How to prove security of communication protocols? A discussion on the soundness of formal models w.r.t. computational ones.*

Hubert Comon-Lundh[1] and Véronique Cortier[2]

1 **ENS Cachan& CNRS & INRIA Saclay Ile-de-France**
2 **LORIA, CNRS**

Abstract

Security protocols are short programs that aim at securing communication over a public network. Their design is known to be error-prone with flaws found years later. That is why they deserve a careful security analysis, with rigorous proofs. Two main lines of research have been (independently) developed to analyse the security of protocols. On the one hand, formal methods provide with symbolic models and often automatic proofs. On the other hand, cryptographic models propose a tighter modeling but proofs are more difficult to write and to check. An approach developed during the last decade consists in bridging the two approaches, showing that symbolic models are *sound* w.r.t. symbolic ones, yielding strong security guarantees using automatic tools. These results have been developed for several cryptographic primitives (e.g. symmetric and asymmetric encryption, signatures, hash) and security properties.

While proving soundness of symbolic models is a very promising approach, several technical details are often not satisfactory. Focusing on symmetric encryption, we describe the difficulties and limitations of the available results.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Verification, security, cryptography

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.29

1 Introduction

Security protocols aim at securing communications over public networks. They are typically designed for bank transfers over the Internet, establishing private channels, or authenticating remote sites. They are also used in more recent applications such as e-voting procedures. Depending on the application, they are supposed to ensure security properties such as confidentiality, privacy or authentication, even when the network is (at least partially) controlled by malicious users, who may intercept, forge and send new messages. While the specification of such protocols is usually short and rather natural, designing a secure protocol is notoriously difficult and flaws may be found several years later. A famous example is the "man-in-the-middle" attack found by G. Lowe against the Needham-Schroder public key protocol [41]. A more recent example is the flaw discovered in Gmail (and now fixed) by Armando *et. al.* [9].

* This work has been supported by the ANR-07-SeSur-002 AVOTÉ project.

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 29–44

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

During the two last decades, formal methods have demonstrated their usefulness when designing and analyzing security protocols. They indeed provide with rigorous frameworks and techniques that allow to discover new flaws. For instance, the two previously mentioned flaws have been discovered while trying to prove the security of the protocol in a formal setting. Following the seminal work of Dolev and Yao [33], many techniques have been developed for analysing the security of protocols, often automatically. For example, the AVISPA platform [8] and the ProVerif tool [20] are both efficient and practical tools for automatically proving security properties or finding bugs if any. The security of protocols is undecidable in general [34]. Checking the secrecy and authentication-like properties is however NP-complete when the number of sessions is fixed [44]. Several extensions have been designed, considering more security properties or more security primitives [2, 25, 28, 24, 37]. Bruno Blanchet has developed an (incomplete) procedure based on clause resolution [19] for analyzing protocols for an unbounded number of sessions. All these approaches rely on a common representation for messages: they are symbolically modeled by *terms* where each function symbol represents a cryptographic primitive, some of their algebraic properties being reflected in an equational theory. Then protocols are modeled using or adapting existing frameworks such as fragments of logic, process algebras or constraint systems.

While the symbolic approaches were successful in finding attacks, the security proofs in these models are questionable, because of the level of abstraction: most cryptographic details are ignored. This might be a problem: for instance, it is shown in [45] that a protocol can be proved in a formal, symbolic, model, while there is an attack, that also exploits some finer details of the actual implementation of the encryption scheme. In contrast, cryptographic models are more accurate: the security of protocols is based on the security of the underlying primitives, which in turn is proved assuming the hardness of various computational tasks such as factoring or computing discrete logarithms. The messages are bitstrings. The proofs in the computational model imply strong guarantees (security holds in the presence of an arbitrary probabilistic polynomial-time adversary). However, security reductions for even moderately-sized protocols become extremely long, difficult, and tedious. Recently, a significant research effort [6, 43, 13, 15, 11, 26] has been directed towards bridging the gap between the symbolic and the cryptographic approaches. Such *soundness* results typically show that, under reasonable cryptographic assumptions such as IND-CCA2 for the encryption scheme, proofs in symbolic models directly imply proofs in the more detailed cryptographic models. These approaches are very promising: they allow to reconcile two distinct and independently developed views for modeling and analysing security protocols. Second and more importantly, they allow to obtain the best of the two worlds: strong security guarantees through the simpler symbolic models, that are amenable to automatic proofs.

However, such soundness results also assume many other properties regarding the implementation or even regarding the key infrastructure. In this paper, we discuss these usually under-looked assumptions, pointing the limitations of current results. In particular, we provide with several protocols (counter) examples, for which IND-CCA2 does not imply the security, as soon as a malicious user may chose its own keys at its will. These examples show that standard symbolic models are not sound w.r.t. cryptographic ones when using symmetric encryption. We also discuss how to symbolically represent the length of messages and what are the implications on the implementation. All these examples will be discussed within the the applied-pi calculus [3], but the counter-examples do not depend on this particular process algebra: the discussion will stay at a rather informal level and can be understood without familiarity with the applied-pi calculus.

Related work. Many soundness results have been established in various settings. We discuss some of them in Section 3. Fewer works are dedicated to the limitations. Backes and Pfitzmann have shown that primitives such as Exclusive Or or hash functions cannot be soundly abstracted in their simulatability library [14]. This is related to the impossibility of constructing some universally composable primitives [40]. This witnesses the difficulty of designing sound and accurate models for some primitives. [1] compares CryptoVerif [21], an automatic tool designed for performing proofs directly in the cryptographic model, and the use of soundness results, emphasizing the current limitations of the latter.

2 Setting

We recall here briefly part of the syntax and the operational semantics of the applied π-calculus of [3]. We are going to use a small fragment of this calculus for the formal definition of the protocols.

