PAPER *Special Section on Security, Privacy and Anonymity in Computation, Communication and Storage Systems*

On Randomness Exposure Resilience of Group Signatures

Tomoyoshi ONO[†*], ***Nonmember and*** **Kazuki YONEYAMA**[†a)], ***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-

Manuscript received December 15, 2016.
Manuscript revised April 20, 2017.
Manuscript publicized July 21, 2017.
[†]The authors are with Ibaraki University, Hitachi-shi, 316–8511 Japan.
[*]Presently, the author is with the University of Tsukuba.
a) E-mail: kazuki.yoneyama.sec@vc.ibaraki.ac.jp
DOI: 10.1587/transinf.2016INP0015

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 $\mathcal{ALG}$ is an algorithm, then by $y \leftarrow \mathcal{ALG}(x; r)$ we denote that y is output by $\mathcal{ALG}$ on input x and randomness r (if $\mathcal{ALG}$ is deterministic, r is empty). If $\mathcal{A}$ is an adversary, then by $y \leftarrow \mathcal{A}(x : \mathsf{Oracle})$ 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 $\mathcal{GS}$ 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 κ 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 isuuer 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 κ 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, an outgoing message M_{out} and a decision $dec \in \{accept, reject, cont\}$ where $accept$ means that the interaction is accepted, $reject$ means that the interaction is accepted and $cont$ 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 $reg[i]$ for the user i, in a registration table 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 σ 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 σ, and outputs 1 if σ is valid for the massage m, otherwise outputs 0.

```
AddU($i$)
If $i \in$ CU or $i \in$ HU then return $\perp$
HU $\leftarrow$ HU $\cup \{i\}$
$dec^i \leftarrow cont$; $gsk[i] \leftarrow \perp$
$(upk[i], usk[i]) \leftarrow$ UKg$(1^\kappa)$
$St^i_{jn} \leftarrow (gpk, upk[i], usk[i])$
$St^i_{iss} \leftarrow (gpk, ik, i, upk[i])$; $M_{jn} \leftarrow \perp$
$(St^i_{jn}, M_{iss}, dec^i) \leftarrow$ Join$(St^i_{jn}, M_{jn})$
While $dec^i = cont$ do
    $(St^i_{iss}, M_{jn}, dec^i) \leftarrow$ Iss$(St^i_{iss}, M_{iss})$
    If $dec^i = accept$ then $reg[i] \leftarrow St^i_{iss}$
    $(St^i_{jn}, M_{iss}, dec^i) \leftarrow$ Join$(St^i_{jn}, M_{jn})$
Endwhile
$gsk[i] \leftarrow St^i_{jn}$
Return $upk[i]$

CrptU($i, upk$)
If $i \in$ HU $\cup$ CU then return $\perp$
CU $\leftarrow$ CU $\cup \{i\}$
$upk[i] \leftarrow upk$
$dec^i \leftarrow cont$
$St^i_{iss} \leftarrow (gpk, ik, i, upk[i])$;
Return 1

Op($m, \sigma$)
If $(m, \sigma) \in$ Gset then return $\perp$
Return Open$(gpk, ok, reg, m, \sigma)$

Sig($i, m$)
If $i \notin$ HU then return $\perp$
If $gsk[i] = \perp$ then return $\perp$
$\sigma \leftarrow$ GSig$(gpk, gsk[i], m; r)$
$\sigma$set $\leftarrow \sigma$set $\cup (i, m, \sigma, r)$
Return $\sigma$

SndToI($i, M_{in}$)
If $i \notin$ CU then return $\perp$
$(St^i_{iss}, M_{out}, dec^i) \leftarrow$ Iss$(St^i_{iss}, M_{in})$
If $dec^i = accept$ then $reg[i] \leftarrow St^i_{iss}$
Return $M_{out}$

SndToU($i, M_{in}$)
If $i \notin$ HU then
HU $\leftarrow$ HU $\cup \{i\}$
$(upk[i], usk[i]) \leftarrow$ UKg$(1^\kappa)$
$gsk[i] \leftarrow \perp$
$St^i_{jn} \leftarrow (gpk, upk[i], usk[i])$
$(St^i_{jn}, M_{out}, dec^i) \leftarrow$ Join$(St^i_{jn}, M_{in})$
If $dec^i = accept$ then $gsk[i] \leftarrow St^i_{jn}$
Return $(M_{out}, dec^i)$

USK($i$)
Return $(gsk[i], usk[i])$

RReg($i$)
Return $reg[i]$

WReg($i, \rho$)
$reg[i] \leftarrow \rho$

Ch$_b$($i_0, i_1, m$)
If $i_0 \notin$ HU or $i_1 \notin$ HU then return $\perp$
If $gsk[i_0] = \perp$ or $gsk[i_1] = \perp$ then return $\perp$
$\sigma \leftarrow$ GSig$(gpk, gsk[i_b], m; r)$
Gset $\leftarrow$ Gset $\cup (m, \sigma)$
$\sigma$set $\leftarrow \sigma$set $\cup (i_b, m, \sigma, r)$
Return $\sigma$

Rev($m, \sigma$)
If $(i, m, \sigma, r) \notin \sigma$set then return $\perp$
Return $r$
```

