Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 186 (2007) 121–139
www.elsevier.com/locate/entcs

Computational Soundness of Symbolic Analysis for Protocols Using Hash Functions

Romain Janvier and Yassine Lakhnech and Laurent Mazaré[1]

VERIMAG, 2 av. de Vignates
38610 Gieres, FRANCE

Abstract

In this paper, we consider a Dolev-Yao model with hash functions and establish its soundness with respect to the computational model. Soundness means that the absence of attacks in the Dolev-Yao model implies that the probability for an adversary to perform an attack in the computational model is negligible. Classical requirements for deterministic hash functions (e.g. one-wayness, collision freeness) are not sufficient for proving this result. Therefore we introduce new security requirements that are sufficient to prove the soundness result and that are verified by random oracles.

Keywords: Security, Cryptographic Protocols, Formal Encryption, Probabilistic Encryption, Keyed Hash Function, Dolev-Yao Model, Computational Model.

1 Introduction

Motivation.

Historically, verification of cryptographic protocols has been separated in two distinct branches. *Symbolic verification* (also called formal verification) of cryptographic protocols, originates from the work of Dolev and Yao [14]. The essential part of this approach is the perfect cryptography assumption that can be roughly summarized as follows: messages are represented as algebraic terms, fresh nonce creation is perfect, that is, nonces range over an infinite domain and freshness is absolute, the same holds for key creation. Moreover, there is no way to guess a nonce or a key and no information can be extracted from an encrypted message unless the inverse of the key used to encrypt the message is known. In this approach there is a single attacker that is modeled as an infinite process without bounds on its computational resources. Despite the strong assumptions concerning the cryptographic primitives, flaws have been found on protocols that were believed to be

[1] Email: {romain.janvier,yassine.lakhnech,laurent.mazare}@imag.fr

doi:10.1016/j.entcs.2007.01.066

secure (the most famous one has been exposed by G. Lowe in [18], some of them are listed in [10]). The good news about this approach is that a rich collection of automatic verification methods and tools have been developed [21,8,7,2].

In the *computational approach*, cryptographic primitives operate on strings of bits and their security is defined in terms of high complexity and weak probability of success (e.g. [16] for encryption) of any attacker. Protocols as well as attackers are randomized polynomial-time Turing machines. This computational approach is recognized as more realistic than the formal approach, however, its complexity makes it very difficult to design automatic verification tools.

Establishing computational soundness of symbolic verification allows to have the best of both world, i.e. automatic verification on one hand and computational correctness on the other hand.

Related work.

In the last years, several works have focussed on bridging the gap that separates these two approaches. In their seminal paper [1], Abadi and Rogaway prove that a notion of message *indistinguishability* in the symbolic model is valid in the computational model provided that the encryption scheme used to implement messages satisfies semantic security. This means that if two messages are not distinguishable in the symbolic model, then their computational implementations cannot be separated by a Turing machine in a reasonable (polynomial) time. This paper deals with passive attackers that can only eavesdrop but not alter or block messages.

Active attackers are considered in [19,12,17]. These papers prove that if the encryption scheme satisfies semantic security against chosen cipher-text attacks, then security in the symbolic model implies security in the computational model. Cortier and Warinschi prove in [12], extending the results of [19], safety of the symbolic model for protocols that use asymmetric encryption and digital signature. A similar result for protocols where secret key transmission is allowed has been formulated independently in [17]. Cortier, Kremer, Küsters and Warinschi [11] have extended these results with hash functions using the random oracle model.

Soundness of symbolic analysis has been proven in the black-box reactive simulatability (BRSIM) framework by Backes, Pfitzmann and Waidner [3]. Recently this result has been extended [4] to include hash functions in the random oracle model, it has also been proven that such an extension is not possible in the standard model. Given the strength of the BRSIM model, this impossibility result does not automatically carry over to the Dolev-Yao model used in [11].

A soundness result for hash functions that does not rely on the random oracle model is given in [15]. However only passive adversaries are considered and hash functions have to be probabilistic [9].

Contributions.

We establish our result in a symbolic model based on [19]. This result does not directly rely on the random oracle model but instead we introduce new security requirements for keyed hash functions that are easily achievable in the random

oracle model. While classical requirements for hash functions are collision freeness and pre-image resistance, we keep collision freeness but ask for an unforgeability requirement instead of pre-image resistance. Since hash functions are deterministic, it is not reasonable to require that an adversary cannot produce the hash of a message of his choice. Instead we ask that it is hard for an adversary to compute the hash of a message that contains a randomly sampled nonce. This nonce is not given to the adversary but he can request hashes of messages containing this nonce.

Using these new requirements for hash functions and semantic security against chosen cipher-text attacks for asymmetric encryption, we prove computational soundness of the symbolic model for protocols that use both asymmetric cryptography and hash functions.

Paper organization.

The next section recalls the necessary preliminaries for using the computational model. In section 3, protocols are introduced as well as their symbolic and computational semantics. In section 4, we introduce our requirements for cryptographic primitives. Section 5 contains our main result: computational soundness for protocols using hash functions. We conclude the paper with a short discussion.

2 Preliminaries

2.1 Cryptographic schemes

An *asymmetric encryption scheme* $\mathcal{AE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is defined by three algorithms. The key generation algorithm $\mathcal{KG}$ is a randomized function which given a security parameter η outputs a pair of keys (pk, sk), where pk is a public key and sk the associated secret key. The encryption algorithm $\mathcal{E}$ is also a randomized function which given a message and a public key outputs the encryption of the message by the public key. Finally the decryption algorithm $\mathcal{D}$ takes as input a secret key and a cipher-text and outputs the corresponding plain-text, i.e., $\mathcal{D}(\mathcal{E}(m, pk), sk) = m$ provided that (pk, sk) has been generated by the key generation algorithm. The execution time of the three algorithms is assumed polynomially bounded by η.

A keyed hash function $\mathcal{H}$ [6] is a deterministic function that takes as input a key k and a bit-string bs and outputs a bit-string of the same length as k. Hash keys can be any bit-string of length η.

In this paper, we only consider *cryptographic library* $\mathcal{CL}$ that contain an asymmetric encryption scheme and a keyed hash function, we also only consider encryption schemes that are length preserving meaning that the length of a cipher-text only depends on the length of the plain-text and the security parameter η.