2.1 Syntax

In any symbolic model for security protocols, messages are modeled by *terms*, which are built on a set of function symbols Σ, that represent the cryptographic primitives (e.g. encryption, pairing, decryption). Given an infinite set $\mathcal{N}$ of *names* and an infinite set $\mathcal{X}$ of *variables*, $\mathcal{T}(\mathcal{N}, \mathcal{X})$ is the set of *terms*:

s, t, u ::=	terms	
x, y, z	variable	
a, b, c, k, n, r	name	
$f(s_1, \ldots, s_k)$	function application	$f \in \Sigma$ and k is the arity of f.

Terms represent messages and names stand for (randomly) generated data. We assume the existence of a length function l, which is a Σ-morphism from $T(\mathcal{N})$ to $\mathbb{N}$.

In what follows, we will consider symmetric encryption and pairing. Let Σ_0 consist of the binary pairing $< \cdot, \cdot >$, the two associated projections π_1, π_2, the binary decryption dec and the ternary symbol $\{\cdot\}_\cdot^\cdot$ for symmetric encryption: $\{x\}_k^r$ stands for the encryption of x with the key k and the random r. Σ_0 also contains constants, in particular a constant 0^l of length l for every l.

The syntax of processes is displayed in Figure 1. In what follows, we restrict ourselves to processes with public channels: there is no restriction on name channel. We assume a set $\mathcal{P}$ of predicate symbols with an arity. Such a definition, as well as its operational semantics coincides with [3], except for one minor point introduced in [26]: we consider conditionals with arbitrary predicates. This leaves some flexibility in modeling various levels of assumptions on the cryptographic primitives.

In what follows, we may use expressions of the form $\mathsf{let}\ \ldots\ \mathsf{in}\ \ldots$ as a syntactic sugar to help readability.

2.2 Operational semantics

We briefly recall the operational semantics of the applied pi-calculus (see [3, 26] for details). E is a set of equations on the signature Σ, defining an equivalence relation $=_E$ on $\mathcal{T}(\mathcal{N})$, which is closed under context. $=_E$ is meant to capture several representations of the same message. This yields a quotient algebra $\mathcal{T}(\mathcal{N})/=_E$, representing the messages. Predicate symbols are interpreted as relations over $\mathcal{T}(\mathcal{N})/=_E$. This yields a structure $\mathcal{M}$.

Φ_1, Φ_2 ::=	conditions
$p(s_1, \ldots, s_n)$	predicate application
$\Phi_1 \wedge \Phi_2$	conjunction
P, Q, R ::=	processes
$c(x).P$	input
$\overline{c}(s).P$	output
$\mathbf{0}$	terminated process
$P \parallel Q$	parallel composition
$!P$	replication
$(\nu\alpha)P$	restriction
`if` Φ `then` P `else` Q	conditional

Figure 1 Syntax of processes

In what follows, we will consider the equational theory E_0 on Σ_0 defined by the equations corresponding to encryption and pairing:

$$\mathsf{dec}(\{x\}^z_y, y) = x \qquad \pi_1(< x, y >) = x \qquad \pi_2(< x, y >) = y$$

These equations can be oriented, yielding a convergent rewrite system: every term s has a unique normal form $s\downarrow$.

We also consider the following predicates introduced in [26].

- M checks that a term is well formed. Formally, M is unary and holds on a (ground) term s iff $s\downarrow$ does not contain any projection nor decryption symbols and for any $\{u\}^r_v$ subterm of s, v and r must be names. This forbids compound keys for instance.
- EQ checks the equality of well-formed terms. EQ is binary and holds on s, t iff $M(s), M(t)$ and $s\downarrow = t\downarrow$: this is a strict interpretation of equality.
- $\mathsf{P}_{\mathsf{samekey}}$ is binary and holds on ciphertexts using the same encryption key: $\mathcal{M} \models \mathsf{P}_{\mathsf{samekey}}(s, t)$ iff $\exists k, u, v, r, r'.EQ(s, \{u\}^r_k) \wedge EQ(t, \{v\}^{r'}_k)$.
- EL is binary and holds on s, t iff $M(s), M(t)$ and s, t have the same length.

▶ **Example 2.1.** The Wide Mouth Frog [22] is a simple protocol where a server transmits a session key K_{ab} from an agent A to an agent B. This toy example is also used in [1] as a case study for both CryptoVerif and soundness techniques. For the sake of illustration, we propose here a flawed version of this protocol.

$$\begin{aligned} A \to S &: A, B, \{N_a, K_{ab}\}_{K_{as}} \\ S \to B &: A, \{N_s, K_{ab}\}_{K_{bs}} \end{aligned}$$

The server is assumed to share long-term secret keys with each agent. For example, K_{as} denotes the long-term key between A and the server. In this protocol, the agent A establishes a freshly generated key K_{ab} with B, using the server for securely transmitting the key to B.

A session l_a of role A played by agent a with key k_{as} can be modeled by the process

$$A(a, b, k_{as}, l_a) \stackrel{\text{def}}{=} (\nu r, n_a)\, \overline{\mathsf{c_{out}}}(< l_a, < a, < b, \{< n_a, k_{ab} >\}^r_{k_{as}} >>>) \cdot \mathbf{0}$$

Similarly a session of role S played for agents a, b with corresponding keys k_{as} and k_{bs}, can be modeled by

$$S(a,b,k_{as},k_{bs},l_s) \stackrel{\text{def}}{=} (\nu n_s, r)\ \mathsf{c_{in}}(x).\ \texttt{if}\ EQ(\pi_1(x), l_s)\ \texttt{then}\ \mathsf{let}\ y = \pi_2(\mathsf{dec}(\pi_2(\pi_2(\pi_2(x))), k_{as}))\ \mathsf{in}$$
$$\texttt{if}\ \ \pi_1(\pi_2(x)) = a\ \ \wedge\ \ \pi_1(\pi_2(\pi_2(x)) = b\ \ \wedge\ \ M(y)\ \texttt{then}$$
$$\overline{\mathsf{c_{out}}}(< l_s, < a, \{< n_s, y >\}^r_{k_{bs}} >>) \cdot \mathbf{0}$$
$$\texttt{else}\ \overline{\mathsf{c_{out}}}(\bot) \cdot \mathbf{0}\ \texttt{else}\ \overline{\mathsf{c_{out}}}(\bot) \cdot \mathbf{0}$$

where l_s is the session identifier of the process.

