On Randomness Exposure Resilience of Group Signatures

Tomoyoshi ONO†∗, Nonmember and Kazuki YONEYAMA††, Member

SUMMARY Group signature (GS) schemes guarantee anonymity of the actual signer among group members. Previous GS schemes assume that randomness in signing is never exposed. However, in the real world, full randomness exposure can be caused by implementation problems (e.g., using a bad random number generator). In this paper, we study (im)possibility of achieving anonymity against full randomness exposure. First, we formulate a new security model for GS schemes capturing full randomness exposure. Next, we clarify that it is impossible to achieve full-anonymity against full randomness exposure without any secure component (e.g., a tamper-proof module or a trusted outside storage). Finally, we show a possibility result that selfless-anonymity can be achieved against full randomness exposure. While selfless-anonymity is weaker than full-anonymity, it is strong enough in practice. Our transformation is quite simple; and thus, previous GS schemes used in real-world systems can be easily replaced by a slight modification to strengthen the security.

key words: group signature, full-anonymity, selfless-anonymity, randomness exposure

1. Introduction

In our daily life, we use various web-services, and these services often require authentication to verify if a user is qualified. Anonymity is an important security notion in authentication systems when sensitive information is handled. For example, let us consider an electronic first-price sealed-bid auction. All bidders submit sealed bids of their own valuations, and the highest bidder wins and pays the price they submitted. In this case, it is required that no bidder links a valuation with any other participant. Thus, we need to verify that a valuation is certainly submitted by one of bidders (i.e., authenticity), but it must be hidden who submits the valuation (i.e., anonymity). Group signature (GS) [1] is a special kind of digital signatures to provide both authenticity and anonymity. In GS, a group manager issues signing keys of a group of signers, and a signer generates a group signature with his/her signing key. Any entity except the group manager (or an opener) cannot distinguish the signer from other group members. Hence, a verifier can verify the validity of the group signature without knowing the actual signer. If we apply GS to the auction, bidders generate group signatures of their valuations, and only the auctioneer can verify signatures and know who is the highest bidder though bidders and third parties cannot know it even with information of signatures and their signing keys.

The first unified formal security model of GS is defined by Bellare et al. [2]. Though this model is for the static group case (i.e., a group of signers is fixed before signing), it is extended to the dynamic group case [3] (i.e., a new signer can join the group after signing). In these security models, anonymity is defined as full-anonymity (FAnon). Intuitively, FAnon guarantees that no adversary can distinguish two signatures generated by distinct signers even if all signing keys of the group are given. Thus, FAnon means that even the actual signer cannot decide if a signature is generated by his/her own. On the other hand, there is a weaker anonymity definition, called selfless-anonymity (SLAnon) [4]. SLAnon is the same as the FAnon except signing keys of two signers for the challenge signature are not given to the adversary. Thus, SLAnon means that the actual signer can decide if a signature is generated by his/her own, but other signers (and third parties) cannot. Though SLAnon is weaker than FAnon, it seems strong enough for most of applications in practice.

1.1 Motivating Problem

Bellare et al. [2], [3] also propose generic constructions of static or dynamic GS. In the past decade, various GS schemes are introduced based on these generic constructions, e.g., the static group setting [5]–[7] and the dynamic group setting [8]–[13]. Some GS schemes (or its variants supporting user revocation) satisfy SLAnon [14]–[18]. These GS schemes are provably secure in formal security models [2]–[4]. However, all of these security models suppose that randomness in generating signatures is never exposed. In the real world, randomness will be fully exposed in various situations. For example, if a bad random number generator (RNG) is implemented in a system using GS, then outputs of the RNG are easily predicted by adversaries; and thus, randomness is fully exposed. (e.g., the incident of Debian’s OpenSSL package [19]) Also, even if the RNG is not weak, inappropriate implementations like reuse of nonce or randomness cause full randomness exposure. If there is a problem on the randomness generation in a system, randomness in the system setup and key generation procedure can be protected because it is generated once in the setup timing, usually, outside of the system under the off-line manner. Conversely, exposure of randomness in the signature generation procedure is not prevented because it must be gener-
ated in the system under the on-line manner. Therefore, it is desirable that GS schemes are still secure if randomness in the signature generation is fully exposed. However, since previous security models do not capture randomness exposure, it is unclear if previous GS schemes are secure against randomness exposure. Though leakage-resilient cryptography (e.g., leakage-resilient signature [20]) or hedged cryptography (e.g., hedged encryption [21]) seems a solution for this problem, these approaches are not applicable if randomness is fully exposed because these approaches assume that some uncertainty of randomness is left for adversaries.

1.2 Our Contribution

This paper clarifies the influence of full randomness exposure for GS schemes, and how to make GS schemes secure against randomness exposure.

First, we introduce a new security model of GS, which captures randomness exposure. Especially, randomness exposure affects anonymity. We call FAnon and SLAnon against full randomness exposure FAnon-RE and SLAnon-RE, respectively. In our model, the adversary can access an additional oracle (Rev) to provide randomness in generating the challenge signature as well as oracles in the previous security model [3]. In this paper, though we consider the dynamic group case, our modelling method is trivially applied to the static group case.

Next, we show that it is impossible to achieve FAnon-RE without any secure component. The secure component means that the adversary cannot obtain any information from it even with all oracles, and can be realized by a tamper-proof module or a trusted outside storage. In other words, if the adversary can obtain all secret information of the actual signer, the adversary can distinguish the actual signer from others by using randomness for any GS schemes. Specifically, we show a general attack procedure.

Finally, we propose a generic transformations of GS to satisfy SLAnon-RE. Our transformation is quite simple: applying a pseudo-random function (PRF) to generate new randomness with a secret salt. The salt is stored in a part of the signing key. Similar techniques are used to resist secret exposure attacks in authenticated key exchange schemes like [22]. Thanks to its simplicity, we can replace GS schemes implemented in systems by a slight modification in order to provide randomness exposure. We prove that a transformed GS scheme satisfies SLAnon-RE if the original GS scheme satisfies SLAnon.

Furthermore, we show that FAnon-RE is also achievable with a physical assumption. Specifically, by relying on a secure component, we can avoid our impossibility and construct a generic transformation of FAnon-RE GS schemes. In this case, the salt of the PRF is stored in the secure component. Though such an assumption is not natural, it may be useful in applications that some secure component (e.g., a smart card, a private cloud storage) can be used.

2. Security Model

In this section, we introduce a new security model for dynamic GS, which captures security against full randomness exposure in the signature generation. The main difference from the previous model is to add a new oracle in order to expose randomness. We note that our security model can be easily modified to the static group setting.

Throughout this paper we use the following notations. If ALG is an algorithm, then by \( y \leftarrow ALG(x; r) \) we denote that \( y \) is output by \( ALG \) on input \( x \) and randomness \( r \) (if \( ALG \) is deterministic, \( r \) is empty). If \( \mathcal{A} \) is an adversary, then by \( y \leftarrow \mathcal{A}(x : \text{Oracle}) \) we denote that \( y \) is output by \( \mathcal{A} \) on input \( x \) and \( \mathcal{A} \) can pose queries to Oracle.

2.1 Syntax of Dynamic Group Signatures

A dynamic group signature scheme GS consists of eight algorithms (GKg, UKg, GSig, GVf, Open, Judge, Join, Iss).

