Computationally Sound Symbolic Secrecy in the Presence of Hash Functions*

Véronique Cortier[1], Steve Kremer[2], Ralf Küsters[3], and Bogdan Warinschi[4]

[1] Loria, CNRS & INRIA project Cassis, France
[2] LSV, CNRS & ENS Cachan & INRIA project Secsi, France
[3] ETH Zurich, Switzerland
[4] Loria, Univerité Henri Poincaré & INRIA project Cassis, France

Abstract. The standard symbolic, deducibility-based notions of secrecy are in general insufficient from a cryptographic point of view, especially in presence of hash functions. In this paper we devise and motivate a more appropriate secrecy criterion which exactly captures a standard cryptographic notion of secrecy for protocols involving public-key enryption and hash functions: protocols that satisfy it are computationally secure while any violation of our criterion directly leads to an attack. Furthermore, we prove that our criterion is decidable via an NP decision procedure. Our results hold for standard security notions for encryption and hash functions modeled as random oracles.

1 Introduction

Two distinct kinds of models have been developed for the rigorous design and analysis of cryptographic protocols: the so-called Dolev-Yao, symbolic, or formal models on the one hand and the cryptographic, computational, or concrete models on the other hand. In symbolic models messages are considered as formal terms and the adversary can manipulate these terms based on a fixed set of operations. The main advantage of the symbolic approach is its relative simplicity which makes it amenable to automated analysis tools (see, e.g., [6, 15]). In cryptographic models, messages are actual bit strings and the adversary is an arbitrary probabilistic polynomial-time (ppt) Turing machine. While proofs in this kind of models yield strong security guarantees, the proofs are often quite involved and only rarely suitable for automation (see, e.g., [11, 5]).

Starting with the seminal work of Abadi and Rogaway [1], a significant amount of research has been directed at bridging the gap between the two approaches. The goal is to obtain the best of both worlds: simple, automated security proofs that entail strong security guarantees. The typical approach is to show that the executions of the computational adversaries correspond to executions of the symbolic adversaries, and then use this result to show how to translate security notions from the symbolic world to the computational world.

For some security notions like integrity and authentication, the derivation of computational guarantees out of symbolic ones can be done with relative simplicity [3, 14]. In contrast, analogous results for the basic notion of secrecy proved significantly more

* Work partly supported by ACI Jeunes Chercheurs JC9005 and ARA SSIA Formacrypt.

elusive and have appeared only recently [4, 10, 12, 7]. The apparent reason for this situation is the striking difference between the definitional ideas used in the two different models. Symbolic secrecy typically states that the adversary cannot deduce the entire secret from the messages it gathers in an execution. On the other hand, computational secrecy requires that not only the secret, but also no partial information is leaked to the adversary. A typical formulation that is used requires the adversary to distinguish between the secret and a completely unrelated alternative.

OUR CONTRIBUTIONS. In this paper we investigate soundness results for symbolic secrecy in the presence of hash functions. One of the main motivations for considering hash functions, which have not been considered in the aforementioned results[5], is that they present a new challenge in linking symbolic and cryptographic secrecy: Unlike ciphertexts, hashes have to be publicly verifiable, i.e., any third party can verify if a value h is the hash value corresponding to a given message m. This implies that a simple minded extension of previous results on symbolic and computational secrecy fails. Assume, for example, that in some protocol the hash $h = h(s)$ of some secret s is sent in clear over the network. Then, while virtually all symbolic models would conclude that s remains secret (and this is also a naive assumption often made in practice), a trivial attack works in computational models: given s, s' and h, compare h with $h(s)$ and $h(s')$, and therefore recover s. Similar verifiability properties also occur in other settings, e.g. digital signatures which do not reveal the message signed.

In this paper we propose a new symbolic definition for nonce secrecy in protocols that use party identities, nonces, hash functions, and public key encryption. The definition that we give is based on the intuitively appealing concept of patterns [1].

The central aspect of our criterion is that it captures precisely security in the computational world in the sense that it is both sound and complete. More specifically, nonces that are secret according to our *symbolic* criterion are also secret according to a standard *computational* definition. Furthermore, there exist successful attacks against the secrecy of any nonce that does not satisfy our definition. Our theorems hold for protocols implemented with encryption schemes that satisfy standard notions of security, and for hash functions modeled as random oracles. In the proofs we combine different techniques from cryptography and make direct use of a (non-trivial) extension of the mapping theorem of [14] to hash functions.

Our second important result is to prove the decidability of our symbolic secrecy criterion (w.r.t. a bounded number of sessions). This is a crucial result that enables the automatic verification of computational secrecy for nonces. We give an NP-decision procedure based on constraint solving, a technique that is suitable for practical implementations [2]. While the constraint solving technique is standard in automatic protocol analysis, we had to adapt it for our symbolic secrecy criterion: For the standard deducibility-based secrecy definition it suffices to transform constraint systems until one obtains a so-called simple form. However, for our symbolic secrecy criterion further transformations might be required in order for the procedure to be complete. Identifying a sufficient set of such transformations and proving that they are sufficient turned out to be non-trivial.

[5] One exception is [12] where hash functions are allowed, but only as randomness extractors.

RELATED WORK. The papers that are immediately related to our work are those of Cortier and Warinschi [10], Backes and Pfitzmann [4], and Canetti and Herzog [7], who study computationally sound secrecy properties, as well as the paper by Janvier et al. [13], that presents a soundness result in the presence of hash functions. In this context, our work is the first to tackle computationally sound secrecy in the presence of hashes. We study the translation of symbolic secrecy into a computational version in a setting closely related to that in [10]. However, the use of hashes requires, as explained above, new notions and non-trivial extensions of the results proved there. In [13], Janvier et al. present a soundness result that differs however from this one. On the one hand they do not consider computational secrecy of nonces sent under hash functions. On the other hand, they present a new security criterion for hash functions, which is not the random oracle, although no implementation of a hash function satisfying their criterion is currently known. The work in [4] and [7] is concerned with secrecy properties of key-exchange protocols in the context of simulation-based security, and hence, they study different computational settings. Interestingly, the symbolic criterion used in [7] is also formalized using patterns, but their use is unrelated to ours. None of the mentioned works considers decidability issues.

