Regular Paper

Ring Signatures: Universally Composable Definitions and Constructions

Kazuki Yoneyama† and Kazuo Ohta†

Though anonymity of ring signature schemes has been studied in many publications, these papers gave different definitions and there has been no consensus. Recently, Bender, et al. proposed two new anonymity definitions of ring signature schemes which are stronger than the previous definitions, that are called anonymity against attribution attacks/full key exposure. In addition, ring signature schemes have two levels of definitions for unforgeability definitions, i.e., existential unforgeability and strong existential unforgeability. In this paper, we will redefine anonymities and unforgeabilities within the universally composable (UC) security framework. First, we will give new ideal functionalities of ring signature schemes for each security level separately. Next, we will show the relations between game-based security definitions and our UC definitions. Finally, we will give another proof for the security of the Bender, et al.'s ring signature scheme within the UC framework. A simulator we constructed in this proof can easily simulate an adversary of existential unforgeability, which can be adaptable to the case of strong existential unforgeability if we assume the exploited signature scheme is a standard single strong existentially unforgeable signature scheme.

1. Introduction

Currently, there are several digital signature schemes which require an *anonymity* property, i.e., a verifier can be convinced that a signature is valid although he cannot identify the true signer among many possible signers. The ring signature scheme is the sort of scheme which is suitable for this kind of situation. For any signed message, ring signature schemes hide the true signer of the message among several signer candidates. A ring signature is realized by allowing the true signer to create the signature by using his own signing key and other members' verification keys who are the members of a group of signer candidates. In 2001, Rivest, et al. [10] proposed the first concrete construction of ring signature schemes with RSA public-keys. Their scheme is based on trapdoor one-way permutations and an ideal block cipher. As an improvement, Abe, et al. [1] proposed a construction which is less computational costs and less storage costs compared to that of Rivest, et al.'s one [10] based on the random oracle model. Dodis, et al. [6] constructed an efficient scheme which has a constant signature size for the number of the group members. Chow, et al. [4] and Bender, et al. [2], independently, first showed constructions which have security proof without relying on the random oracle assumption.

1.1 Ring Signatures in the Universal Composability Framework

In recent years, the *universal composability* (UC) framework [3] as a new technique of security evaluation has been studied. The UC framework is an approach to guarantee the security of protocol A by proving that the *real-life execution* of a protocol A, denoted by ϕ_A, is emulating the *ideal process* for an ideal functionality of protocol A. The ideal functionality $\mathcal{F}_A$ is a generic procedure which captures all features and all necessary security requirements of protocol A. If the proof is given, then protocol A is said to satisfy *UC-security*. The advantage of the UC framework over traditional frameworks is that UC provides strong and robust secure composability, i.e., the security of a primitive (which is UC secure in a stand-alone manner) will always be preserved even when it is executed concurrently with other unlimited numbers of UC secure primitives in an adversarial controlled manner.

The formulation of the ring signature in the UC framework is firstly introduced by Hanatani, et al. [9] and Yoneyama, et al. [11]. They proposed ideal functionalities for the ring signature and proved that a protocol of a ring signature scheme securely realizes functionalities if and only if the scheme satisfies unforgeability and anonymity. However, their results aren't enough since the functionalities only capture a kind of unforgeability and anonymity. In particular, they only considered basic anonymity. Furthermore, though the

† The University of Electro-Communications

strong cryptographic notion of anonymity is defined by Bender, et al. [2], the strong notion in UC framework isn't given. In this paper, we will propose formulations of the ring signature functionality corresponding to strong cryptographic security notions, and a construction which actually satisfies UC-security.

1.2 Security Notions of Ring Signature Schemes

The security of ring signature schemes is discussed with respect to two requirements, i.e., unforgeability and anonymity. Moreover, we should consider a unique attack scenario based on the particularity of ring signature schemes.

In the standard single signature schemes where there is only one true signer, the *adaptively chosen message attacks* (ACMA) [8] is generally recognized as the strongest attack scenario. However, in the case of ring signature schemes where a signature might correspond to more than one signer, in addition to ACMA, we also have to consider an attack scenario where the adversary can adaptively choose the groups which a signature will be considered valid for. This additional adversarial attack is called *adaptively chosen verification key attacks* (ACVKA) [1]. This attack allows an adversary to add any verification key to the list of verification keys of signers.

Therefore, by considering ACVKA, the desirable notion of unforgeability on ring signature schemes is existential unforgeability against adaptively chosen message and verification key attacks (eUF-ACMA&ACVKA) (Definition 2.3). This notion requires that no adversary can create a signature $\widetilde{\sigma}$ of never signed message $\widetilde{m}$ except with negligible probability, such that $\widetilde{\sigma}$ is verified as valid for $\widetilde{m}$ with respect to the correct verification key list. Note that, we have to consider unforgeability toward the verification key list of a group as well as a message in order to prevent a forge for ever signed message with different groups since the verification key list (of the group) is specified from all signer candidates as an input of signature generation.

Furthermore, we can consider the stronger notion of unforgeability, known as strongly existential unforgeability against adaptively chosen message and verification key attacks (sUF-ACMA&ACVKA) (Definition 2.4). This notion differs from eUF-ACMA&ACVKA in the following point: In addition to the requirement of eUF-ACMA&ACVKA, sUF-ACMA&ACVKA requires that any adversary can't even create a valid signature $\widetilde{\sigma}$ of already signed message $\widetilde{m}$ with respect to the verification key list except with negligible probability. Most previous schemes [2),4),6),10)] adopt eUF-ACMA&ACVKA and it seems enough from a practical viewpoint. However, eUF-ACMA&ACVKA give no guarantee when a forger who has a valid signature of a message with respect to a verification key list creates another valid signature of the same message with respect to the same verification key list. So, sUF-ACMA&ACVKA may be needed for some applications or situations.

With regard to the notion of anonymity, prior works provide various definitions. The way of defining anonymity has mainly two different views.

One view requires that the adversary should be unable to "know" who is the true signer in a ring nor be able to "distinguish" whether a signature of the ring is generated by the true signer. In this case, at least two honest parties are required in the ring for guaranteeing signer anonymity completely. If all members in the ring are corrupted except one honest true signer, or all verification keys are chosen by ACVKA except the true signer's one, then the adversary can "know" who is the true signer regardless of whether she distinguishes the signature since the adversary didn't generate the signature himself. Most previous works are established from this viewpoint (Definition 2.5).

The other view is proposed by Bender, et al. [2]. This view requires the security that the adversary cannot show any evidence that a signature is the true signer's one even when the adversary "knows" the true signer. That is, it requires that the adversary who corrupts all members of the ring except one honest true signer should be unable to prove to a third party that the true signer has generated the signature. This situation means that the adversary can prove to a third party by using all internal states of corrupted parties. Such a situation requires, in other words, that the signature of the true signer is distinguishable from signatures of other members by using all internal states of corrupted parties. Therefore, we have to consider the security in the situation where randomnesses used at key generation of the other members of the ring except the true signer are exposed. This notion is called anonymity against *attribution attacks* (Definition 2.6). Furthermore, we can also consider the stronger variant that the security is guar-

anteed even if all randomnesses including the true signer's one in the ring are exposed. This notion is called anonymity against *full key exposure* (Definition 2.6).

1.3 Our Contribution

This paper has for the main part a definitional contribution.

Universally composable definition The previous definition of the UC ring signature [11] only captures sUF-ACMA&ACVKA and basic anonymity. Also, though stronger definitions of anonymity are studied by Bender, et al. [2], they only gave cryptographic definitions, i.e., no UC definitions. In this paper, we formulate definitions of a UC ring signature with regard to various security levels. In particular, we propose an ideal ring signature functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ which is convertible by a level of unforgeability *uf* and a level of anonymity *anon*. We are able to choose eUF or sUF as *uf*, and basic anonymity, anonymity against attribution attacks or anonymity against full key exposure as *anon*. So, our functionality can represent six kinds of security notions by the combination of unforgeability and anonymity. This convertibility of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ is useful in order to capture a necessary security property for analyzing a protocol. Moreover, we show the relations between cryptographic definitions and our functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ at all security levels. As a result, realizing $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ is equivalent to ensuring traditional cryptographic security notions of a ring signature. Therefore, our UC definitions of a ring signature are well-designed.

Universally composable construction In this paper, we also show concrete constructions which securely realize our functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. We adapt Bender, et al.'s scheme [2] (BKM scheme) which is secure without relying on the random oracle assumption.

2. Preliminaries

We briefly show the protocol syntax in the UC definition, and we will present the intuitive framework of ring signature schemes. Next, we define a specific description with respect to ring signature, attack models and security notion.

