Computational and Symbolic Anonymity in an Unbounded Network

Hubert Comon-Lundh\textsuperscript{1,2}, Yusuke Kawamoto\textsuperscript{3} and Hideki Sakurada\textsuperscript{4}

Ecole Normale Supérieure de Cachan, \textsuperscript{1}
Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, Akihabara-Daibiru Room 1102, 1–18–13, Sotokanda, Chiyoda-ku, Tokyo 101–0021, Japan\textsuperscript{2}
Graduate School of Information Science and Technology, The University of Tokyo, 7–3–1 Hongo, Bunkyo-ku, Tokyo 113–8656, Japan\textsuperscript{3}
NTT Communication Science Laboratories, NTT Corporation, 3–1 Morinosato-Wakamiya, Atsugi, Kanagawa 243–0198, Japan\textsuperscript{4}

E-mail h.comon-lundh at aist.go.jp

Received October 17, 2008, Accepted February 24, 2009 (INVITED PAPER)

Abstract
We provide a formal model for protocols using ring signatures and prove that this model is computationally sound: if there is an attack in the computational world, then there is an attack in the formal (abstract) model. Our original contribution is that we consider security properties, such as anonymity, which are not properties of a single execution trace, while considering an unbounded number of sessions of the protocol.

Keywords computational soundness, security protocols, communicating processes, ring signatures

Research Activity Group Formal Approach to Information Security

1. Introduction

There are two main approaches to protocol security. The first approach considers an attacker modeled as a probabilistic polynomial time interactive Turing machine (PPT) and the protocol is an unbounded number of copies of PPTs. The attacker is assumed to control the network and can schedule the communications and send fake messages. The security property is defined as an indistinguishability game: the protocol is secure if, for any attacker $A$, the probability that $A$ gets an advantage in this game is negligible. A typical example is the anonymity property, by which an attacker should not be able to distinguish between two networks in one of which identities have been switched. The problem with such computational security notions is the difficulty in obtaining detailed proofs: they are in general unmanageable, and cannot be verified by automatic tools.

The second approach relies on a formal model: bit-strings are abstracted by formal expressions (terms), the attacker is any formal process, and security properties, such as anonymity, can be expressed by the observational equivalence of processes. This model is much simpler: there is no coin tossing, no complexity bounds, and the attacker is given only a fixed set of primitive operations (the function symbols in the term algebra). Therefore it is not surprising that security proofs become much simpler and can sometimes be automatized. However, the drawback is that we might miss some attack because the model might be too rough.

Starting with the seminal work of Abadi and Rogaway [1], there have been several results showing the computational soundness of the formal models: we do not miss any attacks when considering the abstract model, provided that the security primitives satisfy certain properties; for instance IND-CPA or IND-CCA in the case of encryption. Such results allow to perform formal symbolic proofs, while yielding computational security guarantees. It is therefore an approach, that is relevant, in principle, to all protocol security proofs.

The present paper is a contribution to this line of research. Until recently, only a few security properties were considered in the soundness results. Roughly speaking, only passive attackers or the properties of execution traces were considered. However, several properties, such as anonymity, cannot be expressed as a property satisfied by all execution traces. Here we consider an active attacker and indistinguishability properties. In [1], the authors only consider a passive attacker and encryption schemes, while we are considering ring signatures and active intruders: we cannot rely on their results in the present paper.

This problem has been discussed in two recent papers. In [2], we reported a soundness result for the anonymity of ring signatures. However, we assumed only a fixed number of instances of the protocol, which is a strong simplification. Furthermore, the symbolic model gave quite a lot of power to the attacker and the soundness proof was dedicated to anonymity. In [3], there are no such restrictions, however the results are limited to symmetric encryption, which does not provide any hint as regards an adequate formal model for ring signatures.

The current paper bridges these two recent studies: we...
consider a formal model for ring signatures and prove the
soundingness of observational equivalence for an unbounded
number of sessions.

2. Ring Signatures

The aim of a ring signature is to enable the verification
without revealing the signer’s identity with a group of
signers.

