A Dolev-Yao model for Zero Knowledge

A. Baskar, R. Ramanujam, and S. P. Suresh

[1] Chennai Mathematical Institute, Chennai. abaskar@cmi.ac.in
[2] Institute of Mathematical Sciences, Chennai. jam@imsc.res.in
[3] Chennai Mathematical Institute, Chennai. spsuresh@cmi.ac.in

Abstract. In cryptographic protocols, zero knowledge proofs are employed for a principal A to communicate some non-trivial information t to B while at the same time ensuring that B cannot derive any information "stronger" than t. Often this implies that B can verify that some property holds without being able to produce a proof of this. While a rich theory of zero knowledge proofs exists, there are few formal models addressing verification questions. We propose an extension of the standard Dolev-Yao model of cryptographic protocols which involves not only constructibility of terms but also a form of verifiability. We present a proof system for term derivability, which is employed to yield a decision procedure for checking whether a given protocol meets its zero knowledge specification.

1 Introduction

Zero-knowledge proofs were introduced by [GMR89] and were immediately seen to be an attractive technique for implementing information hiding. [GMW91] gave a variety of zero-knowledge proofs demonstrating the applicability of the technique and today, it is extensively used in electronic voting protocols, contract signing protocols and designated verifier proofs ([JCJ05], [LBD+04], [CGS97]).

On the other hand, security protocols are known to be difficult to design and hard to analyze, principally due to the concurrent execution of such protocols, leading to information transfer across many interleaved runs. Discovery of design flaws in early key distribution and authentication protocols ([NS78], [Low96]) led to the advent of formal methods in verification of cryptographic protocols ([MS01], [RT03], [CLS03], to name a few references). A key abstraction in such employment of formal methods is that of the *Dolev-Yao* model, in which cryptographic operations are idealized and proofs can be carried out without the complications relating to implementation of cryptographic primitives, random numbers and error probabilities. Symbolic abstraction has proved to be useful in the study of a range of security protocols (see [Bel03], [DKR09], [Cre08], for instance). Recent research has further demonstrated that such symbolic models can be proved to be sound with repect to underlying computational models (see, for instance, [AR02], [Her05], [CKKW06]) thus achieving a satisfactory two-level layering of security proofs.

The standard Dolev-Yao model offers a term algebra with operators for encryption and pairing. Recently, a number of extensions have been studied in different contexts [CDL06]. The model has also been extended to include analysis of flaws such as guessing attacks [Bau05]. In this context, it is natural to consider extensions of the Dolev-Yao model for analysis of cryptographic protocols that employ zero-knowledge proofs as well.

Two approaches suggest themselves. One is to symbolically represent a zero-knowledge proof as an explicit constructor in the term algebra, say *zk* with appropriate parameters, and study protocols using such terms in communications. Separately, one ensures that guarantees given by proofs with

such symbolic abstraction are applicable to protocol implementations where the abstractions are realised by cryptographic zero-knowledge proofs. This is the line of work successfully carried out by [BMU08] and [BU08].

Another approach is to so extend the Dolev-Yao model such as to formalize aspects of cryptographic zero-knowledge proof construction as well, within the symbolic abstraction. The underlying idea is that an idealization of the sequence of operations that constitute a cryptographic zero-knowledge proof is itself possible within such an extended Dolev-Yao model. This is in the spirit of logical studies of security protocols, where we wish to identify patterns of reasoning relating to information transfer and hiding. The advantage of such an approach is that computational soundness of underlying abstractions can be 'lifted' to the construction. This is the approach we follow in this paper.

In general, a zero-knowledge proof is a proof of a statement α. A principal A tries to convince another principal B that α is true, but in such a way that for any β such that $\beta \supset \alpha$ and $\neg(\alpha \supset \beta)$, B does not receive any new information as to whether β holds or not. In the context of Dolev-Yao models, α refers to some property of a set of terms X without revealing X itself.

This is the central idea of our abstraction: when a term t is constructible by a principal A in the Dolev-Yao model, she has complete and certain information about t. If she has a way to send t to B whereby B can verify some property of t (for instance t has a pattern as substructure) without getting to know t itself, we can build such capabilities to stitch zero-knowledge proofs together.

How do we bring such a capability in a formal model? We employ another standard notion, that of a **typed term**. This is of the form $t \Downarrow p$, where t is a **term** formed using the operations of pairing and encryption starting from atomic terms, while p is a **pattern** built using pairing, encryption, and disjunction starting from atomic terms, and $\Box$ (denoting an unknown term).

Consider the following example protocol, stated using such typed term:

$$V \to A : \{(v, r)\}_{public(C)} \Downarrow \{(0 \oplus 1, \Box)\}_{public(C)}$$
$$A \to V : \{\{(v, r)\}_{public(C)}\}_{private(A)} \Downarrow \Box$$
$$A \to C : \{\{(v, r)\}_{public(C)}\}_{private(A)} \Downarrow \Box$$

The first message is typed with a disjunctive pattern. The rest of the communications are typed with the pattern $\Box$, to indicate the fact that no non-trivial type information is being passed about them. In this manner, zero knowledge proofs are not communicated, but instead the content of the proof is expressed as a typing for the appropriate term.