PAPER OUTLINE. In the following section, we introduce the symbolic and computational models. Our symbolic secrecy criterion is developed in Section 3. We state and prove the soundness and completeness of this criterion w.r.t. computational secrecy in Section 4, and prove its decidability in Section 5. Details about the models and proofs can be found in [9].

2 The Symbolic and Concrete Protocol and Intruder Models

2.1 The Symbolic Model

We define (symbolic) messages and terms, how honest agents and the (Dolev-Yao-style) intruder can derive messages from a set of messages, and how protocols are specified.

MESSAGES AND TERMS. To define messages, we consider an infinite set A of agent identities, infinite sets Nonce_{ag}, Nonce_{adv}, Rand_{ag}, and Rand_{adv} (nonces and random coins generated by the agents and the adversary, respectively), and an infinite set $\mathsf{Garbage}$ representing garbage messages. All of these sets are assumed to be pairwise disjoint. We set $\mathsf{Nonce} = \mathsf{Nonce}_{ag} \cup \mathsf{Nonce}_{adv}$ and $\mathsf{Rand} = \mathsf{Rand}_{ag} \cup \mathsf{Rand}_{adv}$.

The set of messages M (w.r.t. A, Nonce, and Rand) is defined by the following grammar: $\mathsf{M} ::= \mathsf{A} \mid \mathsf{Nonce} \mid \mathsf{ek}(\mathsf{A}) \mid \mathsf{dk}(\mathsf{A}) \mid \langle \mathsf{M}, \mathsf{M} \rangle \mid \{\mathsf{M}\}^{\mathsf{Rand}}_{\mathsf{ek}(\mathsf{A})} \mid \mathsf{h}(\mathsf{M}) \mid \mathsf{Garbage}$ where $\mathsf{ek}(a)$ and $\mathsf{dk}(a)$ with $a \in \mathsf{A}$ denote the public and private key of a, respectively, $\langle m, m' \rangle$ denotes pairing of m and m', $\{m\}^r_{\mathsf{ek}(a)}$ denotes the message m encrypted with $\mathsf{ek}(a)$ using the random coins r, and $h(m)$ is the hash of m. We define the following subsets of M: EKey, DKey, $\mathsf{Ciphertext}$, Hash, and Pair are the sets of all messages starting with $\mathsf{ek}(\cdot)$, $\mathsf{dk}(\cdot)$, $\{\cdot\}^{\cdot}_{\cdot}$, $\mathsf{h}(\cdot)$, and $\langle \cdot, \cdot \rangle$, respectively. We sometimes refer to the sets introduced above as types.

We assume an infinite set of typed variables X where the types are as above and for a variable of a certain type only messages of this type may be substituted. In particular, we assume variables A_i, $i \in \{1, \ldots, k\}$, for agent identities and variables $X^j_{A_i}$, $L^j_{A_i}$

$j \in \mathbb{N}$, for fresh nonces and random coins generated by A_i. The set of terms $\mathsf{T}(\mathsf{X})$ over X is defined analogously to the set of messages.

DERIVING MESSAGES. terms. The set of terms that can be derived from ϕ is defined by the deduction rules given in Figure 1. We write $\phi \vdash t$ to say that t can be derived from ϕ. For example, $\{\langle \mathsf{dk}(a), \{c\}^r_{\mathsf{ek}(a)}\rangle\} \cup \mathsf{A} \vdash \{c\}^{r'}_{\mathsf{ek}(b)}$ where $b \in \mathsf{A}$ and $r' \in \mathsf{Rand}_{adv}$.

$$\frac{}{\phi \vdash m}\; m \in \phi \quad \frac{\phi \vdash b}{\phi \vdash \mathsf{ek}(b)}\; b \in \mathsf{A} \cup \mathsf{X}.a \quad \frac{\phi \vdash m_1 \quad \phi \vdash m_2}{\phi \vdash \langle m_1, m_2\rangle} \quad \frac{\phi \vdash \langle m_1, m_2\rangle}{\phi \vdash m_i}\; i \in \{1,2\}$$

$$\frac{\phi \vdash \mathsf{ek}(b), \phi \vdash m}{\phi \vdash \{m\}^r_{\mathsf{ek}(b)}}\; r \in \mathsf{Rand}_{adv} \quad \frac{\phi \vdash \{m\}^r_{\mathsf{ek}(b)} \quad \phi \vdash \mathsf{dk}(b)}{\phi \vdash m} \quad \frac{\phi \vdash m}{\phi \vdash \mathsf{h}(m)}$$

Fig. 1. Deduction rules

PROTOCOLS. Roles are usually specified by a sequence of input/output actions. In order to model branching protocols, the *roles* we consider are ordered edge-labeled finite trees where every edge is labeled by an *agent rule* (l, r), where $l, r \in \mathsf{T}(\mathsf{X})$ are messages with variables, and certain syntactic conditions are satisfied such that the actions can actually be carried out (in a computational interpretation). A *k-party protocol* is a mapping $\Pi : [k] \to \mathsf{Roles}$ where $[k] = \{1, \ldots, k\}$ and Roles denotes the set of roles.