Then an unbounded number of sessions of this protocol, in which A plays a (with b) and s plays S (with a, b and also with b, c) can be represented by the following process

$$P_{\mathsf{ex}} = \nu(k_{as}, k_{bs})\ \ (\ \ !((\nu k_{ab}, l_a)\overline{\mathsf{c_{out}}}(l_a).A(a,b,k_{as},l_a,r))$$
$$\|\ !((\nu l_s)\overline{\mathsf{c_{out}}}(l_s).S(a,b,k_{as},k_{bs},l_s))\ \|\ !((\nu l_s)\overline{\mathsf{c_{out}}}(l_s).S(a,c,k_{as},k_{cs},l_s))\ \ )$$

To reflect the fact that c is a dishonest identity, its long-term key k_{cs} shared with the server does not appear under a restriction and is therefore known to an attacker.

The environment is modeled through *evaluation context*, that is a process $C = (\nu\overline{\alpha})([\cdot] \| P)$ where P is a process. We write $C[Q]$ for $(\nu\overline{\alpha})(Q \parallel P)$. A context (resp. a process) C is *closed* when it has no free variables (there might be free names).

Possible evolutions of processes are captured by the relation $\rightarrow$, which is the smallest relation, compatible with the process algebra and such that:

$$\begin{array}{rrcll}
\text{(Com)} & c(x).P \parallel \overline{c}(s).Q & \rightarrow & \{x \mapsto s\} \parallel P \parallel Q & \\
\text{(Cond1)} & \texttt{if}\ \Phi\ \texttt{then}\ P\ \texttt{else}\ Q & \rightarrow & P & \text{if } \mathcal{M} \models \Phi \\
\text{(Cond2)} & \texttt{if}\ \Phi\ \texttt{then}\ P\ \texttt{else}\ Q & \rightarrow & Q & \text{if } \mathcal{M} \not\models \Phi
\end{array}$$

$\stackrel{*}{\rightarrow}$ is the smallest transitive relation on processes containing $\rightarrow$ and some structural equivalence (e.g. reflecting the associativity and commutativity of the composition operator $\|$) and closed by application of contexts.

▶ **Example 2.2.** Continuing Example 2.1, we show an attack, that allows an attacker to learn k_{ab}, the key exchanged between a and b. Indeed, an attacker can listen to the first message $< l_a, < a, < b, \{< n_a, k_{ab} >\}^{r_1}_{k_{as}} >>>$ and replace it with $< l_a, < a, < c, \{< n_a, k_{ab} >\}^{r_1}_{k_{as}} >>>$. Thus the server would think that a wishes to transmit her key k_{ab} to c. Therefore it would reply with $< a, \{< n_s, k_{ab} >\}^{r_2}_{k_{cs}} >$. The attacker can then very easily decrypt the message and learn K_{ab}. This attack corresponds to the context

$$C_{\mathsf{attack}} \stackrel{\text{def}}{=} [\cdot] \parallel \mathsf{c_{out}}(x_{l_a}).\mathsf{c_{out}}(x_{l_s}).\mathsf{c_{out}}(x_{m_a}).\ \text{//listens to sessions ids and the first message}$$
$$\mathsf{let}\ y = \pi_2(\pi_2(x_{m_a}))\ \mathsf{in}$$
$$\overline{\mathsf{c_{in}}}(< x_{l_s}, < a, < c, y >>>).\ \text{//replays the message, with } b \text{ replaced by } c$$
$$\mathsf{c_{out}}(x_{m_s}).\ \text{//listens to the server's reply}$$
$$\mathsf{let}\ y' = \mathsf{dec}(\pi_2(\pi_2(x_{m_s})), k_{cs})\ \mathsf{in}\ \overline{\mathsf{c_{in}}}(\pi_2(y')).0\ \text{//outputs the secret}$$

Then the attack is reflected by the transitions $C_{\mathsf{attack}}[P_{\mathsf{ex}}] \stackrel{*}{\rightarrow} \overline{\mathsf{c_{out}}}(k_{ab}) \parallel Q$ for some process Q, yielding the publication of the confidential key k_{ab}.

2.3 Observational equivalence

Observational equivalence is useful to describe many properties such as confidentiality or authentication as exemplified in [5]. It is also crucial for specifying privacy related properties as needed in the context of electronic voting protocols [32].

▶ **Definition 2.3.** The *observational equivalence relation* $\sim_o$ is the largest symmetric relation $\mathcal{S}$ on closed extended processes such that $A\mathcal{S}B$ implies:

1. if, for some context C, term s and process A',
   $A \xrightarrow{*} C[\overline{c}(s) \cdot A']$ then for some context C', term s' and process B', $B \xrightarrow{*} C'[\overline{c}(s') \cdot B']$.
2. if $A \xrightarrow{*} A'$ then, for some B', $B \xrightarrow{*} B'$ and $A'\mathcal{S}B'$
3. $C[A]\mathcal{S}C[B]$ for all closed evaluation contexts C

▶ **Example 2.4** (Group signature)**.** The security of group signature has been defined in [7]. It intuitively ensures that an attacker should not be able to distinguish two signatures performed with two distinct identities when they belong to the same group. It can be modeled as observational equivalence as follows. Let $P(x, i)$ be the protocol for signing message x with identity i. Let $P_0 = c(y).P(\pi_1(y), \pi_1(\pi_2(y)))$ and $P_1 = c(y).P(\pi_1(y), \pi_2(\pi_2(y)))$. Intuitively, the adversary will send $< m, < i_0, i_1 >>$ where m is a message to be signed and i_0, i_1 are two identities. P_0 signs m with i_0 while P_1 signs m with i_1. Then P preserves anonymity iff $P_0 \sim_o P_1$.

