Computationally Sound Symbolic Anonymity of a Ring Signature

Yusuke Kawamoto † Hideki Sakurada ‡ Masami Hagiya †

† Department of Computer Science, Graduate School of Information Science and Technology, University of Tokyo {y_kwmt, hagiya}@is.s.u-tokyo.ac.jp

‡ NTT Communication Science Laboratories, NTT Corporation sakurada@theory.brl.ntt.co.jp

Abstract

This paper investigates the relationship between symbolic and computational approaches to the analysis of the anonymity of cryptographic protocols. We provide two computational soundness results for a symbolic ring signature in the presence of active adversaries, each corresponding to the unforgeability and signer anonymity of a ring signature scheme. The first result is the mapping-style soundness: every computational execution trace corresponds to a symbolic execution trace with overwhelming probability. The second result is the soundness of the symbolic anonymity of protocols: symbolic indistinguishability implies computational indistinguishability. By employing the mapping-style soundness result, we obtain the soundness of the symbolic anonymity when the length of execution traces is bound by a constant independent of the security parameter.

Key words: Formal methods, cryptographic protocols, computational soundness, anonymity.

1 Introduction

Symbolic and computational approaches have developed independently in research related to the security analysis of cryptographic protocols. In the symbolic approach, a message is abstracted into a symbolic expression, called a Dolev-Yao term [10], and an adversary can perform only a fixed set of operations on Dolev-Yao terms. The symbolic analysis of cryptographic protocols assumes perfectly secure cryptographic schemes. On the other hand, in the computational approach, a message is a bit string and an adversary is a probabilistic polynomial-time (PPT) Turing machine. The computational analysis of cryptographic protocols is based on computational complexity theory.

In recent years, many researches have related these two approaches [2, 1, 14] to obtain computational soundness results, whereby symbolic security proofs imply strong security guarantees based on computational complexity theory. The soundness results in the previous studies are divided into two categories: mapping-style soundness and indistinguishability-based soundness.

The former category of studies [14, 9, 8, 16, 4] has shown that every computational execution trace corresponds to a symbolic execution trace with overwhelming probability. This kind of soundness results provides a computational guarantee for the symbolic analysis of unforgeability and authenticity, which are expressed as nondeducibility in symbolic models. However, it cannot fully deal with anonymity, which is

expressed as indistinguishability. The latter category of studies [2, 12, 3, 11, 13] has proven that symbolic indistinguishability implies computational indistinguishability. This indistinguishability-based soundness can deal with anonymity, but is not suited to unforgeability or authenticity.

This paper provides computational soundness results for both the symbolic nondeducibility and indistinguishability of a ring signature scheme [15, 6, 17]. A ring signature scheme is a digital signature scheme that enables an agent to produce a signature of a message that identifies a group of possible signers but hides the actual signer. Since it needs to satisfy both unforgeability and signer anonymity, we combine a mapping-style soundness result with an indistinguishability-based soundness result. Although many computational soundness results have been obtained for various cryptographic schemes, no previous studies have dealt with the computational soundness of a symbolic ring signature, which requires both categories of soundness results. For the sake of brevity, we do not deal with other cryptographic schemes whose soundness results have been provided in previous studies.

To obtain the computational soundness result, we first extend the symbolic model presented in [9, 8] to deal with the symbolic nondeducibility of a ring signature, and prove the mapping-style soundness of the model. Next, we introduce a symbolic indistinguishability into the model, and prove its computational soundness by employing the mapping-style soundness. Here, our indistinguishability-based soundness result is limited to a case where the length of execution traces is bound by a constant independent of the security parameter. However, this is the first step in our investigation of the computationally sound symbolic anonymity of a ring signature, and we will extend our work to more practical cases by using the technique presented in [7].

The organization of this paper is as follows. Section 2 presents the definitions of a ring signature scheme and its computational security. Section 3 introduces a symbolic model, and Section 4 describes a concrete model. Section 5 provides a mapping-style soundness result, and Section 6 presents the soundness of symbolic anonymity. The final section summarizes our work and discusses our future research.

2 Ring Signature

This section briefly reviews the definition of a ring signature scheme and its security notions. Hereafter we denote the set $\{ 1, 2, \cdots, n \}$ by $[n]$.

2.1 Ring Signature Scheme

A ring signature scheme [15] enables an agent to produce a digital signature of a message that identifies a group of possible signers but hides the actual signer. In the scheme, a real signer can produce a signature by using his own private signing key and a set of public verification keys of possible signers. We describe the definition of a ring signature scheme as follows.

Definition 1 A *ring signature scheme* is a triple $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ consisting of the following three algorithms:

- a key generation algorithm $\mathcal{G}$ that takes a security parameter 1^η and a randomness r_{gen}, and outputs a private signing key and a public verification key pair (sk, vk).

- a signing algorithm $\mathcal{S}$ that takes a signing key sk, a set $\mathcal{L}$ of verification keys that includes the key corresponding to sk, a message m, and a randomness r, and outputs the signature of m using sk, $\mathcal{L}$, and r.
- a verification algorithm $\mathcal{V}$ that takes a set $\mathcal{L}$ of verification keys and two bit strings m and σ, and outputs 1 if σ is a valid signature of m using $\mathcal{L}$, or 0 otherwise.

These algorithms must satisfy $\mathcal{V}(\mathcal{L}, m, \mathcal{S}(sk, \mathcal{L}, m, r)) = 1$, and messages must be easily recovered from their signatures.

We assume that an adversary can add to $\mathcal{L}$ any verification keys possibly including fake verification keys, which may not have the corresponding signing keys or cannot generate signatures. We also assume that each public verification key is bound to the identity of its owner by a certificate authority, so that the owner of the verification key can be identified.

2.2 Security Definitions of a Ring Signature Scheme

The ring signature scheme defined above should satisfy two notions of security: unforgeability and anonymity. Although many definitions of security notions have been proposed, we employ the definitions found in [6, 17].

First, we introduce an oracle O given to the adversary.

Definition 2 We define an oracle O that satisfies the following conditions.

- If O receives a signing query $\mathsf{sign}(i, \mathcal{L}, m)$ for an agent i, a set $\mathcal{L}$ of verification keys, and a message m, then it produces the signature of m using i's private signing key sk_i, $\mathcal{L}$, and some fresh randomness r, and returns it to the adversary.
- If O receives a corruption query $\mathsf{corrupt}(i)$ for an agent i, then it returns i's private signing key sk_i to the adversary.

To prove the soundness result, we assume that the adversary sends corruption queries to O before sending signing queries. We will take account of dynamic corruptions in future work.

Next, we define the existential unforgeability of the ring signature scheme against adaptive chosen message and public key attacks with insider corruptions [6].

Definition 3 A ring signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is *existentially unforgeable* if the advantage Adv_A^{EUF} defined below is negligible in every security parameter η for every PPT adversary A having access to the oracle O.

$$
\begin{aligned}
Adv_A^{\mathrm{EUF}}(\eta) = \Pr\ [\ & (sk_1, vk_1), \cdots, (sk_n, vk_n) \leftarrow \mathcal{G}(1^\eta); \\
& \mathcal{L}^{leg} := (vk_1, \cdots, vk_n); \\
& (\mathcal{L}, m, \sigma) := A^{O}(1^\eta, \mathcal{L}^{leg}) : \\
& \mathcal{V}(\mathcal{L}, m, \sigma) = 1,\ \mathcal{L} \subseteq \mathcal{L}^{leg}, \\
& A \text{ has not queried } \mathsf{corrupt}(i) \text{ to } O \text{ for any } i \text{ with } vk_i \in \mathcal{L}, \text{ and} \\
& A \text{ has not queried } \mathsf{sign}(i, \mathcal{L}, m) \text{ to } O \text{ for any } i \text{ with } vk_i \in \mathcal{L} \quad]
\end{aligned}
$$

Note that the definition of existentially unforgeability does not prevent a PPT adversary from producing a signature of a message m using a randomness R from another signature of the same message m using another randomness R'.

Finally, we define the basic anonymity of the ring signature scheme against adaptive chosen message and chosen public key attacks [6, 17].

