# Compiling and securing cryptographic protocols

Yannick Chevalier, Michael Rusinowitch

HAL Id: inria-00426669

https://inria.hal.science/inria-00426669

Submitted on 27 Oct 2009

# Compiling and securing cryptographic protocols

### Yannick Chevalier, Michaël Rusinowitch

Loria, Inria Nancy Grand Est

Campus Scientifique — BP 239 54506 Vandœuvre-lès-Nancy, France

`{chevalie,rusi}@loria.fr`

October 27, 2009

**Abstract**

Protocol narrations are widely used in security as semi-formal notations to specify conversations between roles. We define a translation from a protocol narration to the sequences of operations to be performed by each role. Unlike previous works, we reduce this compilation process to well-known decision problems in formal protocol analysis. This allows one to define a natural notion of prudent translation and to reuse many known results from the literature in order to cover more crypto-primitives. In particular this work is the first one to show how to compile protocols parameterised by the properties of the available operations.

## 1 Introduction

Cryptographic protocols are designed to prescribe message exchanges between agents in hostile environment in order to guarantee some security properties such as confidentiality. There are many apparently similar ways to describe a given security protocol. However one has to be precise when specifying how a message should be interpreted and processed by an agent since overlooking subtle details may lead to dramatic flaws. The main issues are the following:

- What parts of a received message should be extracted and checked by an agent?

- What actions should be performed by an agent to compute an answer?

These questions are often either partially or not at all adressed in common protocol descriptions such as the so-called protocol narrations. A *protocol narration* is the definition of a cryptographic protocol by the intended sequence of messages. For example the well-known Needham-Schroeder Public Key protocol [23] is conveniently specified by the following text:

$$A{\rightarrow}B{:}\mathrm{enc}(\langle A, N_a\rangle, K_B)$$
$$B{\rightarrow}A{:}\mathrm{enc}(\langle N_a, N_b\rangle, K_A)$$
$$A{\rightarrow}B{:}\mathrm{enc}(N_b, K_B)$$
$$\textbf{where}$$
$$A \textbf{ knows } A, B, K_A, K_B, K_A^{-1}$$
$$B \textbf{ knows } A, B, K_A, K_B, K_B^{-1}$$

Protocol narrations are also a textual representation of Message Sequence Charts (MSC), which are employed *e.g.* in RFCs. For more complex protocols, one needs to indicate the internal

```
         Client                              Host
```

| Client | | Host |
|---|---|---|
| $U =$`<username>` | $\rightarrow$ | |
| | $\leftarrow$ | $s =$`<salt from passwd file>` |

Upon identifying himself to the host, the client will receive the salt stored on the host under his username.

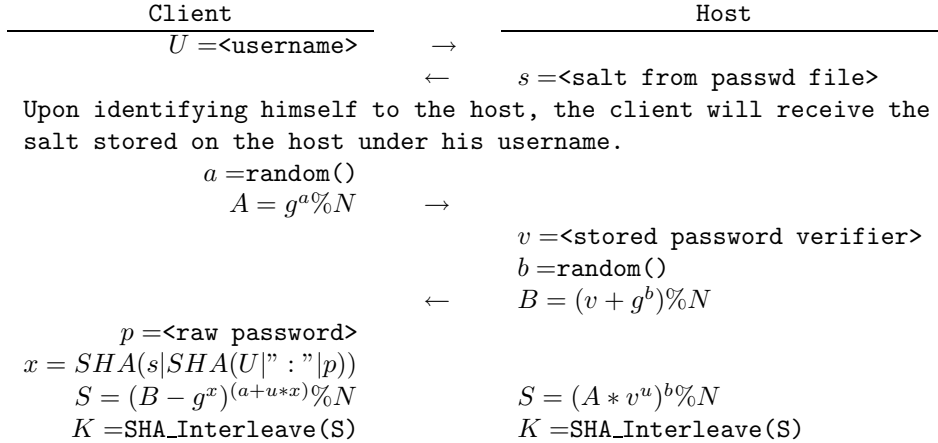| Client | | Host |
|---|---|---|
| $a =$`random()` | | |
| $A = g^a \% N$ | $\rightarrow$ | |
| | | $v =$`<stored password verifier>` |
| | | $b =$`random()` |
| | $\leftarrow$ | $B = (v + g^b)\%N$ |
| $p =$`<raw password>` | | |
| $x = SHA(s\|SHA(U|" : "|p))$ | | |
| $S = (B - g^x)^{(a+u*x)}\%N$ | | $S = (A * v^u)^b\%N$ |
| $K =$`SHA_Interleave(S)` | | $K =$`SHA_Interleave(S)` |

Figure 1: Annotated message sequence chart extracted from the RFC 2945 (SRP Authentication and Key Exchange System)

computations of each participant either by annotating the MSC or by employing the Lowe operator [17] or otherwise express internal actions that have to be performed, as in the specification of Fig. 1.

We claim that all internal computations specified in Figure 1, and more generally most such annotations, can be computed automatically from the protocol narration. Our goal in this paper is to give an operational semantics to—or, equivalently, to *compile*—protocol narrations so that internal actions (excluding *e.g.* storing a value in a special list for a use external to the protocol) are described.

**Related works** Although many works have been dedicated to verifying cryptographic protocols in various formalisms, only a few have considered the different problems of extracting operational (non ambiguous) role definitions from protocol descriptions. Operational roles are expressed as multiset rewrite rules in CAPSL [20], CASRUL [15], or sequential processes of the spi-calculus with pattern-matching [6]. This extraction is also used for end-point projection [18, 19]. A pioneering work in this area is one by Carlsen [7] that has proposed a system for translating protocol narrations into CKT5 [5], a modal logic of communication, knowledge and time.