Protocol syntax Following the reference [3], a protocol is represented as a system of interactive Turing machines (ITMs). ITM has a session identifier (SID), a party identifier (PID). SID shows belonging session of the ITM. PID shows the party identifier of the ITM. The pair (SID, PID) is unique in the system. We assume that all ITMs run in probabilistic polynomial time.

Ring signature schemes Ring signature schemes allow any party to generate a pair of keys, i.e., a signing key and a verification key. A signer chooses group members from among parties who make public their verification keys. Let G_{all} be the set of parties $P_1, \ldots, P_n$ and let L_{all} be the list of their verification keys. Furthermore, let G be a subset of G_{all} and let L be the list of G's verification keys. Also, a signature of a message is generated by a signer of group G. Though any party can verify the signature using L as a verifier, he cannot identify the signer among G.

Definition 2.1 (Ring Signature Schemes) A ring signature scheme Σ_r consists of the following 3-tuple (**RGen**, **RSign**, **RVer**):

- **RGen** is a key generation algorithm which on input 1^k, where k is the security parameter, outputs a pair of keys (sk, vk). sk and vk are called the signing key and verification key.
- **RSign** is a signature generation algorithm which takes as inputs a message m and a list of verification keys L from the signing key sk_i corresponding to verification key $vk_i \in L$, and outputs a signature σ.
- **RVer** is a verification algorithm which takes as inputs a message m, a list of verification keys L and a signature σ, and output a bit 1 or 0 (i.e., **accept** or **reject**). For any i where $vk_i \in L$, **RVer** satisfies $1 \leftarrow \mathbf{RVer}(m, L, \mathbf{RSign}(m, L, sk_i))$. **RVer** has to be stateful, i.e., if $(m, L, \sigma) = (m', L', \sigma')$, then $\mathbf{RVer}(m, L, \sigma) = \mathbf{RVer}(m', L', \sigma')$.

In ring signature schemes, we should consider the situation where an adversary (i.e., a forger or a distinguisher) adds arbitrary verification keys to the list of verification keys and asks the signing oracle to sign for arbitrary messages with the arbitrary subset of keys. Let L_{valid} be the list of verification keys which are generated by **RGen** as valid keys, L_{invalid} be the list of verification keys which are added by the forger and L_{all} be the list of all registered verification keys (i.e., $L_{\mathrm{all}} = L_{\mathrm{valid}} \cup L_{\mathrm{invalid}}$).

Definition 2.2 (Adaptively Chosen Message and Chosen Verification key Attack) An adversary is allowed to behave arbitrarily as follows:

- **ACMA:** Sends (m, L, i) to the signing

oracle $\mathcal{SO}$. If $L \subseteq L_{\text{all}}$ and $vk_i \in L \cap L_{\text{valid}}$, then $\mathcal{SO}$ returns $\sigma \leftarrow \mathbf{RSign}(m, L, sk_i)$. Else, $\mathcal{SO}$ returns $\perp$. Let $\{(\bar{m}, \bar{L}, \bar{\sigma})\}$ denote all the 3-tuple the adversary obtained from queries to $\mathcal{SO}$.

• **ACVKA:** Adds arbitrary vk to L_{invalid}.

The security of ring signature schemes is characterized by two properties (i.e., *unforgeability* and *anonymity*). First, we formally define the cryptographic notion of unforgeability on two levels.

Definition 2.3 (eUF-ACMA&ACVKA) A ring signature scheme Σ_r is existentially unforgeable against adaptively chosen message and chosen verification key attacks (eUF-ACMA&ACVKA) if the probability that the following experiment holds for a security parameter k is negligible;

(1) The key pair (sk_i, vk_i) for all parties is generated from $\mathbf{RGen}(1^k)$, and the list of all verification keys $L_{\text{all}}(= L_{\text{valid}})$ is given to the forger.
(2) The forger plays ACMA&ACVKA arbitrary and can obtain their secret keys by corrupting parties.
(3) The forger outputs $(\widetilde{m}, \widetilde{L}, \widetilde{\sigma})$ satisfying $\mathbf{RVer}(\widetilde{m}, \widetilde{L}, \widetilde{\sigma}) = 1$, $(\widetilde{m}, \widetilde{L}, *) \notin \{(\bar{m}, \bar{L}, \bar{\sigma})\}$ where $*$ means a wildcard, $\widetilde{L}$ is the verification key list of the group $\widetilde{G}$ and $\widetilde{L} \subseteq L_{\text{valid}}$, and all members of $\widetilde{G}$ are uncorrupted.

Definition 2.4 (sUF-ACMA&ACVKA) A ring signature scheme Σ_r is strong existentially unforgeable against adaptively chosen message and chosen verification key attacks (sUF-ACMA & ACVKA) if the probability that the following experiment holds for a security parameter k is negligible;

(1) The same as (1) in Definition 2.3.
(2) The same as (2) in Definition 2.3.
(3) The forger outputs $(\widetilde{m}, \widetilde{L}, \widetilde{\sigma})$ satisfying $\mathbf{RVer}(\widetilde{m}, \widetilde{L}, \widetilde{\sigma}) = 1$, $(\widetilde{m}, \widetilde{L}, \widetilde{\sigma}) \notin \{(\bar{m}, \bar{L}, \bar{\sigma})\}$, $\widetilde{L}$ is the verification key list of the group $\widetilde{G}$ and $\widetilde{L} \subseteq L_{\text{valid}}$, and all members of $\widetilde{G}$ are uncorrupted.

Next, we define the cryptographic notion of anonymity in the two senses. Informally speaking, the notion of anonymity should guarantee that given a signature, it is not feasible for any distinguisher to tell the signing key which is used to generate the signature from the others in G. In the traditional sense, anonymity is guaranteed only when there are at least two honest parties in the ring. This restriction arises because the adversary "knows" signatures by generating corrupted parties, i.e., the adversary is able to tell which is the signature of the honest party. We call the definition based on this sense *basic* anonymity.

Definition 2.5 (Basic anonymity) A ring signature scheme Σ_r is basic anonymous if the probability that the following experiment holds for a security parameter k is negligibly close to 1/2;

(1) The key pair (sk_i, vk_i) for all parties is generated from $\mathbf{RGen}(1^k)$, and the list of all verification keys $L_{\text{all}}(= L_{\text{valid}})$ is given to the distinguisher.
(2) The distinguisher plays ACMA&ACVKA arbitrarily and by corrupting parties can obtain their secret keys.
(3) The distinguisher outputs (m, i_0, i_1, L) where $vk_{i_0}, vk_{i_1} \in L_{\text{valid}} \cap L$, P_{i_0} and P_{i_1} are uncorrupted, and $(m, L, \mathbf{RSign}(m, L, sk_{i_0}))$, $(m, L, \mathbf{RSign}(m, L, sk_{i_1})) \notin \{(\bar{m}, \bar{L}, \bar{\sigma})\}$. Then, the distinguisher obtains the signature $\sigma_b \leftarrow \mathbf{RSign}(m, L, sk_{i_b})$ where b is a randomly chosen bit.
(4) The distinguisher guesses a bit $\tilde{b}$ and $\tilde{b} = b$.

Though the basic anonymity is adopted in most traditional studies, it's desirable that we can catch the notion of anonymity when there is only one honest party (or there is no honest party) in the ring. However, from the viewpoint of whether the distinguisher knows the true signer or not, the restriction of basic anonymity is inevitable. Therefore, we are able to consider the notion of anonymity from the other viewpoint which requires that the only one honest party isn't framed by other corrupted parties in the ring, i.e., no corrupted parties can show any evidence that the honest party has generated a signature even if they use all the information of their secrets and internal states (e.g., randomness used at key generation). This notion is called anonymity against attribution attacks in Ref. 2). Moreover, the stronger notion which guarantees anonymity even if secrets and internal states of all parties in the ring (including the honest one) are exposed is called anonymity against full key exposure.