Definition 4 A ring signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is *basically anonymous* if the advantage Adv_A^{ANO} defined below is negligible in every security parameter η for every PPT adversary A having access to the oracle O.

$$
\begin{aligned}
Adv_A^{\mathrm{ANO}}(\eta) = \Pr[\ & (sk_1, vk_1), \cdots, (sk_n, vk_n) \leftarrow \mathcal{G}(1^\eta); \\
& \mathcal{L}^{leg} := (vk_1, \cdots, vk_n); \\
& (i_0, i_1, \mathcal{L}, m, \omega) := A^O(1^\eta, \mathcal{L}^{leg}); \\
& \quad (vk_{i_0}, vk_{i_1} \in \mathcal{L} \cap \mathcal{L}^{leg},\ \ i_0 \neq i_1) \\
& r \leftarrow C^\eta.r; \\
& b \leftarrow \{0, 1\}; \\
& \sigma := \mathcal{S}(sk_{i_b}, \mathcal{L}, m, r); \\
& b' := A^O(\omega, \sigma) : \\
& b' = b \text{ and } A \text{ has not queried } \mathsf{corrupt}(i_0) \text{ or } \mathsf{corrupt}(i_1) \text{ to } O\ ] - \tfrac{1}{2}
\end{aligned}
$$

where ω is an internal state of A, and $C^\eta.r$ is the set of all the bit strings for randomness.

3 Symbolic Model

This section defines a symbolic model, which is an extension of the Dolev-Yao model presented in [9, 8]. There are two kinds of parties: agents and an active and adaptive adversary. Agents follow a protocol without deviation, and can run multiple sessions of the protocol simultaneously. Agents are divided into honest agents and corrupted agents, who give their private signing keys to the adversary before executing protocols.

3.1 Terms

We define the following sets of atomic symbols, which are mutually exclusive:

- a set AG of *agent identity symbols*,
- a set $\mathsf{Garbage}$ of *garbage message symbols*,
- a set Nonce_{ag} of *nonce symbols of agents*,
- a set Nonce_{adv} of *nonce symbols of the adversary*,
- a set Rand_{ag} of *randomness symbols of agents*, and
- a set Rand_{adv} of *randomness symbols of the adversary*.

Let $\mathsf{Nonce} = \mathsf{Nonce}_{ag} \cup \mathsf{Nonce}_{adv}$ and $\mathsf{Rand} = \mathsf{Rand}_{ag} \cup \mathsf{Rand}_{adv}$. Intuitively, a randomness symbol denotes the random coin used to generate a signature. Let $\mathsf{Atom} = \mathsf{AG} \cup \mathsf{Garbage} \cup \mathsf{Nonce} \cup \mathsf{Rand}$.

Using these atomic symbols, a *term* is constructed from a signing key $\mathsf{sk}(\_)$, a verification key $\mathsf{vk}(\_)$, a pairing $\langle \_, \_ \rangle$, and a ring signing $[\ \_\ ]^{\_}_{\_,\_}$ as follows:

$$\mathsf{Term} \ni m ::= a \mid g \mid n \mid \mathsf{sk}(a) \mid \mathsf{vk}(a) \mid \mathsf{vk}(g) \mid \langle m, m \rangle \mid [\ m\ ]^R_{\mathsf{sk}(a), \mathsf{VK}(G)}$$

where $a \in \mathsf{AG}$, $g \in \mathsf{Garbage}$, $n \in \mathsf{Nonce}$, $R \in \mathsf{Rand}$, and $\mathsf{VK}(G) = \{\, \mathsf{vk}(x) \mid x \in G \,\}$ for $G \subseteq \mathsf{AG} \cup \mathsf{Garbage}$ with $|G \cap \mathsf{AG}| \geq 2$.

The term $\mathsf{sk}(a)$ denotes the legitimate signing key of an agent $a \in \mathsf{AG}$, the term $\mathsf{vk}(a)$ denotes the legitimate verification key of an agent $a \in \mathsf{AG}$, and the term $\mathsf{vk}(g)$ for $g \in \mathsf{Garbage}$ denotes a fake verification key that looks like a legitimate one. The term $\langle m_1, m_2 \rangle$ denotes the concatenation of two messages m_1 and m_2. The term $[\ m\ ]^R_{\mathsf{sk}(a), \mathsf{VK}(G)}$ denotes the signature of a message m by using a signing key $\mathsf{sk}(a)$, a set of verification keys $\mathsf{VK}(G)$, and a randomness R. Note that $\mathsf{VK}(G)$ may contain fake verification keys.

$$\frac{}{\varphi \vdash m}\ m \in \varphi \qquad \frac{}{\varphi \vdash a,\ \mathsf{vk}(a)}\ a \in \mathsf{AG} \qquad \frac{}{\varphi \vdash g,\ \mathsf{vk}(g)}\ g \in \mathsf{Garbage}$$

$$\frac{\varphi \vdash \mathsf{vk}(x_1),\ \cdots,\ \varphi \vdash \mathsf{vk}(x_n)}{\varphi \vdash \mathsf{VK}(G)}\ G = \{ x_1, \cdots, x_n \} \subseteq \mathsf{AG} \cup \mathsf{Garbage},\ |G \cap \mathsf{AG}| \geq 2 \qquad \frac{\varphi \vdash [\, m \,]^R_{\mathsf{sk}(a),\, \mathsf{VK}(G)}}{\varphi \vdash m}$$

$$\frac{\varphi \vdash m_0 \quad \varphi \vdash m_1}{\varphi \vdash \langle m_0,\, m_1 \rangle} \qquad \frac{\varphi \vdash \langle m_0,\, m_1 \rangle}{\varphi \vdash m_i}\ i = 0,\ 1 \qquad \frac{\varphi \vdash [\, m \,]^R_{\mathsf{sk}(a),\, \mathsf{VK}(G)}}{\varphi \vdash [\, m \,]^{R'}_{\mathsf{sk}(a),\, \mathsf{VK}(G)}}\ R' \in \mathsf{Rand}$$

$$\frac{\varphi \vdash \mathsf{sk}(a) \quad \varphi \vdash \mathsf{VK}(G) \quad \varphi \vdash m}{\varphi \vdash [\, m \,]^R_{\mathsf{sk}(a),\, \mathsf{VK}(G)}}\ R \in \mathsf{Rand},\ \mathsf{vk}(a) \in \mathsf{VK}(G) \qquad \frac{\varphi \vdash [\, m \,]^R_{\mathsf{sk}(a),\, \mathsf{VK}(G)}}{\varphi \vdash [\, m \,]^R_{\mathsf{sk}(b),\, \mathsf{VK}(G)}}\ \mathsf{vk}(b) \in \mathsf{VK}(G)$$

Figure 1: The adversary's deduction rules.

We define the following subsets of Term: SKey, VKey, Pair, and Sign, each denoting the set of all terms starting with $\mathsf{sk}(\_)$, $\mathsf{vk}(\_)$, $\langle \_, \_ \rangle$, and $[\, \_ \,]^{\_}_{\_,\_}$, respectively. Let VKeyList be the set $\{ \mathsf{VK}(G) \mid G \subseteq \mathsf{AG} \cup \mathsf{Garbage},\ |G \cap \mathsf{AG}| \geq 2 \}$. We sometimes refer to these sets as types.

We use terms with variables to describe protocols. We introduce the following infinite sets of variables: $\mathsf{X}.a$, $\mathsf{X}.g$, $\mathsf{X}.n$, $\mathsf{X}.r$, $\mathsf{X}.p$, $\mathsf{X}.s$, and $\mathsf{X}.L$ each denoting the set of all variables of the type AG, Garbage, Nonce, Rand, Pair, Sign, and VKeyList, respectively. Let $\mathsf{X} = \mathsf{X}.a \cup \mathsf{X}.g \cup \mathsf{X}.n \cup \mathsf{X}.r \cup \mathsf{X}.p \cup \mathsf{X}.s \cup \mathsf{X}.L$.