Compiling narrations to roles has been extended beyond perfect encryption primitives to algebraic theories in [10, 22]. We can note that, although these works admit very similar goals, all their operational role computations are ad-hoc and lack of a uniform principle. In particular they essentially re-implemented previously known techniques. An advantage of [22] is that it supports implicit decryption which may lead to more efficient secrecy decision procedures.

We propose here a uniform approach to role computation that allows us to relate the problem to well-known decision results in formal cryptographic protocols analysis, namely the *reachability problem*. Moreover this approach is also used successfully for the automatic computation of prudent security wrapper (a.k.a. security tests) for filtering messages received by principals. We show how to reduce this computation to known results about the standard notion of *static equivalence*.

# 2 Role-based Protocol Specifications

First we show how from Alice& Bob notation we can derive a plain role-based specification. Then the specification will be refined in the following Sections.

## 2.1 Specification of messages and basic operations

**Terms** We consider an infinite set of free constants $C$ and an infinite set of variables $\mathcal{X}$. For each signature $\mathcal{F}$ (*i.e.* a set of function symbols with arities), we denote by $\mathrm{T}(\mathcal{F})$ (resp. $\mathrm{T}(\mathcal{F}, \mathcal{X})$) the set of terms over $\mathcal{F} \cup C$ (resp. $\mathcal{F} \cup C \cup \mathcal{X}$). The former is called the set of ground terms over $\mathcal{F}$, while the later is simply called the set of terms over $\mathcal{F}$. Variables are denoted by $x$, $y$, terms are denoted by $s$, $t$, $u$, $v$, and finite sets of terms are written $E, F, ...,$ and decorations thereof, respectively. We abbreviate $E \cup F$ by $E, F$, the union $E \cup \{t\}$ by $E, t$ and $E \setminus \{t\}$ by $E \setminus t$.

In a signature $\mathcal{F}$ a *constant* is either a free constant or a function symbol of arity 0 in $\mathcal{F}$. Given a term $t$ we denote by $\mathrm{Var}(t)$ the set of variables occurring in $t$ and by $\mathrm{Cons}(t)$ the set of constants occurring in $t$. A substitution $\sigma$ is an idempotent mapping from $\mathcal{X}$ to $\mathrm{T}(\mathcal{F}, \mathcal{X})$ such that $\mathrm{Supp}(\sigma) = \{x \,|\, \sigma(x) \neq x\}$, the *support* of $\sigma$, is a finite set. The application of a substitution $\sigma$ to a term $t$ (resp. a set of terms $E$) is denoted $t\sigma$ (resp. $E\sigma$) and is equal to the term $t$ (resp. $E$) where all variables $x$ have been replaced by the term $x\sigma$. A substitution $\sigma$ is *ground* if for each $x \in \mathrm{Supp}(\sigma)$ we have $x\sigma \in \mathrm{T}(\mathcal{F})$.

*Operations.* Terms are manipulated by applying *operations* on them. These operations are defined by a subset of the signature $\mathcal{F}$ called the *set of public constructors*. A context $C[x_1, \ldots, x_n]$ is a term in which all symbols are public and such that its nullary symbols are either public non-free constants or variables.

*Equational theories.* An *equational presentation* $\mathcal{E} = (\mathcal{F}, E)$ is defined by a set $E$ of equations $u = v$ with $u, v \in \mathrm{T}(\mathcal{F}, \mathcal{X})$. The equational theory generated by $(\mathcal{F}, E)$ on $\mathrm{T}(\mathcal{F}, \mathcal{X})$ is the smallest congruence containing all instances of axioms of $E$ (free constants can also be used for building instances). We write $s =_{\mathcal{E}} t$ as the congruence relation between two terms $s$ and $t$. By abuse of terminology we also call $\mathcal{E}$ the equational theory generated by the presentation $\mathcal{E}$ when there is no ambiguity. This equational theory is introduced in order to specify the effects of operations on the messages and the properties of messages.

*Deduction systems.* A *deduction system* is defined by a triple $(\mathcal{E}, \mathcal{F}, \mathcal{F}_p)$ where $\mathcal{E}$ is an equational presentation on a signature $\mathcal{F}$ and $\mathcal{F}_p$ a subset of *public* constructors in $\mathcal{F}$. For instance the following deduction system models public key cryptography: $(\{\mathrm{dec}(\mathrm{enc}(x, y), y^{-1}) = x\}, \{\mathrm{dec}(\_, \_), \mathrm{enc}(\_, \_), \_^{-1}\}, \{\mathrm{dec}(\_, \_), \mathrm{enc}(\_, \_)\})$ The equational theory is reduced here to a single equation that expresses that one can decrypt a ciphertext when the inverse key is available.

## 2.2 Role Specification

We present in this subsection how protocol narrations are transformed into sets of *roles*. A role can be viewed as the projection of the protocol on a principal. The core of a role is a *strand* which is a standard notion in cryptographic protocol modeling [14].

A *strand* is a finite sequence of messages each with label (or polarity) ! or ?. Messages with label ! (resp. ?) are said to be "sent" (resp. "received"). A strand is *positive* iff all its labels are !. Given a list of message $l = m_1, \ldots, m_n$ we write $?l$ (resp. $!l$) as a short-hand for $?m_1, \ldots, ?m_n$, (resp. $!m_1, \ldots, !m_n$).