2.2 Randomized Turing Machines with Oracle

Adversaries in the computational world are probabilistic polynomial Turing machines (PPTM) that may access some oracles. To define an adversary $\mathcal{A}$ that can use an oracle $\mathcal{O}$, we write:

Adversary $\mathcal{A}/\mathcal{O}$:
Code of $\mathcal{A}$ that can access $\mathcal{O}$
e.g. $bs \leftarrow \mathcal{O}(x)$

The execution of $\mathcal{A}$ with an implementation of the oracle done by PPTM F is denoted by $\mathcal{A}/\lambda s.F(s)$.

A function $g : \mathbb{R} \to \mathbb{R}$ is *negligible*, if for all $c > 0$ there exists N_c such that $g(x) < x^{-c}$, for all $x > N_c$.

3 Protocols Syntax and Semantics

In this section, we consider protocols that allow parties to exchange messages built from identities and randomly generated numbers using public key encryption and a hash function. We assume that the same hash key is used by every participant using the protocol hence we do not represent this key in the syntax of the protocol. Three types of instructions can be performed during protocol execution: receiving a message, sending a message or testing a hash. To describe a specific instruction, we use terms in the free algebra with the following sorts:

- The sort Nonce for nonces.
- The sort Ident for principal identities.
- The sorts Pubkey and Privkey for, respectively, public keys and private keys.
- The sort Term that includes all other sorts.

The signature includes the following function symbols with their corresponding arities:

- $\mathsf{enc} : \mathsf{Term} \times \mathsf{Pubkey} \to \mathsf{Term}$. We use $\{t\}_k$ as a shorthand for $\mathsf{enc}(t, k)$.
- $h : \mathsf{Term} \to \mathsf{Term}$.
- $\mathsf{pair} : \mathsf{Term} \times \mathsf{Term} \to \mathsf{Term}$ for pairing, where $\langle t_1, t_2 \rangle$ is used for $\mathsf{pair}(t_1, t_2)$.

We assume a one-to-one onto mapping from Pubkey to Privkey associating each key k to its inverse k^{-1}. This mapping is extended from Privkey to Pubkey in such a way that $(k^{-1})^{-1} = k$.

Protocols are specified using terms in this algebra and typed variables. That is, we allow disjoint sets of variables that range over the different sorts. A term is called *atomic*, if it is a key, a nonce or a variable. Grounds terms, i.e. variable free terms, are called *messages*.

3.1 Protocols

Usually, a protocol is given by a finite sequence of principal identities with associated roles. A role is specified by a list of instructions that can be executed by the principal running this role. Thus, an n-party protocol is a mapping $\Pi : [0, \cdots, n-1] \to \mathsf{Ident} \times \mathit{Role}$, where $\mathit{Role} = \mathit{inst}^*$ and the set inst of instructions is defined as follows:

$$\mathit{inst} ::= \mathsf{Rec}(t) \mid \mathsf{Snd}(t) \mid [h(t) = x]$$

where t is a term and x a variable. Instruction $\mathsf{Rec}(t)$ denotes the reception of a message (and its pattern matching using prototype t, which should not contain hash), $\mathsf{Snd}(t)$ denotes the emission of t and finally, $[h(t) = x]$ checks that the value of x is the hash of the value of t. A role R is a finite list of instructions, the set $atoms(R)$ contains every atom that appears in R or whose inverse (for keys) appears in R.

As semantics for parallel composition of roles, we take the usual interleaving semantics. We only consider a bounded number of sessions. Now, it is easy to see that as finitely many roles are considered and as each role consists of a finite number of instructions executed sequentially, the number of possible interleavings for the parallel execution of the different roles is bounded[2]. Therefore, to simplify the presentation and without loss of generality (except that we only consider bounded protocols), we consider protocols that contain only a single role. In order to properly define a protocol using a single role, we also have to specify which atoms used in the role are initially known by the adversary. We assume that the dishonest participants are chosen before the beginning of the execution of the protocol. Therefore, a protocol Π is a pair (R, IK) where R is a role and IK is a subset of $atoms(R)$ that represents the initial knowledge of the adversary. The set IK contains all the identities and public keys for asymmetric encryption from R. It also contains private keys related to dishonest participants.

Let us illustrate our definition and our syntax on the classical Needham-Schroeder protocol [18]. The description of the protocol in the BAN-notation is as follows:

$$A \to B : \{A, N_A\}_{pk_B} \quad B \to A : \{N_A, N_B\}_{pk_A} \quad A \to B : \{N_B\}_{pk_B}$$

Let us consider two sessions: one between A and B and one between A and the adversary I. This protocol involves two roles of which we consider the interleavings, the role of B (communicating with A) and the role of A (communicating with I):

$$Role_B^{AB} : \mathsf{Rec}(\{A, y\}_{pk_B}) \,.\, \mathsf{Snd}(\{y, N_B\}_{pk_A})$$
$$Role_A^{AI} : \mathsf{Snd}(\{A, N'_A\}_{pk_I}) \,.\, \mathsf{Rec}(\{N'_A, x\}_{pk_A}) \,.\, \mathsf{Snd}(\{x\}_{pk_I})$$

The protocol consists in the parallel composition of the two roles $Role_B^{AB}$ and $Role_A^{AI}$. Let us consider the single-role protocol $\Pi = (R, \mathsf{IK})$ which corresponds to the man-in-the-middle attack of [18]. Role R is:

$\mathsf{Snd}(\{A, N'_A\}_{pk_I})$	A sends its first message in session A, I
$\mathsf{Rec}(\{A, y\}_{pk_B}) \,.\, \mathsf{Snd}(\{y, N_B\}_{pk_A})$	B receives its first message in session A, B and answers
$\mathsf{Rec}(\{N'_A, x\}_{pk_A}) \,.\, \mathsf{Snd}(\{x\}_{pk_I})$	A receives its answer in session A, I

[2] Although this number is exponential in the number of sessions, numerous automatic tools are designed to verify protocols for an unbounded number of sessions. Security results given by such tools ensure security for a bounded number of sessions.