To represent agents' messages, we define the set ATerm(X) of *agent terms*:

$$\mathsf{ATerm}(\mathsf{X}) \ni m ::= X \mid \mathsf{vk}(X') \mid \langle m,\, m \rangle \mid [\, m \,]^{X.r}_{\mathsf{sk}(A),\, X.L}$$

where $X \in \mathsf{X}.a \cup \mathsf{X}.n \cup \mathsf{X}.p \cup \mathsf{X}.s$, $X' \in \mathsf{X}.a \cup \mathsf{X}.g$, $A \in \mathsf{X}.a$, $X.r \in \mathsf{X}.r$, and $X.L \in \mathsf{X}.L$. By using agent terms in ATerm(X), we describe protocols where agents do not try to construct ill-formed messages using garbage messages, or to send signing keys or randomnesses as plaintexts. Note that we do not need symbols in AG∪Nonce∪Rand in the definition of ATerm(X), because we assign a term to each variable of $\mathsf{X}.a \cup \mathsf{X}.n \cup \mathsf{X}.r$ occurring in a session of the protocol execution defined below when the session is created. In later sections, we use variables $A_i \in \mathsf{X}.a$, $X.n^j_{A_i} \in \mathsf{X}.n$, $X.r^j_{A_i} \in \mathsf{X}.r$, and $L^j_i \in \mathsf{X}.L$ to describe agents' messages.

3.2 Deduction Rules

We introduce deduction rules in Figure 1 to restrict the behavior of the adversary in the symbolic setting. In the deduction rules, a set φ of terms represents the adversary's knowledge of messages. These rules are standard [9] except for ring signatures. The rule for deriving $[\, m \,]^R_{\mathsf{sk}(b),\, \mathsf{VK}(G)}$ from $[\, m \,]^R_{\mathsf{sk}(a),\, \mathsf{VK}(G)}$ expresses that the definition of existential unforgeability does not prevent the adversary from producing an agent b's signature of a message m from another agent a's signature of m when $a, b \in G$. The rule for deriving $[\, m \,]^{R'}_{\mathsf{sk}(a),\, \mathsf{VK}(G)}$ from $[\, m \,]^R_{\mathsf{sk}(a),\, \mathsf{VK}(G)}$ expresses that the definition of existential unforgeability does not prevent the adversary from rerandomizing a signature, i.e., replacing a randomness used to produce a signature by another randomness. Although these rules are too strong for us to obtain a *computational completeness* of symbolic nondeducibility, they allow us to obtain a computational soundness result in Section 5.

3.3 Symbolic Execution Model

We introduce roles to define a symbolic execution model. Our definition is standard and similar to that in [9, 8]. Hereafter we denote the domain of a function f by $dom(f)$.

A *k-party protocol* Π is a function that maps each $i \in [k]$ to a role tree $\Pi(i)$. A *role tree* $\Pi(i)$ for an agent $A_i \in \mathsf{X}.a$ is a finite ordered edge-labeled tree where each edge is labeled by an *agent rule* (l, r) consisting of $l \in \mathsf{ATerm(X)}$ and $r \in \mathsf{ATerm(X)} \cup \{\perp\}$, where the symbol $\perp$ represents the abort of the protocol Π. Here, l and r represent messages sent from the adversary to the agent A_i and from A_i to the adversary, respectively. If A_i sends $\perp$ to the adversary, then the execution of the protocol Π stops. We denote the root node of a role tree by ε. In an execution of a protocol Π, each agent A_i stands at a node of his role tree $\Pi(i)$. We assume that the number of children of each node in each role tree is independent of the security parameter η introduced in Section 4.

A *symbolic execution trace* of a k-party protocol Π is a finite sequence of global states with the restrictions given below. A *global state* is a triple $(\mathsf{SId}, f, \varphi)$ consisting of a finite set SId of session ids, a function f that maps each session id in SId to the current local state of the corresponding session, and the set φ of terms representing the current adversary's knowledge. A *local state* of a session $\mathsf{sid} \in \mathsf{SId}$ is a quadruple $(i, \sigma, p, (a_1, \cdots, a_k))$ where $i \in [k]$ is the index of the role tree $\Pi(i)$, σ is a partial function from X to Term, p is a node in the role tree $\Pi(i)$, each $a_j \in \mathsf{AG}$ is the agent carrying out the session sid, and $\sigma(A_j) = a_j$ holds for each $j \in [k]$.

The partial function $\sigma\colon \mathsf{X} \to \mathsf{Term}$ is updated in the execution of the protocol Π. Let p be a node in the role tree $\Pi(i)$. Now, if the agent a_i receives a term m at p, then a_i produces a term m' as follows. Until a_i outputs some term, each agent rule that labels an outgoing edge of the node p is applied to m in order from the left-most rule to the right-most rule. Given an agent rule (l, r), m is matched against $l\sigma$. If m and $l\sigma$ match, we denote the matcher by η. Then, the agent produces the output $m' := r\sigma\eta$, and follows the edge to the child node, where σ is updated to $\sigma\eta$. Note that m' does not contain a variable, because of the conditions introduced below. If m and $l\sigma$ do not match, the next agent rule is applied. If there is no rule (l, r) such that m and $l\sigma$ match, then the agent outputs no term and σ remains unchanged. We describe the restriction to the role tree at the end of this section.

To define transitions between global states, we introduce some notations. We denote distinct nonces by $n^{a,j,s} \in \mathsf{Nonce}_{ag}$ with $a \in \mathsf{AG}$ and $j, s \in \mathbb{N}$, and distinct randomnesses by $r^{a,j,s} \in \mathsf{Rand}_{ag}$ with $a \in \mathsf{AG}$, $j \in \mathsf{Rand}_{ag}$, and $s \in \mathbb{N}$. Let $\tau_{a,s}$ be a function that maps each $j \in \mathsf{Rand}_{ag}$ to $r^{a,j,s}$. For a term $t \in \mathsf{ATerm(X)}$, we write $t\tau_{a,s}$ to denote the term obtained from t by replacing every occurrence of $j \in \mathsf{Rand}_{ag}$ in t with $\tau_{a,s}(j)$ simultaneously.

The initial global state is $q_I = (\emptyset, \emptyset, \mathsf{Nonce}_{adv})$. We allow three kinds of transitions between global states: **corrupt**, **new**, and **send**.

- The adversary can specify a set of agents and obtain their private signing keys before the execution of the protocol: $q_I \xrightarrow{\mathbf{corrupt}(a_1, \cdots, a_l)} (\emptyset, \emptyset, \mathsf{Nonce}_{adv} \cup C_{\mathsf{SKey}})$ where $C_{\mathsf{SKey}} = \{\mathsf{sk}(a_i) \mid i \in [l]\}$.
- The adversary can initiate a new session for an agent A_i and specify the sets of verification keys used to produce ring signatures: $(\mathsf{SId}, f, \varphi) \xrightarrow{\mathbf{new}(i, a_1, \cdots, a_k, \mathsf{VK}(G_i^1), \cdots, \mathsf{VK}(G_i^m))} (\mathsf{SId}', f', \varphi)$ where $\mathsf{VK}(G_i^1), \cdots, \mathsf{VK}(G_i^m) \in \mathsf{VKeyList}$, and SId' and f' are defined as follows.
  - Let SId' be the set $\mathsf{SId} \cup \{\mathsf{sid}\}$ for the new session $\mathsf{sid} = |\mathsf{SId}| + 1$.

– Let f' be the function that satisfies $f'(\mathsf{sid}') = f(\mathsf{sid}')$ for every $\mathsf{sid}' \in \mathsf{SId}$ and $f'(\mathsf{sid}) = (i, \sigma, \varepsilon, (a_1, \cdots, a_k))$ where $dom(\sigma) = \{A_j \mid j \in [k]\} \cup \{X.n^j_{A_i} \mid j \in \mathbb{N}\} \cup \{X.r^j_{A_i} \mid j \in \mathsf{Rand}_{ag}\} \cup \{L^j_i \mid j \in [m]\}$, $\sigma(A_j) = a_j$ for each $j \in [k]$, $\sigma(X.n^j_{A_i}) = n^{a_i, j, \mathsf{sid}}$ for each $j \in \mathbb{N}$, $\sigma(X.r^j_{A_i}) = r^{a_i, j, \mathsf{sid}}$ for each $j \in \mathsf{Rand}_{ag}$, and $\sigma(L^j_i) = \mathsf{VK}(G^j_i)$ for each $L^j_i \in \mathsf{X}.L$ with $j \in [m]$. If a variable $L \in \mathsf{X}.L$ satisfying $L \neq L^j_i$ for each $j \in [m]$ occurs in the agent A_i's role tree $\Pi(i)$, then the adversary cannot create the new session sid.