Definition 2.6 (Anonymity against attribution attacks/full key exposure) A ring signature scheme Σ_r is anonymous against attribution attacks/full key exposure if the probability that the following experiment holds for a security parameter k is negligibly close to 1/2;

(1) The key pair (sk_i, vk_i) for all parties is generated from $\mathbf{RGen}(1^k; \omega_i)$ where ω_i is randomness, and the list of all verification keys $L_{\mathrm{all}}(= L_{\mathrm{valid}})$ is given to the distinguisher.
(2) The distinguisher plays ACMA&ACVKA arbitrarily.
(3) The distinguisher outputs (m, i_0, i_1, L) where $vk_{i_0}, vk_{i_1} \in L_{\mathrm{valid}} \cap L$ and $(m, L, \mathbf{RSign}(m, L, sk_{i_0}))$, $(m, L, \mathbf{RSign}(m, L, sk_{i_1})) \notin \{(\bar{m}, \bar{L}, \bar{\sigma})\}$. Then, the distinguisher obtains the signature $\sigma_b \leftarrow \mathbf{RSign}(m, L, sk_{i_b})$ where b is a randomly chosen bit and randomnesses $\{\omega\}_{i \neq i_0}$ (in the case of full key exposure, all randomnesses is given).
(4) The distinguisher guesses a bit $\tilde{b}$ and $\tilde{b} = b$.

3. Ring Signature Functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$

In this section, we will introduce a new definition of ring signature schemes in UC framework, i.e., a new ideal ring signature functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$.

3.1 Definition of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$

$\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ receives three instructions for the basic function of ring signature schemes (Key Generation, Signature Generation and Signature Verification requests) and one instruction for the adversary (Attribution request). $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ covers six kinds of security levels, i.e., combination of two levels of unforgeability and three levels of anonymity. That is, $(uf, anon)$ represents a security level as uf is parameterized by eUF and sUF, and $anon$ is parameterized by basic anonymity, anonymity against attribution attacks and anonymity against full key exposure. From now on, for $anon$, we use basic as basic anonymity, attribution as anonymity against attribution attacks and full-key as anonymity against full key exposure for short. Specifically, a Signature Verification request concerns unforgeability and an Attribution request concerns anonymity.

$\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ is the standard corruption functionality. If the adversary corrupts some party P_j and P_j finished Key Generation request, then $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ outputs all internal state of P_j to the adversary.

Also, we will show the relations between cryptographic security notions as Section 2 and our functionality. **Figure 1** shows the functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$.

Determining algorithms As with the basic signature functionality [3], the values of verification keys and legitimated signatures are determined by the adversary via algorithms **RV** and **RS** since the notion of the security of ring signature schemes doesn't make any requirements on these values. That is, the signature values may depend on the identity of the signer or all signature values are identical. Making sure that signature values are independent from the identity of the signer or are different from each other is only a technical tool which allows realizing the abstract requirement in an algorithmic way. Naturally, we can reflect these requirements in the formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$, only we adapt the general formulation in this paper.

Guaranteeing completeness and statefulness As Definition 2.1, ring signature schemes have to satisfy completeness, i.e., $1 \leftarrow \mathbf{RVer}(m, L, \mathbf{RSign}(m, L, sk_i))$, and statefulness, i.e., if $(m, L, \sigma) = (m', L', \sigma')$, then $\mathbf{RVer}(m, L, \sigma) = \mathbf{RVer}(m', L', \sigma')$. $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ guarantees both completeness and statefulness as follows: If a signature σ is honestly generated, then $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ exactly outputs 1 at Signature Verification request since $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ already checks $\bigwedge_{j \in G} \mathbf{RV}_j(m, L, \sigma) \stackrel{?}{=} 1$ for σ at Signature Generation request. Also, for the same verification request $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ outputs the consistent value since all **RV** are deterministic algorithms.

Guaranteeing unforgeability If an event corresponding to the condition of unforgeability occurs, $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ outputs that the signature is invalid to the Signature Verification request. Also, the difference of the condition between eUF-ACMA&ACVKA (Definition 2.3) and sUF-ACMA&ACVKA (Definition 2.4) appears at the Signature Verification request since the difference of two notions is only the range of signatures which should be dealt with as forged signatures, i.e., $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ decides forged signatures.

Guaranteeing anonymity The adversary obtains no information about the linkage between the identity of a signer and a generated signature at the Signature Generation request since $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ generates the signature by using the signing algorithm **RS** without any interaction with the adversary. Therefore, as long as the adversary doesn't corrupt the signer, perfect

Functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$

Key Generation: On input ("KeyGen", sid) from party P_i, forward ("KeyGen", sid) to the adversary. If it is the first "KeyGen" input to the adversary, then obtain a ("Algorithms", $sid, \mathbf{RS}, \mathbf{RV}_i$) from the adversary, where $\mathbf{RS}$ is a description of a PPT ITM and $\mathbf{RV}_i$ is a description of a deterministic polytime ITM. Otherwise, then obtain a ("Algorithms", $sid, \mathbf{RV}_i$) from the adversary, where $\mathbf{RV}_i$ is a description of a deterministic polytime ITM. Finally, output ("Verification Algorithms", $sid, \mathbf{RV}_i$) to P_i, record $L_{\mathrm{all}} \leftarrow L_{\mathrm{all}} \cup \{\mathbf{RV}_i\}$.

Signature Generation: On input ("Sign", sid, m, L) from P_i, check $(L \subseteq L_{\mathrm{all}}) \wedge (\mathbf{RV}_i \in L)$. If not, ignore the input. Else, let $\sigma = \mathbf{RS}(m, L)$. If $\bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(m, L, \sigma) = 1$, then record (m, L, σ, P_i) and output ("Signature", sid, m, L, σ). Else, output an error message.

Signature Verification: On input ("Verify", sid, m, L, σ) from some party P_k, output ("Verified", sid, m, f) where:

- Case of $uf = \mathsf{eUF}$
  - **Event 1.** If $L \subseteq L_{\mathrm{all}}$, $(m, L, *, *)$ is not recorded and any party in G is not corrupted, then set $f = 0$.
  - **Event 2.** Else, set $f = \bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(m, L, \sigma)$.
- Case of $uf = \mathsf{sUF}$
  - **Event 1.** If $L \subseteq L_{\mathrm{all}}$, $(m, L, \sigma, *)$ is not recorded and any party in G is not corrupted, then set $f = 0$.
  - **Event 2.** Else, set $f = \bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(m, L, \sigma)$.

Attribution: On input ("Attribute", sid, m, L, σ) from the adversary, check that there exists P_i s.t. (m, L, σ, P_i) is recorded. If not, ignore the input. Otherwise, output a message to the adversary where:

- Case of $anon = \mathsf{basic}$
  - **Event 1.** If P_i is uncorrupted and there is the other one honest party in G, then output error message.
  - **Event 2.** Else, output ("Attributed", sid, m, σ, P_i).
- Case of $anon = \mathsf{attribution}$
  - **Event 1.** If there is honest party in G, then output error message.
  - **Event 2.** Else, output ("Attributed", sid, m, σ, P_i).
- Case of $anon = \mathsf{full\text{-}key}$
  - Always output error message.

Fig. 1 Ring signature functionality $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$.

anonymity is guaranteed for the signature. Intuitively, the Attribution request represents the attribution attack by the adversary. For this attack, the requirements of anonymity property differ in each definitions. In the case of basic anonymity (Definition 2.5), if there aren't at least two uncorrupted parties in the ring, then Event 2 at the Attribution request in $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ occurs, i.e., the anonymity isn't guaranteed. Also, in the case of anonymity against attribution attacks (one of Definition 2.6), if all parties are corrupted, then Event 2 occurs. In the case of anonymity against full key exposure (the other of Definition 2.6), the adversary can't obtain any information regarding the true signer even if all parties are corrupted.

Allowing adaptively chosen verification key attacks When the forger E and the distinguisher D perform adaptively chosen verification key attack, they can generate an arbitrary verification key and add it to the list of verification key L at Key Generation request by corrupting some party by formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. This situation represents ACVKA. From the formulation of Event 1 at the Signature Verification request, the unforgeability property is preserved even if ACVKA is allowed.

3.2 Equivalence Relations between Cryptographic Notions and $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$

Here, we show the relations between realizing $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ and ensuring both unforgeability and anonymity which correspond to security levels.

Let Σ_r be a ring signature scheme. Then, we

Protocol π_{Σ_r}

Key Generation: On input ("KeyGen", sid), P_i runs algorithm **RGen** and records the pair (sk_i, vk_i).

Signature Generation: On input ("Sign", sid, m, L), P_i runs algorithm **RSign**, obtains $\sigma = \mathbf{RSign}(m, L, sk_i)$ and outputs ("Signature", sid, m, L, σ).

Signature Verification: On input ("Verify", sid, m, L, σ), any party outputs ("Verified", $sid, m, \mathbf{RVer}(m, L, \sigma)$).

Fig. 2 The generic protocol of ring signature schemes π_{Σ_r}.

describe a generic protocol π_{Σ_r} corresponding to Σ_r. **Figure 2** shows the protocol π_{Σ_r}.