The intruder's initial knowledge IK contains identities A, B and I as well as public keys pk_A, pk_B and pk_I and ecret key $sk_I = pk_I^{-1}$ as participant I is dishonest.

3.2 Executable Protocols

We now define *executable* protocols.

Definition 3.1 A protocol $\Pi = (R, \mathsf{IK})$ is *executable*, if:

(i) Any variable that appears in a send action $\mathsf{Snd}(t)$ has to occur before in a reception action $\mathsf{Rec}(t')$. This ensures that t evaluates to a message (a groud term) when $\mathsf{Snd}(t)$ is executed.

(ii) Any variable that appears in an action $[h(t) = x]$ has to occur before in a reception action $\mathsf{Rec}(t')$.

Moreover we assume that secret keys do not appear in sent messages.

We consider two different protocol semantics: one in the symbolic model and one in the computational model. Both define the behavior of the protocol confronted to an adversary. This adversary has total control of the network. Moreover the adversary impersonates the dishonest participants. In the symbolic case, the protocol and the adversary exchange symbolic messages. Deductions that can be made by the adversary are defined by a deduction relation. In the computational case, they exchange strings of bits. The adversary is a polynomial random Turing machine and can therefore perform any operation it wants.

3.3 Symbolic Semantics

To define symbolic semantics of protocols, we introduce the *entailment* relation $E \vdash m$, where E is a finite set of messages and m a message. Intuitively, $E \vdash m$ means that m can be deduced from the set of messages E [14].

Henceforth, let Π be a protocol given by (R, IK). The relation $E \vdash m$ is defined as the least binary relation verifying:

(i) If an atom a appears in IK or does not occur in $atoms(R)$, then $E \vdash a$.

(ii) If $m \in E$, then $E \vdash m$.

(iii) If $E \vdash m$ and $E \vdash n$, then $E \vdash \langle m, n \rangle$.

(iv) If $E \vdash \langle m, n \rangle$, then $E \vdash m$ and $E \vdash n$.

(v) If $E \vdash m$ and $E \vdash k$, then $E \vdash \{m\}_k$.

(vi) If $E \vdash \{m\}_k$ and $E \vdash k^{-1}$, then $E \vdash m$.

(vii) If $E \vdash m$, then $E \vdash h(m)$.

Remark 3.2 The first rule introduces the initial knowledge of the adversary. The adversary can deduce any atom in IK and also can generate fresh atoms (i.e. atoms that do not occur in $atoms(R)$).

A *symbolic trace* is a list of emissions and receptions of messages represented re-

spectively by $\mathsf{Snd}(m)$ and $\mathsf{Rec}(m)$ where m is a message. Clearly, there are some sequences of messages which are not feasible.

Definition 3.3 [Valid substitutions and valid traces] Let $\Pi = (R, \mathsf{IK})$ be a protocol. Let σ be a ground substitution and $R' = inst_1, \cdots, inst_n$ a prefix of R. For any $i = 1, \cdots, n$ such that $inst_i = \mathsf{Rec}(t)$, let $T_i = \{t' \mid \exists j < i \cdot inst_j = \mathsf{Snd}(t')\}$. Then, σ is a *valid substitution* for Π if the following conditions are satisfied, for $i = 1, \cdots, n$:

(i) If $inst_i = \mathsf{Rec}(t)$ then $t\sigma$ is deducible from messages sent before, that is, $T_i\sigma \vdash t\sigma$.

(ii) If $inst_i = [h(t) = x]$ then $x\sigma = h(t\sigma)$.

In case σ is a valid substitution for Π, we call the sequence of messages $m_1, \cdots, m_k$ obtained from $R'\sigma$ by deleting all verification tests a *valid trace* of Π. Moreover, the set of all valid traces of Π is denoted by $\mathsf{Traces}(\Pi)$. □

Example 3.4 Let us consider once more the Needham-Schroeder protocol introduced previously. The man-in-the-middle attack presented in [18] uses only the following prefix of the role:

$\mathsf{Snd}(\{A, N'_A\}_{pk_I})$	A sends its first message in session A, I
$\mathsf{Rec}(\{A, y\}_{pk_B})$. $\mathsf{Snd}(\{y, N_B\}_{pk_A})$	B receives its first message and answers
$\mathsf{Rec}(\{N'_A, x\}_{pk_A})$. $\mathsf{Snd}(\{x\}_{pk_I})$	A receives its answer in session A, I

Then the attack corresponds to valid substitution σ defined by:

$$\sigma = \left(x \to N_B, y \to N'_A \right)$$

The corresponding trace is given hereafter. It is easy to check that this trace is valid.

$$\mathsf{Snd}(\{A, N_A\}_{pk_I}).\mathsf{Rec}(\{A, N_A\}_{pk_B}).\mathsf{Snd}(\{N_A, N_B\}_{pk_A}).$$
$$\mathsf{Rec}(\{N_A, N_B\}_{pk_A}).\mathsf{Snd}(\{N_B\}_{pk_I}).$$

□

3.4 Computational Semantics

We want to let the adversary have arbitrary control over the network, as in the symbolic model, and hence we eliminate the network. Moreover, the adversary drives the computation by sending messages to the other players and receiving messages from them. In the *computational model*, the messages that are exchanged are bit-strings (and depend on the security parameter η). However, we assume types corresponding to the different ranges of the cryptographic primitives: M_{ag}, M_n, M_{pk}, M_{sk}, M_a, M_h and M_p for principal identities, nonces, public keys, secret keys, cipher-texts, hashes and pairs, respectively. Moreover, we assume a polynomial time type retrieval function $\mathsf{type} : \mathsf{M} \to \mathsf{Types}$, where Types is the set of all types. We also assume that equality of bit-strings holds only when types coincide, i.e. $bs = bs'$ implies $\mathsf{type}(bs) = \mathsf{type}(bs')$. Concerning pairs, we assume that there are two deterministic polynomial algorithms pr_1 and pr_2 such that $pr_i(bs_1 \cdot bs_2) = bs_i$.

Now, a *computational trace* is a sequence of emissions and receptions of bit-strings denoted respectively by $\mathsf{Snd}(bs)$ and $\mathsf{Rec}(bs)$, where $\mathsf{Snd}(bs)$ is a message sent by a principal and received by the adversary, while $\mathsf{Rec}(bs)$ is a message received by a principal and sent by the adversary.

