_2)$.*
3. *If $z \neq x$, and c is $x \overset{r}{\leftarrow} \mathcal{U}$ or $x := e'$ with $e' \in \{t||y, t \oplus y, f(y), H(y), t \oplus H(y)\}$, then*
   $(G1)^f$ $\{\mathsf{Indis}_f(\nu z; V_1; V_2)\}\ \mathsf{c}\ \{\mathsf{Indis}_f(\nu z; V_1; V_2)\}$,
   provided that $x \notin V_1 \cup V_2$ or e' is constructible from $(V_1 \smallsetminus \{z\}; V_2 \smallsetminus \{z\})$.

4. $(\text{R3})^f$ $\{\mathit{Indis}_f(\nu y;\, V_1;\, V_2)\}\ x \overset{r}{\leftarrow} \mathcal{U}\ \{\mathit{Indis}_f(\nu y;\, V_1 \cup \{x\};\, V_2)\}$, *provided* $x \neq y$.
5. $(\text{H7})^f$ $\{\mathit{Indis}_f(\nu z;\, V_1 \cup \{z\};\, V_2) \wedge \mathit{WS}(y;\, V_1 \cup \{z\};\, V_2) \wedge \mathit{H}(H, y)\}\ x := H(y)$ $\{\mathit{Indis}_f(\nu z;\, V_1 \cup \{z, x\};\, V_2)\}$, *provided that* $x \neq y$ *and* $z \neq x$.
6. $(\text{X2})^f$ $\{\mathit{Indis}_f(\nu t;\, V_1 \cup \{y, z\};\, V_2)\} x := y \oplus z \{\mathit{Indis}_f(\nu t;\, V_1 \cup \{x, y, z\};\, V_2)\}$, *provided that* $t \neq x, y, z$.

The reader should notice that some rules that hold for Indis can not be generalized to Indis_f. It is the case for (P1), (X1), (C2), etc.

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 $(x, y) \mapsto f(x||y)$:

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

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 [19], which can be written shortly as $f(r||H(\text{in}_e||s))||(\text{in}_e||s) \oplus G(r)$. We note $\mathsf{Var} = \{\text{in}_e, \text{out}_e, x_\sigma, r, s, w, h, a, b\}$.

```
**var** $r; s; w; h; a; b$;
*true*
1. $r \overset{r}{\leftarrow} \{0,1\}^{n_0}$                                using $(R1)$ and $(R2)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{H}(G, r)$
2. $s \overset{r}{\leftarrow} \{0,1\}^{n_0}$                                using $(R3)$, $(R1)$, $(G3)$ and $(R2)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{Indis}(\nu s; \mathsf{Var}) \wedge$
    $\mathsf{H}(G, r) \wedge \mathsf{H}(H, \text{in}_e||s)$
3. $w := \text{in}_e||s$                                              using $(C3)$, $(C1)$, $(G3)$, and $(G3)$
$\mathsf{Indis}(\nu r; \mathsf{Var}) \wedge \mathsf{WS}(w; \mathsf{Var} \setminus \{s, w\}) \wedge$
    $\mathsf{H}(G, r) \wedge \mathsf{H}(H, w)$
4. $h := H(w)$                                                    using $(H7)$, $(H1)$, and $(G3)$
$\mathsf{Indis}(\nu r; \mathsf{Var} \setminus \{w, s\}) \wedge$
    $\mathsf{Indis}(\nu h; \mathsf{Var} \setminus \{w, s\}) \wedge \mathsf{H}(G, r)$
5. $a := f(r||h)$                                                 using the new rule $(PO1)$ and $(G3)$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} \setminus \{r, s, w, h\}) \wedge$
    $\mathsf{WS}(r; \mathsf{Var} \setminus \{r, s, w, h\}) \wedge \mathsf{H}(G, r)$
6. $b := w \oplus G(r)$                                           using $(G1)^f$ and $(H1)$
$\mathsf{Indis}_f(\nu a; \mathsf{Var} \setminus \{r, s, w, h\}) \wedge$
    $\mathsf{Indis}(\nu b; \mathsf{Var} \setminus \{r, s, w, h\})$
By the Consequence rule using (†)
$\mathsf{Indis}_f(\nu a; a, b, \mathsf{Var} \setminus \{r, s, w, h, \text{out}_e\}) \wedge$
    $\mathsf{Indis}(\nu b; a, b, \mathsf{Var} \setminus \{r, s, w, h, \text{out}_e\})$
```

7. $\mathrm{out}_e := a||b$
$\mathsf{Indis}_f(\nu a; a, b, \mathsf{Var} \setminus \{r, s, w, h, \mathrm{out}_e\}) \wedge$ using $(G1)^f$ and $(G1)$
$\mathsf{Indis}(\nu b; a, b, \mathsf{Var} \setminus \{r, s, w, h, \mathrm{out}_e\})$

($\dagger$) $\mathsf{Indis}_f(\nu a; a, V) \wedge \mathsf{Indis}(\nu b; a, b, V)$ implies $\mathsf{Indis}_f(\nu a; a, b, V \setminus \{x\})$

5 Automation

We can now fully automate our verification procedure of IND-CPA for the encryption schemes. The idea is, for a given program, to compute invariants backwards, starting with the invariant $\mathsf{Indis}(\nu \mathrm{out}_e; \mathrm{out}_e, \mathrm{in}_e, x_\sigma)$ at the end of the program.