2.4 Computational interpretation

We assume given an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ where $\mathcal{G}$ is the generating function for keys, $\mathcal{E}$ is the encryption function and $\mathcal{D}$ the decryption function. We also assume given a pairing function. The encryption, decryption, and pairing functions and their corresponding projectors form respectively the computational interpretation of the symbols $\{\cdot\}, \mathsf{dec}, <, >, \pi_1, \pi_2$. We assume that the decryption and projection functions return an error message $\perp$ when they fail. Then, given an interpretation τ of names as bitstrings, $[\![\cdot]\!]_\tau$ is the (unique) Σ-morphism extending τ to $T(\mathcal{N})$; $[\![t]\!]_\tau$ is the *computational interpretation* of t. When τ is randomly drawn, according to a distribution that depends on a security parameter η, we may write $[\![t]\!]_\eta$ for the corresponding distribution and $[\![t]\!]$ for the corresponding family of distributions. Then here are possible interpretations of the predicates:

- $[\![M]\!]$ is the set of bitstrings, which are distinct from $\perp$. Intuitively $[\![M]\!]$ implements M if the encryption scheme is *confusion-free* (a consequence of INT-CTXT [42]).
- $[\![EQ]\!]$ is the set of pairs of identical bitstrings, which are distinct from $\perp$. It is an implementation of EQ as soon as $[\![M]\!]$ implements M.
- $[\![\mathsf{P}_{\mathsf{samekey}}]\!]$ is the set of pairs of bitstrings that have the same encryption tag.
- $[\![EL]\!]$ is the set of pairs of bitstrings of same length.

Processes can also be interpreted as communicating Turing machines. Such machines have been introduced in [16, 38] for modeling communicating systems. They are probabilistic Turing machines with input/output tapes. Those tapes are intuitively used for reading and sending messages. We will not describe them here and we refer to [26] for more details.

Now, given a process P without replication, one can interpret it as a (polynomial time) communicating Turing machine. The computational interpretation of P is denoted by $[\![P]\!]$ and is intuitively defined by applying the computational counterpart of each function and

predicate symbols. Then the replicated process $!P$ can also be interpreted by letting the adversary play with as many copies of $[\![P]\!]$ as he wants.

Indistinguishability. In computational models, security properties are often stated as *indistinguishability* of games. Two families of machines are indistinguishable if an adversary cannot tell them apart except with non negligible probability.

A function $f : \mathbb{N} \to \mathbb{N}$ is *negligible* if, for every polynomial P, $\exists N \in \mathbb{N}, \forall \eta > N, f(\eta) < \frac{1}{P(\eta)}$. We write $\mathbf{Pr}\{x : P(x)\}$ the probability of event $P(x)$ when the sample x is drawn according to an appropriate distribution (the key distribution or the uniform distribution; this is kept implicit).

▶ **Definition 2.5.** Two environments $\mathcal{F}$ and $\mathcal{F}'$ are *indistinguishable*, denoted by $\mathcal{F} \approx \mathcal{F}'$, if, for every polynomial time communicating Turing Machine A (i.e. for any attacker),

$$|\mathbf{Pr}\{\overline{r}, r : (\mathcal{F}(\overline{r}) \parallel A(r))(0^\eta) = 1\} - \mathbf{Pr}\{\overline{r}, r : (\mathcal{F}'(\overline{r}) \parallel A(r))(0^\eta) = 1\}|$$

is negligible. $\overline{r}$ is the sequence of random inputs of the machines in $\mathcal{F}$ (resp. $\mathcal{F}'$), including keys. r is the random input of the attacker.

For example, anonymity of group signatures as discussed in Example 2.4 is defined in [7] through the following game: the adversary chooses a message m and two identities i_0 and i_1. Then in $\mathcal{F}_0$, the machines sign m with identity i_0 while in $\mathcal{F}_1$, the machines sign m with identity i_1. Then the anonymity is defined by $\mathcal{F}_0 \approx \mathcal{F}_1$. Note that, for $i = 1, 2$, $\mathcal{F}_i$ can be defined as $[\![P_i]\!]$, implementation of the process P_i of the Example 2.4.

More generally, security properties can be defined by specifying the ideal behavior P_{ideal} of a protocol P and requiring that the two protocols are indistinguishable. For example, in [4], authenticity is defined through the specification of a process where the party B magically received the message sent by the party A. This process should be indistinguishable from the initial one.

3 Soundness results

Computational models are much more detailed than symbolic ones. In particular, the adversary is very general as it can be any (polynomial) communicating Turing machine. Despite the important difference between symbolic and computational models, it is possible to show that symbolic models are *sound* w.r.t. computational ones.

3.1 A brief survey

There is a huge amount of work on simulatability/universal composability, especially the work of Backes *et. al.* and Canetti [23, 13, 15, 11]. When the ideal functionality is the symbolic version of the protocol, then the black-box simulatability implies the trace mapping property [11], therefore showing a safe abstraction. Such results can be applied to trace properties such as authentication but not to indistinguishability. In a recent paper [12], Backes and Unruh show that the whole applied-pi calculus can be embedded in CoSP, a framework in which they prove soundness of public-key encryption and digital signatures, again for trace properties.

Besides [26], which we discuss in more detail below, one of the only results that prove soundness for indistinguishability properties is [39], for some specific properties (see the end of section 4 for more details).