We now sketch further examples:

- If someone has provided a proof that a term t is of the form $\{t'\}_k$ (for a fixed k, and arbitrary t'), this might be represented by the typed term $t \Downarrow \{\Box\}_k$.
- A proof of the fact that two terms t and t' are encrypted using the same key k might be represented by $(t, t') \Downarrow (\{\Box\}_k, \{\Box\}_k)$.
- That t is either $\{0\}_k$ or $\{1\}_k$ is represented by $t \Downarrow \{0\}_k \oplus \{1\}_k$.
- That either t or t' is of the form r can be represented by (for instance) $(t, t') \Downarrow (r, \Box) \oplus (\Box, r)$.
- That t is encrypted by either k or k' can be represented by $\{t\}_k \Downarrow \{t\}_{k \oplus k'}$. If t is a term known to A (a nonce generated by A, say) and the inverse of key is also known to her, then she can in fact

verify more of the structure of $\{t\}_k$ (in particular, that the given term is indeed encrypted using k and not k'). Such considerations are important in modelling examples like the whistle blower's problem [Cla].

In general, protocols in our model communicate a typed term $t \Downarrow p$ where p is a disjunctive pattern $p_1 \oplus p_2$. The typed term should be derivable by the sender, but the receiver can in general not resolve the disjunction. Then interesting questions about "leak of knowledge" can be asked. If the protocol intends only $t \Downarrow p$ to be known by an agent A, but she gets to know $t \Downarrow p'$ for a "stronger pattern" p', that would constitute a leak.

Thus the central contribution of the paper is the setting up of a proof system for when a given typed term is derivable from a set of typed terms, and its use in proving the decidability of the verification problem for "information leakage". What is notable about this system is that we do non-trivial reasoning with disjunction and contradiction. In contrast, the typical proof systems that one encounters in relation to security protocols essentially reason with conjunction and some form of implication, with some algebraic rules.

2 The term model and derivations

In this section, we present the basic term model, and also the rules for deriving typed judgements.

Fix a finite set of **agents** Ag, which includes the **intruder** I, which is an abstraction that quantifies over the malicious forces at work to compromise security. The intruder is assumed to have unbounded memory, has access to all that travels on the public channel, can forge and block messages.

Fix a countable set of **fresh secrets** $\mathcal{N}$. (This includes *random, unguessable nonces* as well as *temporary session keys*.) $\mathcal{B} = \mathcal{N} \cup Ag$ is the set of **basic terms**. The set of *keys*, $\mathcal{K}$, is defined to be $\mathcal{N} \cup \{public(A), private(A), shared(A,B) | A, B \in Ag\}$. Here $public(A)$, $private(A)$, and $shared(A,B)$ denote the public key of A, private key of A, and (long-term) shared key between A and B, respectively. We define $inv(k)$ for every $k \in \mathcal{K}$ as follows: $inv(public(A)) = private(A)$, $inv(private(A)) = public(A)$, and $inv(k) = k$ for every other $k \in \mathcal{K}$. The set $\mathcal{T}$ of **terms** is given by the following syntax:

$$\mathcal{T} ::= m \mid (t, t') \mid \{t\}_{t'}$$

where m ranges over $\mathcal{K} \cup Ag$, and t, t' over $\mathcal{T}$. Here (t, t') denotes the pair consisting of t and t', and $\{t\}_{t'}$ denotes the term t encrypted using t'.

Given a term, its set of subterms $st(t)$ is defined in the usual manner: $st(m) = m$; $st((t, t')) = \{(t, t')\} \cup st(t) \cup st(t')$; $st(\{t\}_{t'}) = \{\{t\}_{t'}\} \cup st(t) \cup st(t')$. The notion of **inverse** is extended to all terms t by letting $inv(t) = t$ for all $t \notin \mathcal{K}$.

Terms conform to certain **patterns**. A pattern has the same structure as a term, except for two important differences: we use a special pattern $\square$ to signify that nothing is known about the structure of a given term; and we use *disjunction*, in the form $p \oplus p'$, to signify that a given term has either the structure specified in p or the structure specified in p'. Patterns are given by the following syntax:

$$\mathcal{P} ::= \square \mid m \mid (p, p') \mid \{p\}_{p'} \mid p \oplus p'$$

where $m \in \mathcal{K} \cup Ag$ and $p, p' \in \mathcal{P}$.

We extend the notion of inverse to an arbitrary basic pattern $p \in \mathscr{P}$ by letting $inv(p) = p$ if $p \notin \mathscr{K}$ and p is not of the form $p_0 \oplus p_1$. We define $inv(p \oplus p')$ to be $inv(p) \oplus inv(p')$. The notion of **subpatterns** of a pattern p, $st(p)$, is also defined in the usual manner.