SYMBOLIC EXECUTION OF A PROTOCOL. The symbolic execution of a k-party protocol is modeled as a finite sequence of global states. A *global state* is a triple $(\mathsf{SId}, f, \varphi)$ where φ is a finite set of messages (the *current intruder knowledge*), SId is a finite set of session ids, and f maps every session id in SId to the current state of the corresponding session. This state is called the *local state* and is of the form $(i, \sigma, p, (a_1, a_2, \ldots, a_k))$ where $i \in [k]$ is the index of the role that is executed in this session, σ is a substitution whose domain is a subset of the variables occurring in $\Pi(i)$ (i.e., σ determines the messages assigned to variables so far in the current session), p is a node of $\Pi(i)$ and determines at what node the agent currently stands, and $(a_1, a_2, \ldots, a_k) \in \mathsf{A}^k$ is the tuple of names of the agents that are involved in the session, where a_i is the agent carrying out the current session (supposedly with the mentioned agents a_j, $j \neq i$). The initial state is $q_I = (\emptyset, \emptyset, \mathsf{A} \cup \mathsf{EKey} \cup \mathsf{Nonce}_{adv})$, i.e., the intruder knows all names and public keys of agents as well as the infinite set of intruder nonces.

We allow three kinds of transitions between global states.

- The adversary corrupts a set of parties and thereby learns the private keys of the agents: $q_I \xrightarrow{\mathbf{corrupt}(a_1,\ldots,a_l)} (\emptyset, \emptyset, \mathsf{A} \cup \mathsf{EKey} \cup \{\mathsf{dk}(a_j) \mid 1 \leq j \leq l\})$. Note that this transition can only be applied at the beginning (static corruption).
- The adversary can initiate new sessions: $(\mathsf{SId}, f, \varphi) \xrightarrow{\mathbf{new}(i,a_1,\ldots,a_k)} (\mathsf{SId}', f', \varphi)$ where SId' and f' are defined as follows. Let $\mathsf{sid} = |\mathsf{SId}|+1$ be the session identifier of the new session where $|\mathsf{SId}|$ denotes the cardinality of SId. We define $\mathsf{SId}' = \mathsf{SId} \cup \{\mathsf{sid}\}$. The function f' is defined as: $f'(\mathsf{sid}') = f(\mathsf{sid}')$ for every $\mathsf{sid}' \in \mathsf{SId}$

and $f'(\mathsf{sid}) = (i, \sigma, \varepsilon, (a_1, \ldots, a_k))$ where ϵ denotes the root of the role tree and $\sigma(A_j) = a_j$ for $1 \leq j \leq k$ and $\sigma(X^j_{A_i}) = n^{a_i,j,s}$, $\sigma(L^j_{A_i}) = l^{a_i,j,s}$ for $j \in \mathbb{N}$.

– The adversary can send messages: $(\mathsf{SId}, f, \varphi) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}, f', \varphi')$ where $\mathsf{sid} \in \mathsf{SId}$, $m \in \mathsf{M}$, and φ' and f' are defined as follows. We define $f'(\mathsf{sid}') = f(\mathsf{sid}')$ for every $\mathsf{sid}' \neq \mathsf{sid}$. Suppose that $f(\mathsf{sid}) = (i, \sigma, p, (a_1, \ldots, a_k))$ and $(l_1, r_1), \ldots, (l_h, r_h)$ are the labels of edges leaving p (in this order). We distinguish two cases:
  - there does not exist a j such that m and $l_j\sigma$ match. Then, we define $f'(\mathsf{sid}) = f(\mathsf{sid})$ and $\varphi' = \varphi$ (the state remains unchanged);
  - else, let j be minimal s. t. m and $l_j\sigma$ match. Let θ be the matcher, i.e., $m = (l_j\sigma)\theta$. We define $f'(\mathsf{sid}) = (i, \sigma\cup\theta, pj, (a_1, \ldots, a_k))$ and $\varphi' = \varphi\cup\{(r_j\tau_{a_i,\mathsf{sid}})\sigma\theta\}$.

A finite sequence of global states is called a *symbolic execution trace* (for a protocol Π) if it starts with the initial global state q_I and two consecutive global states in this sequence are connected via one of the above transitions. We say that a trace is *valid* if every send transition $(\mathsf{SId}, f, \varphi) \xrightarrow{\mathbf{send}(\mathsf{sid},m)} (\mathsf{SId}, f', \varphi')$ verifies that the adversary could actually deduce m, that is $\varphi \vdash m$. The set of valid symbolic execution traces (for a protocol Π) is denoted by $\mathsf{Exec}^s(\Pi)$. The set of valid set of messages is defined by $\mathsf{Msg}^s(\Pi) = \{\varphi \mid (\mathsf{SId}, f, \varphi)$ is the last state of a valid execution trace$\}$.

2.2 The Concrete Model

The concrete model is defined w.r.t. an encryption scheme $\mathcal{AE} = (\mathsf{K_e}, \mathsf{Enc}, \mathsf{Dec})$, which we now fix once and for all. Hashing is modeled by the random oracle.

CONCRETE MESSAGES. *Concrete messages* are bit strings which carry type informationwhich can be efficiently computed. In bit strings of type Pair, the two components can be efficiently retrievedand strings of type $\mathsf{Ciphertext}$ carry the public key that supposedly was used to encrypt the plaintext. The set of bit strings is denoted by $\mathcal{C}^\eta$. This set depends on the security parameter η as this parameter determines the length of agent names, nonces, and keys. Substitutions now map variables (of some type) to concrete messages (of the same type).