**Definition 1** *A role specification is an expression $A(\vec{l}) : \nu\vec{n}.(S)$ where $A$ is a name, $\vec{l}$ is a sequence of constants (called the role parameters), $\vec{n}$ is a sequence of constants (called the nonces*

*of the role), and S is a strand. Given a role r we denote by* $\text{nonces}(r)$ *the nonces* $\vec{n}$ *of r and* $\text{strand}(r)$ *the strand S of r.*

**Example 1** *For example, the initiator of the NSPK protocol is modeled, at this point, with the role:*

$$\nu N_a.(?N_a, ?A, ?B, ?K_A, ?K_B, ?K_A^{-1},$$
$$!\text{msg}(B, \text{enc}(\langle A, N_a \rangle, K_B)), ?\text{msg}(B, \text{enc}(\langle N_a, N_b \rangle, K_A)),$$
$$!\text{msg}(B, \text{enc}(N_b, K_B)))$$

*with the equational theory of public key cryptography, plus the equations* $\{\pi_1(\langle x, y \rangle) = x, \pi_2(\langle x, y \rangle) = y\}.$

Note that nothing guarantees in general that a protocol defined as a set of roles is executable. For instance some analysis is necessary to see whether a role can derive the required inverse keys for examining the content of a received ciphertext. We also stress that role specfications do not contain any variables. The symbols $N_a, A, \dots$ in the above example are constants, and the messages occurring in the role specification are all ground terms.

**Plain roles extracted from a narration**  From a protocol narration where each nonce originates uniquely we can extract almost directly a set of roles, called plain roles as follows. The constants occurring in the initial knowledge of a role are the parameters of the strand describing this role. We model this initial knowledge by a sequence of receptions (from an unspecified agent) of each term in the initial knowledge. In order to encode narrations we assume that we have in the signature three public function symbols $\text{msg}(\_, \_)$, $\text{partner}(\_)$ and $\text{payload}(\_)$ satisfying the equational theory:

$$\begin{cases} \text{partner}(\text{msg}(x, y)) & = & x \\ \text{payload}(\text{msg}(x, y)) & = & y \end{cases}$$

For every agent name $A$ in the protocol narration, a role specification for $A$ is $A(l)$ : $\nu\text{nonces}(S).(?\text{nonces}(S), ?K, S^A)$, where $K$ is such that $A$ **knows** $K$ occurs in the protocol narration, $l$ is the set of constants in $K$. $\text{nonces}(S)$ and strand $S^A$ are computed as follows:

**Computation of** $S^A$**:** Init $S_0^A = \emptyset$
  On the $(n+1)$-th line $S \to R : M$ do

$$S_{n+1}^A = \begin{cases} S_n, !\text{msg}(R, M) & \text{If } A = S \\ S_n, ?\text{msg}(S, M) & \text{If } A = R \\ S_n^A & \text{Otherwise} \end{cases}$$

**Computation of** $\text{nonces}(A)$**:** This set contains each constant $N$ that appears in the strand $?K, S^A$ inside a message labelled ! and such that $N$ does not occur in previous messages (with any polarity).

This computation always extracts role specifications from a given protocol narration and it has the property that every constant appears in a received message before appearing in a sent message. Since a nonce is to be created within an instance of a role, we reject protocol narrations from which the algorithm described above extracts two different roles $A$ and $B$ with $\text{nonces}(A) \cap \text{nonces}(B) \neq \emptyset$.

Example 1 is a plain role that can be derived by applying the algorithm to the NSPK protocol narration. We now define the *input* of a role specification which informally is the sequence of messages *sent to* a role as defined by the protocol narration.

**Definition 2** *Let $r = \nu N.(\overset{!}{?} M_i)_{1 \leq i \leq n}$ be a role specification, and let $(R_1, \ldots, R_k)$ be the subsequence of the messages $M_i$ labeled with ?. The* input *of $r$ is denoted* input$(r)$ *and is the positive strand $(!R_1, \ldots, !R_k)$.*

In the next section we define a target for the compilation of role specifications. Then we compute constraints to be satisfied by sent and received messages. and by adding the constraints to the specification this one gets executable in the safest way as possible w.r.t. to its initial specification.

# 3 Operational semantics for roles

In Section 2 we have defined roles and shown how they can be extracted from protocol narrations. In this section we define what an implementation of a role is and in Section 4 we will show how to compute such an implementation from a protocol narration.

**Unification systems** Intuitively an operational model for a role has to reflect the possible manipulations on messages performed by a program implementing the role. These operations are specified here by a deduction system $\mathcal{D} = (\mathcal{E}, \mathcal{F}, \mathcal{S})$ where the set of public functions $\mathcal{S}$, a subset of the signature $\mathcal{F}$, is defined by equations in $\mathcal{E}$. Beside defining function computations, the equations $\mathcal{E}$ specify some properties.

**Definition 3** *Let $\mathcal{E}$ be an equational theory. An $\mathcal{E}$-Unification system $S$ is a finite set of equations denoted by $(u_i \overset{?}{=} v_i)_{i \in \{1, \ldots, n\}}$ with terms $u_i, v_i \in \mathrm{T}(\mathcal{F}, \mathcal{X})$. It is satisfied by a substitution $\sigma$, and we note $\sigma \models S$, if for all $i \in \{1, \ldots, n\}$ $u_i\sigma =_{\mathcal{E}} v_i\sigma$.*

**Active frames** We introduce now the set of implementations of a role specification as *active frames*. An active frame extends the role notion by specifying how a message to be sent is constructed from already known messages, and how a received message is checked to ascertain its conformity w.r.t. already known messages. The notation $!v_i$ (resp. $?v_i$) refers to a message stored in variable $v_i$ which is sent (resp. received).