A **typed term** is of the form $t \Downarrow^i p$ where t is a term, p is a pattern, and $i \in \{0, 1\}$. (We will explain the need for the superscript i shortly.)

Given two patterns p and q, we define when p is **incompatible** with q (in symbols: $p\#q$) by the following rules:

if $m \neq n$ then $m\#n$ for $m, n \in \mathscr{B}$
$m\#(q, q')$ and $m\#\{q\}_{q'}$ for $m \in \mathscr{B}$
$(p, p')\#\{q\}_{q'}$
$p\#q \oplus q'$ if $p\#q$ and $p\#q'$
$p\#q$ if $q\#p$

We say that a term t is **compatible** with a pattern p if $\neg(t\#p)$. We say that a typed term $t \Downarrow^i p$ is **well-formed** if t is compatible with p.

We next describe rules that let one derive new terms from old. The rules, given in Figure 1, involve **sequents** of the form $X \vdash t \Downarrow^i p$ where $X \cup \{t \Downarrow^i p\}$ is a set of typed terms. We use $X, t \Downarrow^i p$ as a shorthand for $X \cup \{t \Downarrow^i p\}$. We use $X \vdash t \Downarrow^i p$ to also denote the fact that the sequent $X \vdash t \Downarrow^i p$ has a derivation. We let $\overline{X} = \{t \Downarrow^i p \mid X \vdash t \Downarrow^i p\}$.

Some of the rules are routine and some are quite complicated. We discuss the important aspects of the proof system in some detail.

1. A has complete knowledge of the terms she has generated, and thus for such **constructed terms** t, she has $t \Downarrow^1 t$ in her database. She believes in the typing judgement sent across by the other agents, presumably because the typing judgement comes with a zero knowledge proof which A can verify. For such **verified terms** t, she has $t \Downarrow^0 p$ in her database.
2. Constructed terms are automatically deemed to have been verified, but not all verified terms need be constructed.
3. Only constructed terms can be used in constructing further new terms, while verified terms can be used to verify the structure of further terms.
4. It is possible to verify that a given term has more structure than what is revealed by the sender. For instance, if A has received a typed term $\{m\}_k \Downarrow \{\Box\}_{k \oplus k'}$ where m is a term already constructed by her, and decrypts this term using the inverse of k (which she has generated or constructed), she can verify that the term is of type $\{m\}_k$.
5. Important to inferring more structure for a given term is the disjunction elimination rule. We just rule out some of the cases that lead to a contradiction.
6. We do not have a rule for disjunction introduction, or any other form of weakening of types, to keep the proof system simpler. But it is important to allow some form of weakening in the model. This is to allow agents to reveal only some partial structure of a term another agent. We take care of this in the model by saying that a typed message of the form $t \Downarrow^i p$ can be communicated even when $t \Downarrow^i p'$ is known to the sender for a "stronger" pattern p'.

We note two important properties of the proof system below. The proofs are by a more-or-less routine induction on the structure of derivations.