Throughout this section, let $\Pi = (\langle inst_1, \cdots, inst_\ell \rangle, \mathsf{IK})$ be a fixed arbitrary protocol. Moreover, let $\mathcal{CL}$ be a cryptographic library composed by an asymmetric encryption scheme $\mathcal{AE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ and a keyed hash function $\mathcal{H}$.

The computational semantics is defined by the random algorithm *Exec* given in Figure 3.1.Some explanatory remarks are of order.

- *Exec* is given as input an adversary $\mathcal{A}$, a protocol $\Pi = (\langle inst_1, \cdots, inst_\ell \rangle, \mathsf{IK})$ and an initial mapping θ from bit-strings to nonces, keys and identities from $atoms(\langle inst_1, \cdots, inst_\ell \rangle)$. The value of θ is computed using the key generation algorithm of $\mathcal{CL}$ and an algorithm for generating values for nonces. Henceforth, we write $\theta \leftarrow Init(\Pi)$ to denote the sampling of the initial value of θ. We assume that θ also associates a hash key to k_H.

  Let us call $\mathsf{IK}(\theta)$ the restriction of θ to atoms in IK and to k_H.
- The output of *Exec* is a computational trace recording the interaction between the adversary and the principals running the protocol, the final substitution θ and the final memory *mem* of the adversary. Our main theorem in Section 5 states that the produced computational trace corresponds to a *valid* symbolic trace with overwhelming probability. The other outputs, θ and *mem*, are used when considering properties over protocols.
- *Exec* modifies θ, it also uses a list *trace* that contains the computational trace eventually output by *Exec*. *Exec* uses two auxiliary functions:

  (i) *concr* that takes as arguments a term t and a computational substitution θ and returns a computational value, i.e. a bit-string for t.

  (ii) *parse* that takes as arguments a bit-string bs, a term t and a computational substitution θ and returns an updated version of θ (obtained by matching bs with t).

  Due to space contraints, we do not present the algorithms of *concr* and *parse* which are quite obvious. The two algorithms use the cryptographic primitives of $\mathcal{CL}$.
- Some errors may be raised during execution. When an error is raised, execution is aborted and control flow is transferred to the final **return** statement in *Exec*. Errors may be raised by *Exec* when a hash verification test fails or by the *parse* function. This function may raise errors in the following situations. The value of x in θ is needed but $\theta(x)$ is undefined or does not have the expected value; in this case, the protocol cannot be executed. An error is also raised when the type of a bit-string is not the expected one or when a call to the decryption algorithm or to one of the projection algorithms fails.

Figure 3.1 Exec Algorithm

Exec$(\mathcal{A}, \mathsf{IK}, \langle inst_1 ... inst_\ell \rangle, \theta)$:
$trace := []$; $mem := \mathsf{IK}(\theta)$; Initialisation
for i **from** 1 **to** ℓ
  match $inst_i$ **with**
    $[[h(t_1) = t_2]]$ Hash verification
      if $\mathcal{H}(concr(k_H, \theta), concr(t_1, \theta)) \neq concr(t_2, \theta)$
      then raise test-failed
    $[\mathsf{Rec}(t)]$ Message reception
      $(bs, mem) := \mathcal{A}(mem)$
      $\theta := parse(bs, t, \theta)$
      $trace := trace :: \mathsf{Rec}(bs)$
    $[\mathsf{Snd}(t)]$ Message emission
      $bs := concr(t, \theta)$
      $trace := trace :: \mathsf{Snd}(bs)$
      $mem := \mathcal{A}(bs, mem)$
  endmatch
endfor
return $(trace, \theta, mem)$

3.5 Protocol Properties

For both the symbolic and the computational model, a trace property is given by a set of traces; a set of symbolic traces in the first case and a set of computational traces in the second.

3.5.1 Symbolic Properties

In the case of the symbolic semantics, a protocol Π satisfies a trace property ψ, if all valid traces of Π are in ψ.

Definition 3.5 Let Π be a protocol and let ψ be a set of symbolic traces. Then, Π satisfies ψ, denoted by $\Pi \models_f \psi$, if $\mathsf{Traces}(\Pi) \subseteq \psi$.

Authentication properties, such as aliveness, weak agreement, non-injective agreement, are typical trace properties. Another trace property extensively studied is non-deductibility.

Definition 3.6 Let m be a symbolic message. Let Π be a protocol. We denote by $\mathsf{Secret}(m)$ the set of symbolic traces $m_1, \cdots, m_n$ such that $\{m_1, \cdots, m_n\} \not\vdash m$ and we say that Π satisfies $\mathsf{Secret}(m)$, if $\Pi \models_f \mathsf{Secret}(m)$.

3.5.2 Computational Properties

A computational trace property is a set of computational traces that is a set of sequences of bit-strings. A protocol Π satisfies a trace property, if for any adversary

the probability to obtain a trace that does not satisfy the property is negligible. More formally, we have the following:

Definition 3.7 Let $\Pi = (R, \mathsf{IK})$ be a protocol and let ϕ be a set of computational traces. Then, Π satisfies ϕ, denoted by $\Pi \models_c \phi$, if for every adversary $\mathcal{A}$,

$$Pr[\theta \leftarrow Init(\Pi)\ ;\ Exec(\mathcal{A}, \mathsf{IK}, R, \theta) \notin \phi]$$

is negligible as a function of η. Notice that the probability is taken over the coin tosses used for computing θ, the coin tosses of $\mathcal{A}$ and the coin tosses of the algorithms in $\mathcal{CL}$.