CONCRETE EXECUTION OF A PROTOCOL. A *concrete global state* is a 4-tuple $(\mathsf{SId}, f, \varphi, \mathcal{H})$ where φ is a finite set of bit strings, SId is a finite set of session ids, and f maps every session id in SId to the current state of the corresponding session (the concrete local states). A *concrete local state* is defined just as a symbolic one, except that variables are now mapped to bit strings and agent names are also bit strings. The fourth component carries the state of the random oracle: $\mathcal{H}$ is a set of couples (m, h) where m is a bit string and h its corresponding hash value. A protocol is executed by running a ppt Turing machine, the (concrete) adversary, which may make queries corresponding to the transitions in the symbolic model. We allow four kinds of transitions between global states, which we will refer to by *corrupt*, *new*, *send transitions*, and *hash queries*. The semantics of the first three queries is defined by analogy with the formal execution model. In addition, the adversary may also make queries to the random oracle: $(\mathsf{SId}, f, \varphi, \mathcal{H}) \xrightarrow{\mathbf{hash}(m)} (\mathsf{SId}, f, \varphi, \mathcal{H}')$ where $\mathcal{H}'$ is defined as follows. If there exists n such that $(m, n) \in \mathcal{H}$, then $\mathcal{H}' = \mathcal{H}$ and we define $h = n$. Else a hash value h is generated at random for m and $\mathcal{H}' = \mathcal{H} \cup \{(m, h)\}$. In any case, h is returned to the adversary. A finite sequence of concrete global states is called a *concrete execution trace* if it starts with the initial global state. Obviously, since the adversary is a ppt Turing machine the length of the trace is bounded by a polynomial in the security parameter η. Also, the sequence of random coins R_Π used in the execution by the honest agents and the random oracle as well as the sequence of random coins $R_\mathcal{A}$ used by the adversary can be bounded in length by polynomials $g_\mathcal{A}(\eta)$ and $p_\mathcal{A}(\eta)$, respectively. Clearly, if R_Π and $R_\mathcal{A}$ are fixed, we obtain a uniquely determined concrete trace, which we denote by $\mathsf{Exec}_{\Pi(R_\Pi),\mathcal{A}(R_\mathcal{A})}(\eta)$.

3 Symbolic and Computational Secrecy Properties

In this section we recall the computational definition of secrecy and introduce our new symbolic definition for secrecy.

COMPUTATIONAL SECRECY. Computational secrecy requires that no partial information is leaked to the adversary. The typical way to formalize this idea is to require that the secret s is indistinguishable from an unrelated random bitstring s' chosen (from an appropriate distribution). The secrecy of nonce variable X_{A_i} (the nonce generated by A_i in the ith role of the protocol) in protocol Π is defined as follows.

Definition 1. *Consider the experiment $\mathbf{Exp}^{\mathrm{sec\_}b}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta)$ parametrized by a bit b and that involves an adversary $\mathcal{A}$ against protocol Π. The experiment takes as input a security parameter η and starts by generating two random nonces n_0 and n_1 in $\mathcal{C}^\eta.n$. Then the adversary $\mathcal{A}$ starts interacting with the protocol Π as in the execution described by $\mathsf{Exec}_{\Pi,\mathcal{A}}(\eta)$. At some point in the execution the adversary initiates a session s in which the role of A_i is executed, and declares this session under attack. In this session, the variable $X^j_{A_i}$ is instantiated with n_b. The rest of the execution is exactly as in $\mathsf{Exec}_{\Pi,\mathcal{A}}(\eta)$. At some point the adversary requires the two nonces n_0 and n_1 and has to output a guess d. The bit d is the result of the experiment. We define the advantage of the adversary $\mathcal{A}$ by:*

$$\mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta) = \Pr\left[\mathbf{Exp}^{\mathrm{sec\_1}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta){=}1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{sec\_0}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta){=}1\right]$$

We say that nonce $X^j_{A_i}$ is computationally secret in protocol Π, and we write $\Pi \models^{\mathsf{c}} \mathsf{SecNonce}(i,j)$ if for every p.p.t. adversary $\mathcal{A}$ its advantage is negligible.

SYMBOLIC SECRECY. As explained in the introduction, weak secrecy is not sufficient to capture the standard indistinguishability-based notion used in computational settings. The new notion of secrecy we propose here relies on the intuitively appealing concept of patterns [1]. Roughly, the pattern of an expression is obtained by replacing with $\Box$, all the subterms of the expression that are secret. In our case, a subterm T of T' is secret if, even when given T the adversary cannot verify that T has been used to construct T'. Formally, we add T to the knowledge set ϕ in the deduction relation. The ideas behind our definition of patterns are related to offline guessing attacks, where the adversary is given the weak secret and should be unable to test whether the given weak secret is indeed the one used in the observed messages.

Definition 2 (Patterns). *Given a set of closed terms $\phi = \{M_1, M_2, \ldots, M_k\}$ and a term T, we define $\mathsf{Pat}_T(\phi) = \{\mathsf{Pat}_T^\phi(M_1), \mathsf{Pat}_T^\phi(M_2), \ldots, \mathsf{Pat}_T^\phi(M_k)\}$, where $\mathsf{Pat}_T^\phi(M)$ defined recursively by:*

$$
\begin{aligned}
\mathsf{Pat}_T^\phi(a) &= \begin{cases} a & \textit{if } \phi, T \vdash a \\ \square & \textit{otherwise} \end{cases} \\
\mathsf{Pat}_T^\phi(\langle M_1, M_2\rangle) &= \langle \mathsf{Pat}_T^\phi(M_1), \mathsf{Pat}_T^\phi(M_2)\rangle \\
\mathsf{Pat}_T^\phi(\{M\}^r_{\mathsf{ek}(a)}) &= \begin{cases} \{\mathsf{Pat}_T^\phi(M)\}^r_{\mathsf{ek}(a)} & \textit{if } \phi, T \vdash \mathsf{dk}(a) \textit{ or if } r \in \mathsf{Rand}_{adv} \\ \square & \textit{otherwise} \end{cases} \\
\mathsf{Pat}_T^\phi(h(M)) &= \begin{cases} h(\mathsf{Pat}_T^\phi(M)) & \textit{if } \phi, T \vdash M \\ \square & \textit{otherwise} \end{cases}
\end{aligned}
$$

Pat_T^ϕ *is extended to set of messages as expected:* $\mathsf{Pat}_T^\phi(S) = \bigcup_{t \in S} \mathsf{Pat}_T^\phi(t)$.