Fig. 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 σ, and outputs i and τ where τ is a proof that σ 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 σ and a proof τ, and outputs 1 if τ generated by Open is valid, otherwise outputs 0.

2.2 Security Definition

2.2.1 Oracles

First, we define the adversarial capacity as the access to oracles. In the previous model [3], ten oracles are defined: add user AddU, corrupt user CrptU, send to issuer SndToI, send to user SndToU, reveal user secret keys USK, read registration table RReg, write registration table WReg, sign Sig, open Op and challenge Ch. In addition to these oracles, we newly add oracle Rev which returns randomness r used to produce σ. Let HU be a set of honest users, CU be a set of corrupted users, Gset be a set of pair (m, σ) of message/signature produced by Ch$_b$ and σset be a set of a tuple (i, m, σ, r) of user/message/signature/randomness produced by Sig and Ch$_b$.

Figure 1 shows definitions of oracles. We must carefully define Rev. If inputs of Rev contain a user i as well as a message and a signature, then the adversary can check if the user i actually generates the signature. Thus, anonymity is trivially broken by using Rev. We formulate Rev to avoid such trivial attacks.

2.2.2 Correctness

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

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

```
Experiment $\mathsf{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(\kappa)$:
    CU $\leftarrow \emptyset$; HU $\leftarrow \emptyset$
    $(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^\kappa)$
    $(i, m) \leftarrow \mathcal{A}(gpk : \mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot))$
    If $i \notin \mathsf{HU}$ then return 0
    If $gsk[i] = \perp$ then return 0
    $\sigma \leftarrow \mathsf{GSig}(gpk, gsk[i], m)$
    If $0 \leftarrow \mathsf{GVf}(gpk, m, \sigma)$ then return 1
    $(j, \tau) \leftarrow \mathsf{Open}(gpk, ok, reg, m, \sigma)$
    If $i \neq j$ or $0 \leftarrow \mathsf{Judge}(gpk, i, upk[i], m, \sigma, \tau)$
    then return 1 else return 0
```

The advantage of $\mathcal{A}$ is defined as: $\mathsf{Adv}^{corr}_{\mathcal{GS},\mathcal{A}}(\kappa) = \Pr[\mathsf{Exp}^{corr}_{\mathcal{GS},\mathcal{A}}(\kappa) = 1]$. We say that the dynamic group signature scheme $\mathcal{GS}$ is correct if $\mathsf{Adv}^{corr}_{\mathcal{GS},\mathcal{A}}(\kappa) = 0$ for any PPT adversary $\mathcal{A}$.