In a series of papers starting with Micciancio and Warinschi [43] and continued with e.g. [31, 36], the authors show trace mapping properties: for some selected primitives (public-key encryption and signatures in the above-cited papers) they show that a computational trace is an instance of a symbolic trace, with overwhelming probability. But, again, this does not show that indistinguishability properties can be soundly abstracted, except for the special case of computational secrecy that can be handled in [31] and also in [29] for hash functions in the random oracle model.

We refer to [30] for a more complete survey of soundness results.

3.2 Observational equivalence implies indistinguishability

The main result of [26] consists in establishing that observational equivalence implies indistinguishability:

▶ **Theorem 3.1.** *Let P_1 and P_2 be two simple processes such that each P_i admits a key hierarchy. Assume that the encryption scheme is joint IND-CPA and INT-CTXT. Then $P_1 \sim_o P_2$ implies that $[\![P_1]\!] \approx [\![P_2]\!]$.*

This result assumes some hypotheses, some of which are explicitly stated above and informally discussed below (the reader is referred to [26] for the full details). There are additional assumptions, that are discussed in more details in the next section.

Simple processes are a fragment (introduced in [26]) of the applied-pi calculus. It intuitively consists of parallel composition of (possibly replicated) *basic processes*, that do not involve replication, parallel composition or else branches. Simple processes capture most protocols without else branch, for an unbounded number of sessions. For example, the process P_{ex} introduced in Example 2.1 is a basic process.

The most annoying restriction is the absence of conditional branching. Ongoing works should overcome this limitation, at the price of some additional computational assumptions. But the extension to the full applied π-calculus is really challenging, because of possible restrictions on channel names. Such restrictions indeed allow "private" computations, of which an attacker only observes the computing time (which is not part of the model).

Key hierarchy ensures that no key cycles can be produced on honest keys, even with the interaction of the adversary. This hypothesis is needed because current security assumptions such as IND-CPA do not support key cycles (most encryption schemes are not provably secure in the presence of key cycles). In [26], it is assumed that there exists a strict ordering on key such that no key encrypts a greater key.

Checking such conditions, for any possible interaction with the attacker, is in general undecidable, though a proof can be found in many practical cases. (And it becomes decidable when there is no replication [27]).

No dynamic corruption assumes that keys are either immediately revealed (e.g. corrupted keys) or remain secret. Showing a soundness result in case of dynamic key corruption is a challenging open question, that might require stronger assumptions on the encryption scheme.

IND-CPA and INT-CTXT are standard security assumptions on encryption schemes. The IND-CPA assumption intuitively ensures that an attacker cannot distinguish the encryption of any message with an encryption of zeros of the same length. INT-CTXT ensures that an

adversary cannot produce a valid ciphertext without having the encryption key. IND-CPA and INT-CTXT are standard security assumptions [18].

4 Current limitations

We discuss in this section the additional assumptions of Theorem 3.1. Let us emphasize that such assumptions are not specific to this result: other soundness results have similar restrictions and/or provide with a weaker result.

4.1 Parsing

Parsing the bitstrings into terms is used in the proof of the soundness results; this function has actually to computable in polynomial time, since this is part the construction of a Turing machine used in a reduction. Therefore, Theorem 3.1 assumes that the pairing, key generation and encryption functions add a typing tag (which can be changed by the attacker), that indicates which operator has been used and further includes which key is used in case of encryption. This can be achieved by assuming that a symmetric key k consists of two parts (k_1, k_2), k_1 being generated by some standard key generation algorithm and k_2 selected at random. Then one encrypts with k_1 and tags the ciphertext with k_2.

These parsing assumptions are easy to implement and do not restrict the computational power of an adversary. Adding tags can only add more security to the protocol. However, current implementations of protocols do not follow these typing hypotheses, in particular regarding the encryption. Therefore Theorem 3.1 requires a reasonable but non standard and slightly heavy implementation in order to be applicable.

The parsing assumption might be not necessary. There are ongoing works trying to drop it.

4.2 Length function

As explained in Section 2, Theorem 3.1 assumes the existence of a length function l, which is a morphism from $T(\mathcal{N})$ to $\mathbb{N}$. This length function is needed to distinguish between ciphertexts of different lengths. For example, the two ciphertext $\{< n_1, n_2 >\}_k$ and $\{n_1\}_k$ should be distinguishable while $\{< n_1, n_2 >\}_k$ and $\{< n_1, n_1 >\}_k$ should not. There are however cases where it is unclear whether the ciphertexts should be distinguishable or not:

$$\nu k.\overline{\mathsf{c_{out}}}(\{< n_1, n_2 >\}_k) \stackrel{?}{\sim}_o \nu k.\overline{\mathsf{c_{out}}}(\{\{n_1\}_k\}_k).$$

Whether these two ciphertexts are distinguishable typically depends on the implementation and the security parameter: their implementation may (or not) yield bitstrings of equal length. However, for a soundness result, we need to distinguish (or not) these ciphertexts *independently* of the implementation.
In other words, if we let length be the length of a bitstring, we (roughly) need the equivalence:

$$l(s) = l(t) \qquad \text{iff} \qquad \forall \tau.\ \mathsf{length}([\![s]\!]_\tau) = \mathsf{length}([\![t]\!]_\tau)$$

This requires some length-regularity of the cryptographic primitives. But even more, this requires length to be homogenous w.r.t. the security parameter η. To see this, consider the case where length is an affine morphism:

$$\begin{aligned} \mathsf{length}([\![\{t_1\}_k^r]\!]_\tau) &= \mathsf{length}([\![t_1]\!]_\tau) + \gamma \times \eta + \alpha \\ \mathsf{length}([\![< t_1, t_2 >]\!]_\tau) &= \mathsf{length}([\![t_1]\!]_\tau) + \mathsf{length}([\![t_2]\!]_\tau) + \beta \\ \mathsf{length}(\tau(n)) &= \delta \times \eta \end{aligned}$$

where β cannot be null since some bits are needed to mark the separators between the two strings $[\![t_1]\!]$ and $[\![t_2]\!]$. (Also, $\gamma, \delta > 0$.)