**Definition 4** *Given a deduction system $\mathcal{D}$ with equational theory $\mathcal{E}$, a $\mathcal{D}$-active frame is a sequence $(T_i)_{1 \leq i \leq k}$ where*

$$T_i = \begin{cases} !v_i \text{ with } v_i \overset{?}{=} C_i[v_1, \ldots, v_{i-1}] & \text{(send)} \\ \quad\quad\quad\quad \textbf{or} \\ ?v_i \text{ with } S_i(v_1, \ldots, v_i) & \text{(receive)} \end{cases}$$

*where $C_i[v_1, \ldots, v_{i-1}]$ denotes a context over variables $v_1, \ldots, v_{i-1}$ and $S_i(v_1, \ldots, v_i)$ denotes a $\mathcal{E}$-unification system over variables $v_1, \ldots, v_i$. Each variable $v_i$ occuring with polarity ? is an* input variable *of the active frame.*

**Example 2** *The following is an active frame denoted $\phi_a$ that can be employed to model the role $A$ in the NSPK protocol:*

$$(?v_{N_a} ?v_A, ?v_B, ?v_{K_A}, ?v_{K_B}, ?v_{K_A^{-1}},$$
$$!v_{msg_1} \text{ with } v_{msg_1} \overset{?}{=} \mathrm{msg}(v_B, \mathrm{enc}(\langle v_A, v_{N_a} \rangle, v_{K_B})),$$
$$?v_r \text{ with } \emptyset$$
$$!v_{msg_2} \text{ with } v_{msg_2} \overset{?}{=} \mathrm{msg}(v_B, \mathrm{enc}(\pi_2(\mathrm{dec}(v_r, v_{K_A^{-1}})), v_{K_B})))$$

Compilation is the computation of an active frame from a role specification such that, when receiving messages as intended by the role specification, the active frame emits responses equal modulo the equational theory to the responses issued in the role specification. More formally, we have the following:

**Definition 5** *Let $\mathcal{D}$ be a deduction system with equational theory $\mathcal{E}$. Let $\varphi = (T_i)_{1 \leq i \leq k}$ be an active frame, where the $T_i$'s are as in Definition 4, and where the input variables are $r_1, \ldots, r_n$. Let $s$ be a positive strand $!M_1, \ldots, !M_n$. Let $\sigma_{\varphi,s}$ be the substitution $\{r_i \mapsto M_i\}$ and $S$ be the union of the $\mathcal{E}$-unification systems in $\varphi$. The* evaluation *of $\varphi$ on $s$ is denoted $\varphi \cdot s$ and is the strand $(m_1, \ldots, m_k)$ where:*

$$m_i = \begin{cases} !C_i[m_1, \ldots, m_{i-1}] & \textit{If } v_i \textit{ has label ! in } T_i \\ ?v_i \sigma_{\varphi,s} & \textit{If } v_i \textit{ has label ? in } T_i \end{cases}$$

*We say that $\varphi$ accepts $s$ if $S\sigma_{\varphi,s}$ is satisfiable.*

To simplify notations, the application of a $\mathcal{D}$-context $C[x_1, \ldots, x_n]$ on a positive strand $s = (!t_1, \ldots, !t_n)$ of length $n$ is denoted $C \cdot s$ and is the term $C[t_1, \ldots, t_n]$.

**Example 3** *Let $r$ be the role specification of role $A$ in NSPK as given in Ex. 1 and $\phi_A$ be the active frame of Ex. 2. We have:*

$$\mathrm{input}(r) = (!N_a, !A, !B, !K_A, !K_B, !K_A^{-1}, !\mathrm{msg}(B, \mathrm{enc}(\langle N_a, N_b \rangle, K_A)))$$

*and $\phi_A \cdot \mathrm{input}(r)$ is the strand:*

$(?N_a, ?A, ?B, ?K_A, ?K_B, ?K_A^{-1}, !\mathrm{msg}(B, \mathrm{enc}(\langle A, N_a \rangle, K_B)), ?\mathrm{msg}(B, \mathrm{enc}(\langle N_a, N_b \rangle, K_A)),$
$!\mathrm{msg}(B, \mathrm{enc}(\pi_2(\mathrm{dec}(\mathrm{payload}(\mathrm{msg}(B, \mathrm{enc}(\langle N_a, N_b \rangle, K_A))), K_A^{-1})), K_B))$

*Modulo the equational theory, this strand is equal to the strand:*

$(?N_a, ?A, ?B, ?K_A, ?K_B, ?K_A^{-1}, !\mathrm{msg}(B, \mathrm{enc}(\langle A, N_a \rangle, K_B)), ?\mathrm{msg}(B, \mathrm{enc}(\langle N_a, N_b \rangle, K_A)),$
$!\mathrm{msg}(B, \mathrm{enc}(N_b, K_B))$

It is not coincidental that in Ex. 3 the strands $\varphi \cdot \mathrm{input}(r)$ and $\mathrm{strand}(r)$ are equal as it means that within the active frame, the sent messages are composed from received ones in such a way that when receiving the messages expected in the protocol narration, the role responds with the messages intended by the protocol narration. This fact gives us a criterion to define functional implementations of a role.

**Definition 6** *An active frame $\varphi$ is an* implementation *of a role specification $r$ if $\varphi$ accepts $\mathrm{input}(r)$ and $\varphi \cdot \mathrm{input}(r) =_\mathcal{E} \mathrm{strand}(r)$. If a role admits an implementation we say this role is* executable.

**Example** $\phi_a$ defined above is a possible implementation of the initiator role in NSPK. However this implementation does not check the conformity of the messages with the intended patterns, *e.g.* it neither checks that $v_r$ is really an encryption with the public key $v_{K_A}$ of a pair, nor that the first argument of the encrypted pair has the same value as the nonce $v_{N_a}$. In Section 4 we show not only how to compute an active frame when the role specification is executable, but also to ensure that all the possible checks are performed.

# 4 Compilation of role specifications

Usually the compilation of a specification is defined by a *compilation algorithm*. An originality of this work is that we present the result of the compilation as the solution to decision problems. This has the advantage of providing for free a notion of prudent implementation as explained below.