Theorem 3.1 π_{Σ_r} securely realizes $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ if and only if Σ_r satisfies both unforgeability and anonymity according to uf and $anon$.

[Proof]

First, we prove "only if" direction. The proof outline is that we can construct a real model adversary $\mathcal{A}$ and an environment $\mathcal{Z}$ which successfully distinguishes the real model from the ideal model, that is, $\mathcal{Z}$ distinguishes an interaction with $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ and π_{Σ_r} if Σ_r does not satisfy unforgeability or anonymity ☆.

(I). For proving the case of no unforgeability of uf, we consider a successful forger E_{uf}. Let $\mathcal{Z}$ internally runs a copy of E_{uf} with L_{all} obtained from parties as the input. Whenever E_{uf} plays ACVKA to add a verification algorithm $\mathbf{RV}_j$ to L_{all}, $\mathcal{Z}$ instructs the adversary to corrupt some party P_j, to request ("KeyGen", sid) and to output $\mathbf{RV}_j$. Also, whenever E_{uf} sends (m, L, i) to the signing oracle $\mathcal{SO}$, $\mathcal{Z}$ activates P_i with ("Sign", sid, m, L) and returns the obtained signature σ to E_{uf}. If E_{uf} requires corrupting a party, $\mathcal{Z}$ instructs the adversary to corrupt the party and outputs the obtained signing key (or algorithm) from the adversary to E_{uf}.

When E_{uf} outputs a forged tuple $(\widetilde{m}, \widetilde{L}, \widetilde{\sigma})$ where $\widetilde{L} \subseteq L_{\mathrm{all}}$, $\mathcal{Z}$ proceeds as follows: If uf = eUF, $(\widetilde{m}, \widetilde{L}, *)$ was queried to $\mathcal{SO}$ where $*$ means a wildcard or there is a corrupted member in $\widetilde{G}$, then $\mathcal{Z}$ outputs 0 and halts. Also, if uf = sUF, $(\widetilde{m}, \widetilde{L}, \widetilde{\sigma})$ was queried to $\mathcal{SO}$ or there is a corrupted member in $\widetilde{G}$, then $\mathcal{Z}$ outputs 0 and halts. Else, $\mathcal{Z}$ sends ("Verify", $sid, \widetilde{m}, \widetilde{L}, \widetilde{\sigma}$) to a party and outputs the verification result.

In the real model, $\mathcal{Z}$ outputs 1 with non-negligible probability since $\mathcal{Z}$ perfectly gives interfaces for ACMA&ACVKA to E_{uf} in cases of both uf = eUF and sUF, and we assume E_{uf} is a successful forger. However, in the ideal model, $\mathcal{Z}$ exactly outputs 0 since events which $\mathcal{Z}$ outputs 1 don't occur in cases of both uf = eUF and sUF at Signature Verification request because of the formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. Therefore, we can construct an environment $\mathcal{Z}$ which successfully distinguishes the real model from the ideal model.

(II). For proving the case of no anonymity of $anon$, we consider a successful distinguisher D_{anon}. Let $\mathcal{Z}$ internally runs a copy of D_{anon} with L_{all} obtained from parties as the input. When D_{anon} requires something to $\mathcal{Z}$, i.e., ACMA, ACVKA and corruption, $\mathcal{Z}$ returns outputs like **(I)**. In addition, when D_{anon} wants to examine whether a signature is generated by a certain party with a message m and a verification key list L, $\mathcal{Z}$ sends ("Attribute", sid, m, L, σ) to the adversary and returns the output obtained from the adversary.

When D_{anon} queries a challenge (m, i_0, i_1, L) where $vk_{i_0}, vk_{i_1} \in L_{\mathrm{valid}} \cap L$, $\mathcal{Z}$ randomly chooses a bit b, sends ("Sign", sid, m, L) to P_b and returns the obtained signature σ_b to D_{anon}. Then, if $anon$ = attribution, then $\mathcal{Z}$ corrupts $\{P_i\}_{i \neq i_0}$ and hands obtained signing keys (or algorithms) to D_{anon}. If $anon$ = full-key, then $\mathcal{Z}$ corrupts all parties and hands obtained signing keys (or algorithms) to D_{anon}. When D_{anon} outputs a bit b, $\mathcal{Z}$ proceeds as follows: If $anon$ = basic, and P_{i_0} or P_{i_1} is corrupted or (m, L, σ_b) was queried to $\mathcal{SO}$ then $\mathcal{Z}$ outputs 0 and halts. Also, if $anon$ = attribution, and P_{i_0} and P_{i_1}

☆ In the cases of no completeness or no statefulness, trivially we can construct $\mathcal{Z}$ which successfully distinguishes the real model from the ideal model.

Simulator $\mathcal{S}$

◇ Interaction with $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$

- First, run the algorithm **RGen**, obtain the pair (sk', vk') and record $\mathbf{RS} := \mathbf{RSign}(\cdot, (vk', \cdots), sk')$. On input ("KeyGen", sid) from $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$, run algorithm **RGen**, obtain the pair (sk, vk) and record $\mathbf{RV} := \mathbf{RVer}(\cdot, (vk', vk, \cdots), \cdot)$. If it is the first "KeyGen" input from $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$, then send ("Algorithms", $sid, \mathbf{RS}, \mathbf{RV}$) back to $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. Otherwise, then send ("Algorithms", $sid, \mathbf{RV}$) back to $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$.

◇ Interaction with $\mathcal{Z}$

- When being ordered by $\mathcal{Z}$ to corrupt some party P_j, corrupt P_j, inform a corruption to $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ and report the obtained information from $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ to $\mathcal{Z}$. (If P_j already finished Key Generation request, the information includes the signing algorithm **RS**.)
- When being ordered by $\mathcal{Z}$ to attribute a signature σ to some party, report the obtained information from $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ to $\mathcal{Z}$.

Fig. 3 Simulator $\mathcal{S}$.

are corrupted or (m, L, σ_b) was queried to $\mathcal{SO}$ then $\mathcal{Z}$ outputs 0 and halts. Also, if $anon = \mathsf{full\text{-}key}$ and (m, L, σ_b) was queried to $\mathcal{SO}$ then $\mathcal{Z}$ outputs 0 and halts. Else, $\mathcal{Z}$ outputs 1 if $\tilde{b} = b$.

In the real model, $\mathcal{Z}$ outputs 1 with half plus non-negligible probability since $\mathcal{Z}$ perfectly gives interfaces for ACMA&ACVKA and corruption to D_{anon} in cases of all $anon = \mathsf{basic}, \mathsf{attribution}$ and $\mathsf{full\text{-}key}$, and we assume D_{anon} is a successful distinguisher. However, in the ideal model, $\mathcal{Z}$ never outputs 1 over half plus non-negligible probability since D_{anon} cannot correctly guess the bit b with at most half probability in cases of all $anon = \mathsf{basic}$, $\mathsf{attribution}$ and $\mathsf{full\text{-}key}$ even if the adversary uses the Attribution request because of the formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. Therefore, we can construct an environment $\mathcal{Z}$ which successfully distinguishes the real model from the ideal model.

From **(I)** and **(II)**, "only if" direction is proven.

Next, we prove "if" direction. The proof outline is that Σ_r doesn't satisfy unforgeability or anonymity if there exists a real model adversary $\mathcal{A}$ for any simulator (ideal model adversary) $\mathcal{S}$ such that there is an environment $\mathcal{Z}$ which successfully distinguishes two interactions with $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ or π_{Σ_r}. $\mathcal{S}$ has to be a generic simulator corresponding to the adversary $\mathcal{A}$. **Figure 3** shows the description of $\mathcal{S}$.

Now, we consider a fixed environment $\mathcal{Z}$. $\mathcal{Z}$ interacts with $\mathcal{S}$ to corrupt some party and to attribute a signature to a party. By formulation of $\mathcal{S}$ as Fig. 3, $\mathcal{S}$ carries out the same corruption in the ideal model as $\mathcal{A}$ carries out in the real model. The difference for $\mathcal{Z}$ only consists in interfaces between two interactions with parties and the adversary in the ideal model and in the real model. That is, $\mathcal{Z}$ distinguishes the real model from the ideal model when outputs of parties and the adversary are different at the Key Generation request, Signature Generation request, Signature Verification request or Attribution request.