- The adversary can send messages: $(\mathsf{SId},\ f,\ \varphi) \xrightarrow{\mathbf{send}(\overline{\mathsf{sid}},\, m)} (\mathsf{SId},\ f',\ \varphi')$ where $\overline{\mathsf{sid}}$ is a tuple $(\mathsf{sid}_1, \cdots, \mathsf{sid}_v)$ of elements in SId with $v \geq 2$, $m \in \mathsf{Term}$, $\varphi \vdash m$, and f' and φ' are defined as follows. The function f' initially satisfies $f'(\mathsf{sid}') = f(\mathsf{sid}')$ for every $\mathsf{sid}' \in \mathsf{SId}$. In this transition, the term m is first sent to the session sid_1. For each $u \in [v-1]$, if m is not received by sid_u, then it is sent to sid_{u+1}. For a session $\mathsf{sid}_u \in \overline{\mathsf{sid}}$, suppose that $f(\mathsf{sid}_u) = (i, \sigma, p, (a_1, \cdots, a_k))$ and $((l_1, r_1), \cdots, (l_h, r_h))$ are the labels of edges leaving p in this order.
  – If there does not exist j such that m and $l_j\sigma$ match, then we have $\varphi' = \varphi$, and m is sent to the session sid_{u+1}. If m is not received by any sessions in $\overline{\mathsf{sid}}$, then the adversary receives nothing.
  – Otherwise, let j be minimal such that m and $l_j\sigma$ match. Let θ be the matcher. Then, we update f' to obtain $f'(\mathsf{sid}) = (i, \sigma\theta, p_j, (a_1, \cdots, a_k))$ and $\varphi' = \varphi \cup \{r_j\tau_{a_i, \mathsf{sid}}\sigma\theta\}$ where p_j is the j-th child node of p. Here, the adversary receives the message $r_j\tau_{a_i, \mathsf{sid}}\sigma\theta$, but does not learn which session he receives it from. Thus, this transition represents an anonymous communication.

The set of symbolic execution traces for a protocol Π is denoted by $\mathsf{Exec}^s(\Pi)$.

We introduce the following restrictions on role trees. For a node $p \neq \varepsilon$ in the role tree $\Pi(i)$, let $Rules_{\leq p}$ be the sequence of agent rules with which the edges on the path from the root node ε to p are labeled. We write $Rules^l_{\leq p}$ and $Rules^r_{\leq p}$ to denote the sequence of the left/right-hand side of the rules $Rules_{\leq p}$, respectively. If $p \neq \varepsilon$ holds, we write $rule_p$ or $(rule^l_p,\ rule^r_p)$ to denote the agent rule with which the edge leading to p is labeled. Let $K^i_p = \{\mathsf{sk}(A_i)\} \cup \{A_j \mid j \in [k]\} \cup \{X.n^j_{A_i} \mid j \in \mathbb{N}\} \cup \{L^j_i \mid j \in [m]\} \cup Rules^l_{\leq p}$.

For each node $p \neq \varepsilon$ in the role tree $\Pi(i)$, we require the following conditions.

1. If a term of the form $[\,t\,]^{X.r}_{\mathsf{sk}(A),\, X.L}$ occurs in $rule^r_p$, then either it occurs in $Rules^l_{\leq p}$, or $A = A_i$ holds and $X.r \in \mathsf{X}.r$ never occurs in $Rules^l_{\leq p}$ or $Rules^r_{\leq p}$.
2. For every $A \in \mathsf{X}.a$, if $\mathsf{sk}(A)$ occurs in $Rules^l_{\leq p}$, then $A \neq A_j$ holds for $j \neq i$ and A occurs in $Rules^l_{\leq p}$ at most once in a term of the form $[\,t\,]^{X.r}_{\mathsf{sk}(A),\, X.L}$ for some $t \in \mathsf{ATerm}(\mathsf{X})$, $X.r \in \mathsf{X}.r$, and $X.L \in \mathsf{X}.L$.
3. Every $X.r \in \mathsf{X}.r$ occurs in $Rules^l_{\leq p}$ at most once in a term of the form $[\,t\,]^{X.r}_{\mathsf{sk}(A),\, X.L}$ for some $t \in \mathsf{ATerm}(\mathsf{X})$, $A \in \mathsf{X}.a$, and $X.L \in \mathsf{X}.L$.
4. For every variable $X \in \mathsf{X} \setminus \mathsf{X}.a \cup \mathsf{X}.n \cup \mathsf{X}.r$, if X occurs in $rule^r_p$, X occurs in $Rules^l_{\leq p}$.
5. We have $K^i_p \vdash rule^r_p$.

Intuitively, the first condition prescribes that no agent can forge another agent's signature, and that each agent should always use a fresh randomness when he produces a signature. The second condition prescribes that agents cannot break signer anonymity. The third condition prescribes that agents cannot recognize the different signatures

produced using the same randomness. Note that we can describe the protocols where agents recognize the same signatures, because we can use variables in $\mathsf{X}.s$. The last two conditions prescribe that each agent can produce terms only from previously received terms.

4 Concrete Model

This section defines a concrete model similar to that in [9, 8].

In the concrete model, bit strings are tagged with type tags, and the adversary can efficiently recognize the following types of bit strings: agent ids, nonces, randomnesses, garbages, signing keys, verification keys, pairs, signatures, and a set of verification keys including at least two legitimate verification keys. These concrete types are denoted by $C^\eta.a$, $C^\eta.n$, $C^\eta.r$, $C^\eta.g$, $C^\eta.sk$, $C^\eta.vk$, $C^\eta.p$, $C^\eta.s$, and $C^\eta.L$, respectively, where η is a security parameter. We denote the set of all bit strings by C^η.

A *concrete execution trace* of a k-party protocol Π is a finite sequence of concrete global states with the restrictions presented below. A *concrete global state* is a pair (SId, g) consisting of a finite set SId of session ids, and a function g that maps each session id in SId to the current concrete local state of the corresponding session. A *concrete local state* of a session $\mathsf{sid} \in \mathsf{SId}$ is a quadruple $(i, \rho, p, (a_1, \cdots, a_k))$ where $i \in [k]$ is the index of the role tree $\Pi(i)$, ρ is a partial function from X to C^η, p is a node in the role tree $\Pi(i)$, each $a_j \in C^\eta.a$ is the agent carrying out the session sid, and $\rho(A_j) = a_j$ holds for each $j \in [k]$. The partial function $\rho\colon \mathsf{X} \to C^\eta$ is updated in the concrete execution of the protocol Π in the same way as in the definition of the partial function $\sigma\colon \mathsf{X} \to \mathsf{Term}$ in the symbolic model except that ρ assigns bit strings instead of terms.

In a concrete execution of a protocol Π, agents and an active and adaptive adversary $\mathcal{A}$ are probabilistic polynomial time (PPT) Turing machines with respect to the security parameter η, and if one of the machines reaches the computation bounds then the execution of Π stops. The sequence of all random coins used by agents and the oracle $\mathcal{O}$ is denoted by R_Π, and the sequence of all random coins used by the adversary $\mathcal{A}$ is denoted by $R_\mathcal{A}$. If R_Π and $R_\mathcal{A}$ are fixed, then the concrete execution trace is uniquely determined, and is written as $\mathsf{Exec}^c_{\Pi(R_\Pi),\mathcal{A}(R_\mathcal{A})}(\eta)$.