We now define another property corresponding to the strong secrecy of nonces SecNonce [12]. This property states that it is hard for an adversary to distinguish the nonce that is used in the execution of the protocol from a random nonce. In the $\mathsf{SecNonce}^b$ experiment, after execution of the protocol, the adversary is given two nonce values bs_0 and bs_1 and has to decide which was used in the execution.

$$\begin{aligned}
&\mathsf{SecNonce}^b(\mathcal{A}, \mathsf{IK}, R, \theta, N) : \\
&\qquad bs_0 \xleftarrow{R} \{0,1\}^\eta \\
&\qquad bs_1 \xleftarrow{R} \{0,1\}^\eta \\
&\qquad \theta \leftarrow \theta[bs_b/N] \\
&\qquad (t_c, \theta', mem) \xleftarrow{R} Exec(\mathcal{A}, \mathsf{IK}, R, \theta) \\
&\qquad d \xleftarrow{R} \mathcal{A}(t_c, bs_0, bs_1, mem)
\end{aligned}$$

Definition 3.8 Let Π be a protocol (R, IK) and N a nonce of Π. For any adversary $\mathcal{A}$, we define the $\mathsf{SecNonce}$-advantage of A as follows:

$$\begin{aligned}
Adv^{SecNonce,N}_{\mathcal{A},\mathsf{IK},R}(\eta) = 2Pr[\ &\theta \xleftarrow{R} Init(\Pi)\ ;\ b \xleftarrow{R} \{0,1\}\ ; \\
&d \xleftarrow{R} \mathsf{SecNonce}^b(\mathcal{A}, \mathsf{IK}, R, \theta, N) : d = b] - 1
\end{aligned}$$

A protocol Π satisfies $\mathsf{SecNonce}(N)$, if $Adv^{SecNonce,N}_{\mathcal{A},\mathsf{IK},R}(\eta)$ is negligible, for any adversary $\mathcal{A}$.

4 Security Definitions for Cryptographic Primitives

Security definitions are introduced using *security criteria* and associated security games (experiments). A security criterion defines a game involving an adversary. The experiment proceeds as follows. First some parameters θ are randomly generated. The adversary is executed and can use an oracle F which depends on θ. At the end, the adversary has to answer a string of bits which is verified by an

algorithm V which also uses θ (e.g. θ includes a bit b and the adversary has to output the value of b).

Definition 4.1 A *security criterion* γ is a triple $(\Theta; F; V)$ where

- Θ is a PPTM that given a security value η randomly generates some challenge θ (for example, a bit b and a pair of keys (pk, sk)).
- F is a PPTM that takes as arguments a string of bits s and a challenge θ and outputs a new string of bits. F represents the oracles that an adversary can call to solve its challenge.
- V is a PPTM that takes as arguments a string of bits s and a challenge θ and outputs either true or false. It represents the verification made on the result computed by the adversary. The answer true (resp. false) means that the adversary solved (resp. did not solve) the challenge.

Remark 4.2 Note that Θ can generate an arbitrary number of parameters and F can represent an arbitrary number of oracles. Thus, it is possible to define criteria with multiple Θ and F.

The advantage of a PPTM $\mathcal{A}$ against γ is

$$\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) = 2\big(Pr[\mathbf{G}^{\gamma}_{\mathcal{A}}(\eta) = true] - PrRand^{\gamma}\big)$$

where $\mathbf{G}$ is the Turing machine defined by:

Game $\mathbf{G}^{\gamma}_{\mathcal{A}}(\eta)$:
$\theta \leftarrow \Theta(\eta)$
$d \leftarrow \mathcal{A}(\eta)/\lambda s.F(s,\theta)$
return $V(d,\theta)$

and $PrRand^{\gamma}$ is the best probability to solve the challenge that an adversary can have without using oracle F. Formally, $PrRand^{\gamma}$ is the maximum of $Pr[\mathbf{G}^{\gamma'}_{\mathcal{A}}(\eta) = true]$ where $\mathcal{A}$ ranges over any possible PPTM and γ' is the criterion $(\Theta; \epsilon; V)$.

We also consider criteria with multiple verifiers denoted by $\gamma = (\Theta; F; V_1, ..., V_n)$. Then the advantage of an adversary $\mathcal{A}$ is defined by:

$$\mathbf{Adv}^{\gamma}_{\mathcal{A}}(\eta) = \max_{1 \le i \le n} \big(\mathbf{Adv}^{(\Theta;F;V_i)}_{\mathcal{A}}(\eta)\big)$$

A criterion γ is said *safe* iff the advantage of any adversary against γ is negligible.

4.1 Asymmetric Encryption

For the asymmetric encryption scheme we use the N-IND-CCA criterion which is equivalent to the classical IND-CCA criterion [5].

Definition 4.3 [N-IND-CCA]
Let $\mathcal{AE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ be an asymmetric encryption scheme. Then, the N-IND-CCA criterion associated with $\mathcal{AE}$, denoted by γ_N, is given by $(\Theta; F; V)$, where:

(i) Θ generates N pairs of keys (pk_1, sk_1) to (pk_N, sk_N) using $\mathcal{KG}$ and a bit b at random;

(ii) V verifies that the adversary outputs the right value for bit b; and

(iii) F gives access to three oracles for each i between 1 and N:
   (a) a left-right encryption oracle that takes as arguments two bit-strings of equal length (bs_0, bs_1) and outputs the encryption of bs_b with pk_i;
   (b) a decryption oracle that decodes any message not produced by the former encryption oracle;
   (c) an oracle that simply makes the public key available.

An asymmetric encryption scheme $\mathcal{AE}$ is said N-IND-CCA iff γ_N is safe.

4.2 Hash Functions

Hash functions are typically used to reduce the size of a message: all the possible outputs of the hash functions have the same size. Moreover, as hash algorithm are deterministic, it is possible to test that a hash result and a bit-string correspond. However hashed messages can be used to make commitments on a bit-string, thus it should be difficult to find two message that have the same hash, this is called a *collision*. It should also be difficult from a hashed message to recover the underlying bit-string. Previous works in provable cryptography have introduced numerous security requirements for hash functions [20]. Among these requirements, the three most commonly used criteria are:

- **Preimage resistance (Pre):** given a hash Y and the key K used, it should be impossible, except with non-negligible probability, to find a bit-string M' such that $\mathcal{H}(K, M') = Y$. That is, the following probability must be negligible in η for any adversary $\mathcal{A}$ and a given polynomial p:

$$Pr[\, K \leftarrow \{0,1\}^\eta;\ M \leftarrow \{0,1\}^{p(\eta)};\\ Y \leftarrow \mathcal{H}(K, M);\ M' \leftarrow \mathcal{A}(K, Y) :\ \mathcal{H}(K, M') = Y]$$