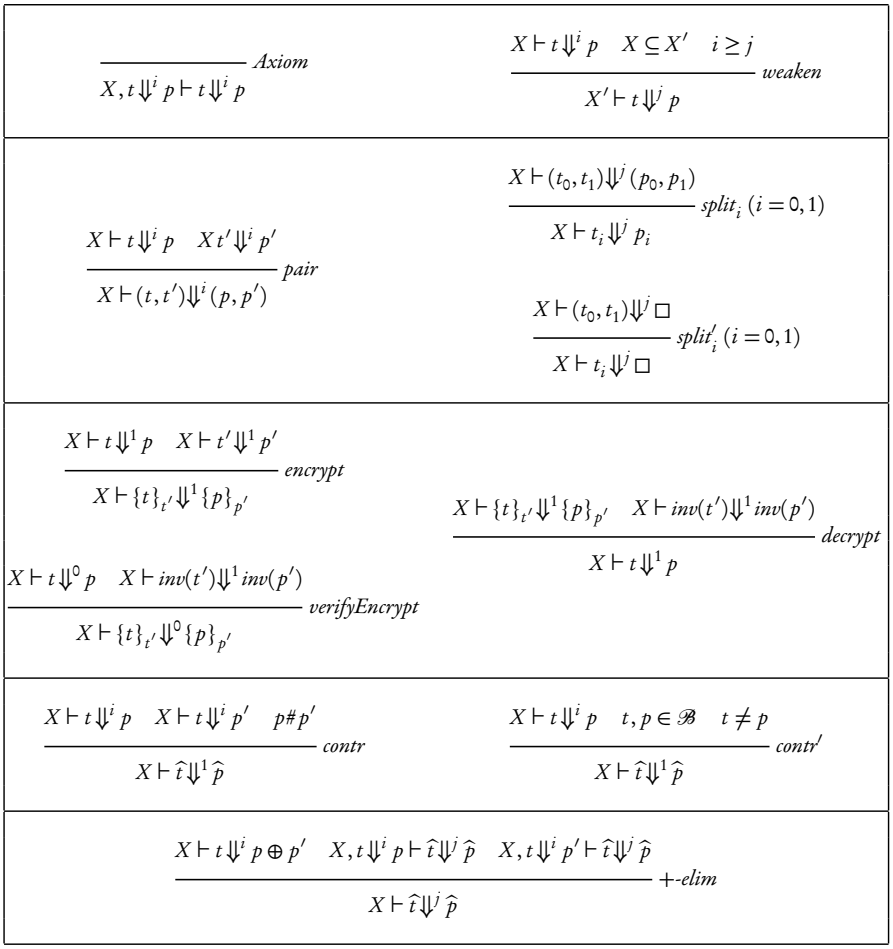

Fig. 1. The derivation rules

Proposition 1. *If X is a set of well-formed typed terms, then any $t\Downarrow^i p\in\overline{X}$ is well-formed.*

Proposition 2. *If X is a set of well-formed typed terms such that for all $u\Downarrow^1 q\in X$, $u=q$, and if $X\vdash t\Downarrow^1 p$, then $t=p$.*

The **derivability problem** asks, given a set of typed terms X and a typed term $t\Downarrow^i p$, whether $t\Downarrow^i p\in\overline{X}$. The following theorem is central to the application of this proof system to protocols. The proof of it will occupy us in Section 4.

Theorem 3. *The derivability problem is decidable.*

3 The protocol model

The model we present here is based on the papers [RS05] and [RS06]. The term model has already been discussed in detail in Section 2. We shall look at the definition of protocols and their runs.

Protocols

Protocols are typically given as a sequence of communications of the form $A \rightarrow B : t$, which denotes the sending of t by A and its receipt by B. But when a protocol is executed, one needs to consider the sends and receives as separate actions. Further, we need notation to indicate when certain nonces are supposed to be generated afresh every time an instance of a message is sent by some agent during a run of the protocol.

The new element in the model in this paper is that we allow typed terms in the communications. This is essential to model the use of zero-knowledge proofs in protocols. The protocol steps which communicate zero-knowledge proofs for a particular fact (that t and t' are both encrypted terms, where the encryption key is k, for instance) is replaced by the communication of a typed term $((t, t') \Downarrow (\{\Box\}_k, \{\Box\}_k)$, in this case). The other terms are typed with the pattern $\Box$.

The important point to note is that the communications of the protocol do not mention whether the terms communicated are constructed or verified. That is just part of the analysis.

We introduce the notion of **actions** first. An action is either a **send action** of the form $A!B : (M)t \Downarrow p$ or a **receive action** of the form $A?B : t \Downarrow p$, where $t \Downarrow p$ is *a well-formed typed term*, and A and B are agent names. By an A-action, we mean $A!B : (M)t \Downarrow p$ or $A?B : t \Downarrow p$, for some B and $t \Downarrow p$. In all the send actions, M is a set of typed terms of the form $m \Downarrow m$ where $m \in st(t)$. These are the nonces supposed to be freshly generated as part of the send action. For an action a of the form $A!B : (M)t \Downarrow p$ or $A?B : t \Downarrow p$, $term(a)$ is defined to be $t \Downarrow p$.

We emphasize that while the sender name in a send action, and a receiver name in a receive action denote the actual agents that send and receive the messages, respectively, in a send action we can only name the *intended receiver*, and in a receive action we can only name the *purported sender*. As we will see later, every send action is an instantaneous receive by the intruder, and similarly, every receive action is an instantaneous send by the intruder.

A **communication** is of the form $A \rightarrow B : (M)t \Downarrow p$. If $c = A \rightarrow B : (M)t \Downarrow p$, then $send(c) = A!B : (M)t \Downarrow p$, and $rec(c) = B?A : t \Downarrow p$.

A **protocol specification** (or simply protocol) Pr is a tuple $(const, c_1 \cdots c_\ell, P, N)$ which satisfies the following conditions:

- for each $i \leq \ell$, c_i is a communication $A_i \rightarrow B_i : (M_i)t_i \Downarrow p_i$
- $\{t_1 \Downarrow p_1, \ldots, t_\ell \Downarrow p_\ell\}$ is a set of well-formed typed terms, and
- $const \subseteq \mathcal{B}$ is a set of **constants** of the protocol,
- P is a set each of whose elements is of the form $(A, t \Downarrow p)$ for some $A \in Ag$ and some typed term $t \Downarrow p$. This specifies the **positive knowledge requirements**.
- N is a set each of whose elements is of the form $(A, t \Downarrow p)$ for some $A \in Ag$ and some typed term $t \Downarrow p$ that occurs in Pr. This specifies the **negative knowledge requirements**.

For ease of notation, we refer to protocols using the sequence of communications. The idea is that $const(Pr)$ should be interpreted the same way throughout any run of the protocol, while the other basic terms can get different interpretations in different **sessions** of a single run of a protocol.