As several rules can lead to the same postcondition, we in fact compute a set of sufficient conditions at all points of the program: for each set (of postconditions) $\{\phi_1, \dots, \phi_n\}$ and each instruction c, we can compute a set of assertions (preconditions) $\{\phi'_1, \dots, \phi'_m\}$ such that, for each $i = 1, \dots, n$, there exists a subset $J \subseteq [1, \dots, m]$ such that $\{\bigwedge_{j \in J} \phi'_j\}c\{\phi_i\}$ can be derived using the rules given Section 4.3,

The set $\{\phi'_1, \dots, \phi'_m\}$ is computed by applying two steps:

1. First, a set of assertions are computed by matching the command and assertion ϕ_i with postconditions of the Hoare axioms. This allows to compute for each assertion ϕ_i a set of preconditions $\textsc{Pre}(\phi_i)$.
2. Next, the consequence rule is applied using Lemma 2, i.e., the assertions in $\textsc{Pre}(\phi_i)$ are replaced by stronger assertions, leading to the assertions in $\{\phi'_1, \dots, \phi'_m\}$.

Since the commands we consider do not include loops, our verification procedure always terminates. However, this verification is potentially exponential in the number of instructions in the encryption command as each postcondition may potentially have several preconditions. This does not seem to be a problem in practice. Indeed, checking Bellare & Rogaway generic construction, for instance, is instantaneous. We implemented that procedure as an Objective Caml program, taking as input a representation of the encryption program.

6 Conclusion

In this paper we proposed an automatic method to prove IND-CPA security of generic encryption schemes in the random oracle model. Then, IND-CCA can be proved using a general method for proving plaintext awareness as described in [11]. 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.

Acknowledgements We thank the anonymous and non-anonymous reviewers for their valuable comments that greatly helped improving the paper.

References

1. Barthe, G., Cederquist, J., Tarento, S.: A machine-checked formalization of the generic model and the random oracle model. In: Basin, D., Rusinowitch, M. (eds.) Proceedings of IJCAR'04, vol. 3097 of LNCS, pp. 385–399 (2004)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS, pp. 394–403 (1997)
3. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: 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, pp. 26–45, London, UK. Springer, Heidelberg (1998)
4. Barthe, G., Grégoire, B., Zanella Béguelin, S.: Formal certification of code-based cryptographic proofs. In: POPL '09: Proceedings of the 36th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 90–101. ACM, New York (2009)
5. Blanchet, B.: A computationally sound mechanized prover for security protocols. In: IEEE Symposium on Security and Privacy (S&P 2006), 21–24, pp. 140–154. IEEE Computer Society, Washington (2006)
6. Blanchet, B., Pointcheval, D.: Automated security proofs with sequences of games. In: Dwork, C. (ed.) CRYPTO, vol. 4117 of Lecture Notes in Computer Science, pp. 537–554. Springer, Heidelberg (2006)
7. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS '93: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM, New York (1993)
8. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) EUROCRYPT, vol. 950 of Lecture Notes in Computer Science, pp. 92–111. Springer, Heidelberg (1994)
9. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331. http://eprint.iacr.org/ (2004)
10. Barthe, G., Tarento, S.: A machine-checked formalization of the random oracle model. In: Filliâtre, J.-C., Paulin-Mohring, C., Werner, B. (eds.) Proceedings of TYPES'04, vol. 3839 of Lecture Notes in Computer Science, pp. 33–49. Springer, Heidelberg (2004)
11. Courant, J., Daubignard, M., Ene, C., Lafourcade, P., Lahknech, Y.: Towards automated proofs for asymmetric encryption schemes in the random oracle model. Technical report, Verimag, Verimag, Centre Équation, 38610 Gières (2009)
12. Corin, R., den Hartog, J.: A probabilistic Hoare-style logic for game-based cryptographic proofs. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP (2), vol. 4052 of Lecture Notes in Computer Science, pp. 252–263. Springer, Heidelberg (2006)
13. Damgard, I.: 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, pp. 445–456. Springer, London (1992)
14. Datta, A., Derek, A., Mitchell, J.C., Warinschi, B.: Computationally sound compositional logic for key exchange protocols. In: CSFW '06: Proceedings of the 19th IEEE Workshop on Computer Security Foundations, pp. 321–334. IEEE Computer Society, Washington (2006)
15. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **IT-22**, 644–654 (1976)
16. Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. J. Cryptol. **1**(2), 77–94 (1988)
17. Halevi, S.: A plausible approach to computer-aided cryptographic proofs. http://theory.lcs.mit.edu/~shaih/pubs.html (2005)
18. Okamoto, T., Pointcheval, D.: React: Rapid enhanced-security asymmetric cryptosystem transform. In: CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology, pp. 159–175. Springer, London(2001)
19. Pointcheval, D.: 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, pp. 129–146. Springer, London (2000)
20. Rabin, M.O.: Digitalized signatures as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Cambridge (1979)
21. Shoup, V.: OAEP reconsidered. J. Cryptol. **15**(4), 223–249 (2002)
22. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs http://eprint.iacr.org/2004/332 (2004)

23. Soldera, D., Seberry, J., Qu, C.: The analysis of Zheng–Seberry scheme. In: Batten, L.M., Seberry, J. (eds.) ACISP, vol. 2384 of Lecture Notes in Computer Science, pp. 159–168. Springer, Heidelberg (2002)
24. Tarento, S.: Machine-checked security proofs of cryptographic signature schemes. In: De Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) Computer Security–ESORICS 2005, vol. 3679 of Lecture Notes in Computer Science, pp. 140–158. Springer, Heidelberg (2005)
25. Zheng, Y., Seberry, J.: Immunizing public key cryptosystems against chosen ciphertext attacks. IEEE J. Sel. Areas Commun. **11**(5), 715–724 (1993)