Now, if we consider an arbitrary term t, $\mathsf{length}([\![t]\!]_\tau) = n_1 \times \gamma \times \eta + n_1 \times \alpha + n_2 \times \beta + n_3 \times \delta \times \eta$. $\frac{\mathsf{length}([\![s]\!]_\tau)}{\mathsf{length}([\![t]\!]_\tau)}$ must be independent of η, hence there must exist $\alpha', \beta' \in \mathbb{N}, \beta' > 0$ such that $\alpha = \alpha' \times \eta$ and $\beta = \beta' \times \eta$.

This implies in particular that the pairing function always adds η bits (or a greater multiple of η) to the bitstrings. Similarly, a ciphertext should be the size of its plaintext plus a number of bits which is the a multiple of η.

While it is possible to design an implementation that achieves such constraints, this is not always the case in practice and it may yield a heavy implementation, in particular in conjunction with the parsing assumptions. Moreover, on the symbolic side, adding a length function raises non trivial decidability issues.

Adding a symbolic length function is needed for proving indistinguishability as illustrated by the former examples. It is worth noticing that several soundness results such as [13, 15, 31, 29, 12] do not need to consider a length function. The reason is that they focus on *trace properties* such as authentication but they cannot considered indistinguishability-based properties (except computational secrecy for some of them).

4.3 Dishonest keys

Theorem 3.1 assumes the adversary only uses correctly generated keys. In particular, the adversary cannot choose his keys at its will, depending on the observed messages. The parties are supposed to check that the keys they are using have been properly generated. The assumption could be achieved by assuming that keys are provided by a trusted server that properly generates keys together with a certificate. Then when a party receives a key, it would check that it comes with a valid certificate, guaranteeing that the key has been issued by the server. Of course, the adversary could obtain from the server as many valid keys as he wants.

However, this assumption is strong compared to usual implementation of symmetric keys and it is probably the less realistic assumptions among those needed for Theorem 3.1. We discuss alternative assumptions at the end of this section. It is worth noticing that in all soundness results for asymmetric encryption, it is also assumed that the adversary only uses correctly generated keys. Such an assumption is more realistic in an asymmetric setting as a server could certify public keys. However, this does not reflect most current implementations for public key infrastructure, where agents generate their keys on their own.

We now explain why such an assumption is needed to obtain soundness.The intuitive reason is that IND-CCA does not provide any guarantee on the encryption scheme when keys are dishonestly generated.

▶ **Example 4.1.** Consider the following protocol. A sends out a message of the form $\{c\}_{K_{ab}}$ where c is a constant. This can be formally represented by the process

$$A = (\nu r)\overline{\mathsf{c_{out}}}(< c, \{c\}^r_{k_{ab}} >).\mathbf{0}$$

Then B expects a key y and a message of the form $\{\{b\}_y\}_{K_{ab}}$ where b is the identity of B, in which case, it sends out a secret s (or goes in a bad state).

$$\begin{array}{rll} & A & \to B: \quad (\nu r)\ c, \{c\}^r_{K_{ab}} \\ B : k, \{\{b\}_k\}_{K_{ab}} & & \to A: \quad s \end{array}$$

This can be formally modeled by the process

$$B = \mathsf{c_{in}}(z).\ \texttt{if}\ \ EQ(b, \mathsf{dec}(\mathsf{dec}(\pi_2(z), k_{ab}), \pi_1(z))\ \ \texttt{then}\ \overline{\mathsf{c_{out}}}(s)\ \ \texttt{else}\ \mathbf{0}$$

Then symbolically, the process $(\nu k_{ab})(\nu s)A \| B$ never emits s. However, a computational adversary can forge a key k such that any bitstring can be successfully decrypted to b using k. In particular, at the computational level, we have $\mathsf{dec}(c, k) = b$. Thus by sending $< k, \{c\}^r_{k_{ab}} >$ to B, the adversary would obtain the secret s.

This is due to the fact that security of encryption schemes only provides guarantees on *properly generated* keys. More precisely, given an IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, it is easy to build another IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$, which allows to mount the previous attack: consider $\mathcal{G}' = 0 \cdot \mathcal{G}$ (all honest keys begin with the bit 0), $\mathcal{E}'(m, i.k) = \mathcal{E}(m, k)$ for $i \in \{0, 1\}$ and $\mathcal{D}'$ defined as follows:

- $\mathcal{D}'(c, k) = \mathcal{D}(c, k')$ if $k = 0 \cdot k'$
- $\mathcal{D}'(c, k) = k'$ if $k = 1 \cdot k'$

It is easy to check that $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ remains IND-CCA2 and allows an adversary to choose a key that decrypts to anything he wants.

To capture this kind of computational attacks, an idea (from M. Backes [10]) is to enrich the symbolic setting with a rule that allows an intruder, given a ciphertext c and a message m, to forge a key such that c decrypts to m. This could be modeled e.g. by adding a functional symbol $\mathsf{fakekey}$ of arity 2 together with the equation

$$\mathsf{dec}(x, \mathsf{fakekey}(x, y)) = y$$

Going back to Example 4.1, this would allow a symbolic intruder to send the message $< \mathsf{fakekey}(c, b), \{c\}^r_{k_{ab}} >$ to the B process and the process $(\nu k_{ab})(\nu s)A \| B$ would emit s.

This solution appears however to be insufficient to cover the next examples.