## 4.1 Computation of a "vanilla" implementation

Let us first present how to compute an implementation of a role specification in which no check is performed, as given in the preceding example. To build such an implementation we need to compute for every sent message $m$ a context $C_m$ that evaluates to $m$ when applied to the previously received ones. This reachability problem is unsolvable in general. Hence we have to consider systems that admit a reachability algorithm, formally defined below:

**Definition 7** *Given a deduction system $\mathcal{D}$ with equational theory $\mathcal{E}$, a $\mathcal{D}$-reachability algorithm $\mathcal{A}_{\mathcal{D}}$ computes, given a positive strand $s$ of length $n$ and a term $t$, a $\mathcal{D}$-context $\mathcal{A}_{\mathcal{D}}(s,t) = C[x_1, \ldots, x_n]$ such that $C \cdot s =_{\mathcal{E}} t$ iff there exists such a context and $\bot$ otherwise.*

We will show that several interesting theories admit a reachability algorithm. This algorithm can be employed as an oracle to compute the contexts in sent messages and therefore to derive an implementation of a role specification $r$. We thus have the following theorem.

**Theorem 1** *If there exists a $\mathcal{D}$-reachability algorithm then it can be decided whether a role specifications $r$ is executable and, if so one can compute an implementation of $r$.*

**Proof sketch.** Let $r = (\frac{!}{?} M_i)_{i \in \{1, \ldots, n\}}$ be an executable role specification. By definition there exists an active frame $\varphi$ that implements $r$, *i.e.* for each sent message $M_i$, there exists a context $C_i$ such that $C_i[M_1, \ldots, M_{i-1}]$ is equal to $M_i$ modulo the equational theory. Thus if there exists a $\mathcal{D}$-reachability algorithm $\mathcal{A}_{\mathcal{D}}$, the result $\mathcal{A}_{\mathcal{D}}(M_1, \ldots, M_{i-1}), M_i)$ cannot be $\bot$ by definition. As a consequence, $\mathcal{A}_{\mathcal{D}}((M_1, \ldots, M_{i-1}), M_i)$ is a context $C'_i[x_1, \ldots, x_n]$. Thus for all index $i$ such that $M_i$ is sent we can compute a context $C'_i$ that, when applied on previous messages, yields the message to send. We thus have an implementation of the role specification.

## 4.2 Computation of a prudent implementation

Computing an active frame is not enough since one would want to model that received messages are checked as thoroughly as possible. For instance in Example 1, a prudent implementation of the message reception "$?v_r$ with $\emptyset$" should be:

$$?v_r \text{ with } \pi_1(\text{dec}(\text{payload}(v_r), v_{K_A^{-1}})) \overset{?}{=} v_{N_a} \wedge \text{partner}(v_r) \overset{?}{=} v_B$$

Let us first formalize this by a refinement relation on sequences of messages. We will say a strand $s$ refines a strand $s'$ if any observable equality of subterms in strand $s$ can be observed in $s'$ using the same tests. To put it formally:

**Definition 8** *A positive strand $s = (!M_1, \ldots, !M_n)$ refines a positive strand $s' = (!M'_1, \ldots, !M'_n)$ if, for any pair of contexts $(C_1[x_1, \ldots, x_n], C_2[x_1, \ldots, x_n])$ one has $C_1 \cdot s' = C_2 \cdot s'$ implies $C_1 \cdot s = C_2 \cdot s$.*

For instance the strand $s = (!\mathrm{enc}(\mathrm{enc}(a, k'), k), !\mathrm{enc}(a, k'), !k, !k', !a)$ refines $s' = (!\mathrm{enc}(\mathrm{enc}(a, k'), k), !\mathrm{enc}(a, k'), !k, !k'', !a)$ since all equalities that can be checked on $s'$ can be checked on $s$. We can now define an implementation to be *prudent* if every equality satisfied by the sequence of messages of the protocol specification is satisfied by *any* accepted sequence of messages.

**Definition 9** *Let $r$ be a role specification and $\varphi$ be an implementation of $r$. We say that $\varphi$ is prudent if any positive strand $s$ accepted by $\varphi$ is a refinement of* $\mathrm{input}(r)$.

As we shall see in Section 5, most deduction systems considered in the context of cryptographic protocols analysis have the property that it is possible to compute, given a positive strand, a finite set of context pairs that summarizes all possible equalities in the sense of the next definition. Let us first introduce a notation: Given a positive strand $s$ we let $P_s$ be the set of context pairs $(C_1, C_2)$ such that $C_1 \cdot s = C_2 \cdot s$.

**Definition 10** *A deduction system $\mathcal{D}$ has the* finite basis property *if for each positive strand $s$ one can compute a finite set $P_s^f$ of pairs of $\mathcal{D}$-contexts such that, for each positive strand $s'$:*

$$P_s \subseteq P_{s'} \quad \mathit{iff} \quad P_s^f \subseteq P_{s'}$$

Let us now assume that a deduction system $\mathcal{D}$ has the finite basis property. There thus exists an algorithm $\mathcal{A}'_{\mathcal{D}}(s)$ that takes a positive strand $s$ as input, computes a finite set $P_s^f$ of context pairs $(C[x_1, \ldots, x_n], C'[x_1, \ldots, x_n])$ and returns as a result the $\mathcal{E}$-unification system $S_s$ : $\left\{ C[x_1, \ldots, x_n] \stackrel{?}{=} C'[x_1, \ldots, x_n] \mid (C, C') \in P_s^f \right\}$. For any positive strand $s' = (!m_1, \ldots, !m_n)$ of length $n$, let $\sigma_{s'}$ be the substitution $\{x_i \mapsto m_i\}_{1 \leq i \leq n}$. By definition of $S_s$ we have that $\sigma_{s'} \models S_s$ if and only if $s'$ is a refinement of $s$. Given the preceding definition of $\mathcal{A}_{\mathcal{D}}(s, t)$, we are now ready to present our algorithm for the compilation of role specifications into active frames.