Given a protocol $Pr = c_1 \cdots c_\ell$, one can extract its set of roles $\{\eta_1, \ldots, \eta_n\}$ as follows: consider the sequence of actions $\eta = send(c_1)rec(c_1) \cdots send(c_\ell)rec(c_\ell)$, and for every A that is either a sender or a recipient in the protocol, consider the subsequence of all A-actions in η. This is the A-role of Pr. The

idea is that an agent participating in the protocol can execute many *sessions* of a role in the course of a single run, by instantiating the non-constants in many different ways.

Runs of a protocol

A **substitution** σ is a map from $\mathscr{B}$ to $\mathscr{T}$ such that $\sigma(Ag) \subseteq Ag$ and $\sigma(I) = I$ and $\sigma(\mathscr{N}) \subseteq \mathscr{N}$. We say that a substitution σ is suitable for a protocol Pr if for every basic term m specified to be a constant of the protocol, $\sigma(m) = m$. For an arbitrary term t and pattern p, $\sigma(t)$ and $\sigma(p)$ are defined in the obvious manner. For a typed term $t \Downarrow p$, $\sigma(t \Downarrow p) = \sigma(t) \Downarrow \sigma(p)$.

An **event** of a protocol Pr is a triple $e = (\eta, \sigma, lp)$ where η is a role of Pr, σ is a substitution suitable for Pr, and $1 \leq lp \leq |\eta|$. For events $e = (\eta, \sigma, lp)$ and $e' = (\eta', \sigma', lp')$ of Pr, we say that $e \prec e'$ (meaning that e is in the *local past* of e') if $\eta = \eta'$, $\sigma = \sigma'$, and $lp < lp'$. For an event $e = (\eta, \sigma, lp)$ of Pr, the action of e, $act(e)$ is defined to be $\sigma(a_{lp})$, where $\eta = a_1 \cdots a_k$, and $term(e) = term(act(e))$.

An **state** is a tuple $(s_A)_{A \in Ag}$, where $s_A \subseteq \mathscr{T}$ for each $A \in Ag$. The **initial state** of Pr, denoted by $initstate(Pr)$, is the tuple $(s_A)_{A \in Ag}$ such that for all $A \in Ag$,

$$s_A = \{m \Downarrow^1 m \mid m \in Ag \cup \{private(A), public(A)\} \cup \{public(B), shared(A,B) \mid A \neq B\}\}.$$

We need to define when the various send and receive actions are enabled, and the state updates that happen as a result of the communications. This is on standard lines, but some points need to be highlighted. For a send action to be enabled, all the nonces that are designated as fresh should not be derived by anyone in that state. These will get added to the sender's state as part of the send action. Further, the communicated term should be *constructible* by the agent, that is, $t \Downarrow^1 p$ should be derivable by the agent. But she is allowed to communicate a weaker pattern p. Further, the intruder (who plays the part of the network) gets $t \Downarrow^2 p$ into its state, which will be erased once the message is delivered. This is to model the fact that zero knowledge proofs cannot be replayed.

The notions of an action **enabled** at a state, and $update(s, a)$, the **update** of a state s on an action a, are defined as follows:

- A send action a of the form $A!B : (M)t \Downarrow p$ is **enabled** at any state s iff $t \Downarrow^1 t \in \overline{s_A \cup M}$, and for all $m \Downarrow m \in M$, $m \Downarrow^i m \notin \overline{s_C}$ for every C (including A).
- A receive action $a = A?B : t \Downarrow p$ is enabled at s iff one of the following two cases holds:
  - $t \Downarrow^2 p \in s_I$,
  - $t \Downarrow^1 t \in \overline{s_I}$.
- $update(s, A!B : (M)t \Downarrow p) = s'$ where $s'_A = s_A \cup M$, $s'_I = s_I \cup \{t \Downarrow^2 p\}$, and $s'_C = s_C$ for $C \neq I$.
- $update(s, A?B : t \Downarrow p) = s'$ where $s'_B = s_B \cup \{t \Downarrow^0 p\}$, $s'_I = (s_I \setminus t \Downarrow^2 p) \cup t \Downarrow^0 p$, and $s'_C = s_C$ for $C \notin \{B, I\}$.

$update(s, \eta)$ for a state s and a sequence of actions η is defined in the obvious manner. Given a protocol Pr and a sequence of its events ξ, $infstate(\xi)$ is defined to be $update(initstate(Pr), act(\xi))$.

Given a protocol Pr, a sequence $e_1 \cdots e_k$ of events of Pr is said to be an **run** of Pr iff the following conditions hold:

for all $i, j \leq k$ such that $i \neq j$, $e_i \neq e_j$,
for all $i \leq k$ and for all $e \prec e_i$, there exists $j < i$ such that $e_j = e$, and
for all $i \leq k$, $act(e_i)$ is enabled at $infstate(e_1 \cdots e_{i-1})$.

A run $\xi = e_1 \cdots e_k$ of Pr (with $e_i = (\eta_i, \sigma_i, lp_i)$ for each $i \leq k$) is a b-bounded run (for b a natural number) if $|\{(\eta_i, \sigma_i) \mid i \leq k\}| \leq b$.