In the concrete model, we fix a ring signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$. Let $\mathcal{L}^{leg}$ be a set of all legitimate verification keys that is initially empty and available to the adversary. Whenever a verification key is generated by running $\mathcal{G}(1^\eta, R_\Pi)$, it is added to $\mathcal{L}^{leg}$.

We define transitions between concrete global states in $\mathsf{Exec}^c_{\Pi(R_\Pi),\mathcal{A}(R_\mathcal{A})}(\eta)$ as follows. The initial concrete global state is $q_I = (\emptyset, \emptyset)$. We allow three kinds of transitions between concrete global states: **corrupt**, **new**, and **send**.

- The adversary can specify a set of agents and obtain their private signing keys before the execution of the protocol: $q_I \xrightarrow{\mathbf{corrupt}(a_1,\cdots,a_l)} q_I$ where $a_1, \cdots, a_l \in C^\eta.a$. Then, the signing and verification keys are generated for the agents by running $\mathcal{G}(1^\eta, R_\Pi)$, and are given to the adversary $\mathcal{A}$.
- The adversary can initiate a new session for an agent A_i and specify the sets of verification keys used to produce ring signatures: $(\mathsf{SId}, g) \xrightarrow{\mathbf{new}(i,a_1,\cdots,a_k,\mathcal{L}_i^1,\cdots,\mathcal{L}_i^m)} (\mathsf{SId}', g')$ where $i \in [k]$, $a_1, \cdots, a_k \in C^\eta.a$, $\mathcal{L}_i^1, \cdots, \mathcal{L}_i^m \in C^\eta.L$ and SId' and g' are defined as follows.
  - Let SId' be the set $\mathsf{SId} \cup \{\mathsf{sid}\}$ for the new session $\mathsf{sid} = |\mathsf{SId}| + 1$.

– Let g' be the function that satisfies $g'(\mathsf{sid}') = g(\mathsf{sid}')$ for every $\mathsf{sid}' \in \mathsf{SId}$ and $g'(\mathsf{sid}) = (i, \rho, \varepsilon, (a_1, \cdots, a_k))$ where $dom(\rho) = \{A_j \mid j \in [k]\} \cup \{X.n^j_{A_i} \mid j \in \mathbb{N}\} \cup \{X.r^j_{A_i} \mid j \in \mathsf{Rand}_{ag}\} \cup \{L^j_i \mid j \in [m]\}$, $\rho(A_j) = a_j$ for each $j \in [k]$, $\rho(X.n^j_{A_i}) \xleftarrow{R_\Pi} C^\eta.n$ for each $j \in \mathbb{N}$, $\rho(X.r^j_{A_i}) \xleftarrow{R_\Pi} C^\eta.r$ for each $j \in \mathsf{Rand}_{ag}$, and $\rho(L^j_i) = \mathcal{L}^j_i$ for each $L^j_i \in \mathsf{X}.L$ with $j \in [m]$. If a variable $L \in \mathsf{X}.L$ satisfying $L \neq L^j_i$ for each $j \in [m]$ occurs in the agent A_i's role tree $\Pi(i)$, then the adversary $\mathcal{A}$ cannot create the new session sid.

As a result of this transition, each a_j's signing and verification keys are generated if they have not been produced yet, and all the verification keys are given to $\mathcal{A}$.

- The adversary can send messages: $(\mathsf{SId}, g) \xrightarrow{\mathbf{send}(\overline{\mathsf{sid}}, m)} (\mathsf{SId}, g')$ where $\overline{\mathsf{sid}}$ is a tuple $(\mathsf{sid}_1, \cdots, \mathsf{sid}_v)$ of elements in SId with $v \geq 2$, $m \in C^\eta$, and g' is defined as follows. The function g' initially satisfies $g'(\mathsf{sid}') = g(\mathsf{sid}')$ for every $\mathsf{sid}' \in \mathsf{SId}$. In this transition, the term m is first sent to the session sid_1. For each $u \in [v-1]$, if m is not received by sid_u, then it is sent to sid_{u+1}. For a session $\mathsf{sid}_u \in \overline{\mathsf{sid}}$, suppose that $g(\mathsf{sid}) = (i, \rho, p, (a_1, \cdots, a_k))$ and $((l_1, r_1), \cdots, (l_h, r_h))$ are the labels of edges leaving p in this order.
  - If there does not exist j such that m and $l_j\rho$ match, then m is sent to the session sid_{u+1}. If m is not received by any sessions in $\overline{\mathsf{sid}}$, then the adversary $\mathcal{A}$ receives nothing.
  - Otherwise, let j be minimal such that m and $l_j\rho$ match. Let θ be the matcher. Then, we update g' to obtain $g'(\mathsf{sid}) = (i, \rho', p_j, (a_1, \cdots, a_k))$ and the agent a_i outputs the bit string corresponding to $r_j\tau_{a_i, \mathsf{sid}}\rho'$ where $\rho' = \rho\theta$ and p_j is the j-th child node of p. Although the adversary $\mathcal{A}$ receives the bit string, he does not learn which session he receives it from.

  Matching $l_j\rho$ against m is performed recursively on the structure of l_j.
  - If $l_j \in \mathsf{X}$, and the types of l_j and m match, then
    - if $\rho(l_j) = m$, then $\rho' = \rho$,
    - if $\rho(l_j)$ is not defined, then $\rho'(X) = \rho(X)$ for every $X \neq l_j$ and $\rho'(l_j) = m$.
  - If $l_j = \mathsf{vk}(X')$ for some $X' \in \mathsf{X}.a \cup \mathsf{X}.g$, and m is a legitimate/fake verification key, then $l_j\rho$ is matched against m. If the matcher θ exists, then $\rho' = \rho\theta$.
  - If $l_j = \langle l_{j1}, l_{j2} \rangle$ and m is the pair consisting of m_1 and m_2, then $l_{j1}\rho$ is matched against m_1. If the matcher θ_1 exists, then $l_{j2}\rho\theta_1$ is matched against m_2. If the matcher θ_2 exists, then $\rho' = \rho\theta_1\theta_2$.
  - If $l_j = [\, l_{m'} \,]^{l_R}_{\mathsf{sk}(A), l_L}$ and m is a signature of m' using a set L of verification keys, then $l_{m'}\rho$ is matched against m'. If the matcher θ_1 exists, then $l_L\rho\theta_1$ is matched against L. If the matcher θ_2 exists, then $\rho' = \rho\theta_1\theta_2$. Note that $\mathsf{sk}(A)$ and l_R do not occur as terms in r_j.
  - Otherwise, the matching fails.

5 Computational Soundness of Nondeducibility

This section provides a mapping-style computational soundness result for the above symbolic execution model.

First, we define a partial function c that maps each term to a bit string, and a concrete instantiation of a symbolic execution trace in the same way as in [9, 8].

Definition 5 Let $t^s = (\mathsf{SId}_1^s,\ f_1,\ \varphi_1),\ \cdots,\ (\mathsf{SId}_n^s,\ f_n,\ \varphi_n)$ be a symbolic execution trace, and $t^c = (\mathsf{SId}_1^c,\ g_1),\ \cdots,\ (\mathsf{SId}_n^c,\ g_n)$ be a concrete execution trace. The trace t^c is a *concrete instantiation* of t^s with an injective partial mapping $c\colon \mathsf{Term} \to \mathcal{C}^\eta$, written as $t^s \preceq_c t^c$, if for every $\ell \in [n]$, it holds that

- $\mathsf{SId}_\ell^s = \mathsf{SId}_\ell^c$, and
- for every $\mathsf{sid} \in \mathsf{SId}_\ell^s$, if $f_\ell(\mathsf{sid}) = (i^{\mathsf{sid}},\ \sigma^{\mathsf{sid}},\ p^{\mathsf{sid}},\ (a_1,\ \cdots,\ a_n))$ and $g_\ell(\mathsf{sid}) = (j^{\mathsf{sid}},\ \rho^{\mathsf{sid}},\ q^{\mathsf{sid}},\ (a_1,\ \cdots,\ a_n))$, then $i^{\mathsf{sid}} = j^{\mathsf{sid}}$, $\rho^{\mathsf{sid}} = c\sigma^{\mathsf{sid}}$, and $p^{\mathsf{sid}} = q^{\mathsf{sid}}$.