- **Second-preimage resistance (Sec):** given a bit-string M and a key K, it should be impossible, except with non-negligible probability, to find a different bit-string M' such that $\mathcal{H}(K, M) = \mathcal{H}(K, M')$. That is, the following probability must be negligible in η for any adversary $\mathcal{A}$ and a given polynomial p:

$$Pr[\, K \leftarrow \{0,1\}^\eta;\ M \leftarrow \{0,1\}^{p(\eta)};\\ M' \leftarrow \mathcal{A}(K, M) :\ M \neq M' \ \wedge\ \mathcal{H}(K, M) = \mathcal{H}(K, M')]$$

- **Collision resistance: (Col)** it should be impossible, except with non-negligible probability, given a key K, to find two different bit-strings M and M' such that $\mathcal{H}(K, M) = \mathcal{H}(K, M')$. That is, the following probability must be negligible in η for any adversary $\mathcal{A}$:

$$Pr[K \leftarrow \{0,1\}^\eta;\ (M, M') \leftarrow \mathcal{A}(K) :\ M \neq M' \ \wedge\ \mathcal{H}(K, M) = \mathcal{H}(K, M')]$$

There are some well-known implications between the three previous criteria [20]: criterion **Pre** is implied by criterion **Sec** and **Sec** is itself implied by **Col**.

As there are no collisions in the symbolic setting, it is natural to require non-collision when proving computational soundness.

In the symbolic setting, the adversary cannot deduce from a hashed message $h(m)$ the underlying message m. In the computational setting, we have to ensure that an adversary is not able to deduce significative parts of a bit-string from its hashed value. For example, given $\mathcal{H}(k_H, bs_1 \cdot bs_2)$, an adversary should not be able, with non-negligible probability, to deduce bs_1, or to produce a new hash of a message containing bs_1, if the length of bs_1 depends on the security parameter. The **Pre** and **Sec** requirements are not sufficient in this case as they only provide that the adversary is not able to produce $bs_1 \cdot bs_2$.

Another approach to ensure security of hash functions is to use probabilistic hash functions citecanetti-hash. Whilst the security notion described in citecanetti-hash ensure that any hash preserves any partial information over the hashed message, it does not ensure that given a hash containing a secret nonce, it is impossible to forge a new hash containing the secret nonce. Moreover the hash functions considered are probabilistic.

Hence we define a new criterion for hash functions. We want to ensure that hash functions satisfy a form of semantic security. However it is not possible to directly adapt classical definitions as hash functions are deterministic. The adversary cannot have full access to a left-right hash oracle. Therefore we introduce a new security game where first some challenge nonces are randomly sampled. The adversary has access to a left-right oracle but his queries are not directly hashed: the challenge nonces are inserted in these queries before applying the hash function.

For that purpose, the left-right oracle takes as arguments pattern terms instead of bit-strings. Pattern terms as defined as follows.

4.2.1 Patterns

Pattern terms are terms where new atomic constants have been added: pattern variables. These variables represent challenge nonces whose values are not known by the adversary.

$$pat ::= \langle pat, pat \rangle \mid bs \mid [i]$$

The computation (evaluation) made by the oracle is easily defined recursively by replacing each variable $[i]$ by its value denoted by $\theta(i)$. The concatenation operator is still denoted by $\cdot$.

$$v(bs, \theta) = bs \qquad v([i], \theta) = \theta(i) \qquad v(\langle p_1, p_2 \rangle, \theta) = v(p_1, \theta) \cdot v(p_2, \theta)$$

We define the *length* of a pattern pat as the length of any result of $v(pat, \theta)$, for a given θ. The evaluation algorithm v is close to the *concr* algorithm . However as the main use for v is to perform operations on secret keys and *concr* is more general, we use two different algorithms.

A pattern is *hollow* if it contains at least one pattern variable.

Definition 4.4 [HASH]

Let $\mathcal{H}$ be a keyed hash function [6]. Then, the HASH criterion associated with $\mathcal{H}$, denoted by γ_H, is given by $(\Theta; F; V_{UNF}, V_{NC})$, where:

(i) Θ randomly generates a nonce N^H of length η and a hash key k.

(ii) F gives access to three oracles:
- a hashing oracle $\mathcal{H}_N$ which takes as input a hollow pattern *pat*, which has never been given to the oracle before, and returns the hash with k of *pat* completed with N^H;
- an oracle *Store* which keeps in memory each bit-string issued by the adversary;
- a hash key oracle which returns k.

(iii) V_{UNF} verifies if a bit-string *bs* has been produced by the hashing oracle whilst it has been given prior to the oracle *Store*.

(iv) V_{NC} takes as input two different patterns, not necessarily hollow, and verifies if, completed with N^H, they have the same hash with the key k.

A hash function $\mathcal{H}$ is said HASH iff γ_H is safe.

We denote by HASH/UNF=$(\Theta; F; V_{UNF})$ the subcriterion related to the unforgeability of hashes if some part of the hashed message is unknown and HASH/NC=$(\Theta; F; V_{NC})$ the subcriterion related to the collision criterion.

We note that HASH/NC is equivalent to the criterion **Col** and that HASH implies **Pre**. We also remark that in the random oracle model, the existence of a hash function that satisfies HASH is trivial (the hash function is given by the random oracle). On the other hand, in the standard model, we are not aware of an implementation satisfying HASH.

Similarly to IND-CCA, HASH can be extended to an equivalent criterion denoted by N-HASH where N challenge nonces are generated and used by the hashing oracle.

Proposition 4.5 *A hash function is HASH if an only if it is N-HASH, for any given* $N > 0$.

Proof. The fact that N-HASH implies *HASH* is obvious. To prove the converse, let us consider an adversary $\mathcal{A}$ against N-HASH/UNF. We build an adversary $\mathcal{B}$ against HASH/UNF. The adversary $\mathcal{A}$ has to produce a new hash of a message containing at least one of her challenge nonces. The adversary $\mathcal{B}$ randomly chooses an integer i between 1 and N. She simulates $\mathcal{A}$ by generating $N-1$ nonces and assigning it to the challenge nonces of $\mathcal{A}$, except for the i-th challenge nonce, which is associated with the challenge nonce of $\mathcal{A}$. This way, $\mathcal{B}$ wins her challenge whenever $\mathcal{A}$ wins her challenge by producing a new hash of a message containing the i-th challenge nonce. Hence

$$\mathbf{Adv}_{\mathcal{A}}^{N-HASH/UNF}(\eta) \leq N.\mathbf{Adv}_{\mathcal{B}}^{HASH/UNF}(\eta)$$

The implication of N-HASH/NC by HASH/NC is similar.