(I). First, we will show the reduction to unforgeability for uf. Let $B_{E_{\mathsf{eUF}}}$ denote the event that at some point of running π_{Σ_r}, all members $P_i \in G$ is uncorrupted, for $i = 1$ to n P_i generates a verification key vk_i, $\mathcal{Z}$ requests some party to verify a signature σ of a message m with L such that $vk_i \in L$ and $1 \leftarrow \mathbf{RVer}(m, L, \sigma)$ where $(m, L, *)$ is not issued. Let $B_{E_{\mathsf{sUF}}}$ denote the same event as $B_{E_{\mathsf{eUF}}}$ except $1 \leftarrow \mathbf{RVer}(m, L, \sigma)$ where (m, L, σ) is not issued. Here, we assume Σ_r satisfies completeness, statefulness and anonymity ☆. Then, we show that for $\mathcal{Z}$ the difference between the real model and the ideal model is only detected at event $B_{E_{uf}}$ by formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ and π_{Σ_r}. That is represented by an equation as

$$\begin{aligned}&\Pr[\mathcal{Z} \text{ outputs } 1 \mid \neg B_{E_{uf}} \\ &\qquad \wedge\ \mathcal{Z} \text{ interacts with } \pi_{\Sigma_r} \text{ and } \mathcal{A}] \\ &= \Pr[\mathcal{Z} \text{ outputs } 1 \mid \neg B_{E_{uf}} \\ &\qquad \wedge\ \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S}]. \end{aligned} \tag{1}$$

Here, we will prove Eq. (1). At the Key Generation request, $\mathcal{S}$ can perfectly sim-

☆ Indeed, as later, Σ_r does not satisfy unforgeability regardless of whether Σ_r satisfies or not completeness, statefulness and anonymity if Event $B_{E_{uf}}$ occurs. In order to concentrate the relation between Event $B_{E_{uf}}$ and unforgeability, we assume Σ_r satisfies completeness, statefulness and anonymity.

ulate outputs of parties and the adversary regardless of whether there is a corrupted party or not since completeness and statefulness are satisfied. At the Signature Generation request, if the signer is uncorrupted, outputs of the signer are the same both in the real and ideal model since the adversary can do nothing. If the signer is corrupted, outputs of the signer are also the same since it is decided by the adversary and $\mathcal{S}$ can perfectly simulate $\mathcal{A}$. Also, at the Attribution request, $\mathcal{A}$ has the same capacity on *anon* as $\mathcal{S}$ with respect to distinguishing signatures since anonymity is satisfied and because of the formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. So, the difference may exist only at a Signature Verification request.

At a Signature Verification request, if a member $P_i \in G$ is corrupted, then all verification results can be decided by $\mathcal{S}$ in the ideal model regardless of whether Σ_r satisfies completeness, statefulness and anonymity since verification results are derived from all verification algorithms of members in the ring including the corrupted one. In the real model, P_i can also output arbitrary verification. Therefore, there is no difference for $\mathcal{Z}$.

If all signers $P_i \in G$ are uncorrupted, in the cases of both $uf = \mathsf{eUF}$ and sUF, when Event 2 at Signature Verification request in $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ occurs, it corresponds to completeness, statefulness or no guarantee for verification results. So, Σ_r satisfies completeness and statefulness, there is no difference for $\mathcal{Z}$ between the real model and the ideal model. When Event 1 at a Signature Verification request occurs, $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ always outputs 0. However, in the real model, the output of a verifier may be different from the corresponding output of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ (i.e., event $B_{E_{uf}}$ occurs in the real model with non-negligible probability). Therefore, the difference for $\mathcal{Z}$ is only detected at event $B_{E_{uf}}$ by formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ and we have Eq. (1).

Here, we consider a forger E_{uf} on Σ_r. We construct E_{uf} from $\mathcal{Z}$ (i.e., E_{uf} runs simulated copy of $\mathcal{Z}$). At Key Generation request for a party P_i, instead of running **RGen**, E_{uf} hands $\mathcal{Z}\,\mathbf{RV}_i := \mathbf{RVer}(\cdot,(vk_i,\cdot),\cdot)$ where vk_i is a verification key of the input L. At a Signature Generation request with a message m for L, instead of running **RSign**, forwards this request to signing oracle $\mathcal{SO}$. At a Signature Verification request with (m,L,σ) for a party P_j, E_{uf} runs $\bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(m,L,\sigma)$. At an Attribution request with (m,L,σ), E_{uf} outputs the signer identity if the condition of corruption doesn't guarantee anonymity for *anon*. In the case of $uf = \mathsf{eUF}$, if $\bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(\widetilde{m},L,\widetilde{\sigma})$ outputs 1 and $\widetilde{m}$ was never signed for L, then E_{uf} completes the simulation and returns $(\widetilde{m},L,\widetilde{\sigma})$. Else, then E_{uf} hands $\mathcal{Z}$ $\bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(m,L,\sigma)$. In the case of $uf = \mathsf{sUF}$, if $\bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(\widetilde{m},L,\widetilde{\sigma})$ outputs 1 and $\widetilde{\sigma}$ was never generated as signature of $\widetilde{m}$ for L, then E_{uf} completes the simulation and returns $(\widetilde{m},L,\widetilde{\sigma})$. Else, then E_{uf} hands $\mathcal{Z}$ $\bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(m,L,\sigma)$. Furthermore, when $\mathcal{Z}$ halts or the adversary corrupts a party in G, then E_{uf} fails.

Thus, by construction of E_{uf} and Eq. (1), we have

$$
\begin{aligned}
&|\Pr[\mathcal{Z} \text{ outputs } 1 \mid \mathcal{Z} \text{ interacts with } \pi_{\Sigma_r} \text{ and } \mathcal{A}] - \Pr[\mathcal{Z} \text{ outputs } 1 \mid \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S}]| \\
&= \Big(\Pr[\,B_{E_{uf}}\,] \cdot \Pr[\,\mathcal{Z} \text{ outputs } 1 \mid B_{E_{uf}} \wedge \mathcal{Z} \text{ interacts with } \pi_\Sigma \text{ and } \mathcal{A}\,] \\
&\quad + \Pr[\,\neg B_{E_{uf}}\,] \cdot \Pr[\,\mathcal{Z} \text{ outputs } 1 \mid \neg B_{E_{uf}} \wedge \mathcal{Z} \text{ interacts with } \pi_\Sigma \text{ and } \mathcal{A}\,]\Big) \\
&\quad - \Big(\Pr[\,B_{E_{uf}}\,] \cdot \Pr[\,\mathcal{Z} \text{ outputs } 1 \mid B_{E_{uf}} \wedge \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S}\,] \\
&\quad + \Pr[\,\neg B_{E_{uf}}\,] \cdot \Pr[\,\mathcal{Z} \text{ outputs } 1 \mid \neg B_{E_{uf}} \wedge \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S}\,]\Big) \\
&= \Pr[B_{E_{uf}}] \cdot \Big(\Pr[\mathcal{Z} \text{ outputs } 1 \mid B_{E_{uf}} \wedge \mathcal{Z} \text{ interacts with } \pi_\Sigma \text{ and } \mathcal{A}] \\
&\quad - \Pr[\mathcal{Z} \text{ outputs } 1 \mid B_{E_{uf}} \wedge \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S}]\Big) \\
&\leq \Pr[B_{E_{uf}}] \\
&= \Pr[E_{uf} \text{ success}].
\end{aligned}
$$

Therefore, if there exists $\mathcal{A}$ for any $\mathcal{S}$ such that there is an environment $\mathcal{Z}$ which successfully distinguishes interaction with $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ or π_{Σ_r}, then E_{uf} successfully forges signatures.