▶ **Example 4.2** (hidden cyphertext)**.** The same kind of attacks can be mounted even when the ciphertext is unknown to the adversary. We consider a protocol where A sends $<< A, k >, \{\{k'\}^{r'}_k\}^r_{k_{ab}} >$ where k and k' are freshly generated keys. B recovers k' and sends it encrypted with k_{ab}. In case A receives her name encrypted with k_{ab}, A emits a secret s.

$$\begin{array}{rll} A & \to B: & (\nu k, k', r_1, r_2)\ A, k, \{\{k'\}^{r_1}_k\}^{r_2}_{K_{ab}} \\ B & \to A: & (\nu r_3)\ \{k'\}^{r_3}_{K_{ab}} \\ A : \{A\}_{K_{ab}} & \to B: & s \end{array}$$

Then symbolically, the process $(\nu k_{ab})(\nu s)A \| B$ would never emit s while again, a computational adversary can forge a key k such that any bitstring can be successfully decrypted to a using k.

This attack could be captured, allowing the forged key to be independent of the ciphertext. This can be modeled by the equation

$$\mathsf{dec}(x, \mathsf{fakekey}(y)) = y$$

where $\mathsf{fakekey}$ is now a primitive of arity 1. Some attacks may however require the decryption to depend from the cyphertext as shown in the next example.

▶ **Example 4.3** (simultaneous cyphertexts)**.** Consider the following protocol where A sends to B p cyphertexts $c_1, \ldots, c_p$. Then B encrypts all ciphertexts with a shared key k_{ab} together

with a fresh value n_b and commits to p other nonces $N_1, \ldots, N_p$. Then A simply forwards the cyphertext together with a fresh key k. Then B checks whether each cyphertext c_i decrypts to N_i using the key k received from A, in which case he sends out a secret s.

$$\begin{aligned} A &\to B: \quad c_1, \ldots, c_p \\ B &\to A: \quad \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, N_1, \ldots, N_p \\ A &\to B: \quad \{N_b, c_1, \ldots, c_p\}_{K_{ab}}, k \\ B: \{N_b, \{N_1\}_k, \ldots, \{N_p\}_k\}_{K_{ab}}, k &\to A: s \end{aligned}$$

Then symbolically, the process $(\nu k_{ab})(\nu s)A \| B$ would never emit s since the N_i are generated after having received the c_i and the nonce N_b protects the protocol from replay attacks. However, having seen the c_i and the N_i, a computational adversary can forge a key k such that each bitstring c_i can be successfully decrypted to N_i using k. More precisely, given an IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, it is easy to build another IND-CCA2 (authenticated) encryption scheme $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$, which allows to mount the previous attack. Indeed, consider $\mathcal{G}' = 0 \cdot \mathcal{G}$ (all honest keys begin with the bit 0), $\mathcal{E}'(m, i.k) = \mathcal{E}(m, k)$ for $i \in \{0, 1\}$ and $\mathcal{D}'$ defined as follows:

- $\mathcal{D}'(c, k) = \mathcal{D}(c, k')$ if $k = 0 \cdot k'$
- $\mathcal{D}'(c, k) = n$ if $k = 1 \cdot c_1, n_1, \cdots c, n \cdots c_p, n_p$
- $\mathcal{D}'(c, k) = \perp$ otherwise

For dishonest keys, the decryption function $\mathcal{D}'(c, k)$ searches for c in k and outputs the following component when c is found in k. It is easy to check that $(\mathcal{G}', \mathcal{E}', \mathcal{D}')$ remains IND-CCA2 and allows an adversary to chose a key that decrypts to anything he wants, but with different possible outputs depending on the ciphertext.

To capture this attack, we need to consider a symbol of arity $2p$ for any p and an equation of the form

$$\mathsf{dec}(x_i, \mathsf{fakekey}(x_1, \ldots, x_p, y_1, \ldots, y_p)) = y_i$$

But this is still not be sufficient as the outcome may also depend on the ciphertext that is under decryption and on public data. Intuitively, decrypting with an adversarial key may produce a function depending on the underlying plaintext and on any previously known data.

Related work. To our best understanding of [13], these examples seem to form counter-examples of the soundness results for symmetric encryption as presented in [13]. An implicit assumption that solves this issue [10] consists in forbidding dishonest keys to be used for encryption or decryption (the simulator would stop as soon as it received a dishonest keys). As a consequence, only protocols using keys as nonces could be proved secure.

The only work overcoming this limitation in a realistic way is the work of Kuesters and Tuengerthal [39] where the authors show computational soundness for key exchange protocols with symmetric encryption, without restricting key generation for the adversary. Instead, they assume that sessions identifiers are added to plaintexts before encryption. This assumption is non standard but achievable. It would however not be sufficient in general as shown by the examples. In their case, such an assumption suffices because the result is tailored to key exchange protocols and realization of a certain key exchange functionality.

5 Conclusion

Among all the limitations we discuss in this paper, the main one is to consider only honestly generated keys (or a certifying infrastructure), which is completely unrealistic. There are

(at least) two main ways to overcome this assumption. A first possibility, already sketched in the paper, consists in enriching the symbolic model by letting the adversary create new symbolic equalities when building new (dishonest) keys. In this way, many protocols should still be provably secure under the IND-CCA assumption, yet benefiting from a symbolic setting for writing the proof.

A second option is to seek for stronger security assumptions by further requesting non-malleability. The idea is that a ciphertext should not be opened to a different plaintext, even when using dishonest keys. This could be achieved by adding a commitment to the encryption scheme [35].

However all these limitations also demonstrate that it is difficult to make symbolic and computational models coincide. Even for standard security primitives, soundness results are very strong since they provide with a generic security proof for *any* possible protocol (contrary to CryptoVerif). For primitives with many algebraic properties like Exclusive Or or modular exponentiation, the gap between symbolic and computation models is even larger and would require a lot of efforts.

We still believe that computational proofs could benefit from the simplicity of symbolic models, yielding automated proofs. An alternative approach to soundness results could consist in computing, out of a given protocol, the minimal computational hypotheses needed for its security. This is for example the approach explored in [17], though the symbolic model is still very complex.