A ring signature scheme $RS = (G, S, V)$ consists of
two probabilistic algorithms $G$ and $S$, and a determinis-
tic algorithm $V$:

- The key-generation algorithm $G$, given a security
  parameter $1^n$, outputs a private signing key and a
  public verification key.
- The signing algorithm $S$, given a signing key, a set of
  verification keys and a message, outputs a signature
  for the message.
- The verification algorithm $V$, given a set of verifica-
tion keys, a message, and a signature, outputs 0 or
  1.

If a signature is produced by $S$ with keys generated by
$G$, then the verification of the signature always succeeds.

We consider two security notions for ring signature
schemes: existential unforgeability and basic anonymity
[4]. A ring signature scheme $RS$ is existentially unforge-
able if a signature cannot be forged without knowing the
signing key: for any PPT attacker $A$ having access to an
oracle $O$, the following probability is negligible in $n$:

$$\Pr \left[ \left( (sk_1, vk_1) \rightarrow G(1^n) ; \cdots ; (sk_n, vk_n) \rightarrow G(1^n) ; \right) \right.$$

$$\left. \left. L^{\text{leg}} := \{vk_1, \ldots , vk_n\} ; \right. \right.$$

$$\left. (L, m, \sigma) \rightarrow A^D(1^n, L^{\text{leg}}) ; \right. \right.$$

$$\left. L \subseteq L^{\text{leg}} \text{ and } V(L, m, \sigma) = 1 \text{ and} \right. \right.$$

$$\left. \text{for any } i \text{ with } vk_i \in L, \text{ neither } \text{sign}(i, L, m) \right. \right.$$

$$\left. \text{nor } \text{corrupt}(i) \text{ has been queried to } O \right]$$

where the oracle $O$ returns $\sigma \leftarrow S(sk_i, L, m)$ when
queried with $\text{sign}(i, L, m)$ and returns $sk_i$ when queried
with $\text{corrupt}(i)$.

$RS$ is basically anonymous if the signer of a message
cannot be inferred: for any PPT attacker $A$ having access
to an oracle $O$ (as above), the following probability is
negligible in $n$:

$$\Pr \left[ \left( (sk_1, vk_1) \rightarrow G(1^n) ; \cdots ; (sk_n, vk_n) \rightarrow G(1^n) ; \right) \right.$$

$$\left. \left. L^{\text{leg}} := \{vk_1, \ldots , vk_n\} ; \right. \right.$$

$$\left. (i_0, i_1, L, m, \omega) \rightarrow A^D(1^n, L^{\text{leg}}) ; \right. \right.$$

$$\left. b \in \{0, 1\} ; \right. \right.$$

$$\left. \sigma \leftarrow S(sk_{i_0}, L, m) ; \right. \right.$$

$$\left. b' \leftarrow A^D(\omega, \sigma) ; \right. \right.$$

$$\left. \text{neither } \text{corrupt}(i_0) \text{ nor } \text{corrupt}(i_1) \right. \right.$$

$$\left. \text{has been queried to } O ; \right. \right.$$

$$\left. b = b' \right]$$

In addition, we assume unpredictability, which means
that no PPT attacker, even with the signing keys, can
predict the output of the signing algorithm. Unpredic-
tability is also assumed in the soundness of symbolic
zero-knowledge proofs [5]. It is easily obtained by adding
extra random bits to signatures.

3. Symbolic Model

We use a fragment [3] of the applied pi-calculus [6].
Below, we only give the definitions related to ring signa-
tures: for other constructions, refer to [3].