**Algorithm** Let $r$ be a role specification with $\mathrm{strand}(r) = (\frac{!}{?}M_1, \ldots, \frac{!}{?}M_n)$ and let $s = (!M_1, \ldots, !M_n)$. Let us introduce two notations to simplify the writing of the algorithm, *i.e.* we write $r(i)$ to denote the $i$-th labelled message $\frac{!}{?}M_i$ in $r$, and $s^i$ to denote the prefix $(!M_1, \ldots, !M_i)$ of $s$. Compute, for $1 \leq i \leq n$:

$$T_i = \begin{cases} !v_i \text{ with } v_i \stackrel{?}{=} \mathcal{A}_{\mathcal{D}}(s^{i-1}, M_i) & \text{If } r(i) = !M_i \\ ?v_i \text{ with } \mathcal{A}'_{\mathcal{D}}(s^i) & \text{If } r(i) = ?M_i \end{cases}$$

and return the active frame $\varphi_r = (T_i)_{1 \leq i \leq n}$. By construction we have the following theorem.

**Theorem 2** *Let $\mathcal{D}$ be a deduction system such that $\mathcal{D}$-ground reachability is decidable and $\mathcal{D}$ has the finite basis property. Then for any executable role specification $r$ one can compute a prudent implementation $\varphi$.*

# 5  Examples and Applications

Many theories that are relevant to cryptographic protocol design satisfy the hypothesis of Theorem 2. For instance let us introduce the *convergent subterm theory*:

**Definition 11** *An equational theory is* convergent subterm *if it admits a presentation by a set of equations $\mathcal{E} = \bigcup_{i=1}^{n} \{l_i = r_i\}$ such that $\mathcal{E}$ is a convergent set of rules such that each $r_i$ is either a proper subterm of $l_i$ or a ground term.*

It is known (see e.g. [2]) that reachability is decidable for subterm convergent theories. It was proved in [1] that any subterm convergent theory has the finite basis property too. This is a consequence of Proposition 11 in [1] that is used by the authors to decide the so-called static equivalence property for this class of theories. We give more details in the Appendix.

Many interesting theories are subterm convergent. For instance consider the Dolev-Yao equational theory:

$$\mathcal{E}_{DY} \left\{ \begin{array}{rcll} \pi_1(\langle x, y \rangle) & = & x & (P_1) \\ \pi_2(\langle x, y \rangle) & = & y & (P_2) \\ \mathrm{dec}(\mathrm{enc}(x,y), y^{-1}) & = & x & (D) \\ \mathrm{symtest}(\mathrm{enc}(x,y), y) & = & \mathrm{TRUE} & (T_e) \\ \mathrm{pairtest}(\langle x, y \rangle) & = & \mathrm{TRUE} & (T_p) \end{array} \right.$$

where $(P_1)$ (resp. $(P_2)$) models the projections on the arguments of a pair, $(D)$ models the decryption using the inverse key and $(T_e)$ (resp. $(T_p)$) models that anyone knowing a public key can test whether a message is encrypted with this key (resp. that anyone can test whether a message is a pair.) As a consequence of our Theorem 2, for every protocol expressed with functions satisfying this theory we can compute a prudent implementation.

The equational theory of the *eXclusive-OR* operator $\cdot \oplus \cdot$ is given by the following set of equations $\mathcal{E}_\oplus$ where $0$ is a constant and $\oplus, 0$ are public functions:

$$\mathcal{E}_\oplus \left\{ \begin{array}{rcl} (x \oplus y) \oplus z & = & x \oplus (y \oplus z) \\ x \oplus y & = & y \oplus x \\ 0 \oplus x & = & x \\ x \oplus x & = & 0 \end{array} \right.$$

This example can be generalized to monoidal theories as follows. Assume that all symbols are public and that the signature of a deduction system is equal to $\mathcal{F} = \{+, 0, h_1, \ldots, h_n\}$ or $\{+, -, 0, h_1, \ldots, h_n\}$ where $+$ is a binary associative-commutative symbol, $0$ is the identity for $+$, the symbol $-$ is unary and satisfies the equation $x + (-x) = 0$ and $h_1, \ldots, h_n$ (for $n \geq 0$) are unary commuting homomorphism on $+$ (*i.e.* such that $h_i(x + y) = h_i(x) + h_i(y)$ and $h_i(h_j(x)) = h_j(h_i(x))$ for $1 \leq i, j \leq n$). Let us add to the signature $a_1, \ldots, a_k$ the constants appearing in the protocol narration.

Reachability for this resulting deduction system is decidable (see e.g. [11, 12]). We can also show that the deduction system has the finite basis property: Each ground term in the narration can be interpreted as an element of the module $(Z[X_1, \ldots, X_n])^k$ as follows:

- $[\![a_i]\!]$ is the vector in which only the $i$-th coordinate is non-null, and is equal to $1$;

- $[\![h_i(t)]\!] = X_i \cdot [\![t]\!]$, and $[\![t_1 + t_2]\!] = [\![t_1]\!] + [\![t_2]\!]$, and $[\![-t]\!] = -[\![t]\!]$.