4 The decidability of the proof system

We follow the standard approach of reducing every derivation in our system to a normal derivation, and then proving a **subterm property** for normal proofs. This bounds the size of normal proofs for a given sequent $X \vdash t \Downarrow^i p$. Thus we only need to search over a bounded set of proofs to check whether $t \Downarrow^i p \in \overline{X}$.

The normalization rules are quite standard. We basically avoid detours (an application of the *split* rule immediately following an application of the *pair* rule, for example), and permute the application of rules so that no major premise of any rule of a proof is the conclusion of a disjunction elimination or a contradiction rule. We say that π is a **normal proof** if no further normalization rules can be applied to it. A representative sample of the normalization rules is given below:

– Replace any proof of the form

$$\dfrac{\dfrac{\dfrac{\vdots\,\pi}{X \vdash t \Downarrow^i p} \quad \dfrac{\vdots\,\pi'}{X \vdash t' \Downarrow^i p'}\,pair}{X \vdash (t,t') \Downarrow^i (p,p')}\,split_0}{X \vdash t \Downarrow^i p}$$

by the following proof:

$$\dfrac{\vdots\,\pi}{X \vdash t \Downarrow^i p}$$

– Replace any proof of the form

$$\dfrac{\dfrac{\dfrac{\vdots\,\pi}{X \vdash t \Downarrow^1 p} \quad \dfrac{\vdots\,\pi'}{X \vdash t' \Downarrow^1 p'}\,encrypt}{X \vdash \{t\}_{t'} \Downarrow^1 \{p\}_{p'}} \quad \dfrac{\vdots\,\varpi}{X \vdash inv(t') \Downarrow^1 inv(p')}\,decrypt}{X \vdash t \Downarrow^1 p}$$

by the following proof:

$$\dfrac{\vdots\,\pi}{X \vdash t \Downarrow^1 p}$$

– Replace any proof of the form

$$\dfrac{\dfrac{\dfrac{\vdots\,\pi}{X \vdash t \Downarrow^i p \oplus p'} \quad \dfrac{\vdots\,\varpi}{X, t \Downarrow^i p \vdash u \Downarrow^j q} \quad \dfrac{\vdots\,\varpi'}{X, t \Downarrow^i p' \vdash u \Downarrow^j q}\,+\text{-}elim}{X \vdash u \Downarrow^j q} \quad \dfrac{\vdots\,\pi'}{X \vdash u' \Downarrow^j q'}\,\mathsf{r}}{X \vdash \widehat{u} \Downarrow^j \widehat{q}}$$

by the following proof:

$$\dfrac{\begin{array}{c}\vdots\,\pi\\ X\vdash t\Downarrow^i p\oplus p'\end{array}\qquad \dfrac{\begin{array}{c}\vdots\,\varpi\\ X,t\Downarrow^i p\vdash u\Downarrow^j q\end{array}\quad\begin{array}{c}\vdots\,\pi'\\ X\vdash u'\Downarrow^j q'\end{array}}{X\vdash \widehat{u}\Downarrow^j\widehat{q}}\,\mathsf{r}\qquad \dfrac{\begin{array}{c}\vdots\,\varpi'\\ X,t\Downarrow^i p'\vdash u\Downarrow^j q\end{array}\quad\begin{array}{c}\vdots\,\pi'\\ X\vdash u'\Downarrow^j q'\end{array}}{X\vdash \widehat{u}\Downarrow^j\widehat{q}}\,\mathsf{r}}{X\vdash \widehat{u}\Downarrow^j\widehat{q}}\,{+}\textit{-elim}$$

– Replace any proof of the form

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi\\ X\vdash t\Downarrow^i p\end{array}\qquad t,p\in\mathscr{B}\quad t\neq p}{X\vdash u\Downarrow^1 q}\,\textit{contr}'\qquad\begin{array}{c}\vdots\,\pi'\\ X\vdash u'\Downarrow^1 q'\end{array}}{X\vdash \widehat{u}\Downarrow^1\widehat{q}}\,\mathsf{r}$$

by the following proof:

$$\dfrac{\begin{array}{c}\vdots\,\pi\\ X\vdash t\Downarrow^i p\end{array}\qquad t,p\in\mathscr{B}\quad t\neq p}{X\vdash \widehat{u}\Downarrow^1\widehat{q}}\,\textit{contr}'$$

Theorem 4. *Every proof can be converted to a normal proof.*

We omit a proof of this fact. It is an easy adaptation of a standard proof that can be found in [GLT89], for instance. The following theorem states an important property of normal proofs.

Theorem 5. *Let π be a normal proof of $X\vdash t\Downarrow^i p$ and let $u\Downarrow^j q$ be a typed term occurring in π. Then $r\in st(X\cup\{t\})$.*

Proof. We will only sketch the proof, as it involves a standard argument. We shall prove a stronger statement: if $X\vdash u\Downarrow^j q$ is the conclusion of a *split* rule or a *decrypt* rule, $r\in st(X)$, and if it is the conclusion of some other rule, $r\in st(X\cup\{t\})$.