3.1 Terms, predicates and equational theory

The names are split into several disjoint sets:

- identities: $K$: we confuse the identities and the pri-
  vate signing keys held by those identities.
- random symbols: $R$
- nonces: $N$

The set $T$ of ground terms is obtained from the names
by applying the following function symbols, with some
restrictions on the types of their arguments:

- $vk(k)$ constructs a verification key from a signing
  key $k \in K$,
- $(u, v)$ is a pair consisting of two terms $u, v,$
- $\text{check}(u, VK)$ checks the validity of a signature $u$
w.r.t. a set of verification keys $VK = \{vk(k_1), \ldots ,
\text{vk}(k_n)\}$,
- $[u]_k^{VK}$ constructs a signature for $u \in T$ with
  a signing key $k \in K$, verification keys $VK = \{vk(k_1), \ldots ,
\text{vk}(k_n)\}$ and randomness $r \in R$; two
signature terms with the same random symbol $r$
must be identical,
- $\text{RR}(u, r)$ modifies the random number used in a sig-
nature $u$, replacing it with $r \in R$,
- $\pi_1(u), \pi_2(u)$ retrieve the components of a pair.

These function symbols satisfy certain equations,
which we turn into rewrite rules:

$$\left\{ \begin{array}{l}
\pi_1((x, y)) \rightarrow x \\
\pi_2((x, y)) \rightarrow y \\
\text{check}([x]_{y, Z}^y, Z) \rightarrow x \quad \text{if } \text{vk}(y) \in Z \\
\text{RR}([x]_{y, Z}^y, r') \rightarrow [x]_{y, Z}^y \\
\end{array} \right\}$$

This defines an (infinite) convergent term rewriting
system on terms. The normal form of $u$ is written as $u$.
We also introduce predicate symbols that reflect the
(maximal) distinguishing capabilities of an attacker:

- $M$ is the well-formedness predicate on ground terms:
  $M(u)$ is true if $u$ is in normal form and $u$ does not
  contain the symbols $\pi_1, \pi_2, \text{check}, \text{RR}$.
- $EQ$ is the strict equality predicate: $EQ(u, v)$ holds
  if $u = v$ and both terms are well-formed.
- $SK$ is true on pairs of well-formed terms $(k, [s]_k^{VK})$:
  an attacker who knows a signing key can check
  whether that key is used for signing a given mes-
  sage.

3.2 Frames and static equivalence

A frame is a sequence of ground terms in which
some names (typically secret keys) $\pi$ are hidden: $\phi =
\nu \pi. n_1, \ldots , n_k$. We let $bn(\phi)$ be $\pi$. The frames
will record the sequences of messages sent over the network.
With each frame $\phi = \nu \pi. s_1, \ldots , s_m$, we associate a substitu-
tion $\sigma_\phi$ that replaces the variable $x_i$ with $s_i$. 
A term $s$ is deducible from a frame $\phi$, which we write as $\phi \vdash s$, if there is a term $t$ with $m$ variables, not using the names hidden in $\phi$ and such that $ut_{\sigma}$ is $s$. This captures the possible attacker's computations on a sequence of messages.

Two frames $\phi_1, \phi_2$ are equivalent, which is written as $\phi_1 \sim \phi_2$, if, for any terms $u, v$ (with $m$ variables and not using the names hidden by the frames), $M(ut_{\sigma_1})$ holds if $M(ut_{\sigma_2})$ holds and, for $P \in \{EQ, SK\}$, $P(ut_{\sigma_1})$ holds if $P(ut_{\sigma_2})$ holds. In words: when we apply any combination of functions to the two frames, the results always look similar.

Examples
\[ \nu_{n, k, r, k', r', [n]'_{k', V}, [n]'_{k, V}} = \nu_{n', k, r, k', r', [n]'_{k', V}, [n]'_{k, V}} \]

since the attacker can only observe an equality between the two signed messages.

\[ [n]'_{k, V} \not\sim [n]'_{k', V} \]

as soon as $n \neq n'$ since, unlike the previous example, $n, n'$ are not hidden, and so can be used by the attacker: $E\{(check(x, V), n)\}$ holds on the first message and not on the second.

\[ \nu_{k, v', r, [n]'_{k', V}, k \neq \nu_{k, v', r, [n]'_{k, V}, k'} \]

since $SK$ is true on the first sequence and not on the second.

3.3 Computation trees, symbolic equivalence
If $\phi$ is a frame, we let $K(\phi)$ be the set of keys deducible from $\phi$.