2.2.3 Anonymity

Anonymity means that the corrupted issuer and users should be unable to distinguish the actual signer from others with a signature. Especially, if even the actual signer cannot decide if the signature is generated by him/herself, then we call full-anonymity, FAnon. If the actual signer can decide if the signature is generated by him/herself, then we call selfless-anonymity, SLAnon.

Definition 2.2 (Anonymity): We define the experiment $\mathsf{Exp}^{anon-b}_{\mathcal{GS},\mathcal{A}}(\kappa)$ for a PPT adversary $\mathcal{A}$ and a security parameter κ as follows:

```
Experimet $\mathsf{Exp}^{anon-b}_{\mathcal{GS},\mathcal{A}}(\kappa)$:
    $\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$; $\mathsf{Gset} \leftarrow \emptyset$; $\sigma\mathsf{set} \leftarrow \emptyset$
    $(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^\kappa)$
    $b' \leftarrow \mathcal{A}(gpk, ik : \mathsf{Op}(\cdot,\cdot), \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot),$
            $\mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{Sig}(\cdot,\cdot), \mathsf{Ch}_b(\cdot,\cdot,\cdot), \mathsf{Rev}(\cdot,\cdot))$
    Return $b'$
```

The advantage of $\mathcal{A}$ is defined as: $\mathsf{Adv}^{anon}_{\mathcal{GS},\mathcal{A}}(\kappa) = |\Pr[\mathsf{Exp}^{anon-1}_{\mathcal{GS},\mathcal{A}}(\kappa) = 1] - \Pr[\mathsf{Exp}^{anon-0}_{\mathcal{GS},\mathcal{A}}(\kappa) = 1]|$. We say that the dynamic group signature scheme $\mathcal{GS}$ is FAnon-RE if $\mathsf{Adv}^{anon}_{\mathcal{GS},\mathcal{A}}(\kappa)$ is negligible. We say that the dynamic group signature scheme $\mathcal{GS}$ is SLAnon-RE if $\mathcal{A}$ poses neither $\mathsf{USK}(i_0)$ nor $\mathsf{USK}(i_1)$, and $\mathsf{Adv}^{anon}_{\mathcal{GS},\mathcal{A}}(\kappa)$ is negligible for any PPT adversary $\mathcal{A}$.

Remark 2.1: When Rev oracle is removed from $\mathsf{Exp}^{anon-b}_{\mathcal{GS},\mathcal{A}}(\kappa)$, 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 Judge.

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

```
Experiment $\mathsf{Exp}^{trace}_{\mathcal{GS},\mathcal{A}}(\kappa)$:
    $\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$; $\sigma\mathsf{set} \leftarrow \emptyset$
    $(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^\kappa)$
    $(m, \sigma) \leftarrow \mathcal{A}(gpk, ok : \mathsf{SndToI}(\cdot,\cdot), \mathsf{AddU}(\cdot),$
            $\mathsf{RReg}(\cdot), \mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{Rev}(\cdot,\cdot))$
    If $0 \leftarrow \mathsf{GVf}(gpk, m, \sigma)$ then return 0
    $(i, \tau) \leftarrow \mathsf{Open}(gpk, ok, reg, m, \sigma)$
    If $i = \perp$ or $0 \leftarrow \mathsf{Judge}(gpk, i, upk[i], m, \sigma, \tau)$
    then return 1 else return 0
```

The advantage of $\mathcal{A}$ is defined as: $\mathsf{Adv}^{trace}_{\mathcal{GS},\mathcal{A}}(\kappa) = \Pr[\mathsf{Exp}^{trace}_{\mathcal{GS},\mathcal{A}}(\kappa) = 1]$. We say that the dynamic group signature scheme $\mathcal{GS}$ is traceable against randomness exposure if $\mathsf{Adv}^{trace}_{\mathcal{GS},\mathcal{A}}(\kappa)$ 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 Judge for an honest user.