We crucially use the fact that in a normal proof, the conclusion $u\Downarrow^j q$ of a *contr* or *contr'* or $+$*-elim* rule is not the major premise of any other rule. This means that u is either the conclusion t, or the minor premise of the *decrypt* rule or the $+$*-elim* rule. But this means that u is a subterm of the major premise of that rule.

Another thing worth noting is that, since there is no disjunction introduction rule, the major premise $v\Downarrow^k r$ of a disjunction elimination rule is the conclusion of a *decrypt* rule or a *split* rule, and hence $v\in st(X)$. Similarly if $v\Downarrow^k r$ is the major premise of a *decrypt* rule, $v\in st(X)$.

Based on these observations, a straightforward induction on the structure of derivations helps us prove the theorem. ⊣

We have bounded the set of terms that can occur in a proof of $X\vdash t\Downarrow^i p$, but what about the set of *typed terms*? We show below that this is also bounded. More specifically, given a set of typed terms X, and a term t, we show that the number of patterns p such that $X\vdash t\Downarrow^i p$ is bounded.

Given a set of typed terms X, we define $closure_1(X)$ to be the least set such that:

- $X \subseteq closure_1(X)$,
- if $(t, t') \Downarrow^i (p, p') \in closure_1(X)$, then $t \Downarrow^i p, t' \Downarrow^i p' \in closure_1(X)$,
- if $\{t\}_{t'} \Downarrow^i \{p\}_{p'} \in closure_1(X)$, then $t \Downarrow^i p, t' \Downarrow^i p' \in closure_1(X)$, and
- if $t \Downarrow^i p \oplus p' \in closure_1(X)$, then $t \Downarrow^i p, t \Downarrow^i p' \in closure_1(X)$.

Given a set of typed terms X and a term t, we define $closure_2(X, t)$ as follows:

- $closure_1(X) \subseteq closure_2(X, t)$,
- if $(t, t') \in st(X \cup \{t\})$ and if $t \Downarrow^i p, t' \Downarrow^i p' \in closure_2(X, t)$, then $(t, t') \Downarrow^i (p, p') \in closure_2(X, t)$, and
- if $\{t\}_{t'} \in st(X \cup \{t\})$ and if $t \Downarrow^i p, t' \Downarrow^i p' \in closure_2(X, t)$, then $\{t\}_{t'} \Downarrow^i \{p\}_{p'} \in closure_2(X, t)$.

We make a couple of observations that are routine to prove:

Lemma 6. *$closure_1(closure_2(X, t)) \subseteq closure_2(X, t)$, and $closure_2(closure_2(X, t), t) \subseteq closure_2(X, t)$.*

Lemma 7. *$|closure_1(X)| \leq m \cdot |X|$ for some constant m. $|closure_2(X, t)| \leq (|X \cup \{t\}|)^d$ where d is the maximum depth of any term in $st(X \cup \{t\})$.*

Theorem 8. *Let X be a set of typed terms and t be a term. Let π be a normal proof of $X \vdash t \Downarrow^i p$ for some p and i. Then for any typed term $u \Downarrow^j q$ occurring in π, $u \Downarrow^j q \in closure_2(X, t)$.*

This is again a straightforward proof by induction on the structure of a normal proof.

Theorem 3 is an immediate consequence of the above theorem. There is a standard deterministic algorithm that checks, given X and $t \Downarrow^i p$, whether the term is derivable from X in time polynomial in the size of $closure_2(X, t)$.

Application to the information leakage problem

We describe the **information leakage problem** below. But for that we need a definition of when one pattern p is stronger than another pattern q. We shall formally define it below, but the idea is that for all terms t, if t is compatible with p, it is also compatible with q, but there is at least one term t' that is compatible with q but incompatible with p.

We first define when p and q are **equally strong** (in symbols: $p \sim q$). $\sim$ is the smallest congruence on patterns such that

$\Box \sim \Box \oplus p$,
$p \sim p \oplus p$,
$(p \oplus p', q) \sim (p, q) \oplus (p', q)$,
$(q, p \oplus p') \sim (q, p) \oplus (q, p')$,
$\{p \oplus p'\}_q \sim \{p\}_q \oplus \{p'\}_q$, and
$\{q\}_{p \oplus p'} \sim \{q\}_p \oplus \{q\}_{p'}$.

We now define when p is **as strong as** q (in symbols: $p \succsim q$). $\succsim$ is the least binary relation over the set of patterns that is reflexive, transitive, and such that:

$p \sim q \Rightarrow p \succsim q$
$p \succsim \Box$,
$p \succsim p \oplus q$,
$p \succsim q, p' \succsim q' \Rightarrow (p, p') \succsim (q, q'), \{p\}_{p'} \succsim \{q\}_{q'}$.

We say that p is **stronger than** q (in symbols: $p \succ q$) if $p \succsim q$ and $\neg(p \sim q)$.