A computation tree is a tree whose nodes are labeled with states (out of a set $Q$) and frames, and the edges are labeled with terms. We write $t \xrightarrow{\eta} t'$ if there is an edge labeled with $\eta$ departing from the root of $t$ and yielding the subtree $t'$. $\phi(t)$ is the frame labeling the root of $t$ and $q(t)$ is the state labeling the root of the tree.

\[ \sim \] extended to computation trees: $\sim$ is the largest equivalence relation on trees such that, if $t_1 \sim t_2$, then
- $\phi(t_1) \sim \phi(t_2)$,
- if $t_1 \xrightarrow{\xi_1} t_1'$ then there exist $u_2, t_2'$ such that $t_2 \xrightarrow{u_2} t_2'$ and $t_1 \sim t_2'$, and
- if $t_2 \xrightarrow{\xi_2} t_2'$ then there exist $u_1, t_1'$ such that $t_1 \xrightarrow{u_1} t_1'$ and $t_1 \sim t_2'$.

3.4 Symbolic equivalence of reduced trees
For each sequence of verification keys, we let the first non-compromised key be its representative. When all subterms $[n]'_{k, V}$ of a frame $\phi$ are such that $k$ is the representative of keys in $VK$, we say that $\phi$ is reduced. A computation tree is reduced if all the frames labeling its nodes are reduced.

Let $\simeq$ be the equivalence relation on frames defined by: $\nu_{P_1, P_2} \simeq \nu_{P_1, P_2}$ if there are renamings $P_1 \rightarrow P_1$ and $P_2 \rightarrow P_2$ such that $P_1(\nu_1) = P_2(\nu_2)$ is extended to computation trees in the same way as $\sim$ was extended.

Lemma 1 Let $t_1, t_2$ be two reduced computation trees. Then $t_1 \sim t_2$ iff $t_1 \simeq t_2$.

3.5 Processes
A protocol is specified as a simple process, which is a parallel composition of processes that repeatedly receive a message, test it, and send messages. Each test is specified by a conjunction of atomic predicates. Each message is assumed to include its intended recipient.

Each process $P$ in the calculus can be associated with a computation tree $t_P$ that records all possible interactions with the network: labels of edges are messages from the attacker and nodes are labeled with the state of the network and the record of messages that have already been sent.

4. Computational Interpretation
4.1 Computational interpretation of terms
Given a security parameter $\eta$ and an interpretation $\tau$ of names as bitstrings, a computational interpretation $[t]_{\eta, VK}$ of each term $t$ is defined as in [3]. We assume that the interpretation of a ring signature $[u]'_{k, VK}$ contains the interpretations of $u$ and $VK$ in addition to the signature bitstring $m_s: [u]'_{k, VK} = (\nu_{u', \nu_s, [VK]'_{\eta}})$. We also assume that verification keys come with a certificate: the attacker cannot generate such keys oneself and must get them from an authority.

4.2 Computational indistinguishability of computation trees
Given a security parameter $\eta$ and an interpretation $\tau$ of names as bitstrings, we assume that there is a total injective parsing function $\kappa_{\eta}$ from bitstrings to terms. From injectivity, for every $m$, $[\kappa_{\eta}(m)]_{\eta} = m$.

Given a computation tree $t$ and an assignment $\tau$ of names to bitstrings, the oracle $O_{t, \tau}$ is defined as follows:
- When queried for the first time with a bitstring $m$, it returns $[\phi(t)]_{\eta, VK}$ if $t \xRightarrow{\eta} t'$ and $u = \kappa_{\eta}(m)$.
- If there is no edge labeled with $\kappa_{\eta}(m)$ and departing from the root of $t$, it returns an error message.
- After the first query, it behaves as $O_{t', \tau}$.

$t_1$ and $t_2$ are computationally indistinguishable, which we write as $t_1 \approx t_2$, if, for any PPT $A^\tau,
\Pr [\tau : A^{O_{t_1, \tau}}(0^n) = 1] = \Pr [\tau : A^{O_{t_2, \tau}}(0^n) = 1]]$ is negligible in the security parameter $\eta$.