- **GKg** is a group key generation algorithm which takes as input a security parameter \( \lambda \) and outputs a group public key \( gpk \), an issuer key \( ik \) and an opener key \( ok \). The trusted third party runs the GKg, and the issuer key \( ik \) is provided to the issuer, and the opener key \( ok \) is provided to the opener. The group public key \( gpk \) is published.

- **UKg** is a user key generation algorithm for a user \( i \), which takes as input a security parameter \( \lambda \) and outputs a user public key \( upk[i] \) and a user secret key \( usk[i] \).

- **Join** and **Iss** are interactive algorithms between a user \( i \) and the issuer in order to add the user \( i \) as a group member after the successful interaction. Join and Iss take as input a current state and an incoming message \( M_{in} \), and output an updated state, and an outgoing message \( M_{out} \) and a decision \( \text{dec} \in \{\text{accept}, \text{reject}, \text{cont}\} \) where \text{accept} means that the interaction is accepted, \text{reject} means that the interaction is accepted and \text{cont} means that the interaction continues. We suppose that the user \( i \) sends the first message to the issuer, and initial states for Join and Iss contain \( (gpk, upk[i], usk[i]) \) and \( (gpk, ik, i, upk[i]) \), respectively. If the issuer accepts, it makes an entry \( \text{reg}[i] \) for the user \( i \), in a registration table \( \text{reg} \), and fills this entry with a new membership certificate, which is the final state output by Iss. If the user \( i \) accepts, it stores the final state output by Join as its group signing key \( gsk[i] \). We assume that the communication takes place over a private and authenticated channel.

- **GSig** is a signature generation algorithm which takes as input the group public key \( gpk \), a message \( m \) and the group signing key \( gsk[i] \), and outputs a signature \( \sigma \) on the message \( m \).

- **GVf** is a signature verification algorithm which takes as input the group public key \( gpk \), a message \( m \) and a signature \( \sigma \), and outputs 1 if \( \sigma \) is valid for the message \( m \), otherwise outputs 0.
### 2.2 Security Definition

#### 2.2.1 Oracles

- **Open** is an opening algorithm which takes as input the group public key $gpk$, the opener key $ok$, a registration table $reg$, a message $m$ and a signature $\sigma$, and outputs $i$ and $\tau$ where $\tau$ is a proof that $\sigma$ is generated by the user $i$.
- **Judge** is a judging algorithm which takes as input the group public key $gpk$, a user $i$, a user public key $upk[i]$, a message $m$, a signature $\sigma$ and a proof $\tau$, and outputs $1$ if $\tau$ generated by **Open** is valid, otherwise outputs $0$.

#### 2.2.2 Correctness

Correctness means that a signature generated by an honest group member should be valid. $\text{Open}$ should correctly identify the signer from the signature, and the proof generated by $\text{Open}$ should be accepted by $\text{Judge}$.

**Definition 2.1 (Correctness):** We define the experiment $\text{Exp}_{\text{corr}, \mathcal{A}}(\kappa)$ for a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ and a security parameter $\kappa$ as follows:

$$\text{Exp}_{\text{corr}, \mathcal{A}}(\kappa)$$

- $\text{CU} \leftarrow \emptyset$; $\text{HU} \leftarrow \emptyset$
Exp_{\mathcal{G}}(k) for a PPT adversary $\mathcal{A}$ and a security parameter $\kappa$ as follows:

**Experiment Exp_{\mathcal{G},\mathcal{A}}(\kappa):**

\[
\begin{align*}
\text{CU} &\leftarrow \emptyset; \text{HU} \leftarrow \emptyset; \sigma_{\text{set}} \leftarrow \emptyset \\
(gpk, ik, ok) &\leftarrow \text{GKg}(1^\kappa) \\
b' &\leftarrow \mathcal{A}(\text{gpk}, ik : \text{Op}(-), \text{SndToU}(-), \text{WReg}(-), \text{USK}(-), \text{CrptU}(-), \text{Sig}(-), \text{Ch}_{\text{U}}(-), \text{Rev}(-)) \\
\text{Return } b'
\end{align*}
\]

The advantage of $\mathcal{A}$ is defined as: \(\text{Adv}_{\mathcal{G},\mathcal{A}}^{\text{corr}}(\kappa) = \Pr[\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{corr}}(\kappa) = 1] - \Pr[\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{corr}}(\kappa) = 0]\). We say that the dynamic group signature scheme $\mathcal{G}$ is secure against randomness exposure if $\text{Adv}_{\mathcal{G},\mathcal{A}}^{\text{corr}}(\kappa)$ is negligible for any PPT adversary $\mathcal{A}$.

**Remark 2.1:** When $\text{Rev}$ oracle is removed from $\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{corr}}(\kappa)$, the definition means FAnon or SLAnon in [3], [4] (i.e., without considering randomness exposure).

2.2.4 Traceability

Traceability means that the corrupted opener and users should be unable to generate a valid signature such that the actual signer cannot be opened, and should be unable to generate a valid signature such that the proof generated by the honest opener is rejected by $\text{Judge}$.

**Definition 2.3** (Traceability): We define the experiment $\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{trace}}(\kappa)$ for a PPT adversary $\mathcal{A}$ and a security parameter $\kappa$ as follows:

**Experiment Exp_{\mathcal{G},\mathcal{A}}^{\text{trace}}(\kappa):**

\[
\begin{align*}
\text{CU} &\leftarrow \emptyset; \text{HU} \leftarrow \emptyset; \sigma_{\text{set}} \leftarrow \emptyset \\
(gpk, ik, ok) &\leftarrow \text{GKg}(1^\kappa) \\
(m, \pi, \tau) &\leftarrow \mathcal{A}(\text{gpk}, ok : \text{SndToU}(-), \text{AddU}(-), \text{RReg}(-), \text{USK}(-), \text{CrptU}(-), \text{Sig}(-), \text{Rev}(-)) \\
\text{If } 0 &\leftarrow \text{GVI}(\text{gpk}, m, \sigma) \text{ then return } 0 \\
(i, \pi) &\leftarrow \text{Open}(\text{gpk}, \text{ok}, m, \sigma) \\
\text{If } i = \pi \text{ or } 0 &\leftarrow \text{Judge}(\text{gpk}, i, \text{upk}[i], m, \sigma, \tau) \text{ then return } 1 \text{ else return } 0
\end{align*}
\]

The advantage of $\mathcal{A}$ is defined as: \(\text{Adv}_{\mathcal{G},\mathcal{A}}^{\text{trace}}(\kappa) = \Pr[\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{trace}}(\kappa) = 1] - \Pr[\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{trace}}(\kappa) = 0]\). We say that the dynamic group signature scheme $\mathcal{G}$ is traceable against randomness exposure if $\text{Adv}_{\mathcal{G},\mathcal{A}}^{\text{trace}}(\kappa)$ is negligible for any PPT adversary $\mathcal{A}$.

2.2.5 Non-Frameability

Non-frameability means that the corrupted issuer and opener should be unable to generate a valid signature and a valid proof accepted by $\text{Judge}$ for an honest user.