The messages of ϕ may contain some subterms of the form $\{M\}^r_{\mathsf{ek}(a)}$ where $r \in \mathsf{Rand}_{adv}$. Because of the random coins such messages must have been build by the adversary and M should be deducible. Thus we consider ϕ augmented with such messages: $\overline{\phi} = \phi \cup \{M \mid \{M\}^r_{\mathsf{ek}(a)} \text{ subterm of } \phi\}$. For any valid message set ϕ (that is $\phi \in \mathsf{Msg}^s(\Pi)$ for some protocol Π), we can show that $\phi \vdash M$ for every $M \in \overline{\phi}$.

Definition 3 (Nonce secrecy). *Let Π be a protocol and $X^j_{A_i}$ a nonce variable occurring in some role A_i. We say that $X^j_{A_i}$ is secret in Π and we write $\Pi \models^s \mathsf{SecNonce}(i, j)$, if for every valid set of messages $\phi \in \mathsf{Msg}^s(\Pi)$ it holds that for every session number s, the symbolic nonce $n^{a_i,j,s}$ does not occur in $\mathsf{Pat}_{n^{a_i,j,s}}(\overline{\phi})$.*

To better appreciate these definitions, consider the following examples.

1. Let $\phi_1 = \{h(\langle n_b, n'\rangle)\} = \overline{\phi_1}$. Then $\mathsf{Pat}_{n_b}(\overline{\phi_1}) = \{\square\}$. ϕ_1 preserves the indistinguishability of n_b since, intuitively, n_b is hidden by the secret nonce n'.
2. Let $\phi_2 = \{h(\langle n_b, \{n'\}^r_{\mathsf{ek}(a)}\rangle), n'\}$ where $r \notin \mathsf{Rand}_{adv}$. Then $\overline{\phi_2} = \phi_2$ and $\mathsf{Pat}_{n_b}(\overline{\phi_2}) = \{\square, n'\}$. In this example, the encryption of n' does hide n_b.
3. Let $\phi_3 = \{h(\langle n_b, \{n'\}^r_{\mathsf{ek}(a)}\rangle)\}$ where $r \in \mathsf{Rand}_{adv}$. Then $\overline{\phi_3} = \phi_3 \cup \{n'\}$ and $\mathsf{Pat}_{n_b}(\overline{\phi_3}) = \{h(\langle n_b, \{n'\}^r_{\mathsf{ek}(a)}\rangle), n'\}$. We have that n_b occurs in $\mathsf{Pat}_{n_b}(\overline{\phi_3})$. This corresponds indeed to an attack. As n' has been encrypted by the adversary himself he knows the ciphertext. Given n_0 and n_1 he computes both $h(\langle n_0, \{n'\}^r_{\mathsf{ek}(a)}\rangle)$ and $h(\langle n_1, \{n'\}^r_{\mathsf{ek}(a)}\rangle)$ and compares them to $h(\langle n_b, \{n'\}^r_{\mathsf{ek}(a)}\rangle)$ yielding the attack.
4. Let $\phi_4 = \{\{\langle h(n_b), h(n')\rangle\}^r_{\mathsf{ek}(a)}, \mathsf{dk}(a)\}$ where $r \notin \mathsf{Rand}_{adv}$. Then $\overline{\phi_4} = \phi_4$ and $\mathsf{Pat}_{n_b}(\overline{\phi_4}) = \{\{\langle h(n_b), \square\rangle\}^r_{\mathsf{ek}(a)}, \mathsf{dk}(a)\}$. Again, n_b does occur in $\mathsf{Pat}_{n_b}(\phi_1)$. For this attack an intruder may get $h(n_b)$ by decrypting and projecting the message $\{\langle h(n_b), h(n')\rangle\}^r_{\mathsf{ek}(a)}$ and compare $h(n_b)$ with $h(n_0)$ and $h(n_1)$ that he may compute from n_0 and n_1.

Our notion of secrecy has a useful equivalent formulation described in the following lemma. Informally, the lemma states that all unencrypted occurrences of the secret nonce in a set of messages are such that they occur in a term t that is hashed, and such that t itself can not be computed from ϕ and n.

Lemma 1. *Let ϕ be an arbitrary set of messages and n a nonce symbol that occurs in ϕ. n does not occur in $\mathsf{Pat}_n(\overline{\phi})$ if and only if $\overline{\phi} \not\vdash n$ and $\forall M$ subterm of ϕ such that $\overline{\phi} \vdash M$, $\forall p$ such that $M|_p = n$, so that there is no encryption along p, $\exists p' < p$ such that 1) $M|_{p'} = h(M')$ and 2) $\phi, n \not\vdash M'$.*

4 Symbolic Secrecy is Equivalent to Computational Secrecy

To prove the soundness and the completeness of our secrecy criterion, we proceed in two steps: i) relate symbolic and concrete traces and ii) prove equivalence of the symbolic and computational notions.

RELATING SYMBOLIC AND CONCRETE TRACES. The first step linking security properties in symbolic and concrete models is to exhibit a relation between individual execution traces. The relation is similar to that developed in previous works [14, 10], but our definitions and results have to deal with the use of random oracles in computational executions. In line with common practice in symbolic models, hash applications (explicitly captured as queries to the random oracle by concrete traces) are not reflected by the symbolic traces. Therefore, we define the *hash-query free* trace $\mathsf{clean\_hash}(t^c)$ associated to the concrete trace $t^c = (\mathsf{SId}^c_1, g_1, \varphi_1, \mathcal{H}_1), \ldots, (\mathsf{SId}^c_n, g_n, \varphi_n, \mathcal{H}_n)$. The trace $\mathsf{clean\_hash}(t^c)$ is the concrete trace $(\mathsf{SId}^c_{i_1}, g_{i_1}, \varphi_{i_1}, \mathcal{H}_{i_1}), \ldots, (\mathsf{SId}^c_{i_k}, g_{i_k}, \varphi_{i_k}, \mathcal{H}_{i_k})$, obtained by removing from t^c the states that are the result of a hash request.