4.3 Tree soundness
We consider trees without dynamic corruption. In such a tree, if $\psi$ is labeling any node of $t$, we assume that $K(\psi) = K(\phi(t))$: corrupted keys are identical along all branches of the tree.

Given a frame $\phi$ and a term $u$, $\Psi_{VK, \phi}(u)$ is the term obtained by replacing signatures $[s]'_{k, VK}$ occurring in $u$ with $[s]'_{k, VK}$ if $k'$ is a minimal element in $\{k_1 \in bn(\phi) \mid K(\phi) \not\in \{k_1 \in VK\}\}$. $\Psi_{VK}$ is the function that maps each frame $\phi$ to the frame in which all subterms $u$ of $u$ are replaced with $\Psi_{VK, \phi}(u)$.

$\Psi_{VK}$ is extended to computation trees as follows: $\phi(\Psi_{VK}(t)) = \Psi_{VK}(\phi(t))$, $q(\Psi_{VK}(t)) = q(t)$, and, if $t \xRightarrow{\eta} t'$, then $\Psi_{VK}(t) \xRightarrow{\eta} \Psi_{VK}(t')$.

Note that all labels of edges departing from a node in $\Psi_{VK}(t)$ are distinct, as soon as it is the case for $t$,
because different random symbols must be used for different signatures.

Lemma 2  For any computation tree \( t \) without dynamic corruption, and any set of verification keys \( VK \), \( \Psi_{VK}(t) \approx t \).

Lemma 3  If \( t_1 \simeq t_2 \), then \( t_1 \approx t_2 \).

Lemma 4  Assuming basic anonymity, \( t \approx \Psi_{VK}(t) \).

For this crucial lemma, we need to build a machine \( B \) which breaks the basic anonymity, from a machine \( A \) that distinguishes \( t \) and \( \Psi_{VK}(t) \). Roughly speaking, \( B \) will simulate the network, keeping the state in its memory, and behave as \( A \): when \( A \) sends a query \( m,B \) parses \( m \), computes the next state and obtains the symbolic reply \( u \). Then \( B \) computes \( [u]^t_{\eta} \), possibly sending requests to the signing oracle. When such a request would yield different answers depending on whether \( A \) interacts with \( t \) or \( \Psi_{VK}(t) \), then \( B \) requests a signed message and guesses the signer according to the guess of \( A \).

Lemma 5 (Tree soundness for ring signatures)

Assuming basic anonymity, if \( t,t' \) are computation trees without dynamic corruption such that \( t \approx t' \), then \( t \approx t' \).

Proof sketch We successively apply \( \Psi_{VK} \) to all sets of verification keys \( VK \) occurring in the tree and apply Lemmas 1–4.

(\textit{QED})

5. Soundness of Observational Equivalence

The anonymity of a protocol is specified by the equivalence between, for example, two simple processes \( P_0(k_0)\|P_1(k_1) \) and \( P_0(k_1)\|P_1(k_0) \) where \( k_0 \) and \( k_1 \) are the identities (signing keys) of two agents. (We omit the details of how we publish \( vk(k_0) \) and \( vk(k_1) \).) The symbolic anonymity \( P_0(k_0)\|P_1(k_1) \approx P_0(k_1)\|P_1(k_0) \), implies the computational anonymity \( [P_0(k_0)]_t \| [P_1(k_1)]_t \approx [P_0(k_1)]_t \| [P_1(k_0)]_t \) thanks to the soundness theorem below.

Theorem 1  Assume basic anonymity, unforgeability and unpredictability. Let \( P \) and \( Q \) be simple processes and \( A \) be a PPT attacker. If \( P \approx Q \), then \( [P]_t \approx [Q]_t \).

Proof sketch As shown in [3], \( P \approx Q \) implies \( t_P \approx t_Q \). Then \( t_P \approx t_Q \) follows from Lemma 5. Then \( [P]_t \approx [Q]_t \) follows from Lemma 7.

(\textit{QED})

References