**Definition 2.4** (Non-frameability): We define the experiment $\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{nf}}(\kappa)$ for a PPT adversary $\mathcal{A}$ and a security parameter $\kappa$ as follows:

**Definition 2.4** (Non-frameability): We define the experiment $\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{nf}}(\kappa)$ for a PPT adversary $\mathcal{A}$ and a security parameter $\kappa$ as follows:

\[
\begin{align*}
\text{CU} &\leftarrow \emptyset; \text{HU} \leftarrow \emptyset; \sigma_{\text{set}} \leftarrow \emptyset \\
(gpk, ik, ok) &\leftarrow \text{GKg}(1^\kappa) \\
(m, \pi, \tau) &\leftarrow \mathcal{A}(\text{gpk}, ok : \text{SndToU}(-), \text{WReg}(-), \text{USK}(-), \text{CrptU}(-), \text{Sig}(-), \text{Rev}(-)) \\
\text{If } 0 &\leftarrow \text{GVI}(\text{gpk}, m, \sigma) \text{ then return } 0 \\
\text{If } i \in \text{HU}, \text{gsk}[i] \neq \perp \text{ and } 1 &\leftarrow \text{Judge}(\text{gpk}, i, \text{upk}[i], m, \sigma, \tau) \text{ then return } 1 \text{ else return } 0
\end{align*}
\]

The advantage of $\mathcal{A}$ is defined as: \(\text{Adv}_{\mathcal{G},\mathcal{A}}^{\text{nf}}(\kappa) = \Pr[\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{nf}}(\kappa) = 1] - \Pr[\text{Exp}_{\mathcal{G},\mathcal{A}}^{\text{nf}}(\kappa) = 0]\). We say that the dynamic group signature scheme $\mathcal{G}$ is non-frameable against randomness exposure if $\mathcal{A}$ poses neither USK($i_0$) nor GSig($i_0$, $m$), and $\text{Adv}_{\mathcal{G},\mathcal{A}}^{\text{nf}}(\kappa)$ is negligible for any PPT adversary $\mathcal{A}$.

3. Impossibility of Full-Anonymity against Full Randomness Exposure

In this section, we show that FAnon-RE is not achievable if users do not have any secure component (SC). The notion of SC means that the adversary cannot obtain the content of the SC even with any oracle queries like USK. In the real world, the SC is realized by a local tamper-proof module or a trusted outside storage. In other words, if a user does not have any SC, then the adversary can obtain all secret information of the user by posing USK because secret information must be stored in $\text{usk}[i]$ or $\text{gsk}[i]$ in the security model.
Definition 3.1: We say that a component is the SC if any information stored in the component is not revealed by oracle queries AddU, CrptU, SndTol, SndToU, USK, RReg, WReg, Sig, Op, Ch and Rev.

Theorem 3.1: We assume that users do not have any SC. Then, there exists an adversary who breaks FAnon-RE.

Proof. We construct a generic adversary \( \mathcal{A} \) breaking FAnon-RE as follows:

1. Receive \( gpk \) and \( ik \).
2. Choose the challenge users \( i_0 \) and \( i_1 \), and a message \( m \).
3. Pose USK(\( i_0 \)) and USK(\( i_1 \)), and obtain \((gsk[i_0], usk[i_0])\) and \((gsk[i_1], usk[i_1])\).
4. Pose Ch(\( i_0 \), \( i_1 \), \( m \)), and obtain \( \sigma \).
5. Pose Rev(\( m \), \( \sigma \)), and obtain \( r \).
6. Compute \( \sigma_0 \leftarrow GSig(gpk, gsk[i_0], m; r) \) and \( \sigma_1 \leftarrow GSig(gpk, gsk[i_1], m; r) \).
7. If \( \sigma_0 = \sigma_1 \), then output 0, else if \( \sigma_1 = \sigma \) output 1.

Since user \( i_0 \) and user \( i_1 \) do not have any SC, \( \mathcal{A} \) has the same power as \( i_0 \) and \( i_1 \) (i.e., \( gsk[i_0], gsk[i_1] \) and randomness \( r \)). \( \sigma \) is deterministically generated with \( GSig \) on input \((gpk, gsk[i_0], m; r)\). Thus, \( \sigma \) must be identical to either \( \sigma_0 \) or \( \sigma_1 \).

From Theorem 3.1, we cannot construct a GS scheme satisfying FAnon-RE in the standard setting (i.e., without any SC). On the other hand, the given attack is not applicable to the case of SLAnon-RE because \( \mathcal{A} \) cannot proceed Step 3 due to the restriction of the definition of SLAnon-RE.

4. Generic Transformation for Selfless-Anonymity against Full Randomness Exposure

In this section, we introduce a generic transformation of GS to guarantee anonymity against full randomness exposure. As we prove in Sect. 3, FAnon-RE is not achievable. However, we can achieve SLAnon-RE without any SC. Though anonymity is weaker than FAnon-RE, it is strong enough in practice to use in most of applications because only the actual signer may distinguish the his/her signature from other signatures.

4.1 Building Block

Let \( \kappa \) be a security parameter, which is chosen according to required key length. Let \( F = \{F_k : Dom_\kappa \times Salt_\kappa \to Rng_\kappa \} \) be a function family with a family of domains \( \{Dom_\kappa\}_\kappa \), a family of salt spaces \( \{Salt_\kappa\}_\kappa \), and a family of ranges \( \{Rng_\kappa\}_\kappa \).

Definition 4.1 (Pseudo-Random Function): We say that function family \( F = \{F_k\} \) is the pseudo-random function (PRF) family, if for any PPT distinguisher \( D \), \( |Pr[1 \leftarrow D^{F_k}(\cdot)] - Pr[1 \leftarrow D^{R^k}(\cdot)]| \) is negligible, where \( R^k : Dom_\kappa \to Rng_\kappa \) is a truly random function.

4.2 Transformation \( \pi_{\text{SLAnon}} \) for SLAnon-RE

We give the generic transformation \( \pi_{\text{SLAnon}} \) which provides SLAnon-RE to a SLAnon GS scheme. For a GS scheme \( \mathcal{G}_S = (\mathcal{G}_k, \mathcal{U}_k, \text{Join}, \text{Iss}, \mathcal{G}_S, \text{GVf}, \text{Open}, \text{Judge}) \), we denote the transformed GS scheme by \( \pi_{\text{SLAnon}}(\mathcal{G}_S) = (\mathcal{G}_k', \mathcal{U}_k', \text{Join}', \text{Iss}', \mathcal{G}_S', \text{GVf}', \text{Open}', \text{Judge}') \). We denote randomness used in \( GSig \) by \((r_1, \ldots, r_n)\). \( \pi_{\text{SLAnon}} \) uses PRFs \((F_1, \ldots, F_n)\), where Dom and Rng of \( F_j \) depend on the space of \( r_j \). The protocol of \( \pi_{\text{SLAnon}} \) is as follows:

- \( \mathcal{G}_k'(\cdot) \): Choose PRFs \((F_1, \ldots, F_n)\), generate \((gpk, ik, ok) \leftarrow \mathcal{G}_k(\cdot)\), and set \( gpk' = (gpk, F_1, \ldots, F_n) \). Output \((gpk', ik, ok)\).
- \( \mathcal{U}_k'(\cdot) \): Choose salts \((k_1, \ldots, k_n)\), generate \((upk[i], usk[i]) \leftarrow \mathcal{U}_k(\cdot)\), and set \( usk[i]' = (usk[i], k_1, \ldots, k_n) \). Output \((upk[i], usk[i]')\).
- \( \text{Join}'(s_{1n}, m_{jn}) \) and \( \text{Iss}'(s_{1n}, m_{jn}) \): \( \text{Iss}' \) is the same as \( \text{Iss} \). For \( \text{Join}' \), execute \((s_{1n}, m_{jn}, dec') \leftarrow \text{Iss}(s_{1n}, m_{jn}) \) and \((s_{1n}, M_{jn}, dec') \leftarrow \text{Join}(s_{1n}, m_{jn}) \) until \( dec' = accept \), and get \( gsk[i]' = (gsk[i], k_1, \ldots, k_n) \). Output \((gsk[i]', dec')\). For \( \text{Join}' \), set \( \text{reg}(i) \) for \( \text{Iss}' \).
- \( \mathcal{G}_S'(gpk', gsk[i]'', m, \sigma, \tau) \) and \( \text{Judge}'(gpk, i, upk[i], m, \sigma, \tau) \): \( \mathcal{G}_S', \text{Open}' \) and \( \text{Judge}' \) are the same as \( \mathcal{G}_S, \text{Open} \) and \( \text{Judge} \), respectively.