Definition 4. *Let $t^s = (\mathsf{SId}^s_1, f_1, \phi_1), \ldots, (\mathsf{SId}^s_n, f_n, \phi_n)$ be a symbolic execution trace and let $\mathsf{clean\_hash}(t^c) = (\mathsf{SId}^c_1, g_1, \varphi_1, \mathcal{H}_1), \ldots, (\mathsf{SId}^c_n, g_n, \varphi_n, \mathcal{H}_n)$ be the hash-query free trace of concrete execution trace t_c.*

- *We say that trace t^c is a* concrete instantiation of t^s with (partial) mapping $c : \mathsf{M} \rightarrow \mathcal{C}^\eta$ *and we write $t^s \preceq^{\mathsf{c}} t^{\mathsf{c}}$ if for every ℓ ($1 \leq \ell \leq n$) it holds that $\mathsf{SId}^s_\ell = \mathsf{SId}^c_\ell$ and for every $\mathsf{sid} \in \mathsf{SId}^s_\ell$ if $f_\ell(\mathsf{sid}) = (\sigma^{\mathsf{sid}}, i^{\mathsf{sid}}, p^{\mathsf{sid}}, (a_1, \ldots, a_k))$ and $g_\ell(\mathsf{sid}) = (\tau^{\mathsf{sid}}, j^{\mathsf{sid}}, q^{\mathsf{sid}}, (a_1, \ldots, a_k))$ then $\tau^{\mathsf{sid}} = c \circ \sigma^{\mathsf{sid}}$, $i^{\mathsf{sid}} = j^{\mathsf{sid}}$ and $p^{\mathsf{sid}} = q^{\mathsf{sid}}$.*
- *Trace t^c is a* concrete instantiation with Dolev-Yao hash queries *of t^s and we write $t^s \preceq t^c$ if there exists a partial, injective function $c : \mathsf{M} \rightarrow \mathcal{C}^\eta$ such that $t^s \preceq^{\mathsf{c}} t^{\mathsf{c}}$ and for every $1 \leq k \leq n$, for every message m such that $(m, h) \in \mathcal{H}_k$ for some h, there exists a term M such that $c(M) = m$ and $\phi_k \vdash M$.*

Proposition 1. *Let Π be an executable protocol. If the encryption scheme $\mathcal{AE}$ is* IND-CCA *secure, and the hash functions are random oracles, then for any p.p.t. algorithm $\mathcal{A}$*

$$\Pr\left[\exists t^s \in \mathsf{Exec}^s(\Pi) \mid t^s \preceq \mathsf{Exec}^c_{\Pi(R_\Pi),\mathcal{A}(R_\mathcal{A})}(\eta)\right] \geq 1 - \nu_\mathcal{A}(\eta)$$

where the probability is over the choice $(R_\Pi, R_\mathcal{A}) \xleftarrow{\$} \{0,1\}^{p_\mathcal{A}(\eta)} \times \{0,1\}^{g_\mathcal{A}(\eta)}$ and $\nu_\mathcal{A}(\cdot)$ is some negligible function.

The proof shares many ideas with earlier work [14, 10].

SYMBOLIC SECRECY IS EQUIVALENT TO COMPUTATIONAL SECRECY. The following theorem states that the symbolic secrecy criterion is necessary and sufficient for computational secrecy to hold.

Theorem 1. *Let Π be an executable protocol and let $X^j_{A_i}$ be a nonce variable occurring in some role A_i. If the encryption scheme $\mathcal{AE}$ used in the implementation of Π is* IND-CCA *secure then $\Pi \models^s \mathsf{SecNonce}(i,j)$ if and only if $\Pi \models^c \mathsf{SecNonce}(i,j)$.*

Proof. **The "if" direction.** First, we give an ideal execution of the protocols that replaces real nonces with random strings. We show that no adversary can distinguish the modified execution, which we call the "oracle execution" from the real execution.

Next, we argue that in the oracle execution, nonces that are symbolically secret are information theoretically hidden from the computational adversary. Indeed, if the symbolic secrecy property is satisfied, by Lemma 1 the nonce occurs only in some hashed terms, and the term themselves are secret (i.e., it cannot be computed efficiently). Since in the random oracle model the hash values are independent of the hashed message, the view of the adversary is independent from the value of the secret nonces.

STEP I. We now describe the "oracle execution". Whenever the protocol dictates that an honest party encrypts some bitstring m, the party encrypts instead a randomly selected bitstring r_m of equal length. The execution keeps a table with all association (m, r_m), which we call the random associations table (RAT). The RAT is not made available to the adversary, but only to honest parties. Specifically, whenever an honest party receives encrypted messages, the party performs the appropriate decryption and recovers some plaintext. If the plaintext is some m' such that (m, m') occurs in RAT, the party treats the encryption as an encryption of m and continues its execution as normal. Otherwise, the underlying plaintext is set to m'.

Intuitively, if any adversary behaves differently in the two executions, it is because he can see the difference between encryptions of true, and random ciphertexts. Formally, if we let $\mathsf{Exec}_{\mathcal{A},\Pi}(\eta)$ be the output of adversary $\mathcal{A}$ when executed with protocol Π for security parameter η, and $\mathsf{Exec}^o_{\mathcal{A},\Pi}(\eta)$ the output of the adversary in the associated oracle execution, we have the following lemma.