The trace t^c is a *concrete instantiation* of t^s, written as $t^s \preceq t^c$, if there exists an injective partial mapping $c\colon \mathsf{Term} \to \mathcal{C}^\eta$ that satisfies $t^s \preceq_c t^c$.

Next, we provide a mapping-style computational soundness result.

Theorem 1 *Let Π be a protocol. If a ring signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ satisfies existential unforgeability, then for any PPT adversary $\mathcal{A}$,*

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

where the probability is over the choice $(R_\Pi, R_\mathcal{A}) \leftarrow \{0,1\}^{p_\mathcal{A}(\eta)} \times \{0,1\}^{q_\mathcal{A}(\eta)}$, and $\nu_\mathcal{A}$ is a function that is negligible in η.

We use the following lemma to prove Theorem 1.

Lemma 1 *Let $C = C_{\mathsf{SKey}} \cup \mathsf{Nonce}_{adv}$, $\varphi' \subseteq \mathsf{Term} \setminus C$, $\varphi = \varphi' \cup C$, $m \in \mathsf{Term}$, $\mathsf{Nonce}_{ag}(m) = \{\, n \in \mathsf{Nonce}_{ag} \mid n \sqsubseteq m \,\}$, and $\mathsf{Nonce}_{ag}(\varphi') = \bigcup_{m' \in \varphi'} \mathsf{Nonce}_{ag}(m')$. If $\varphi \nvdash m$, then either*

1. *there exists a term $[\, t \,]^R_{\mathsf{sk}(a),\,\mathsf{VK}(G)}$ with $\mathsf{sk}(a') \notin C_{\mathsf{SKey}}$ for any $a' \in G \cap \mathsf{AG}$ such that we have $[\, t \,]^R_{\mathsf{sk}(a),\,\mathsf{VK}(G)} \sqsubseteq m$ and $[\, t \,]^{R'}_{\mathsf{sk}(a'),\,\mathsf{VK}(G)} \not\sqsubseteq m'$ for every $R' \in \mathsf{Rand}$, every $a' \in G \cap \mathsf{AG}$, and every $m' \in \varphi$,*
2. *there exists $\mathsf{sk}(a) \notin C_{\mathsf{SKey}}$ that occurs in m in a form other than $[\, t \,]^R_{\mathsf{sk}(a),\,\mathsf{VK}(G)}$, or*
3. *$\mathsf{Nonce}_{ag}(m) \setminus \mathsf{Nonce}_{ag}(\varphi') \neq \emptyset$ holds.*

Proof of Theorem 1 (Overview). The proof is similar to [9, 8], and is divided into two steps.

STEP I. First, we associate each concrete execution trace with a symbolic execution trace. By fixing the security parameter η and the random coins R_Π and $R_\mathcal{A}$, we can generate all agent identities, garbage messages, nonces, signing keys, and randomnesses. Hence, we obtain a unique concrete execution trace $t^c = \mathsf{Exec}^c_{\Pi(R_\Pi),\,\mathcal{A}(R_\mathcal{A})}(\eta)$. Then, it is sufficient to show that t^c is an instantiation of a symbolic execution trace t^s, i.e., $t^s \preceq t^c$. We can construct the symbolic execution trace t^s and the function *parse* that maps each bit string occurring in t^c to a term occurring in t^s. This step is similar to that in [9, 8]. If a bit string does not have one of the types defined above, and has not been parsed previously, then we map it to a fresh garbage message symbol in $\mathsf{Garbage}$.

STEP II. Next, we show that with overwhelming probability, the symbolic execution trace t^s constructed above satisfies $t^s \in \mathsf{Exec}^s(\Pi)$, that is, t^s is a Dolev-Yao trace. We assume that in the ℓ-th step of t^s, t^s has already been proven to be a Dolev-Yao trace. For some set φ'_ℓ with $\varphi'_\ell \cap C = \emptyset$, let $\varphi_\ell = \varphi'_\ell \cup C$ be the adversary's knowledge in the ℓ-th step of t^s. Then, it is sufficient to prove that for each **send**($\overline{\mathsf{sid}}$, m) transition made by the PPT adversary $\mathcal{A}$ in the $(\ell+1)$-th step of the concrete execution trace t^c, the adversary makes the query **send**($\overline{\mathsf{sid}}$, $parse(m)$) in the symbolic execution trace t^s, and we have $\varphi_\ell \vdash parse(m)$ with overwhelming probability. To prove this, we assume that $\varphi_\ell \nvdash parse(m)$ holds with non-negligible probability, and derive a contradiction to a computational security such as the unforgeability of the ring signature scheme. By using Lemma 1, we divide this proof into three cases corresponding to the conditions of Lemma 1.

In the first case, there exists a term $[\,t\,]^R_{\mathsf{sk}(a),\mathsf{VK}(G)}$ with $\mathsf{sk}(a') \notin C_{\mathsf{SKey}}$ for any $a' \in G \cap \mathsf{AG}$ such that we have $[\,t\,]^R_{\mathsf{sk}(a),\mathsf{VK}(G)} \sqsubseteq parse(m)$ and $[\,t\,]^{R'}_{\mathsf{sk}(a'),\mathsf{VK}(G)} \not\sqsubseteq m'$ for every $R' \in \mathsf{Rand}$, $a' \in G \cap \mathsf{AG}$, and $m' \in \varphi$. Then, we construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine maintaining the global states of the concrete execution and playing the role of the agents in the execution. By using the oracle O defined in Definition 2, the adversary $\mathcal{B}$ parses the queries made by $\mathcal{A}$ and sends the answers to $\mathcal{A}$. Since $\mathcal{B}$ simulates the execution of protocols on behalf of the agents, $\mathcal{A}$ makes a query **send**($\overline{\mathsf{sid}}$, m) where m contains a ring signature of a message m'' such that no ring signatures of m'' have appeared previously. Then, the adversary $\mathcal{B}$ outputs the ring signature and stops. Hence, $\mathcal{B}$ breaks the existential unforgeability of the ring signature scheme.

In the second case, there exists $\mathsf{sk}(a) \notin C_{\mathsf{SKey}}$ that occurs in $parse(m)$ in a form other than $[\,t\,]^R_{\mathsf{sk}(a),\mathsf{VK}(G)}$. Then, the PPT adversary $\mathcal{A}$ can obtain the signing key corresponding to $\mathsf{sk}(a)$. Therefore, we can construct an adversary that breaks the unforgeability of the ring signature scheme by using the signing key.

In the last case, we have $\mathsf{Nonce}_{ag}(parse(m)) \setminus \mathsf{Nonce}_{ag}(\varphi'_\ell) \neq \emptyset$. Then, the PPT adversary $\mathcal{A}$ can produce an agent's nonce independent of the messages that $\mathcal{A}$ has received. □

Finally, we mention the computational soundness of a ring signature scheme that satisfies *strong existential unforgeability* [1]. To prove the mapping-style soundness theorem similar to Theorem 1, it is sufficient to remove the deduction rule shown in Figure 1 that allows the adversary to produce $[\,m\,]^R_{\mathsf{sk}(a),\mathsf{VK}(G)}$ from $[\,m\,]^{R'}_{\mathsf{sk}(a),\mathsf{VK}(G)}$ for any $R' \in \mathsf{Rand}$.

6 Computational Soundness of Indistinguishability

The mapping-style soundness result in Theorem 1 is insufficient as regards the signer anonymity of the ring signature scheme. In fact, in the proof of Theorem 1, we did not need to use any computational security notion with respect to the signer anonymity of the ring signature scheme. This is because our symbolic model could not deal with indistinguishability-based properties, including signer anonymity, in the deduction rules shown in Figure 1. To deal with signer anonymity in our model, we introduce a symbolic indistinguishability relation and prove its computational soundness.

[1] The strong existential unforgeability of a ring signature scheme [17] forbids anyone to produce a ring signature of a message m using a randomness R from another signature of m using another randomness R'.