(II). Next, we will show the reduction in anonymity. Here, we assume Σ_r satisfies completeness, statefulness and unforgeability. Let $B_{D_{\mathsf{basic}}}$ denote the event that at some point of running π_Σ, all members P_i generates a verification key vk_i for $i = 1$ to n and for the signature of P_{i_0} the environment $\mathcal{Z}$ distinguishes $\sigma_{i_0} \leftarrow \mathbf{RSign}(m, L, sk_{i_0})$ from $\sigma_{i_1} \leftarrow \mathbf{RSign}(m, L, sk_{i_1})$ where $\mathbf{RVer}(m, L, \sigma_{i_0}) = \mathbf{RVer}(m, L, \sigma_{i_1})$ and P_{i_0} and P_{i_1} are uncorrupted. Let $B_{D_{\mathsf{attribution}}}$ denote the event that at some point of running π_Σ, all members P_i generates a verification key vk_i with randomness ω_i for $i = 1$ to n and for the signature of P_{i_0} the environment $\mathcal{Z}$ distinguishes $\sigma_{i_0} \leftarrow \mathbf{RSign}(m, L, sk_{i_0})$ from $\sigma_{i_1} \leftarrow \mathbf{RSign}(m, L, sk_{i_1})$ where $\mathbf{RVer}(m, L, \sigma_{i_0}) = \mathbf{RVer}(m, L, \sigma_{i_1})$ and P_{i_0} is uncorrupted. Let $B_{D_{\mathsf{full\text{-}key}}}$ denote the event that at some point of running π_Σ, all members P_i generates a verification key vk_i with randomness ω_i for $i = 1$ to n and for the signature of P_{i_0} the environment $\mathcal{Z}$ distinguishes $\sigma_{i_0} \leftarrow \mathbf{RSign}(m, L, sk_{i_0})$ from $\sigma_{i_1} \leftarrow \mathbf{RSign}(m, L, sk_{i_1})$ where $\mathbf{RVer}(m, L, \sigma_{i_0}) = \mathbf{RVer}(m, L, \sigma_{i_1})$. Then, we will show that for $\mathcal{Z}$ the difference is only detected at event $B_{D_{anon}}$. This is represented by an equation as

$$\begin{aligned}&\Pr[\mathcal{Z} \text{ outputs } 1 \mid \\ &\quad \neg B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \pi_{\Sigma_r} \text{ and } \mathcal{A}] \\ &= \Pr[\mathcal{Z} \text{ outputs } 1 \mid \\ &\quad \neg B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \\ &\quad \text{and } \mathcal{S}]. \qquad (2)\end{aligned}$$

It is easy to see that Eq. (2) holds. Clearly, there is no difference for $\mathcal{Z}$ between the real model and the ideal model at Signature Verification request since Σ_r satisfies completeness, statefulness and unforgeability. At a Key Generation request, since each parties output only their keys and corrupted parties are identical in the real and ideal model by the formulation of $\mathcal{S}$, for $\mathcal{Z}$ there is no difference. At a Signature Generation request, the correctly generated signature is valid both in the real and ideal model since Σ_r satisfies completeness and statefulness. Also, if the signer is corrupted, the signature can be arbitrarily decided by both $\mathcal{A}$ and $\mathcal{S}$. So, there is no difference for $\mathcal{Z}$ at a Signature Generation request. Therefore, the difference for $\mathcal{Z}$ is only detected at an Attribution request. In the case of $anon = \mathsf{basic}$, in the ideal model, for uncorrupted parties $P_i, P_j \in G$, the probability that for the signature of P_i the case $\sigma_i \leftarrow \mathbf{RS}(m, L)$ is distinguished from $\sigma_j \leftarrow \mathbf{RS}(m, L)$ where $\bigwedge_{\mathbf{RV}_l \in L} \mathbf{RV}_l(m, L, \sigma_i) = \bigwedge_{\mathbf{RV}_l \in L} \mathbf{RV}_l(m, L, \sigma_j)$ holds is $1/2$ because of the formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. In the case of $anon = \mathsf{attribution}$, in the ideal model, for uncorrupted $P_i \in G$ and corrupted $P_j \in G$, the probability that for the signature of P_i the case $\sigma_i \leftarrow \mathbf{RS}(m, L)$ is distinguished from $\sigma_j \leftarrow \mathbf{RS}(m, L)$ where $\bigwedge_{\mathbf{RV}_l \in L} \mathbf{RV}_l(m, L, \sigma_i) = \bigwedge_{\mathbf{RV}_l \in L} \mathbf{RV}_l(m, L, \sigma_j)$ holds is $1/2$ because of the formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. Also, in the case of $anon = \mathsf{full\text{-}key}$, in the ideal model, for corrupted parties $P_i, P_j \in G$, the probability that for the signature of P_i the case $\sigma_i \leftarrow \mathbf{RS}(m, L)$ is distinguished from $\sigma_j \leftarrow \mathbf{RS}(m, L)$ where $\bigwedge_{\mathbf{RV}_l \in L} \mathbf{RV}_l(m, L, \sigma_i) = \bigwedge_{\mathbf{RV}_l \in L} \mathbf{RV}_l(m, L, \sigma_j)$ holds is $1/2$ because of the formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. However, in the real model, $\mathcal{Z}$ may distinguish the signature with a higher probability than $1/2$. Thus, the difference for $\mathcal{Z}$ is only detected at event $B_{D_{anon}}$ by formulation of $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ and we have Eq. (2).

Here, we consider a distinguisher D_{anon}. In the case of $anon = \mathsf{basic}$, L_{all} is given as the input. In the case of $anon = \mathsf{attribution}$, L_{all} and $\{\omega_l\}_{l \neq i}$ are given as the input. Also, in the case of $anon = \mathsf{full\text{-}key}$, L_{all} and all $\{\omega_l\}$ are given as the input. D_{anon} is constructed from $\mathcal{Z}$ (i.e., D_{anon} runs a simulated copy of $\mathcal{Z}$). At a Key Generation request for any parties, instead of running **RGen**, D hands $\mathcal{Z}$ $\mathbf{RV}_i := \mathbf{RVer}(\cdot, (vk_i, \cdot), \cdot)$ where vk_i is a verification key of the input L. At a Signature Generation request with a message m for L, instead of running **RSign**, forwards this request to signing oracle

$\mathcal{SO}$. At a Signature Verification request with (m, L, σ) for a party P_j, D_{anon} runs $\bigwedge_{\mathbf{RV}_j \in L} \mathbf{RV}_j(m, L, \sigma)$. At an Attribution request with (m, L, σ), D_{anon} outputs the signer identity which D_{anon} guesses according to *anon*. Furthermore, at a Signature Generation request for a challenge party P_{i_0} or P_{i_1} with a message m, D_{anon} also forwards this request and "challenge" order to signing oracle $\mathcal{SO}$. Then, $\mathcal{SO}$ returns the signature σ' from $\mathbf{RSign}(m, L, sk_{i_0})$ or $\mathbf{RSign}(m, L, sk_{i_1})$ randomly and D_{anon} hands $\mathcal{Z}$ the challenge signature σ' as the signature of P_{i_0}. If $\mathcal{Z}$ distinguishes the signature σ' and $\mathbf{RVer}(m, L, \sigma') = 1$, then D_{anon} completes the simulation and returns 1. Else, then D_{anon} outputs 0 or 1 randomly.

Therefore, by construction of D_{anon} and Eq. (2), we have

$$
\begin{aligned}
&|\Pr[\mathcal{Z} \text{ outputs } 1 \mid \mathcal{Z} \text{ interacts with } \pi_{\Sigma_r} \text{ and } \mathcal{A}] - \Pr[\mathcal{Z} \text{ outputs } 1 \mid \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S}]| \\
&= (\Pr[\, B_{D_{anon}} \,] \cdot \Pr[\, \mathcal{Z} \text{ outputs } 1 \mid B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \pi_{\Sigma} \text{ and } \mathcal{A} \,] \\
&\quad + \Pr[\, \neg B_{D_{anon}} \,] \cdot \Pr[\, \mathcal{Z} \text{ outputs } 1 \mid \neg B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \pi_{\Sigma} \text{ and } \mathcal{A} \,]) \\
&\quad - (\Pr[\, B_{D_{anon}} \,] \cdot \Pr[\, \mathcal{Z} \text{ outputs } 1 \mid B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S} \,] \\
&\quad + \Pr[\, \neg B_{D_{anon}} \,] \cdot \Pr[\, \mathcal{Z} \text{ outputs } 1 \mid \neg B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S} \,]) \\
&= \Pr[B_{D_{anon}}] \cdot (\Pr[\mathcal{Z} \text{ outputs } 1 \mid B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \pi_{\Sigma} \text{ and } \mathcal{A}] \\
&\quad - \Pr[\mathcal{Z} \text{ outputs } 1 \mid B_{D_{anon}} \wedge \mathcal{Z} \text{ interacts with } \mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)} \text{ and } \mathcal{S}]) \\
&\leq \Pr[B_{D_{anon}}] \\
&= \Pr[D_{anon} \text{ success}].
\end{aligned}
$$

Therefore, if there exists $\mathcal{A}$ for any $\mathcal{S}$ such that there is an environment $\mathcal{Z}$ which successfully distinguishes interaction with $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$ or π_{Σ_r}, then D_{anon} successfully breaks anonymity.

From **(I)** and **(II)**, "if" direction is proven. □

4. Universally Composable Construction without Random Oracles

In this section, we show concrete constructions of ring signature which securely realize $\mathcal{F}_{\mathrm{rSIG}}^{(uf,anon)}$. We adapt constructions of BKM schemes [2]. The security of BKM schemes are proved without relying on random oracle assumption.