Lemma 2. *Let Π be an executable protocol, and $\mathcal{A}$ an arbitrary ppt adversary. Then, if the encryption scheme $\mathcal{AE}$ used in the implementation of Π is* IND-CCA *secure, then $\Pr\left[\,\mathsf{Exec}_{\mathcal{A},\Pi}(\eta) = 1\,\right] - \Pr\left[\,\mathsf{Exec}^o_{\mathcal{A},\Pi}(\eta) = 1\,\right]$ is negligible.*

Notice that we can apply the above lemma for the case when the execution that is considered is used in the experiment $\mathbf{Exp}^{\mathrm{sec}\_b}_{\mathsf{Exec}_{\mathcal{A},\Pi}}(i,j)(\eta)$, for some b, i, j. If we write $\mathbf{Exp}^{\mathrm{sec}\_b}_{\mathsf{Exec}^o_{\mathcal{A},\Pi}}(i,j)(\eta)$ for the corresponding oracle execution, we obtain that there exists some negligible function $\nu_{i,j,b}$ such that

$$\Pr\left[\mathbf{Exp}^{\mathrm{sec}\_b}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{sec}\_b}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(i,j)(\eta) = 1\right] = \nu_{i,j,b}(\eta) \qquad (1)$$

STEP II. Next, we associate symbolic traces to the computational traces of the oracle execution. This enables us to reason about an adversary's success in the oracle execution (which is conceptually simpler). The association is in fact the one in the proof of Proposition 1, with an additional parsing step necessary to take into account the random association table that we detail below. In addition to access to the keys and the randomness of the parties, the parsing procedure also uses access to the random association table, and is as follows: the first step in processing some message m' is a search in the random association table. If (m, m') occurs in the RAT, then the procedure proceeds as before, with m' replaced by m, otherwise the procedure remains unchanged.

Next, we argue that the symbolic traces obtained as above are valid execution traces, and moreover, that they are included among the traces of the execution of Π. The formalization is given in the next lemma.

Lemma 3. *The symbolic traces of* $\mathsf{Exec}^o(\Pi, \mathcal{A})$ *are valid with overwhelming probability and* $\mathsf{Exec}^o_{\mathcal{A},\Pi} \subseteq \mathsf{Exec}_{\mathcal{A},\Pi}$.

STEP III. Finally, we prove that if $\mathcal{AE}$ is $\mathsf{IND\text{-}CCA}$ secure then $\Pi \models \mathsf{SecNonce}^s(i,j) \Rightarrow \Pi \models^{\mathsf{c}} \mathsf{SecNonce}(i,j)$. For an arbitrary adversary $\mathcal{A}$ against the secrecy of nonce $X^j_{A_i}$ recall that we write $\mathbf{Exp}^{\mathrm{sec\_}b}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(\eta)$ for the oracle version of the experiment defining secrecy of nonce $X^j_{A_i}$. Let $\mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}^o_{\mathcal{A},\Pi}}(\eta)$ be the corresponding advantage functions. By definition we have that:

$$\mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta) = \Pr\left[\mathbf{Exp}^{\mathrm{sec\_1}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta){=}1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{sec\_0}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta){=}1\right]$$

$$\mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(i,j)(\eta) = \Pr\left[\mathbf{Exp}^{\mathrm{sec\_1}}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(i,j)(\eta){=}1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{sec\_0}}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(i,j)(\eta){=}f1\right]$$

By subtracting, using Equation 1, we obtain that for some negligible function ν

$$\mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(i,j)(\eta) = \mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(i,j)(\eta) + \nu(\eta) \tag{2}$$

Finally, we show that in the oracle execution the advantage $\mathbf{Adv}^{\mathrm{sec}}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(i,j)(\eta)$ of any adversary $\mathcal{A}$ is negligible since nonces that are symbolically secret are informational theoretically hidden from the adversary. This can be seen as follows.

Consider the symbolic trace ϕ that corresponds to the execution of the experiment $\mathbf{Exp}^{\mathrm{sec\_}b}_{\mathsf{Exec}^o_{\Pi,\mathcal{A}}}(\eta)$, up to the point when the adversary is given the nonces and he is asked to determine the bit b. Let s be the id of the session under attack, and let $n^{a,j,s}$ be the symbolic nonce that corresponds to the nonce under attack. By Lemma 3, the trace ϕ is with overwhelming probability a Dolev-Yao trace of protocol Π. By the hypothesis of the theorem $\Pi \models^s \mathsf{SecNonce}(i,j)$ and therefore by Lemma 1, all occurrences of $n^{a,j,s}$ in ϕ that are not under an honest encryption are in some term T_i that appears under a hash, and T_i is nondeductible from $\phi, n^{i,j,s}$. Let t_i be the bitstrings that correspond to the terms T_i. We conclude by observing that in the real execution, the adversary may observe the values $c_1 = h(t_1), c_2 = h(t_2), \ldots$, but provided that it does not query $t_1, t_2, \ldots$ to the random oracle, their values (and thus in particular the value of the secret nonce) are independent from the $c_1, c_2, \ldots$. Since all queries to the random oracle are the images of deductible terms, we conclude that $\mathcal{A}$ does not request $h(t_i)$, for all i.

The "only if" direction. It is important to observe that if a message M is deducible from a set of messages $M_1, M_2, \ldots, M_n$, the associated deduction tree τ can be translated into an (efficient) program $\overline{\tau}$ which given the bit-string representations of m_i for M_i ($i = 1, 2, \ldots, n$) computes the bit-string representation m of M.

We proceed as follows. Assume that for some symbolic trace ϕ, the symbolic nonce $n^{a_i,j,s}$ occurs in $\mathsf{Pat}_{n^{a_i,j,s}}(\overline{\phi})$, starting from Lemma 1 we can show that there exist a term $M \in \overline{\phi}$ and a deduction tree τ such that: 1) $\tau(\phi, n^{a_i,j,s})$ yields message M and 2) for $n \neq n^{a_i,j,s}$, $\tau(\phi, n)$ does not yield M. Since $M \in \overline{\phi}$, we know that there also exists a deduction tree π such that $\pi(\phi)$ yields M.