Acknowledgement

We wish to thank Michael Backes and Dominique Unruh for very helpful explanations on their work and David Galindo for fruitful discussion on non-malleable encryption schemes.

References

1 M. Abadi, B. Blanchet, and H. Comon-Lundh. Models and proofs of protocol security: A progress report. In A. Bouajjani and O. Maler, editors, *Proceedings of the 21st International Conference on Computer Aided Verification (CAV'09)*, volume 5643 of *Lecture Notes in Computer Science*, pages 35–49, Grenoble, France, June-July 2009. Springer.

2 M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, November 2006.

3 M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, January 2001.

4 M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148(1), 1999.

5 M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. of the 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

6 M. Abadi and P. Rogaway. Reconciling two views of cryptography. In *Proc. of the International Conference on Theoretical Computer Science (IFIP TCS2000)*, pages 3–22, August 2000.

7 M. Abdalla and B. Warinschi. On the minimal assumptions of group signature schemes. In *6th International Conference on Information and Communication Security*, pages 1–13, 2004.

8 A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, 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 K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, 2005. Springer.

9 A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and M. L. Tobarra. Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps. In *6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, pages 1–10, Alexandria, VA, USA, 2008.

10 M. Backes. Private communication, 2007.

11 M. Backes, M. Dürmuth, and R. Küsters. On simulatability soundness and mapping soundness of symbolic cryptography. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2007.

12 M. Backes, D. Hofheinz, and D. Unruh. CoSP a general framework for computational soundness proofs. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, pages 66–78, 2009.

13 M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Science Foundations Workshop (CSFW'04)*, pages 204–218, 2004.

14 M. Backes and B. Pfitzmann. Limits of the Cryptographic Realization of Dolev-Yao-style XOR and Dolev-Yao-Style Hash Functions. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, Lecture Notes in Computer Science, pages 336–354, 2005.

15 M. Backes and B. Pfitzmann. Relating cryptographic und symbolic key secrecy. In *26th IEEE Symposium on Security and Privacy*, pages 171–182, Oakland, CA, 2005.

16 M. Backes, B. Pfitzmann, and M. Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12):1685–1720, 2007.

17 G. Bana, K. Hasebe, and M. Okada. Secrecy-oriented first-order logical analysis of cryptographic protocols. Cryptology ePrint Archive, Report 2010/080, 2010. `http://eprint.iacr.org/`.

18 M. Bellare and C. Namprempre. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology (ASIACRYPT 2000)*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, 2000.

19 B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, June 2001.

20 B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *20th International Conference on Automated Deduction (CADE-20)*, July 2005.

21 B. Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, Oct.–Dec. 2008. Special issue IEEE Symposium on Security and Privacy 2006. Electronic version available at http://doi.ieeecomputersociety.org/10.1109/TDSC.2007.1005.

22 M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proc. of the Royal Society*, volume 426 of *Series A*, pages 233–271. 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

23 R. Canetti and M. Fischlin. Universally composable commitments. In *CRYPTO 2001*, pages 19–40, Santa Barbara, California, 2001.

24 Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. *Theoretical Computer Science*, 338(1-3):247–274, June 2005.

25 Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, and L. Vigneron. Deciding the security of protocols with Diffie-Hellman exponentiation and product in exponents. In *Proc. of the 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, 2003.

26 H. Comon-Lundh and V. Cortier. Computational soundness of observational equivalence. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. ACM Press, 2008.

27 H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties for cryptographic protocols. Application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 11(4):496–520, 2010.

28 H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 271–280, Ottawa, Canada, June 2003. IEEE Computer Society Press.

29 V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In N. Garg and S. Arun-Kumar, editors, *Proceedings of the 26th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 176–187, Kolkata, India, December 2006. Springer.

30 V. Cortier, S. Kremer, and B. Warinschi. A Survey of Symbolic Methods in Computational Analysis of Cryptographic Systems. *Journal of Automated Reasoning (JAR)*, 2010.

31 V. Cortier and B. Warinschi. Computationally Sound, Automated Proofs for Security Protocols. In *Proc. 14th European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 157–171, Edinburgh, U.K, April 2005. Springer.

32 S. Delaune, S. Kremer, and M. D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Computer Security Foundations Workshop (CSFW'06)*, pages 28–39, 2006.

33 D. Dolev and A. Yao. On the security of public key protocols. In *Proc. of the 22nd Symp. on Foundations of ComputerScience*, pages 350–357. IEEE Computer Society Press, 1981.

34 N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.

35 D. Galindo, F. D. Garcia, and P. Van Rossum. Computational soundness of non-malleable commitments. In *Proceedings of the 4th international conference on Information security practice and experience*, ISPEC'08, pages 361–376. Springer-Verlag, 2008.

36 R. Janvier, Y. Lakhnech, and L. Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 172–185. Springer, 2005.

37 R. Küsters and T. Truderung. On the automatic analysis of recursive security protocols with xor. In *Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS'07)*, volume 4393 of *LNCS*, Aachen, Germany, 2007. Springer.

38 R. Küsters and M. Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computations. In *Computer Security Foundations (CSF'08)*, 2008.

39 R. Küsters and M. Tuengerthal. Computational Soundness for Key Exchange Protocols with Symmetric Encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, pages 91–100. ACM Press, 2009.

40 Y. Lindell. General composition and universal composition in secure multiparty computation. In *Proc. 44th IEEE Symp. Foundations of Computer Science (FOCS)*, 2003.

41 G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, march 1996.

42 D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway language of encrypted expressions. *Journal of Computer Security*, 2004.

43 D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151, Cambridge, MA, USA, February 2004. Springer-Verlag.

44 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.

45 B. Warinschi. A computational analysis of the needham-schroeder protocol. In *16th Computer security foundation workshop (CSFW)*, pages 248–262. IEEE, 2003.