It is routine to check that under these assumptions, we have that:

1. A context with $m$ holes is interpreted as a linear form mapping $((Z[X_1, \ldots, X_n])^k)^m$ to $(Z[X_1, \ldots, X_n])^k$ and with coefficients in $Z[X_1, \ldots, X_n]$. These polynomials have positive coefficients whenever $-$ is not a public symbol;

2. In any case we note that any linear form with coefficients in $Z[X_1, \ldots, X_n]$ can be written as the difference of two linear forms with positive coefficients.

Under this interpretation for a positive strand $s$ of length $n$ interpreted as a vector in $(Z[X_1, \ldots, X_n]^k)^n$ and a pair of contexts $C_1, C_2$ we have $[\![C_1 \cdot s = C_2 \cdot s]\!]$ iff $([\![C_1]\!] - [\![C_2]\!])([\![s]\!]) = 0$, *i.e.* there is a mapping from $P_s$ to the set $s^*$ of linear forms $f$ such that $f([\![s]\!]) = 0$. The second

remark above shows that this mapping is surjective. Since $s^*$ is the first syzygy module [21] of a linear equation, $s^*$ is also isomorphic to a submodule of $(Z[X_1, \ldots, X_n]^k)^n$. Since $Z[X_1, \ldots, X_n]$ is noetherian this syzygy submodule has a finite generating set $b_1, \ldots, b_l$ that can be computed by an analogous of Buchberger's algorithm [21].

Given another strand $s'$ of the same length, if $\{b_1, \ldots, b_l\} \subseteq (s')^*$ we have also $s^* \subseteq (s')^*$. In other words we have that $s'$ refines $s$. We thus obtain an algorithm to compute $P_s^f$ for any strand $s$ that consists in computing a generating set $b_1, \ldots, b_l$ of $s^*$, write each $b_i$ as the difference $b_i^+ - b_i^-$ of two linear forms with positive coefficients, and output a set of $n$ pairs of contexts $(C_i^+, C_i^-)$ with $[\![C_i^+]\!] = b_i^+$ and $[\![C_i^-]\!] = b_i^-$ for $1 \le i \le n$.

# 6  Conclusion

We have shown how to link the process of compiling protocols to excutable roles with formal decision problems. This allows us to extend many known results on compilation to the case of protocols that are based on more complex cryptographic primitives, admitting algebraic properties that are beyond the usual Dolev Yao ones.

Moreover if the set of symbols occuring in the protocol can be divided so that each part satisfies an equational theory with decidable $\mathcal{D}$-reachability and $\mathcal{D}$ has the finite basis property, then we can exploit the combination results from [9, 3] to derive the same properties for the union of theories. Therefore the protocol can be prudently compiled in this case too.

# References

[1] M. Abadi, V. Cortier, Deciding knowledge in security protocols under equational theories., Theor. Comput. Sci. 367 (1-2) (2006) 2–32.

[2] S. Anantharaman, P. Narendran, M. Rusinowitch, Intruders with Caps in: Proceedings of RTA'07, LNCS 4533.

[3] M. Arnaud, V. Cortier, S. Delaune, Combining Algorithms for Deciding Knowledge in Security Protocols. FroCos 2007, pp. 103-117 LNCS 4720.

[4] F. Baader, Unification in Commutative Theories, Hilbert's Basis Theorem, and Gröbner Bases. Journal of the ACM 40 (3): 477–503 (1993)

[5] P. Bieber, A Logic of Communication in Hostile Environments. Proceedings of the 1990 IEEE Computer Security Foundations Workshop, pp. 14-22

[6] C. Caleiro, L. Viganò, D. Basin, On the Semantics of Alice&Bob Specifications of Security Protocols. Theoretical Computer Science 367 (1–2): 82–122 (2006)

[7] U. Carlsen, Generating Formal Cryptographic Protocol Specifications Proceedings of the 1994 IEEE Symposium on Security and Privacy.

[8] Y. Chevalier, L. Vigneron, A Tool for Lazy Verification of Security Protocols, in: Proceedings of the Automated Software Engineering Conference (ASE'01), IEEE Computer Society Press, 2001, pp. 373–376.

[9] Y. Chevalier, M. Rusinowitch, Combining intruder theories, ICALP, LNCS 3580, Springer, 2005, pp. 639–651.

[10] Y. Chevalier. Résolution de problèmes d'accessibilité pour la compilation et la validation de protocoles cryptographiques. Thèse de doctorat, Université Henri Poincaré, Nancy, décembre 2003.

[11] Y. Chevalier, R. Küsters, M. Rusinowitch, M. Turuani, "An NP Decision Procedure for Protocol Insecurity with XOR". In *Proceedings of the Logic In Computer Science Conference, LICS'03*, pages 261–270, 2003.

[12] V. Cortier, S. Delaune, Deciding Knowledge in Security Protocols for Monoidal Equational Theories. LPAR 2007: 196-210

[13] D. Dolev, A. Yao, On the Security of Public-Key Protocols, IEEE Transactions on Information Theory 2 (29).

[14] F. Javier T. Fabrega, Jonathan C. Herzog, Joshua D. Guttman. Strand Spaces: Why is a Security Protocol Correct? Proceedings, 1998 IEEE Symposium on Security and Privacy, May 1998.

[15] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and Verifying Security Protocols. Logic for Programming and Automated Reasoning, LNCS 1955, pp. 131-160, St Gilles (Réunion, France), November 2000.

[16] G. Lowe, An Attack on the Needham-Schroeder Public-Key Authentication Protocol. Inf. Process. Lett. 56 (3): 131-133 (1995)

[17] G. Lowe, Casper: A Compiler for the Analysis of Security Protocols. Journal of Computer Security 6(1-2): 53-84 (1998)

[18] J. McCarthy, J. D. Guttman, J. D. Ramsdell, and S. Krishnamurthi. Compiling Cryptographic Protocols for Deployment on the Web. In World Wide Web, pp. 687-696, 2007.