6.1 Symbolic Indistinguishability

First, we introduce a set Pattern of symbolic patterns as follows:

$$\mathsf{Pattern} \ni m ::= a \mid g \mid n \mid \mathsf{sk}(a) \mid \mathsf{vk}(a) \mid \mathsf{vk}(g) \mid \langle m,\, m\rangle \mid [\, m \,]^{R}_{\mathsf{sk}(a),\,\mathsf{VK}(G)} \mid [\, m \,]^{R}_{\mathsf{sk}(\square_G),\,\mathsf{VK}(G)}$$

where $a \in \mathsf{AG}$, $g \in \mathsf{Garbage}$, $n \in \mathsf{Nonce}$, $R \in \mathsf{Rand}$, $\mathsf{VK}(G) = \{\, \mathsf{vk}(x) \mid x \in G \,\}$ for $G \subseteq \mathsf{AG} \cup \mathsf{Garbage}$ with $|G \cap \mathsf{AG}| \geq 2$, and $\square_G$ is a special symbol representing some agent in G. Note that patterns are the same as terms except for those representing the signatures $[\, m \,]^{R}_{\mathsf{sk}(\square_G),\,\mathsf{VK}(G)}$ with signer anonymity.

Next, we associate each term with a pattern.

Definition 6 We define the function $pat\colon \mathsf{Term} \times \mathcal{P}(\mathsf{SKey}) \to \mathsf{Pattern}$ that maps $t \in \mathsf{Term}$ and $KS \subseteq \mathsf{SKey}$ to t's pattern with respect to KS.

- $pat(m,\, KS) = m$ (if $m \in \mathsf{Atom} \cup \mathsf{SKey} \cup \mathsf{VKey}$)
- $pat(\langle m_1,\, m_2\rangle,\, KS) = \langle pat(m_1,\, KS),\, pat(m_2,\, KS)\rangle$
- $pat([\, m \,]^{R}_{\mathsf{sk}(a),\,\mathsf{VK}(G)},\, KS) = \begin{cases} [\, pat(m,\, KS) \,]^{R}_{\mathsf{sk}(a),\,\mathsf{VK}(G)} & (\text{if } \mathsf{sk}(a') \in KS \text{ for some } a' \in G\,) \\ [\, pat(m,\, KS) \,]^{R}_{\mathsf{sk}(\square_G),\,\mathsf{VK}(G)} & (\text{otherwise}) \end{cases}$

Then, we define a symbolic indistinguishability relation between terms.

Definition 7 An *atomic renaming* λ is a type-preserving injection from $\mathsf{Nonce} \cup \mathsf{Rand}$ to $\mathsf{Nonce} \cup \mathsf{Rand}$. A *renaming* $\tilde{\lambda}$ with respect to an atomic renaming λ is a function from Pattern to Pattern defined inductively as follows:

- $\tilde{\lambda}(x) = x \quad (x \in \mathsf{AG} \cup \mathsf{Garbage})$
- $\tilde{\lambda}(n) = \lambda(n) \quad (n \in \mathsf{Nonce})$
- $\tilde{\lambda}(\mathsf{sk}(a)) = \mathsf{sk}(a)$
- $\tilde{\lambda}(\mathsf{vk}(x)) = \mathsf{vk}(x)$
- $\tilde{\lambda}(\langle t_1,\, t_2\rangle) = \langle \tilde{\lambda}(t_1),\, \tilde{\lambda}(t_2)\rangle$
- $\tilde{\lambda}([\, t \,]^{R}_{\mathsf{sk}(a),\,\mathsf{VK}(G)}) = [\, \tilde{\lambda}(t) \,]^{\lambda(R)}_{\mathsf{sk}(a),\,\mathsf{VK}(G)}$
- $\tilde{\lambda}([\, t \,]^{R}_{\mathsf{sk}(\square_G),\,\mathsf{VK}(G)}) = [\, \tilde{\lambda}(t) \,]^{\lambda(R)}_{\mathsf{sk}(\square_G),\,\mathsf{VK}(G)}$

Definition 8 Two terms m and m' are *symbolically indistinguishable* with respect to an atomic renaming λ and a set C_{SKey} of signing key symbols, written as $m \cong_{\lambda,\, C_{\mathsf{SKey}}} m'$, if we have $\tilde{\lambda}(pat(m,\, C_{\mathsf{SKey}})) = pat(m',\, C_{\mathsf{SKey}})$. Two terms m and m' are *symbolically indistinguishable*, written as $m \cong m'$, if there exist an atomic renaming λ and a set C_{SKey} of signing key symbols that satisfy $m \cong_{\lambda,\, C_{\mathsf{SKey}}} m'$.

Finally, we define a symbolic indistinguishability relation between symbolic traces.

Definition 9 Two finite symbolic traces t^s_0 and t^s_1 *have equivalent transition types* under an atomic renaming λ and a set C_{SKey} of signing key symbols if they satisfy the following conditions: If we have $C_{\mathsf{SKey}} = \{\, a_1,\, \cdots,\, a_l \,\}$, then the transition $\mathbf{corrupt}(a_1,\, \cdots,\, a_l)$ takes place both in t^s_0 and t^s_1. If a transition in t^s_0 is by $\mathbf{new}(i,\, a_1,\, \cdots,\, a_k)$, the corresponding transition in t^s_1 is by $\mathbf{new}(i,\, a_1,\, \cdots,\, a_k)$. If a transition in t^s_0 is by $\mathbf{send}(\overline{\mathsf{sid}},\, m_0)$, the corresponding transition in t^s_1 is by $\mathbf{send}(\overline{\mathsf{sid}},\, m_1)$, where m_1 is constructed by the same derivation as that of m_0 except that $\tilde{\lambda}$ is applied.

Definition 10 Two finite symbolic traces t^s_0 and t^s_1 are *symbolically indistinguishable* with respect to an atomic renaming λ and a set C_{SKey} of signing key symbols, written as $t^s_0 \cong_{\lambda,\, C_{\mathsf{SKey}}} t^s_1$, if t^s_0 and t^s_1 have equivalent transition types under λ and C_{SKey}, and each term m_0 received by the adversary in t^s_0 and the corresponding term m_1 in t^s_1 are symbolically indistinguishable with respect to λ and C_{SKey}, i.e., $m_0 \cong_{\lambda,\, C_{\mathsf{SKey}}} m_1$. Two finite symbolic traces t^s_0 and t^s_1 are *symbolically indistinguishable*, written as $t^s_0 \cong t^s_1$, if there exist an atomic renaming λ and a set C_{SKey} of signing key symbols that satisfy $t^s_0 \cong_{\lambda,\, C_{\mathsf{SKey}}} t^s_1$.

6.2 Computational Soundness of Indistinguishability

First, we define the computational indistinguishability between distributions.

Definition 11 Two probability distributions D_η and D'_η are *computationally indistinguishable*, written $D_\eta \approx D'_\eta$, if for every PPT adversary A,

$$\Pr[d \leftarrow D_\eta : A(d, \eta) = 1] - \Pr[d' \leftarrow D'_\eta : A(d', \eta) = 1]$$

is negligible in η.

Then, we present a soundness result for computational indistinguishability. Let Π be a protocol. Let $\mathsf{Exec}^{s,\le h}(\Pi)$ be the set of all symbolic traces each of which is a prefix of a symbolic trace in $\mathsf{Exec}^{s}(\Pi)$ and stops at h or fewer steps. Let $\mathsf{Exec}^{c,h}_{\Pi(\cdot),\mathcal{A}(\cdot)}(\eta)$ be the probability distribution obtained from $\mathsf{Exec}^{c}_{\Pi(\cdot),\mathcal{A}(\cdot)}(\eta)$ such that each concrete trace in $\mathsf{Exec}^{c,h}_{\Pi(\cdot),\mathcal{A}(\cdot)}(\eta)$ is a prefix of a concrete trace in $\mathsf{Exec}^{c}_{\Pi(\cdot),\mathcal{A}(\cdot)}(\eta)$ and stops at h steps.