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

```
Experiment $\mathsf{Exp}^{nf}_{\mathcal{GS},\mathcal{A}}(\kappa)$:
    $\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$; $\sigma\mathsf{set} \leftarrow \emptyset$
    $(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^\kappa)$
    $(m, \sigma, i, \tau) \leftarrow \mathcal{A}(gpk, ok, ik : \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot),$
            $\mathsf{USK}(\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{Sig}(\cdot,\cdot), \mathsf{Rev}(\cdot,\cdot))$
    If $0 \leftarrow \mathsf{GVf}(gpk, m, \sigma)$ then return 0
    If $i \in \mathsf{HU}$, $gsk[i] \neq \perp$ and $1 \leftarrow \mathsf{Judge}(gpk, i, upk[i],$
    $m, \sigma, \tau)$ then return 1 else 0
```

The advantage of $\mathcal{A}$ is defined as: $\mathsf{Adv}^{nf}_{\mathcal{GS},\mathcal{A}}(\kappa) = \Pr[\mathsf{Exp}^{nf}_{\mathcal{GS},\mathcal{A}}(\kappa) = 1]$.

We say that the dynamic group signature scheme is non-frameable against randomness exposure if $\mathcal{A}$ poses neither $\mathsf{USK}(i)$ nor $\mathsf{GSig}(i, m)$, and $\mathsf{Adv}^{nf}_{\mathcal{GS},\mathcal{A}}(\kappa)$ 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 $usk[i]$ or $gsk[i]$ 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, SndToI, 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 $\mathsf{USK}(i_0)$ and $\mathsf{USK}(i_1)$, and obtain $(gsk[i_0], usk[i_0])$ and $(gsk[i_1], usk[i_1])$.
4. Pose $\mathsf{Ch}_b(i_0, i_1, m)$, and obtain σ.
5. Pose $\mathsf{Rev}(m, \sigma)$, and obtain r.
6. Compute $\sigma_0 \leftarrow \mathsf{GSig}(gpk, gsk[i_0], m; r)$ and $\sigma_1 \leftarrow \mathsf{GSig}(gpk, gsk[i_1], m; r)$.
7. If $\sigma_0 = \sigma$, 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). σ is deterministically generated with GSig on input $(gpk, gsk[i_b], m; r)$. Thus, σ must be identical to either σ_0 or σ_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 κ be a security parameter, which is chosen according to required key length. Let $\mathsf{F} = \{F_\kappa : Dom_\kappa \times Salt_\kappa \rightarrow Rng_\kappa\}_\kappa$ 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 $\mathsf{F} = \{F_\kappa\}_\kappa$ is the pseudo-random function (PRF) family, if for any PPT distinguisher $\mathcal{D}$, $|\Pr[1 \leftarrow \mathcal{D}^{F_\kappa(\cdot)}] - \Pr[1 \leftarrow \mathcal{D}^{RF_\kappa(\cdot)}]|$ is negligible, where $RF_\kappa : Dom_\kappa \rightarrow Rng_\kappa$ is a truly random function.

4.2 Transformation π_{SLAnon} for SLAnon-RE

We give the generic transformation π_{SLAnon} which provides SLAnon-RE to a SLAnon GS scheme. For a GS scheme $\mathcal{GS}$ = (GKg, UKg, Join, Iss, GSig, GVf, Open, Judge), we denote the transformed GS scheme by $\pi_{SLAnon}(\mathcal{GS})$ = (GKg′, UKg′, Join′, Iss′, GSig′, GVf′, Open′, Judge′). We denote randomness used in GSig by $(r_1, \ldots, r_n)$. π_{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 π_{SLAnon} is as follows:

$\mathsf{GKg}'(1^\kappa)$**:** Choose PRFs $(F_1, \ldots, F_n)$, generate $(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^\kappa)$, and set $gpk' = (gpk, F_1, \ldots, F_n)$. Output (gpk', ik, ok).