Based on the above, we construct a two-stage adversary against secrecy of nonce $X^j_{A_i}$. In the first stage, the adversary produces a computational representation ϕ^{c} of the trace ϕ (by simply following the instructions of the Dolev-Yao adversary that defines ϕ). Once ϕ is created, it requests the two values of the nonce $n^{a_i,j,s}$ and receives from the experiment n_b and n_{1-b}. Then it computes $m_b = \tau(\phi^{\mathsf{c}}, n_b)$ for $b = 0, 1$ and $m = \pi(\phi^{\mathsf{c}})$, and retrieves b by comparing m with m_0 and m_1.

5 Decidability of Symbolic Secrecy

In this section, we show that our notion of secrecy is decidable. We present an NP-procedure that decides nonce non-secrecy for the case of a bounded number of sessions (that is, adversaries are allowed only a fixed number of $\mathbf{new}$ queries)[6]

Without loss of generality, we assume that all of the $\mathbf{new}$ queries are performed at the beginning of the execution. Our decision procedure starts by guessing the sequence of these requests together with the identities of the agents involved. Then, the procedure guesses an interleaving for the execution. Using standard techniques [15], such executions can be translated to constraint systems. We recall their definition:

Definition 5. *A* constraint system C *is a finite set of expressions* $T_i \Vdash \mathit{tt}$ *or* $T_i \Vdash u_i$*, where* T_i *is a non empty set of terms,* tt *is a special symbol that represents an always deducible term, and (for* $1 \leq i \leq n$*)* u_i *is a term such that:*

- $T_i \subseteq T_{i+1}$*, for all* $1 \leq i \leq n-1$*;*
- *if* $x \in \mathrm{var}(T_i)$ *then* $\exists j < i$ *such that* $T_j = \min\{T \mid T \Vdash u \in C, x \in \mathrm{var}(u)\}$ *(for the inclusion relation) and* $T_j \subsetneq T_i$.

The left-hand side *(*right-hand side*) of a constraint* $T \Vdash u$ *is* T *(respectively* u*). The* left-hand side *of a constraint system* C*, (for which we write* $lhs(C)$*), is the maximal set of messages* T_n*. By* $\perp$ *we denote the unsatisfiable system.*

The left-hand side of a constraint represents the messages already sent on the network, while the right-hand side represents the message expected by an agent in order to perform the next protocol step. A *solution* of a constraint system C is a ground substitution σ such that $T\sigma \vdash_{\mathsf{Rand}_{adv}} u\sigma$ for any $T \Vdash u \in C$. We say that C *preserves nonce secrecy* of n if there does not exist a solution σ of C such that n occurs in $\mathsf{Pat}_n(\overline{lhs(C)\sigma})$.

The transformation of protocols into constraint systems yields systems that are well-formed. A constraint system E is *well-formed* if 1) any subterm of E of the form $\mathsf{dk}(t')$ is such that t' is an agent identity and 2) any subterm of E of the form $\{t_1\}^r_{t_2}$ is such that $r \in \mathsf{Rand}$ and $r \notin \mathsf{Rand}_{adv}$. The following theorem states that our notion of nonce secrecy (Section 3) is decidable for a bounded number of sessions.

Theorem 2. *The following problem is co-NP complete:*
Given: *A well-formed constraint system* C *and a nonce* n*.*
Decide: *Does* C *preserve the nonce secrecy of* n*?*

[6] As for the standard deducibility-based notions, nonce secrecy is undecidable for an unbounded number of sessions.

The decision procedure for nonce secrecy preservation works as follows. First, given an arbitrary constraint system we reduce it to a *solved* system using non-deterministic transformation rules similar to those in [8]. A constraint system is *solved* if it is different from $\perp$ and each of its constraints are of the form $T \Vdash tt$ or $T \Vdash x$ where x is a variable. Second, we check whether n occurs in $\mathsf{Pat}_n(\overline{lhs(C)})$. If not, we check whether C can further be simplified into a solved form that does not preserve nonce secrecy, and so on. Note that although for standard deducibility-based notions decision procedures can stop as soon as the constraint system has been transformed into solved form, for our secrecy notion further transformations might be necessary. NP-hardness is proved analogously to the case of standard deducibility-based notions [16].

References

1. M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *IFIP TCS 2000*, volume 1872 of *LNCS*, pages 3–22, August 2000.
2. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, 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*, volume 3576 of *LNCS*, 2005.
3. M. Backes and I. Christian Jacobi. Cryptographically sound and machine-assisted verification of security protocols. In *STACS 2003*, pages 675–686, 2003.
4. M. Backes and B. Pfitzmann. Relating cryptographic und symbolic key secrecy. In *Proc. 26th IEEE Symposium on Security and Privacy (SSP'05)*, pages 171–182, 2005.
5. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Crypto'93*, volume 773 of *LNCS*, pages 232–249, 1993.
6. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96, 2001.
7. R. Canetti and J. Herzog. Soundness of formal encryption in the presence of active adversaries. In *TCC 2006)*, LNCS, 2006. To appear.
8. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS 2003*, pages 271–280, 2003.
9. V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. Research Report 2006/218, Cryptology ePrint Archive, June 2006. 31 pages.
10. V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *ESOP 2005*, volume 3444 of *LNCS*, pages 157–171, 2005.
11. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
12. P. Gupta and V. Shmatikov. Towards computationally sound symbolic analysis of key exchange protocols. In *FMSE 2005*, pages 23–32, 2005.
13. R. Janvier, Y. Lakhnech, and L. Mazaré. Computational soundness of symbolic analysis for protocols using hash functions. In *ICS'06*, 2006. To appear.
14. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *TCC 2004*, volume 2951 of *LNCS*, pages 133–151, 2004.
15. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS 2001*, pages 166–175, 2001.
16. M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete. *Theoretical Computer Science*, 299:451–475, April 2003.