Theorem 2 *Let Π_1 and Π_2 be any two protocols using an existential unforgeable and basically anonymous ring signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$, and $h > 0$ be a fixed integer independent of η. Assume that for every symbolic trace $t_1^s \in \mathsf{Exec}^{s,\le h}(\Pi_1)$, there exists a symbolic trace $t_2^s \in \mathsf{Exec}^{s,\le h}(\Pi_2)$ with $t_1^s \cong t_2^s$, and that for every symbolic trace $t_2^s \in \mathsf{Exec}^{s,\le h}(\Pi_2)$, there exists a symbolic trace $t_1^s \in \mathsf{Exec}^{s,\le h}(\Pi_1)$ with $t_1^s \cong t_2^s$. Then, we have $\mathsf{Exec}^{c,h}_{\Pi_1(\cdot),\mathcal{A}(\cdot)}(\eta) \approx \mathsf{Exec}^{c,h}_{\Pi_2(\cdot),\mathcal{A}(\cdot)}(\eta)$.*

Proof (Overview). For any PPT adversary A and each $0 \le k \le h$, we define $Adv(1, k)$ and $Adv(2, k)$ as follows:

$$Adv(1, k) = \Pr[t_1^c \leftarrow \mathsf{Exec}^{c,k}_{\Pi_1(\cdot),\mathcal{A}(\cdot)}(\eta) : A(t_1^c, \eta) = 1]$$
$$Adv(2, k) = \Pr[t_2^c \leftarrow \mathsf{Exec}^{c,k}_{\Pi_2(\cdot),\mathcal{A}(\cdot)}(\eta) : A(t_2^c, \eta) = 1].$$

Then, we show by induction on k that $Adv(1, k) - Adv(2, k)$ is negligible in η for any PPT adversary A. Clearly, we have $Adv(1, 0) - Adv(2, 0) = 0$. Let $0 \le k < h$. Assume that $Adv(1, k) - Adv(2, k)$ is negligible in η. Then, it is sufficient to prove that $Adv(1, k + 1) - Adv(2, k + 1)$ is also negligible in η.

Let *Supp* be the function that maps a probability distribution to its support. For any PPT adversary A and $j = 1, 2$, we define $\mathsf{Error}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta)$ and $Adv^{error}(j, k)$ as follows:

$$\mathsf{Error}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta) = \left\{ t^c \in Supp(\mathsf{Exec}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta)) \mid t^s \preceq t^c \text{ holds for no } t^s \in \mathsf{Exec}^{s,\le k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta) \right\}$$
$$Adv^{error}(j, k) = \Pr\left[t_j^c \leftarrow \mathsf{Exec}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta) : t_j^c \in \mathsf{Error}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta) \text{ and } A(t_j^c, \eta) = 1\right].$$

We denote the length of a symbolic/concrete trace t by $|t|$, i.e., t stops at $|t|$ steps. For a symbolic trace t^s, we define $[\![t^s]\!]^k$ as follows:

$$[\![t^s]\!]^k = \{ t'^c \mid t^s \preceq t^c \text{ and } t'^c \text{ is a prefix of } t^c \text{ with } |t'^c| = k \}.$$

For $j = 1, 2$, $0 \le k \le h$, and an index set $I(k)$ dependent on k and independent of η, let $\{ t^s_{j,i} \mid i \in I(k) \}$ be a set of symbolic traces that satisfies

$$Supp(\mathsf{Exec}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta)) = \bigcup_{i \in I(k)} [\![t^s_{j,i}]\!]^k \cup \mathsf{Error}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta).$$

We can take $I(k)$ independently of η, because the number of children of each node in each role tree is independent of η. Here, by assumption, we can assume that $t^s_{1,i} \cong t^s_{2,i}$

holds for each $i \in I(k)$ without losing generality. For $i \in I(k)$, we define $P_i(j, k)$ and $Q_i(j, k)$ by:

$$P_i(j, k) = \Pr\left[t_j^c \leftarrow \mathsf{Exec}^{c,k}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta) \colon t_j^c \in [\![\, t_{j,i}^s \,]\!]^k\right]$$
$$Q_i(j, k) = \Pr\left[t_j^c \leftarrow [\![\, t_{j,i}^s \,]\!]^k \colon A(t_j^c, \eta) = 1\right].$$

Then, we have

$$\begin{aligned} Adv(j, k+1) &= \sum_{i \in I(k+1)} \Pr\left[t_j^c \leftarrow \mathsf{Exec}^{c,k+1}_{\Pi_j(\cdot),\mathcal{A}(\cdot)}(\eta) \colon t_j^c \in [\![\, t_{j,i}^s \,]\!]^{k+1} \text{ and } A(t_j^c, \eta) = 1\right] + Adv^{error}(j, k+1) \\ &= \sum_{i \in I(k+1)} P_i(j, k+1) \cdot Q_i(j, k+1) + Adv^{error}(j, k+1). \end{aligned}$$

By the mapping soundness result shown in Theorem 1, $Adv^{error}(1, k{+}1) - Adv^{error}(2, k+1)$ is negligible in η. In addition, we can prove that $Q_i(1, k+1) - Q_i(2, k+1)$ is negligible in η for each $i \in I(k+1)$ by using a hybrid argument similar to [2], because the ring signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ satisfies basic anonymity. By using the induction hypothesis that $Adv(1, k) - Adv(2, k)$ is negligible in η, we can prove that for each $i \in I(k+1)$, $P_i(1, k+1) - P_i(2, k+1)$ is negligible in η. Since $I(k+1)$ is independent of η, $Adv(1, k+1) - Adv(2, k+1)$ is negligible in η. □

7 Conclusion

We proposed a symbolic model dealing with a ring signature, and presented two kinds of computational soundness results in the presence of active adversaries. The first result is the mapping-style soundness: every concrete execution trace corresponds to a symbolic execution trace with overwhelming probability. The second result is the soundness of symbolic anonymity: symbolic indistinguishability implies computational indistinguishability.

Our soundness result for symbolic anonymity is limited to a case where the length of each trace is a constant independent of the security parameter. However, we will take account of traces each of whose length depends on the security parameter by using the technique presented in [7]. In addition, we will also employ a more generic approach [5, 7] to investigate the computationally sound symbolic anonymity of various protocols.

Acknowledgments

We thank Prof. Hubert Comon-Lundh for his helpful comments and suggestions.

References

[1] M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *TACS '01: Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software*, pages 82–94. Springer-Verlag, 2001.

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

[3] P. Adao, G. Bana, and A. Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *CSFW 2005*, pages 170–184, 2005.

[4] M. Backes and D. Unruh. Computational soundness of symbolic zero-knowledge proofs against active attackers. In *CSF 2008*, 2008. To appear. Available at http://www.infsec.cs.uni-sb.de/˜unruh/.

[5] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In *ICALP*, pages 652–663, 2005.

[6] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC*, pages 60–79, 2006.

[7] H. Comon-Lundh and V. Cortier. Computational indistinguishability and observational equivalence. INRIA Research Report RR-6508, INRIA, 2008.

[8] V. Cortier, S. Kremer, R. Küsters, and B. Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In *FSTTCS*, pages 176–187, 2006.

[9] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *ESOP 2005*, pages 157–171, 2005.

[10] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[11] F. D. Garcia and P. van Rossum. Sound and complete computational interpretation of symbolic hashes in the standard model. *Theoretical Computer Science*, 394(1–2):112–133, 2008.

[12] J. Herzog. A computational interpretation of Dolev-Yao adversaries. *Theoretical Computer Science*, 340(1):57–81, 2005.

[13] Y. Kawamoto, H. Sakurada, and M. Hagiya. Computationally sound formalization of rerandomizable RCCA secure encryption. In *Third Franco-Japanese Computer Security Workshop*, 2008.

[14] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *TCC 2004*, pages 133–151, 2004.

[15] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565, 2001.

[16] H. Sakurada and M. Hagiya. Computationally sound formal blind signature. In *Third Franco-Japanese Computer Security Workshop*, 2008.

[17] K. Yoneyama and K. Ohta. Ring signatures: Universally composable definitions and constructions. *IPSJ Digital Courier*, 3:571–584, 2007.