4.3 Security

We show security statements of the generic transformation \( \pi_{\text{SLAnon}} \).

Theorem 4.1 (Correctness of \( \pi_{\text{SLAnon}} \)): If \( \mathcal{G}_S \) satisfies correctness, then \( \pi_{\text{SLAnon}}(\mathcal{G}_S) \) does as well.

Correctness is obvious from the protocol.

Theorem 4.2 (SLAnon-RE of \( \pi_{\text{SLAnon}} \)): If \( \mathcal{G}_S \) is SLAnon and \((F_1, \ldots, F_n)\) are PRFs, then \( \pi_{\text{SLAnon}}(\mathcal{G}_S) \) is SLAnon-RE.

Theorem 4.3 (Traceability of \( \pi_{\text{SLAnon}} \)): If \( \mathcal{G}_S \) is traceable and \((F_1, \ldots, F_n)\) are PRFs, then \( \pi_{\text{SLAnon}}(\mathcal{G}_S) \) is traceable against randomness exposure.

Theorem 4.4 (Non-Frameability of \( \pi_{\text{SLAnon}} \)): If \( \mathcal{G}_S \) is non-frameable and \((F_1, \ldots, F_n)\) are PRFs, then \( \pi_{\text{SLAnon}}(\mathcal{G}_S) \) is non-frameable against randomness exposure.

Here, we give sketches of proofs.

First, we show the intuition of the proof of Theorem 4.2. The difference between \( \mathcal{G}_S \) and \( \pi_{\text{SLAnon}}(\mathcal{G}_S) \) is how to generate a signature. In \( GSig \) randomness is directly used, and in \( \pi_{\text{SLAnon}}(\mathcal{G}_S) \) outputs of PRFs are used instead of randomness. From pseudo-randomness of PRFs, the adversary cannot distinguish outputs of PRFs from random values because the adversary cannot pose USK(\( i_0 \)) and USK(\( i_1 \)), and cannot obtain salts \((k_1, \ldots, k_n)\) of \( i_0 \) and \( i_1 \). Thus, outputs of PRFs in \( \pi_{\text{SLAnon}}(\mathcal{G}_S) \) can be replaced with random values generated from random functions. In this
situation, \text{Rev} oracle does not help the adversary because outputs of \text{Rev} are independent to random values used in \(\pi_{\text{SLAnon}}(\mathcal{GS})\). Therefore, \text{SLAnon} of \mathcal{GS} implies \text{SLAnon-RE} of \(\pi_{\text{SLAnon}}(\mathcal{GS})\).

Next, we show the intuition of the proof of Theorem 4.3. In the experiment, an adversary does not have the access to \text{Sig} and \text{Ch}_b. Thus, \text{Rev} oracle never returns other than \(\bot\), and is not any help of the adversary. Therefore, traceability of \(\mathcal{GS}\) without \text{Rev} oracle naturally implies traceability of \(\pi_{\text{SLAnon}}(\mathcal{GS})\).

Finally, we show the intuition of the proof of Theorem 4.4. In the experiment, an adversary cannot pose \text{USK} and \text{CrptU} for the target user. Hence, the adversary cannot distinguish outputs of \text{PRFs} from random values because the adversary cannot obtain salts \((k_1, \ldots, k_n)\) of the target user. Outputs of \text{PRFs} in \(\pi_{\text{SLAnon}}(\mathcal{GS})\) can be replaced with random values generated from random functions. A subtle point is that the adversary may pose \text{Sig} query for the target user before outputting the forgery. Thus, the simulator guesses the target user in advance. It is possible because the maximum number of group members is polynomial. Therefore, non-frameability of \(\mathcal{GS}\) without \text{Rev} oracle implies non-frameability of \(\pi_{\text{SLAnon}}(\mathcal{GS})\).

4.3.1 Proof of Theorem 4.2

We change the interface of oracle queries in the experiment of \(\text{Exp}_{\text{anonym}}(\mathcal{GS}, \mathcal{A})\). These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, \text{SLAnon-RE} of \(\pi_{\text{SLAnon}}(\mathcal{GS})\) is guaranteed from \text{SLAnon} of \(\mathcal{GS}\). We denote these hybrid experiments by \(\mathcal{H}_0, \ldots, \mathcal{H}_n\), and the advantage of the adversary \(\mathcal{A}\) when participating in experiment \(\mathcal{H}_i\) by \(\text{Adv}(\mathcal{A}, \mathcal{H}_i)\).

\textbf{Hybrid experiment} \(\mathcal{H}_0\): This experiment denotes \(\text{Exp}_{\text{anonym}}(\mathcal{GS}, \mathcal{A})\), and in this experiment the environment for \(\mathcal{A}\) is as defined in the protocol. Thus, \(\text{Adv}(\mathcal{A}, \mathcal{H}_0)\) is the same as \(\text{Adv}_{\text{anonym}}(\mathcal{GS}, \mathcal{A})\).

\textbf{Hybrid experiment} \(\mathcal{H}_j\) for \(1 \leq j \leq n\): The computation of GS\text{Sig}\((gpk, \text{gs}\{i\}^j, m; (r'_1, \ldots, r'_n))\) in \text{Ch}_b oracle is changed. Instead of computing \(r_j = F_j(r'_j, k_j)\), it is changed as obtaining \(r_j\) from a random function \(RF\).

We construct a distinguisher \(\mathcal{D}\) to distinguish \(r_j\) from \(r_j = F_j(r'_j, k_j)\) by assuming an adversary \(\mathcal{A}\) distinguish \(\mathcal{H}_j\) from \(\mathcal{H}_{j-1}\). \(\mathcal{D}\) performs as follows:

\textbf{Setup.} Generate \((gpk', ik, ok) \leftarrow \text{GKG}'(1^n)\), and set \(\text{CU} \leftarrow \emptyset\), \(\text{H} \leftarrow \emptyset\), \(\text{Gset} \leftarrow \emptyset\) and \(\sigma\text{set} \leftarrow \emptyset\). Give \(gpk'\) and \(ik\) to \(\mathcal{A}\) as input.

\textbf{Simulation.}

- \(\text{Op}(m, \sigma)\): If \((m, \sigma) \in \text{Gset}\), then return \(\bot\). Otherwise, return \(\text{Open}'(gpk', ok, \text{reg}, m, \sigma)\).

- \(\text{SnToU}(i, M_{in})\): If \(i \notin \text{H}\) then \(\text{H} \leftarrow \text{H} \cup \{i\}\). Generate \((upk[i], \text{usg}[i]') \leftarrow \text{UKG}'(1^n)\), and set \(gsk[i]' \leftarrow \bot\) and \(S_{j_m}' \leftarrow (gpk', upk[i], \text{usg}[i]')\). Generate \((S_{j_m}', M_{out}, \text{dec}') \leftarrow \text{Join}(S_{j_m}', M_{in})\). If \(\text{dec}' = \text{accept}\), then \(gsk[i]' \leftarrow S_{j_m}'\). Otherwise, return \((M_{out}, \text{dec}')\).