4.1 Security of BKM Schemes

The basic one of BKM schemes is based on general assumptions, i.e., a semantically-secure public-key encryption scheme, a (standard) existentially unforgeable signature scheme and a zap [7]. From now on, we call this scheme *BKM1 scheme*. The zap is a two-round, public coin witness-indistinguishable protocol for any language in NP and needs no preshared common random string (CRS). Instead of relying on preshared CRS, the zap guarantees both soundness and witness-indistinguishability by using the idea of *reverse randomization*. Reverse randomization means the verifier first sends randomness and the prover returns a proof which is generated from randomness of both the verifier and the prover. Though the zap [7] guarantees soundness and witness-indistinguishability by reverse randomization, in signature schemes any verifier cannot choose randomness since the verifier isn't fixed in advance when the signer generates a signature. Therefore, in BKM1 scheme the signer uses randomness which is determined by a part of the verification key of a member of the ring instead of randomness of the verifier. BKM1 scheme uses the zap as a signature and adopts both perfect soundness and computational witness-indistinguishability of the zap for ensuring unforgeability and anonymity respectively. Roughly speaking, perfect soundness means that the verifier certainly rejects the proof for the statement which isn't in the language, and computational witness-indistinguishability means that a polynomial time adversary cannot tell which of two possible witnesses has been used for generating the proof with non-negligible probability. Under these assumptions, it was proved that BKM1 scheme satisfies eUF-ACMA&ACVKA and anonymity against attribution attacks.

Moreover, it is shown that BKM1 scheme is able to modify in order to satisfy based on anonymity against full key exposure. We call this scheme *BKM2 scheme*. In this case,

the additional assumption which is called a *simulatable* public-key encryption scheme [5] is needed. Roughly speaking, a public-key encryption scheme is simulatable if, in addition to the normal key generation procedure, there is an algorithm to generate a public key without getting to know the corresponding secret key (oblivious public-key generator). Also, it must be possible to sample efficiently a random ciphertext without getting to know the corresponding plaintext (oblivious ciphertext generator). It is known the ElGamal public-key encryption scheme is the simulatable under the Decisional Diffie-Hellman (DDH) assumption. For formal definitions of zaps and simulatable public-key system, please refer to papers [5),7] respectively. There, it was proved that a modified BKM2 scheme satisfies eUF-ACMA&ACVKA and anonymity against full key exposure.

However, it is an open problem whether BKM1 scheme and BKM2 scheme are universally composable or not. It seems that these schemes are able to be proved UC-security.

4.2 UC Security of BKM Schemes

In this section, we will show UC security of BKM schemes. We are able to prove that the protocol of BKM1 scheme and the modified protocol of BKM2 scheme satisfy UC-security by using Theorem 3.1. Specifically, the protocol of BKM1 scheme securely realizes $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{eUF},\mathsf{attribution})}$ since BKM1 scheme satisfies eUF-ACMA&ACVKA and anonymity against attribution attacks, and the protocol of BKM2 scheme securely realizes $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{eUF},\mathsf{full\text{-}key})}$ since BKM2 scheme satisfies eUF-ACMA&ACVKA and anonymity against full key exposure. Therefore, BKM1 scheme and BKM2 scheme are concrete constructions of our functionality.

4.3 New Modified BKM Schemes

Though BKM1 scheme and BKM2 scheme only satisfy eUF-ACMA&ACVKA regarding unforgeability, we will give a new modified BKM scheme which has stronger unforgeability (i.e., sUF-ACMA&ACVKA) than the original BKM schemes and prove UC-security of it similarly. We call this scheme *BKM3 scheme*. We show the description of BKM3 scheme.

Let members of the ring be $G = (P_1, \ldots, P_n)$, and let $(\mathbf{OGen}, \mathbf{Enc}, \mathbf{Dec})$ be a simulatable public-key encryption scheme where **OGen** is an oblivious public-key generator, and let $(\mathbf{SGen}, \mathbf{Sign}, \mathbf{Ver})$ be a (standard) signature scheme. We denote $R_E = \{pk_{E_1}, \ldots, pk_{E_n}\}$ a set of public keys and $C \leftarrow \mathbf{Enc}(\alpha, R_E; \omega)$ a set of ciphertexts such that

$$C = \Big(\mathbf{Enc}\left(\omega_1, pk_{E_1}\right), \ldots, \mathbf{Enc}(\omega_{n-1}, pk_{E_{n-1}}), \mathbf{Enc}(\alpha \bigoplus_{j=1}^{n-1} \omega_j, pk_{E_n}) \Big)$$

where α is a plaintext and $\omega = (\omega_1, \ldots, \omega_{n-1}) \in \left(\{0,1\}^{|m|}\right)^{n-1}$ are randomnesses. Let $\mathcal{L}$ denote the NP language such that

$$\Big\{ (vk_S, m, R_E, C) : \exists \sigma, \omega \ \text{ s.t. } \ C = \mathbf{Enc}(\sigma, R_E; \omega) \bigwedge \mathbf{Ver}(m, vk_S, \sigma) = 1 \Big\}$$

where m is a message, vk_S is a verification key and σ is a standard signature. Also, let $(\ell, \mathcal{P}, \mathcal{V})$ be a zap for the language $\mathcal{L}$ where $\ell(k)$ is the length of firstly chosen randomness, $\mathcal{P}$ is a prover and $\mathcal{V}$ is a verifier. **Figure 4** shows the protocol of BKM3 scheme.

Here, we prove the protocol of BKM3 scheme securely realizes $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$, i.e., sUF-ACMA&ACVKA and anonymity against full-key exposure.

Theorem 4.1 If the simulatable public-key encryption scheme $(\mathbf{OGen}, \mathbf{Enc}, \mathbf{Dec})$ is semantically-secure, the standard signature scheme $(\mathbf{SGen}, \mathbf{Sign}, \mathbf{Ver})$ is strong existentially unforgeable against adaptively chosen message attacks, and $(\ell, \mathcal{P}, \mathcal{V})$ is a zap for the language $\mathcal{L}' = \{(x_1, \ldots, x_n) : \exists i \text{ s.t. } x_i \in \mathcal{L}\}$, then the protocol of BKM3 scheme securely realizes $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$.

[Proof]

Let $\mathcal{A}$ be an adversary in the real-life model. The proof outline is that for any $\mathcal{A}$ we can construct a simulator $\mathcal{S}$ such that any environment $\mathcal{Z}$ cannot successfully distinguish the interaction with $\mathcal{A}$ and parties running Π in the real-life model from the interaction with $\mathcal{S}$ and parties for $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$ in the ideal model. Simulator $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$ and simulates the interface for $\mathcal{A}$. Then, $\mathcal{S}$ forwards all instructions from $\mathcal{Z}$ to $\mathcal{A}$ and back, and whenever $\mathcal{A}$ corrupts a party P_i, then $\mathcal{S}$ corrupts the corresponding party P_i. The concrete construction of $\mathcal{S}$ is as follows:

Simulating $\mathcal{Z}$ When $\mathcal{Z}$ instructs some input to $\mathcal{S}$, $\mathcal{S}$ forwards it to $\mathcal{A}$. And, $\mathcal{S}$ returns any output of $\mathcal{A}$ to $\mathcal{Z}$ as the output of $\mathcal{S}$.

Simulating corruption When $\mathcal{A}$ corrupts

protocol of BKM3 scheme

Key Generation $\left(\mathbf{RGen}(1^k)\right)$ **:** When P_i is activated with input ("KeyGen", sid), P_i does the following.

Step 1. Generate signing key pair $(sk_{S_i}, vk_{S_i}) \leftarrow \mathbf{SGen}(1^k)$ and public key $pk_{E_i} \leftarrow \mathbf{OGen}(1^k)$. (In BKM1 scheme and BKM2 scheme, $(\mathbf{SGen}, \mathbf{Sign}, \mathbf{Ver})$ should be existentially unforgeable against adaptively chosen message attacks. However, in our BKM3 scheme, $(\mathbf{SGen}, \mathbf{Sign}, \mathbf{Ver})$ should be *strong* existentially unforgeable against adaptively chosen message attacks.)

Step 2. Choose randomness of the zap $r_i \leftarrow \{0,1\}^{\ell(k)}$.

Step 3. Output the verification key $vk_i = (vk_{S_i}, pk_{E_i}, r_i)$ and the signing key $sk_i = sk_{S_i}$. (In BKM1 scheme, since **OGen** is a key generation algorithm of a standard public-key encryption, a secret key sk_{E_i} is generated with pk_{E_i} and should be erased in Step 3. However, in our BKM3 scheme, since **OGen** is an oblivious public-key generator, we don't need to do such an operation.)