□

4.3 Security of the Cryptographic Library

Here we define the security of nested cryptographic primitives. This is done using a criterion that combines the two previous ones. That is, N asymmetric keys are generated together with a hash key k, N nonces N_i^H and a challenge bit b. The adversary can access oracles he was granted in the previous criteria (left-right encryption, public key and decryption for the asymmetric scheme, hashing and storing for the hash function) and can win either by deducing the value of b, by finding two different patterns having the same hash or by obtaining a bit-string with his hashing oracle that is already in $Store$. The encryption oracles now accept in input pairs of patterns (pat_0, pat_1) of equal length and output the encryption of the pattern pat_b completed with the corresponding N_i^H. Patterns for these oracle are given by the following grammar:

$$\begin{aligned} pat ::= & \langle pat, pat \rangle \mid bs \mid [i] \\ & \mid \{pat\}_{bs} \quad \text{encryption} \\ & \mid h(pat) \quad \text{hashing} \end{aligned}$$

The evaluation algorithm v is extended in a straightforward way to handle these patterns.

Let γ_N be the criterion including the oracles detailed above and the following verifiers: V_{IND} that returns true for bit b, V_{UNF} that returns true if the hashing oracle outputs a bit-string already in $Store$ and V_{NC} that returns true if it is given two different patterns which have the same hash.

Definition 4.6 A cryptographic library $(\mathcal{AE}, \mathcal{H})$ is said N-PAH iff for any adversary $\mathcal{A}$ in $PPTM$ the advantage $\mathbf{Adv}_{\mathcal{A}}^{\gamma_N}(\eta)$ is negligible and the $PrRand$ related to collision is negligible.

Such a library exists under the hypothesis that there exists an IND-CCA asymmetric encryption scheme and a HASH hash function.

Proposition 4.7 *If a cryptographic library $\mathcal{CL}$ is IND-CCA and HASH, it is also N-PAH.*

Proof. This proof can easily be done using the partition theorem presented in [13]. We only give a proof sketch. Let $\mathcal{A}$ be an adversary against N-PAH. The intuition behind the proof is that if $\mathcal{A}$ is able to win against the indistinguishability criterion, then we can build an adversary $\mathcal{B}_1$ against N-IND-CCA who wins her challenge whenever $\mathcal{B}$ wins her own. If $\mathcal{A}$ is able to win against the non-collision challenge, then an adversary $\mathcal{B}_2$ against N-HASH is able to win her collision challenge by simalting $\mathcal{A}$. The idea is that $\mathcal{B}_2$ does not have to use her challenge nonces to simulate $\mathcal{A}$, she can generates new one, so she can simulate the encryption oracle easily, If $\mathcal{A}$ is able to win against the unforgeability criterion, we build an adversary $\mathcal{B}_3$ against the unforgeability of N-PAH that simulates $\mathcal{A}$ using her own hashing oracle, but using newly generated nonces for the encryption oracle. The idea is that either behavior of $\mathcal{A}$ is not affected by the "incorrect" answers produced by the

encryption oracle, and then the advantage of $\mathcal{B}_3$ is the same that the advantage of $\mathcal{A}$, or the behavior of $\mathcal{A}$ is different, and in this case we can build an adversary $\mathcal{B}_4$ against N-IND-CCA that distinguish the normal behavior of $\mathcal{A}$ from her behavior when she is simulated by $\mathcal{B}_3$. Hence the advantage of $\mathcal{A}$ is bounded by a sum of the advantages of B_1, B_2, $\mathcal{B}_3$ and $\mathcal{B}_4$ which is negligible. □

5 Computational Soundness of the Symbolic Semantics

In this section, we prove that under some hypotheses on the considered protocol and cryptographic primitives any computational trace has a valid symbolic abstraction except for negligible probability. We then exploit this to show a preservation result that states that if a symbolic trace property is satisfied then its computational concretization is also satisfied.

5.1 Hypotheses on protocols

Our hypotheses on protocols are either syntactic or automatically provable in the symbolic semantics. The first hypothesis is that we only consider executable protocols (see Def. 3.1). Second, we only consider protocols that preserve secrecy. That is, we assume that for any atomic message $a \in atom(\Pi)$ either $\Pi \models_f \mathsf{Secret}(a)$ or $a \in \mathsf{IK}$ (see Def. 3.6).

A protocol that satisfies these restrictions is called *well-formed*.

5.2 Relating Computational and Symbolic Traces

In order to state our theorem, we need to define the concretization of a symbolic trace t_f. As we are dealing with cryptographic primitives that are not deterministic there is, in general, a set of concretizations of t_f. Therefore, given a symbolic trace t_f, we denote by $Concr(t_f, \theta)$ the set of computational traces obtained by applying $concr(\cdot, \theta')$ on each symbolic message of t_f where θ' is a mapping from bit-strings to symbolic messages compatible with θ. Then, we can state our main theorem as follows.