- \(\text{WReg}(i, \rho)\): Set \(\text{reg}[i] \leftarrow \rho\).

- \(\text{USK}(i)\): Return \((gsk[i]', \text{usg}[i])\).

- \(\text{CrptU}(i, upk)\): If \(i \in \text{HU} \cup \text{CU}\), then return \(\bot\). Set \(\text{CU} \leftarrow \text{CU} \cup \{i\}\), \(upk[i] \leftarrow upk\), \(\text{dec}' \leftarrow \text{cont}\), and \(S_{i_{ss}} \leftarrow (gpk', ik, i, upk[i])\).

- \(\text{Sig}(i, m)\): If \(i \notin \text{HU}\), then return \(\bot\). If \(gsk[i]' = \bot\), then return \(\bot\). Choose \(r'_1, \ldots, r'_n\) from \(\text{Dom}\) of \text{PRFs} and \(r_{j-1}\) from \(\text{Rng}\) of \text{PRFs}, pose \(r'_j\) to the \text{PRF oracle} (i.e., \(F_j\) or \(RF\)), and obtain \(r_j\). Generate \(\sigma \leftarrow \text{GSig}(gpk, gsk[i], m; \langle r_1, \ldots, r_{j-1}, r_j, F_{j+1}(r'_{j+1}, k_{j+1}), \ldots, F_n(r'_n, k_n)\rangle)\). Set \(\sigma\text{set} \leftarrow \sigma\text{set} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))\). Return \(\sigma\).

- \(\text{Ch}_b(i_0, i_1, m)\): If \(i_0 \notin \text{HU} \cup \text{CU}\), then return \(\bot\). If \(gsk[i_0]' = \bot\) or \(gsk[i_1]' = \bot\), then return \(\bot\). Choose \(r'_1, \ldots, r'_n\) from \(\text{Dom}\) of \text{PRFs} and \(r_{j-1}\) from \(\text{Rng}\) of \text{PRFs}, pose \(r'_j\) to the \text{PRF oracle} (i.e., \(F_j\) or \(RF\)), and obtain \(r_j\). Generate \(\sigma \leftarrow \text{GSig}(gpk, gsk[i], m; \langle r_1, \ldots, r_{j-1}, r_j, F_{j+1}(r'_{j+1}, k_{j+1}), \ldots, F_n(r'_n, k_n)\rangle)\). Set \(\sigma\text{set} \leftarrow \sigma\text{set} \cup (i_0, m, \sigma, (r'_1, \ldots, r'_n))\). Return \(\sigma\).

- \(\text{Rev}(m, \sigma)\): If \((i, m, \sigma, r) \notin \sigma\text{set}\), then return \(\bot\). Otherwise, return \(r\).

\textbf{Output.} If \(\mathcal{A}\) outputs \(b' = b\), then \(\mathcal{D}\) outputs 1. Otherwise, \(\mathcal{D}\) outputs 0.

\textbf{Analysis.} If \(\mathcal{A}\) can check if randomness used in \(\text{Ch}_b\) oracle is outputs of \text{PRFs}, then \(\mathcal{A}\) can distinguish the simulation from the real experiment. However, \(\mathcal{A}\) cannot know salts of \(i_0, i_1\) because \(\text{USK}(i_0)\) and \(\text{USK}(i_1)\) cannot be posed for. For \(\mathcal{A}\), the simulation by \(\mathcal{D}\) is same as the experiment \(\mathcal{H}_{j-1}\) if the \text{PRF oracle} is the \text{PRF} \(F_j\). Otherwise, the simulation by \(\mathcal{D}\) is same as the experiment \(\mathcal{H}_j\). Thus, if the advantage of \(\mathcal{D}\) is negligible, then \(\text{Adv}(\mathcal{A}, \mathcal{H}_j) = \text{Adv}(\mathcal{A}, \mathcal{H}_{j-1})\) is negligible.

\textbf{Bounding} \(\text{Adv}(\mathcal{A}, \mathcal{H}_j)\): In \(\mathcal{H}_n\), all computations of GS\text{Sig}' in \(\text{Ch}_b\) oracle use random \((\bar{r}_1, \ldots, \bar{r}_n)\) instead of \((F_1(r'_1, k_1), \ldots, F_n(r'_n, k_n))\). Thus, outputs of \(\text{Rev} oracle\) \((r'_1, \ldots, r'_n)\) are independent from \(\sigma\).

We construct an adversary \(\mathcal{B}\) for \(\mathcal{GS}\) by assuming an adversary \(\mathcal{A}\) for \(\mathcal{H}_n\). \(\mathcal{B}\) performs as follows:

\textbf{Setup.} Receive \((gpk, ik)\) as the challenge, choose \text{PRFs} \((F_1, \ldots, F_n)\), and set \(gpk' = (gpk^*, F_1, \ldots, F_n)\) and \(\sigma\text{set} \leftarrow \emptyset\). Give \(gpk'\) and \(ik\) to \(\mathcal{A}\) as input.
Simulation.

- **Op**: Pose $(m, \sigma)$ to Op, and return the output of Op.
- **SndToU**: Pose $(i, M_m)$ to SndToU, and return the output of SndToU.
- **WReg**: Pose $(i, \rho)$ to WReg.
- **USK**: Pose $i$ to USK, obtain $(upk[i], usk[i])$, choose salts $(k_1, \ldots, k_n)$, and set $usk[i]' = (usk[i], k_1, \ldots, k_n)$. Return $(upk[i], usk[i]')$.
- **CrptU**: Pose $(i, upk)$ to CrptU, and return the output of CrptU.
- **Sig**: Pose $(i, m)$ to Sig, obtain $\sigma$, and choose $(r_1', \ldots, r_m')$. Set $\sigma$ as $\sigma' \subseteq \sigma \cup (i, m, \sigma, (r_1', \ldots, r_m'))$. Return $\sigma$.
- **Chb**: Pose $(i_0, i_1, m)$ to Chb, obtain $\sigma$, and choose $(r_1', \ldots, r_m')$. Set $\sigma'$ as $\sigma' \subseteq \sigma \cup (i, m, \sigma, (r_1', \ldots, r_m'))$. Return $\sigma'$.
- **Rev**: If $(i, m, \sigma, \rho) \in \sigma'$, then return $\bot$. Otherwise, return $r$.

Output. If $\mathcal{A}$ outputs $b'$, then $\mathcal{B}$ outputs $b'$.