[19] J. McCarthy, Shriram Krishnamurthi. Cryptographic Protocol Explication and End-Point Projection. ESORICS 2008: 533-547.

[20] J. Millen and G. Denker. CAPSL and MuCAPSL. J. Telecommunications and Information Technology, 4/2002, pp. 16-27.

[21] L.J. Miller. Analogs of Gröbner Bases in Polynomial Rings over a Ring. Journal of Symbolic Computation. Volume 21, Issue 2, February 1996, Pages 139-153

[22] S. Mödersheim. "Algebraic Properties in Alice and Bob Notation", Proceedings of Ares 2009. An extended version is available as Technical Report no.RZ3709, IBM Zurich Research Lab, 2008.

[23] R. Needham, M. Schroeder: Using Encryption for Authentication in Large Networks of Computers. Commun. ACM 21(12): 993-999 (1978)

## Appendix

This Appendix has been added to ease the review. We recall some notions and results from [1] and explain why they show that any subterm convergent theory has the finite basis property.

Let $E$ be a subterm convergent theory. The constant $c_E$ introduced in [1] depends only from the equational theory $E$ but its exact value is not important for our discussion. The size of a

term $t$ is the number of vertices in its DAG representation. It is denoted by $|t|$. To any positive strand $s = (!M_1, \ldots, !M_n)$ we can associate a frame with an empty set of free names in the sense of [1]. This frame is $\{M_1/x_1, \ldots, M_n/x_n\}$ and will be denoted $s$ too (assuming some variable enumeration). We will reformulate or simplify the results from [1] by taking into account the fact that there are no nonces in the frames in our case. Note that $s$ can also be viewed as a substitution.

Let $st(s)$ be the set of subterms of $s$. The set $sat(s)$ (see Definition 3 [1]) is the minimal set such that

1. $M_1, \ldots, M_n \in sat(s)$;

2. if $N_1, \ldots, N_k \in sat(s)$ and $f(N_1, \ldots, N_k)$ is a subterm of $s$ then $f(N_1, \ldots, N_k) \in sat(s)$;

3. if $N_1, \ldots, N_k \in sat(s)$ and $C[N_1, \ldots, N_k] \to M$ where $C$ is a context, $|C| \leq c_E$ and $M$ in $st(s)$ then $M \in sat(s)$,

Also Proposition 9 from [1] shows that for every $M \in sat(s)$ there exists a term $\zeta_M$ such that $|\zeta_M|_{DAG} \leq c_E \cdot |s|$ and $\zeta_M \cdot s =_E M$.

We can now restate Definition 4 from [1] in our framework:

**Definition 4** *The set $Eq(s)$ is the set of couples:*

$$(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}], C_2[\zeta_{M_1'}, \ldots, \zeta_{M_l'}])$$

*such that $(C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] =_E C_2[\zeta_{M_1'}, \ldots, \zeta_{M_l'}]) \cdot s$, $|C_1|, |C_2| \leq c_E$ and the terms $M_i, M_i'$ are in $sat(s)$.*

Since there are no nonces the set $Eq(s)$ is finite (up to variable renamings).

We recall Lemma 6 and 7 from [1] with our notations:

**Lemma 8** *Let $s, s'$ be two positive strands such that $Eq(s) \subseteq P_s'$. Then for all contexts $C_1, C_2$ and for all terms $M_i, M_i' \in sat(s)$ if $C_1[M_1, \ldots, M_k] = C_2[M_1', \ldots, M_l']$ then $C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] \cdot s' =_E C_2[\zeta_{M_1'}, \ldots, \zeta_{M_l'}] \cdot s'$.*

**Lemma 9** *Let $s$ be a positive strand. For every context $C_1$, for every $M_i \in sat(s)$ for every term $T$ such that $C_1[M_1, \ldots, M_k] \to_E^* T$ there is a context $C_2$ and terms $M_i' \in sat(s)$ such that $T = C_2[M_1', \ldots, M_l']$ and for every positive strand $s'$ such that $Eq(s) \subseteq P_s'$ we have $C_1[\zeta_{M_1}, \ldots, \zeta_{M_k}] \cdot s' =_E C_2[\zeta_{M_1'}, \ldots, \zeta_{M_l'}] \cdot s'$.*

Now we can now extract from Proposition 11 in [1] the part of the proof that shows our claim:

Assume that $s, s'$ are two positive strands and $Eq(s) \subseteq P_s'$. Assume that we have an equality: $M \cdot s =_E N \cdot s$. Let $T$ be the common normal form of $M \cdot s$ and $N \cdot s$ for the rewrite relation $\to_E$.

By Lemma 7 there exists $M_i \in sat(s)$ and $C_M$ such that
$T = C_M[M_1, \ldots, M_k]$ and $M \cdot s' =_E C_M[\zeta_{M_1}, \ldots, \zeta_{M_k}] \cdot s'$.
By the same lemma there exists $M_i' \in sat(s)$ and $C_N$ such that
$T = C_N[M_1', \ldots, M_l']$ and $N \cdot s' =_E C_N[\zeta_{M_1'}, \ldots, \zeta_{M_l'}] \cdot s'$.

Since $C_M[M_1, \ldots, M_k] = C_N[M_1', \ldots, M_l']$ we derive from Lemma 6 that:

$$C_M[\zeta_{M_1}, \ldots, \zeta_{M_k}] \cdot s' =_E C_N[\zeta_{M_1'}, \ldots, \zeta_{M_l'}] \cdot s'$$

As a consequence we have $M \cdot s' = N \cdot s'$. We can conclude that $P_s \subseteq P_s'$. and that the deduction system has the finite basis property by defining for all $s$, $P_s^f$ to be $Eq(s)$.