Theorem 5.1 *Let $\Pi = (R, \mathsf{IK})$ be a well-formed protocol that uses a Q-PAH secure cryptographic library, where Q is the sum of the numbers of keys and nonces in Π. Let $\mathcal{A}$ be an adversary. Then, the following probability is negligible as a function of η:*

$$Pr[\nexists t_f \in \mathsf{Traces}(\Pi) : Exec(\mathcal{A}, \mathsf{IK}, R, \theta) \in Concr(t_f, \theta)]$$

Proof. The proof is very similar to the proofs found in [19,17,12]. The idea is to build an adversary against N-PAH which executes $\mathcal{A}$ by simulating $Exec$. The adversary $\mathcal{B}$ also build the symbolic execution corresponding to the computational execution observed. Whenever $\mathcal{A}$ is able to produces an execution that does not correspond to a valid symbolic execution, then $\mathcal{B}$ is able to win against one of her challenge. Hence the probability that $\mathcal{A}$ produces such an execution is bounded by the advantage of $\mathcal{B}$, which is negligible by assumption. □

5.3 Relating Symbolic and Computational Properties

Recall that a computational trace property P_c is given by a set of computational traces and a symbolic trace property P_f is given by a set of symbolic traces. We say that P_f is a *faithful abstraction* of P_c for protocol Π, if the probability that a concretization of a symbolic trace in P_f is not in P_c is negligible. In other words, the following probability is negligible:

$$Pr[\exists t_f \in \mathsf{Traces}(\Pi) : t_f \in P_f \wedge Exec(A, \mathsf{IK}, R, \theta) \in Concr(t_f) \wedge Exec(A, \mathsf{IK}, R, \theta) \notin P_c].$$

The following proposition is a preservation result for faithful trace properties. It states that if the symbolic property is a faithful abstraction of the computational property and it is satisfied in the symbolic model then the concrete property is satisfied in the computational model. It has been applied to mutual authentication in [19] in which there is also a longer discussion about symbolic/computational properties.

Proposition 5.2 *Let P_f and P_c be a symbolic, respectively a computational, property. Let Π be a well-formed protocol that uses a Q-PAH secure library. If P_f is a faithful abstraction of P_c for Π and if $\Pi \models_f P_f$, then $\Pi \models_c P_c$. In other words, if the protocol satisfies the property P_f in the symbolic model then it satisfies P_c in the computational model.*

Proof. This proposition is a consequence of Theorem 5.1. Indeed, we have

$$\begin{aligned}
&Pr[Exec(\mathcal{A}, \mathsf{IK}, R, \theta) \notin P_c] = \\
&Pr[\exists t_f \in \mathsf{Traces}(\Pi) : t_f \in P_f \wedge Exec(\mathcal{A}, \mathsf{IK}, R, \theta) \in Concr(t_f)] + \\
&Pr[\exists t_f \in \mathsf{Traces}(\Pi) : t_f \notin P_f \wedge Exec(\mathcal{A}, \mathsf{IK}, R, \theta) \in Concr(t_f)] + \\
&Pr[\nexists t_f \in \mathsf{Traces}(\Pi) : Exec(\mathcal{A}, \mathsf{IK}, R, \theta) \in Concr(t_f)] \\
&= \nu(\eta) + 0 + \nu'(\eta)
\end{aligned}$$

where ν and ν' are negligible. □

A similar result can be proven for SecNonce [12].

Theorem 5.3 *Let Π be a well-formed protocol that uses a Q-PAH secure library, N a nonce in Π never sent under a hash and $\mathcal{A}$ an adversary. If $\Pi \models_f \mathsf{Secret}(N)$ then $\Pi \models_c \mathsf{SecNonce}(N)$.*

Proof. The proof is very similar to Theorem 5.1. □

We remark that we cannot deal with computational secrecy of nonces sent in hashes. It is because we do not assume with our HASH criterion that hash functions ensure indistinguishability. We only assume that a hash does not leak enough information to obtain the entire hashed message. A definition for secrecy of nonce that may be sent in hashes is given in [11].

Conclusion

The main contributions of this paper are the following: a formal definition of a correctness criterion for hash functions (that is easily met in the random oracle model). A proof of correctness of the Dolev-Yao model for protocols that may combine an asymmetric scheme and a hash function. The proof of our theorem makes some restrictions on the protocols that are in practice easily met. As future work, it would be of interest to investigate whether correctness of Dolev-Yao can be proved under weaker assumptions on the cryptographic primitives. Moreover, it would be significant to extend this result to other security properties.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany.

[2] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Moedersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The Avispa tool for the automated validation of internet security protocols and applications. In *CAV 2005, 17th Int. Conf. on Computer Aided Verification*, volume 3576 of *LNCS*, pages 281–285, Edinburgh, Scotland, UK, July 2005. Springer-Verlag.

[3] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 220–230. ACM Press, 2003.

[4] M. Backes, B. Pfitzmann, and M. Waidner. Limits of the reactive simulatability/uc of dolev-yao models with hashes, 2006.

[5] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology-EUROCRYPT 2000, Lecture Notes in Comput. Sci., Vol. 1807*, pages 259–274. Springer, 2000.

[6] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[7] B. Blanchet. Abstracting cryptographic protocols by prolog rules. In *International Static Analysis Symposium*, volume 2126 of *LNCS*, pages 433–436, 2001.

[8] L. Bozga, Y. Lakhnech, and M. Périn. Hermes: An automatic tool for verification of secrecy in security protocols. In *15th International Conference on Computer Aided Verification (CAV)*, volume 2725 of *LNCS*, 2003.

[9] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1997.

[10] J. A. Clark and J. L. Jacob. A survey of authentication protocol literature. Version 1.0, University of York, Department of Computer Science, November 1997.

[11] V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationnaly sound symbolic secrecy in the presence of hash functions, to appear, 2006.

[12] V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *Proceeding of the Fourtennth European Symposium on Programming (ESOP 2005)*, pages 157–171. Springer-Verlag, 2005.

[13] M. Daubignard, R. Janvier, Y. Lakhnech, and L. Mazaré. Game-based criterion partition applied to computational soundness of adaptive security. In *International Workshop on Formal Aspects in Security and Trust (FAST'06)*, Hamilton, Canada, August 2006. To appear.

[14] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[15] F. D. Garcia and P. Rossum. Sound computational interpretation of formal hashes. Cryptology ePrint Archive, Report 2006/014, 2006. http://eprint.iacr.org/.

[16] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[17] R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the Picture: Soundness of Formal Encryption in the Presence of Active Adversaries. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185, Edinburgh, U.K, April 2005. Springer.

[18] G. Lowe. An attack on the Needham-Schroeder public-key authentification protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[19] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proceedings of the Theory of Cryptography Conference*, pages 133–151. Springer, 2004.

[20] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *FSE*, Lecture Notes in Computer Science, pages 371–388. Springer, 2004.

[21] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *IEEE Computer Security Foundations Workshop*, 2001.