Analysis. For $\mathcal{A}$, the simulation by $\mathcal{B}$ is the same as the experiment $H_0$. Thus, if the advantage of $\mathcal{B}$ is negligible, then $\text{Adv}(\mathcal{A}, H_0)$ is negligible. □

4.3.2 Proof of Theorem 4.3

The proof is trivial. In $\text{Exp}_{\text{trace}(\mathcal{G}\mathcal{S}, \mathcal{A})}^{\mathcal{A}}(\kappa)$, $\mathcal{A}$ does not have Sig and Chb oracles. Thus, there is no record in $\sigma$, and Rev oracle always returns $\bot$. It means that Rev oracle is not any help of $\mathcal{A}$. Therefore, even if $\mathcal{G}\mathcal{S}$ is traceable without Rev oracle, $\mathcal{A}_{\text{LAM}}(\mathcal{G}\mathcal{S})$ is also traceable. □

4.3.3 Proof of Theorem 4.4

We change the interface of oracle queries in the experiment of $\text{Exp}_{\text{trace}(\mathcal{G}\mathcal{S}, \mathcal{A})}^{\mathcal{A}}(\kappa)$. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, non-frameability of $\mathcal{A}_{\text{LAM}}(\mathcal{G}\mathcal{S})$ is guaranteed from non-frameability of $\mathcal{G}\mathcal{S}$ without Rev oracle. We denote these hybrid experiments by $H_0, \ldots, H_1$, and the advantage of the adversary $\mathcal{A}$ when participating in experiment $H_i$ by $\text{Adv}(\mathcal{A}, H_i)$.

**Hybrid experiment $H_0$**: This experiment denotes $\text{Exp}_{\text{trace}(\mathcal{G}\mathcal{S}, \mathcal{A})}^{\mathcal{A}}(\kappa)$, and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\text{Adv}(\mathcal{A}, H_0)$ is the same as $\text{Adv}(\mathcal{A}, \mathcal{H}_0)$.

**Hybrid experiment $H_1$**: The experiment fixes a user $i'$ which is chosen from $[1, \ldots, N]$, and if $A$ outputs $i \neq i'$ then halts.

Since the guess of the user matches with $\mathcal{A}$’s choice with probability $1/N$ for the maximum number of group members $N$, $\text{Adv}(\mathcal{A}, H_1) \geq 1/N \cdot \text{Adv}(\mathcal{A}, H_0)$.

**Hybrid experiment $H_j$ for $2 \leq j \leq n + 1$**: The computation of $\text{GSig}(gpk, gsk[i']', m; (r_1', \ldots, r_n'))$ in $\mathcal{S}$ is changed. Instead of computing $r_j = F_{j-1}(r_j', k_j)$, it is changed as obtaining $r_j$ from a random function $RF$.

We construct a distinguisher $D$ to distinguish $r_j$ from $F_{j-1}(r_j', k_j)$ by assuming an adversary $\mathcal{A}$ distinguishes $H_j$ from $H_{j-1}$. $D$ performs as follows:

Setup. Generate $(gpk', ik, ok) \leftarrow \text{Gkgen}(1^t)$, and set $CU \leftarrow 0$ and $HU \leftarrow 0$, $Gset \leftarrow 0$ and $\sigma set \leftarrow 0$. Give $(gpk', ik, ok)$ to $\mathcal{A}$ as input.

Simulation.

- **SndToU**: If $i \notin \text{HU}$ then $HU \leftarrow \text{HU} \cup \{i\}$. Generate $(upk[i], usk[i]') \leftarrow \text{Ukgen}(1^t)$, and set $gsk[i']' \leftarrow \bot$ and $S_j \leftarrow (gpk', upk[i], usk[i]')$. Generate $(S_{j-1}', M_{out}, dec') \leftarrow \text{Join}(S_j, M_{out})$. If $dec' = \text{accept}$, then $gsk[i]' \leftarrow S_{j-1}'$. Otherwise, return $(M_{out}, dec')$.
- **WReg**: Set $reg[i] \leftarrow \rho$.
- **USK**: Return $(gsk[i]', usk[i])$.
- **CrptU**: If $i \in \text{HU} \cup \text{CU}$, then return $\bot$. Set $CU \leftarrow \text{CU} \cup \{i\}$, $upk[i] \leftarrow \text{upk}$, $dec' \leftarrow \bot$, and $S_{j-1}' \leftarrow (gpk', ik, i, upk[i])$.
- **Sig**: If $i \notin \text{HU}$, then return $\bot$. If $gsk[i]' = \bot$, then return $\bot$. If $i = i'$, then choose $r_{i}'', \ldots, r_{n}'$ from Dom of PRFs and $r_{1}', \ldots, r_{j-2}'$ from rng of PRFs, pose $r_{j-1}'$ to the PRF oracle (i.e., $F_{j-1}$ or $RF$), obtain $r_{j-1}$, and generate $\sigma \leftarrow \text{GSig}(gpk', gsk[i]', m; (r_1', \ldots, r_j')', F_{j}(r_{j-1}, k_j), \ldots, F_{n}(r_{n}', k_n)))$. Otherwise, choose $r_{i}'', \ldots, r_{n}'$, and generate $\sigma \leftarrow \text{GSig}(gpk', gsk[i]', m; (r_1', \ldots, r_n')')$. Set $\sigma set \leftarrow \sigma set \cup (i, m, \sigma, (r_1', \ldots, r_n'))$. Return $\sigma$.
- **Rev**: If $(i, m, \sigma, r) \notin \sigma set$, then return $\bot$. Otherwise, return $r$.

Output. If $\mathcal{A}$ outputs $b' = b$, then $D$ outputs 1. Otherwise, $D$ outputs 0.

Analysis. If $\mathcal{A}$ can check if randomness used in Sig oracle is outputs of PRFs, then $\mathcal{A}$ can distinguish the simulation from the real experiment. However, since $\mathcal{A}$ can pose neither USK($i'$) nor CrptU($i'$), $\mathcal{A}$ cannot know salts of $i'$. For $\mathcal{A}$, the simulation by $D$ is same as the experiment $H_{j-1}$ if the PRF oracle is the PRF $F_{j-1}$. Otherwise, the simulation by $D$
is same as the experiment $H_j$. Thus, if the advantage of $D$ is negligible, then $|\text{Adv}(\mathcal{A}, H_j) - \text{Adv}(\mathcal{A}, H_{j-1})|$ is negligible.

**Bounding $\text{Adv}(\mathcal{A}, H_{n+1})$:** In $H_{n+1}$, all computations of $\text{GSig}$ in $\text{Sig}$ oracle use random $(\vec{r}_1, \ldots, \vec{r}_n)$ instead of $(F_1'(\vec{r}_1, k_1), \ldots, F_n'(\vec{r}_n, k_n))$. Thus, outputs of $\text{Rev}$ oracle $(\vec{r}_1', \ldots, \vec{r}_n')$ are independent from $\sigma$.

We construct an adversary $\mathcal{B}$ for $\mathcal{GS}$ by assuming an adversary $\mathcal{A}$ for $H_{n+1}$. $\mathcal{B}$ performs as follows:

**Setup.** Receive $(gpk, ik, ok)$ as the challenge, choose PRFs $(F_1, \ldots, F_n)$, and set $gpk' = (gpk, F_1, \ldots, F_n)$ and $\sigma \leftarrow \emptyset$. Give $(gpk', ik, ok)$ to $\mathcal{A}$ as input.

**Simulation.**

- $\text{SndToU}(i, M_m)$: Pose $(i, M_m)$ to $\text{SndToU}$, and return the output of $\text{SndToU}$.
- $\text{WReg}(i, \rho)$: Pose $(i, \rho)$ to $\text{WReg}$.
- $\text{USK}(i)$: Pose $i$ to $\text{USK}$, obtain $(upk[i], usk[i])$, choose salts $(k_1, \ldots, k_n)$, and set $usk[i]' = (usk[i], k_1, \ldots, k_n)$. Return $(upk[i], usk[i]')$.
- $\text{CrptU}(i, upk)$: Pose $(i, upk)$ to $\text{CrptU}$, and return the output of $\text{CrptU}$.
- $\text{Sig}(i, m)$: Pose $(i, m)$ to $\text{Sig}$, obtain $\sigma$, and choose $(r_1', \ldots, r_n')$. Set $\sigma \leftarrow \sigma \uplus (i, m, \sigma, (r_1', \ldots, r_n'))$. Return $\sigma$.
- $\text{Rev}(m, \sigma)$: If $(+, m, \sigma, r) \notin \sigma \uplus r$, then return $\bot$. Otherwise, return $r$.