Signature Generation $(\mathbf{RSign}(m, vk_1, \ldots, vk_n, sk_i))$ **:** When P_{i^*} is activated with input ("Sign", sid, m, G) where $G = (P_1, \ldots, P_n)$, P_{i^*} does following.

Step 1. Compute the signature $\sigma_{i^*} \leftarrow \mathbf{Sign}(\bar{m}, sk_{S_{i^*}})$ where $\bar{m} := m||vk_1|| \cdots ||vk_n$.

Step 2. Choose random coins $\omega^{(1)}$, ..., $\omega^{(n)}$ and compute $C_{i^*} = \mathbf{Enc}(\sigma_{i^*}, R_E; \omega^{(i^*)})$ for i^* and $C_i = \mathbf{Enc}(0^{|\sigma_{i^*}|}, R_E; \omega^{(i)})$ for $i \in \{1, \ldots, n\} \setminus \{i^*\}$ where $R_E = \{pk_{E_1}, \ldots, pk_{E_n}\}$.

Step 3. For $i \in \{1, \ldots, n\}$, let x_i denote the statement : "$(vk_{S_i}, \bar{m}, R_E, C_i) \in \mathcal{L}$", and let $x := \bigvee_{i=1}^{n} x_i$.

Step 4. Compute the proof $\pi \leftarrow \mathcal{P}\left(x, (\sigma_{i^*}, \omega^{(i^*)}), r_1\right)$ and output the signature $\sigma = (C_1, \ldots, C_n, \pi, r_1)$.

Signature Verification $(\mathbf{RVer}(m, vk_1, \ldots, vk_n, \sigma))$ **:** When V is activated with input ("Verify", sid, m, σ, G), V outputs $\mathcal{V}(x, \pi, r_1)$.

Fig. 4 BKM3 scheme.

some party P_j, $\mathcal{S}$ also corrupts P_j in the ideal model. If P_j hasn't received a Key Generation request yet, $\mathcal{S}$ reveals no information to $\mathcal{A}$. If P_j already has received a Key Generation request, all the internal state of P_j to $\mathcal{A}$.

Simulating attribution When $\mathcal{A}$ examines attribution attacks with (m, L, σ), $\mathcal{S}$ forwards (m, L, σ) to $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$ and returns the output of $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$ to $\mathcal{A}$.

Simulating uncorrupted parties In this case, first, $\mathcal{S}$ generates $(sk'_S, vk'_S) \leftarrow \mathbf{SGen}(1^k)$ and $pk'_E \leftarrow \mathbf{OGen}(1^k)$, and chooses $r' \leftarrow \{0,1\}^{\ell(k)}$. When $\mathcal{S}$ receives a request for generating algorithms from $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$, if it is first "KeyGen" input, then $\mathcal{S}$ generates $(sk_S, vk_S) \leftarrow \mathbf{SGen}(1^k)$ and $pk_E \leftarrow \mathbf{OGen}(1^k)$, chooses $r \leftarrow \{0,1\}^{\ell(k)}$, and returns $\mathbf{RS} := \mathbf{RSign}(\cdot, \{(vk'_S, pk'_E, r'), \cdot\}, sk'_S)$ and $\mathbf{RV} := \mathbf{RVer}(\cdot, \{(vk'_S, pk'_E, r'), (vk_S, pk_E, r), \cdot\}, \cdot)$ to $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$. Otherwise, then $\mathcal{S}$ generates $(sk_S, vk_S) \leftarrow \mathbf{SGen}(1^k)$ and $pk_E \leftarrow \mathbf{OGen}(1^k)$, chooses $r \leftarrow \{0,1\}^{\ell(k)}$, and returns $\mathbf{RV} := \mathbf{RVer}(\cdot, \{(vk'_S, pk'_E, r'), (vk_S, pk_E, r), \cdot\}, \cdot)$ to $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$. Furthermore, $\mathcal{S}$ is easily able to simulate $\mathcal{A}$ since there is no influence of $\mathcal{A}$ for uncorrupted parties.

Simulating corrupted parties When $\mathcal{A}$ makes the corrupted party P_j generate a verification key $vk_j = (vk_{S_j}, pk_{E_j}, r_j)$ and a signing key $sk_j = (sk_{S_j})$, $\mathcal{S}$ computes $\mathbf{RV}_j := \mathbf{RVer}(\cdot, \{(vk_{S_j}, pk_{E_j}, r_j), \cdot\}, \cdot)$ and $\mathbf{RS} := \mathbf{RSign}(\cdot, \{(vk_{S_j}, pk_{E_j}, r_j), \cdot\}, sk_{S_j})$. And, when $\mathcal{S}$ receives "KeyGen" for P_j which is forwarded by $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$, $\mathcal{S}$ answers $\mathbf{RV}$ and $\mathbf{RS}$ to $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$. When $\mathcal{A}$ makes the corrupted party P_j generate a signature $\sigma \leftarrow \mathbf{RSign}(m, L, sk_j)$ for a message m and a verification key list L, $\mathcal{S}$ does nothing directly. However, $\mathcal{S}$ can simulate σ by the answered signing algorithm $\mathbf{RS}$ to $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$ at "KeyGen" query. When $\mathcal{A}$ makes the corrupted party P_j output a verification result (a bit) f for (m, L, σ), $\mathcal{S}$ can also simulate f by the answered verification algorithm vk_j and verification algorithms of other corrupted parties to $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$ at "KeyGen" query.

It is easy to verify the validity of the simulation by using assumptions, i.e., a sUF-ACMA&ACVKA signature scheme, a semantically secure simulatable public-key encryption scheme and a zap for the language $\mathcal{L}'$. The validity of the simulation means, for any $\mathcal{A}$ and environment $\mathcal{Z}$, the output of $\mathcal{Z}$ is distributed identically both in the real model, i.e., an interaction with $\mathcal{A}$ and parties running the protocol Π, and in the ideal model, i.e., an interaction with $\mathcal{S}$ and functionality $\mathcal{F}_{\mathrm{rSIG}}^{(\mathsf{sUF},\mathsf{full\text{-}key})}$. □

References

1) Abe, M., Ohkubo, M. and Suzuki, K.: 1-out-of-n Signatures from a Variety of Keys, *Advances in Cryptology-ASIACRYPT2002*, pp.415–432 (2002).
2) Bender, A., Katz, J. and Morselli, R.: Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles, *TCC 2006*, pp.60–79 (2006).
3) Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols, *FOCS 2001*, pp.136–145 (2001).
4) Chow, S.S.M., Liu, J.K., Wei, V.K. and Yuen, T.H.: Ring Signatures without Random Oracles, *ASIACCS 2006* (2006).
5) Damgård, I. and Nielsen, J.B.: Improved Non-committing Encryption Schemes Based on a General Complexity Assumption, *Advances in Cryptology-CRYPTO 2000*, pp.432–450 (2000).
6) Dodis, Y., Kiayias, A., Nicolosi, A. and Shoup, V.: Anonymous Identification in Ad Hoc Groups, *Advances in Cryptology — EUROCRYPT2004*, pp.609–626 (2004).
7) Dwork, C. and Naor, M.: Zaps and Their Applications, *FOCS 2000*, pp.283–293 (2000).
8) Goldwasser, S., Micali, S. and Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks, *SIAM Journal on Computing*, pp.281–308 (1988).
9) Hanatani, Y., Yoneyama, K., Santoso, B. and Ohta, K.: A Note on Universally Composable 1-out-of-n Signature (in Japanese), *SCIS2005*, pp.643–648 (2005).
10) Rivest, R.L., Shamir, A. and Tauman, Y.: How to Leak a Secret, *Advances in Cryptology-ASIACRYPT2001*, pp.552–565 (2001).
11) Yoneyama, K., Hanatani, Y., Santoso, B. and Ohta, K.: Universally Composable Ring Signature, *IWSEC2006* (2006).

(Received November 24, 2006)
(Accepted June 5, 2007)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.3, pp.571–584.)

Kazuki Yoneyama received the B.E. and M.E. degrees from the University of Electro-Communications, Tokyo, Japan, in 2004 and 2006, respectively. He has been currently a doctor course student at the Graduate School of Electro-Communications since 2006. He is presently engaged in research on cryptography. He is a member of IEICE.

Kazuo Ohta received his B.S., M.S., and Dr.S. degrees from Waseda University, Tokyo, Japan, in 1977, 1979, and 1990, respectively. He has been a professor at the University of Electro-Communications since 2001. He was a researcher at NTT Laboratories between 1979 and 2001. He is presently engaged in research on information security. Dr. Ohta is a member of the International Association for Cryptologic Research, IEICE and IEEE.