The information leakage problem asks, given a protocol $Pr = (const, c_1 \cdots c_\ell, P, N)$, and a b, whether the following holds:

1. for every $(A, t \Downarrow p) \in P$, there exists a b-bounded run $\xi = e_1 \cdots e_k$ of Pr (with $e_k = (\eta, \sigma, lp)$) and $i \in \{0, 1\}$ such that $\sigma(t) \Downarrow^i \sigma(p) \in \overline{infstate_{\sigma(A)}(\xi)}$, and
2. for every $(A, t \Downarrow p) \in N$, there exists no b-bounded run $\xi = e_1 \cdots e_k$ of Pr such that (letting $e_k = (\eta, \sigma, lp)$), there no **stronger pattern** $p' \succ \sigma(p)$ such that $\sigma(t) \Downarrow^0 p' \in \overline{infstate_{\sigma(A)}(\xi)}$.

Theorem 9. *The information leakage problem is decidable.*

Proof. Much of the proof is on standard lines, except for the verification of the positive and negative knowledge requirements.

We first notice that for any given protocol Pr, the set of b-bounded runs of Pr. Moreover this set can be computed as follows: Consider all sequences of events which are of length at most b' (a bound that is determined by b). Such a sequence is a run if and only if it is admissible. The non-trivial component in the check for admissibility is the check for whether send events are enabled. For this one has to verify that $t \Downarrow^1 t \in \overline{X}$ for an appropriate X (which represents the state at the end of all the previous events in the sequence under consideration). Theorem 3 guarantees that we can effectively perform this test.

Once we compute the set of b-bounded runs, we need to check the knowledge requirements. For a positive specification $(A, t \Downarrow^i p)$, search for a b-bounded run $\xi = e_1 \cdots e_k$ of Pr (with $e_k = (\eta, \sigma, lp)$) such that $\sigma(t) \Downarrow^i \sigma(p) \in \overline{infstate_{\sigma(A)}(\xi)}$. Theorem 3 once again ensures that we can do this test effectively.

For a negative specification $(A, t \Downarrow^i p)$, we need to check all b-bounded runs $\xi = e_1 \cdots e_k$ of Pr (letting $e_k = (\eta, \sigma, lp)$), to ensure that there is no **stronger pattern** $p' \succ \sigma(p)$ such that $\sigma(t) \Downarrow^0 p' \in \overline{infstate_{\sigma(A)}(\xi)}$. Once we fix a run ξ and $X = infstate_A(\xi)$, we need to verify that there is no pattern p' stronger than $\sigma(p)$ is derivable for t from X. In general, there is no bound on the set of stronger patterns than a given pattern p. But we only need to check for patterns derived from X. Theorem 8 assures us that there are only boundedly many patterns for t derivable from X. Now we just need to check if any one them is stronger than $\sigma(p)$. Thus one can effectively verify the negative knowledge requirements as well.

Hence the information leakage problem is decidable. ⊣

References

[AR02] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[Bau05] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 16–25, New York, NY, USA, 2005. ACM.

[Bel03] Giampaolo Bella. Inductive Verification of Smartcard Protocols. *Journal of Computer Security*, 11(1):87–132, 2003.

[BMU08] Michael Backes, Matteo Maffei, and Dominique Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *IEEE Symposium on Security and Privacy*, pages 202–215, 2008.

[BU08] Michael Backes and Dominique Unruh. Computational Soundness of Symbolic Zero-Knowledge Proofs Against Active Attackers. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium*, pages 255–269, 2008.

[CDL06] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology - EUROCRYPT 97*, pages 103–118, 1997.

[CKKW06] Véronique Cortier, Steve Kremer, Ralf Küsters, and Bogdan Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In *FSTTCS*, pages 176–187, 2006.

[Cla] Andrew Clausen. Logical composition of zero-knowledge proofs. Electronic version found in `www.cis.upenn.edu/~mkearns/teaching/Crypto/zkp-disj.pdf`.

[CLS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder Deductions, Constraint Solving and Insecurity Decisions in Presence of Exclusive or. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS)*, pages 271–280, June 2003.

[Cre08] C.J.F. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc.*, volume 5123/2008 of *Lecture Notes in Computer Science*, pages 414–418. Springer, 2008.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2009. To appear.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing but their Validity, or All Languages in NP have Zero-Knowledge Proof Systems. *Journal of the ACM*, 38(1):691–729, 1991.

[Her05] Jonathan Herzog. A computational interpretation of dolev-yao adversaries. *Theoretical Computer Science*, 340(1):57–81, 2005.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 61–70, 2005.

[LBD+04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proceedings of Information Security and Cryptology (ICISC'03)*, volume 2971 of *LNCS*, pages 245–258, 2004.

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public key protocol using FDR. In *Proceedings of TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166, 1996.

[MS01] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[NS78] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.

[RS05] R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

[RS06] R. Ramanujam and S. P. Suresh. A (restricted) quantifier elimination for security protocols. *Theoretical Computer Science*, 367:228–256, 2006.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol Insecurity with Finite Number of Sessions and Composed Keys is NP-complete. *Theoretical Computer Science*, 299:451–475, 2003.