**Output.** If $\mathcal{A}$ outputs $b'$, then $\mathcal{B}$ outputs $b'$.

**Analysis.** For $\mathcal{A}$, the simulation by $\mathcal{B}$ is same as the experiment $H_{n+1}$. Thus, if the advantage of $\mathcal{B}$ is negligible, then $\text{Adv}(\mathcal{A}, H_{n+1})$ is negligible.

5. **Achieving Full-Anonymity against Full Randomness Exposure with Physical Assumption**

We show another way to avoid our impossibility in Sect. 3 by relying on a physical assumption (i.e., using the SC). We give the generic transformation $\pi_{F\text{Anon}}$ which provides $\text{FAnon-RE}$ to a $\text{FAnon GS}$ scheme. For a $GS$ scheme $\mathcal{GS} = (\text{GKg}, \text{UKg}, \text{Join}, \text{Iss}, \text{GSig}, \text{GVI}, \text{Open}, \text{Judge})$, we denote the transformed GS scheme by $\pi_{\text{FAnon}}(\mathcal{GS}) = (\text{GKg}', \text{UKg}', \text{Join}', \text{Iss}', \text{GSig}', \text{GVI}', \text{Open}', \text{Judge}')$. We denote randomness used in $\text{GSig}$ by $(r_1, \ldots, r_n)$ (i.e., $\text{GSig}$ internally generates $n$ random values). $\pi_{\text{FAnon}}$ uses PRFs $(F_1, \ldots, F_n)$ and the SC $S_C$, where $\text{Rng}$ of $F_j$ depend on the space of $r_j$. The protocol of $\pi_{\text{FAnon}}$ is as follows:

1If some randomness are chosen from the common space, then we can use the common PRF for such randomness.
other than $\perp$, and is not any help of the adversary. Therefore, traceability of GS without $\texttt{Rev}$ oracle naturally implies traceability of $\pi_{F\text{Anon}}(GS)$.

Finally, we show the intuition of the proof of Theorem 5.4. Like the situation of the proof of Theorem 5.2, the adversary cannot distinguish outputs of PRFs from random values because the adversary cannot obtain salts $(k_1, \ldots, k_n)$ from $SC$ even when USK is posed. Outputs of PRFs in $\pi_{F\text{Anon}}(GS)$ can be replaced with random values generated from random functions. Therefore, non-frameability of GS without $\texttt{Rev}$ oracle implies non-frameability of $\pi_{F\text{Anon}}(GS)$.

**References**


**Appendix A: Security Proofs for $\pi_{F\text{Anon}}$.**

In this section, we prove security of generic transformations.

A.1 Proof of Theorem 5.2

We change the interface of oracle queries in the experiment of $\text{Exp}_{\pi_{F\text{Anon}}(GS), A}^0(\kappa)$. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, $\text{FAnon-RE} of \pi_{F\text{Anon}}(GS)$ is guaranteed from $\text{FAnon} of GS$. We denote these hybrid experiments by $H_0, \ldots, H_n$, and the advantage of the adversary $A$ when participating in experiment $H_i by Adv(A, H_i)$.

**Hybrid experiment $H_0$:** This experiment denotes $\text{Exp}_{\pi_{F\text{Anon}}(GS), A}^0(\kappa)$, and in this experiment the environment for $A$ is as defined in the protocol. Thus, $Adv(A, H_0)$ is the same as $Adv_{\pi_{F\text{Anon}}(GS), A}^0(\kappa)$.

**Hybrid experiment $H_i$ for $1 \leq j \leq n$:** The computation of GSig($gpk, gsk[i]$, $m; (r'_1, \ldots, r'_n)$) in Ch0 oracle is changed. Instead of computing $r_j = F_j(r'_j, k_j)$, it is changed as obtaining $\tilde{r}_j$ from a random function $\tilde{RF}$. We construct a distinguisher $D$ to distinguish $\tilde{r}_j$ from $r_j = F_j(r'_j, k_j)$ by assuming an adversary $A$ distinguish $H_j$ from $H_{j-1}$. $D$ performs as follows:

**Setup.** Generate $(gpk', ik, ok) \leftarrow \text{GKg}(1^\kappa)$, and set $\text{CU} \leftarrow 0$, $\text{HU} \leftarrow 0$, $\text{Gset} \leftarrow 0$ and $\text{crset} \leftarrow 0$. Give $gpk'$ and $ik$ to $A$ as input.

**Simulation.**

- $\text{Op}(m, \sigma)$: If $(m, \sigma) \in \text{Gset}$, then return $\perp$. Otherwise, return $\text{Open'}(gpk', ok, reg, m, \sigma)$.
- $\text{SndToU}(i, M_{out})$: If $i \notin \text{HU}$ then $\text{HU} \leftarrow \text{HU} \cup \{i\}$. Generate $(upk[i], usk[i]) \leftarrow \text{UKg}(1^\kappa)$, and set $gsk[i] \leftarrow \perp$ and $S_{i_{\text{out}}} \leftarrow (gpk', upk[i], usk[i])$. Generate $(S_{i_{\text{out}}}, dec^c) \leftarrow \text{Join}(S_{i_{\text{out}}}, M_{out})$. If $dec^c = \text{accept}$, then $gsk[i] \leftarrow S_{i_{\text{out}}}$. Otherwise, return $(M_{out}, dec^c)$.
- $\text{WReg}(i, \rho)$: Set $\text{reg} \leftarrow \rho$.
- $\text{USK}(i)$: Return $(gsk[i], usk[i])$. 


\textbf{A.2 Proof of Theorem 5.3}

The proof is trivial. In $\text{Exp}^{\text{trace}}_{\pi_{\text{FAnon}}(G\mathcal{S}), \mathcal{A}}(\kappa)$, $\mathcal{A}$ does not have $\text{Sig}$ and $\text{Ch}_b$ oracles. Thus, there is no record in $\sigma_{\text{set}}$, and $\text{Rev}$ oracle always returns $\bot$. It means that $\text{Rev}$ oracle is not any help of $\mathcal{A}$. Therefore, even if $G\mathcal{S}$ is traceable without $\text{Rev}$ oracle, $\pi_{\text{FAnon}}(G\mathcal{S})$ is also traceable. \hfill $\square$

\textbf{A.3 Proof of Theorem 5.4}

We change the interface of oracle queries in the experiment of $\text{Exp}^{\text{trace}}_{\pi_{\text{FAnon}}(G\mathcal{S}), \mathcal{A}}(\kappa)$. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, non-frameability of $\pi_{\text{FAnon}}(G\mathcal{S})$ is guaranteed from non-frameability of $G\mathcal{S}$ without $\text{Rev}$ oracle. We denote these hybrid experiments by $H_0, \ldots, H_n$, and the advantage of the adversary $\mathcal{A}$ when participating in experiment $H_j$ by $\text{Adv}(\mathcal{A}, H_j)$.

\textbf{Hybrid experiment $H_n$:} This experiment denotes $\text{Exp}^{\text{trace}}_{\pi_{\text{FAnon}}(G\mathcal{S}), \mathcal{A}}(\kappa)$, and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\text{Adv}(\mathcal{A}, H_0)$ is the same as $\text{Adv}^{\text{trace}}_{\pi_{\text{FAnon}}(G\mathcal{S}), \mathcal{A}}(\kappa)$.

Hybrid experiment $H_j$ for $1 \leq j \leq n$: The computation of $\text{GSig}(gpk, gsk[i], m; (r'_1, \ldots, r'_n))$ in $\text{Sig}$ oracle is changed. Instead of computing $r_j = F_j(r'_j, k_j)$, it is changed as obtaining $\tilde{r}_j$ from a random function $RF$.

We construct a distinguisher $D$ to distinguish $\tilde{r}_j$ from $r_j = F_j(r'_j, k_j)$ by assuming an adversary $\mathcal{A}$ distinguish $H_j$ from $H_{j-1}$. $D$ performs as follows:

\textbf{Setup.} Generate $(gpk', ik, ok) \leftarrow \text{GKg}'(1^\kappa)$, and set $\text{CU} \leftarrow \text{Op}(m, \sigma)$: Pose $(m, \sigma)$ to $\text{Op}$, and return the output of $\text{Op}$.

\textbf{SndToU(i, M_n):} Pose $(i, M_n)$ to $\text{SndToU}$, and return the output of $\text{SndToU}$.

\textbf{WReg(i, q):} Pose $(i, q)$ to $\text{WReg}$.

\textbf{USK(i):} Pose $i$ to $\text{USK}$, obtain $(upk[i], usk[i])$, choose salts $(k_1, \ldots, k_n)$, and set $usk[i]' = (usk[i], k_1, \ldots, k_n)$. Return $(upk[i], usk[i]')$.

\textbf{CrptU(i, upk):} Pose $(i, upk)$ to $\text{CrptU}$, and return the output of $\text{CrptU}$.

\textbf{Sig(i, m):} Pose $(i, m)$ to $\text{Sig}$, obtain $\sigma$, and choose $(r'_1, \ldots, r'_n)$. Set $\sigma_{\text{set}} \leftarrow \sigma_{\text{set}} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return $\sigma$.

\textbf{Ch_b(i_0, i_1, m):} If $i_0 \notin \text{HU}$ or $i_1 \notin \text{HU}$, then return $\bot$. If $gsk[i_0]' = \bot$, then return $\bot$. Choose $r'_1, \ldots, r'_n$ from $\text{Dom}$ of PRFs and $\tilde{r}_1, \ldots, \tilde{r}_{j-1}$ from $\text{Rng}$ of PRFs, pose $r'_j$ to the PRF oracle (i.e., $F_j$ or $RF$), and obtain $r_j$. Generate $\sigma \leftarrow \text{GSig}(gpk, gsk[i_0], m; (\tilde{r}_1, \ldots, \tilde{r}_{j-1}, r_j, F_{j+1}(r'_{j+1}, k_{j+1}), \ldots, F_n(r'_n, k_n)))$. Set $\sigma_{\text{set}} \leftarrow \sigma_{\text{set}} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return $\sigma$.

\textbf{Rev(m, \sigma):} If $(i, m, \sigma, r) \notin \sigma_{\text{set}}$, then return $\bot$. Otherwise, return $r$.

\textbf{Output.} If $\mathcal{A}$ outputs $b' = b$, then $D$ outputs $1$. Otherwise, $D$ outputs $0$.

\textbf{Analysis.} For $\mathcal{A}$, the simulation by $B$ is same as the experiment $H_0$. Thus, if the advantage of $B$ is negligible, then $\text{Adv}(\mathcal{A}, H_0)$ is negligible. \hfill $\square$
\(\emptyset\) and HU \(\leftarrow\) \(\emptyset\), Gset \(\leftarrow\) \(\emptyset\) and \(\sigma\)set \(\leftarrow\) \(\emptyset\). Give \((gpk', ik, ok)\) to \(\mathcal{A}\) as input.

**Simulation.**

- **SndToU\((i, M_m)\):** If \(i \notin \text{HU}\) then HU \(\leftarrow\) HU \(\cup\{i\}\). Generate \((upk[i], usk[i])' \leftarrow \text{UKg}(k^i1)\) and set \(gsk[i]' \leftarrow \bot\) and \(S_{\text{in}}^i \leftarrow (gpk', upk[i], usk[i]'\). Generate \((S_{\text{out}}^i, M_{\text{out}}, \text{dec}^i) \leftarrow \text{Join}'(S_{\text{in}}^i, M_m)\). If \(\text{dec}^i = \text{accept}\), then \(gsk[i]' \leftarrow S_{\text{in}}^i\). Otherwise, return \((M_{\text{out}}, \text{dec}^i)\).

- **WReg\((i, \rho)\):** Set \(\text{reg}[i] \leftarrow \rho\).

- **USK\((i)\):** Return \((gsk[i]', usk[i])\).

- **CrptU\((i, upk)\):** If \(i \in \text{HU} \cup \text{CU}\), then return \(\bot\). Set \(\text{CU} \leftarrow \text{CU} \cup \{i\}\), \(upk[i] \leftarrow upk\), \(\text{dec}^i \leftarrow \text{cont}\), and \(S_{\text{in}}^i \leftarrow (gpk', i, i, upk[i])\).

- **Sig\((i, m)\):** If \(i \notin \text{HU}\), then return \(\bot\). If \(gsk[i]' = \bot\), then return \(\bot\). Choose \(r'_1, \ldots, r'_n\) and \(\tilde{r}_1, \ldots, \tilde{r}_{j-1}\), pose \(r'_j\) to the PRF oracle \((i.e., F_j \text{ or } RF)\), obtain \(r_j\), and generate \(\sigma \leftarrow \text{GSig}(gpk, gsk[i], m, (\tilde{r}_1, \ldots, \tilde{r}_{j-1}, r_j, F_{j+1}(r'_{j+1}, k_{j+1}), \ldots, F_n(r'_n, k_n));\). Set \(\text{set} \leftarrow \sigma\)set \(\cup\) \(\sigma\) and \((i, m, \sigma, r'_1, \ldots, r'_n)\). Set \(\sigma\).

- **Rev\((m, \sigma)\):** If \((i, m, \sigma, r) \notin \sigma\) set, then return \(\bot\). Otherwise, return \(r\).

**Output.** If \(\mathcal{A}\) outputs \(b' = b\), then \(\mathcal{D}\) outputs 1. Otherwise, \(\mathcal{D}\) outputs 0.

**Analysis.** For \(\mathcal{A}\), the simulation by \(\mathcal{B}\) is same as the experiment \(\mathcal{H}_m\). Thus, if the advantage of \(\mathcal{B}\) is negligible, then \(\text{Adv}(\mathcal{A}, \mathcal{H}_m)\) is negligible.

Tomoyoshi Ono received the B.E., degree from the University of Ibaraki, Ibaraki, Japan, in 2016. He is currently a graduate student at University of Tsukuba in Tsukuba, Japan since 2016.

Kazuki Yoneyama received the B.E., M.E. and Ph.D. degrees from the University of Electro-Communications, Tokyo, Japan, in 2004, 2006 and 2008, respectively. He was a researcher of NTT Secure Platform Laboratories from 2009 to 2015. He is presently engaged in research on cryptography at the Ibaraki University, since 2015. He is a member of the International Association for Cryptologic Research (IACR), IPSJ and JSIAM.