$\mathsf{UKg}'(1^\kappa)$**:** Choose salts $(k_1, \ldots, k_n)$, generate $(upk[i], usk[i]) \leftarrow \mathsf{UKg}(1^\kappa)$, and set $usk[i]' = (usk[i], k_1, \ldots, k_n)$. Output $(upk[i], usk[i]')$.

$\mathsf{Join}'(St^i_{jn}, M_{jn})$ **and** $\mathsf{Iss}'(St^i_{iss}, M_{iss})$**:** Iss′ is the same as Iss. For Join′, execute $(St^i_{iss}, M_{jn}, dec^i) \leftarrow \mathsf{Iss}(St^i_{iss}, M_{iss})$ and $(St^i_{jn}, M_{iss}, dec^i) \leftarrow \mathsf{Join}(St^i_{jn}, M_{jn})$ until $dec^i = accept$, and set $gsk[i]' = (gsk[i], k_1, \ldots, k_n)$. Output $gsk[i]'$ for Join′ and $reg[i]$ for Iss′.

$\mathsf{GSig}'(gpk', gsk[i]', m; (r'_1, \ldots, r'_n))$**:** Parse gpk' into $(gpk, F_1, \ldots, F_n)$ and $gsk[i]'$ into $(gsk[i], k_1, \ldots, k_n)$, compute $\{r_j\} = \{F_j(r'_j, k_j)\}$ for $1 \leq j \leq n$, and generate $\sigma \leftarrow \mathsf{GSig}(gpk, gsk[i], m; (r_1, \ldots, r_n))$. Output σ.

$\mathsf{GVf}'(gpk, m, \sigma)$**,**$\mathsf{Open}'(gpk, ok, reg, m, \sigma)$ **and** $\mathsf{Judge}'(gpk, i, upk[i], m, \sigma, \tau)$: GVf′, Open′ and Judge′ are the same as GVf, Open and Judge, respectively.

4.3 Security

We show security statements of the generic transformation π_{SLAnon}.

Theorem 4.1 (Correctness of π_{SLAnon}): If $\mathcal{GS}$ satisfies correctness, then $\pi_{SLAnon}(\mathcal{GS})$ does as well.

Correctness is obvious from the protocol.

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

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

Theorem 4.4 (Non-Frameability of π_{SLAnon}): If $\mathcal{GS}$ is non-frameable and $(F_1, \ldots, F_n)$ are PRFs, then $\pi_{SLAnon}(\mathcal{GS})$ 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 $\mathcal{GS}$ and $\pi_{SLAnon}(\mathcal{GS})$ is how to generate a signature. In $\mathcal{GS}$ randomness is directly used, and in $\pi_{SLAnon}(\mathcal{GS})$ 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 $\mathsf{USK}(i_0)$ and $\mathsf{USK}(i_1)$, and cannot obtain salts $(k_1, \ldots, k_n)$ of i_0 and i_1. Thus, outputs of PRFs in $\pi_{SLAnon}(\mathcal{GS})$ can be replaced with random values generated from random functions. In this situation, Rev oracle does not help the adversary because outputs of Rev are independent to random values used in $\pi_{SLAnon}(\mathcal{GS})$. Therefore, SLAnon of $\mathcal{GS}$ implies SLAnon-RE of $\pi_{SLAnon}(\mathcal{GS})$.

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

Finally, we show the intuition of the proof of Theorem 4.4. In the experiment, an adversary cannot pose USK and CrptU for the target user. Hence, the adversary cannot distinguish outputs of PRFs from random values because the adversary cannot obtain salts $(k_1, \ldots, k_n)$ of the target user. Outputs of PRFs in $\pi_{SLAnon}(\mathcal{GS})$ can be replaced with random values generated from random functions. A subtle point is that the adversary may pose 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 Rev oracle implies non-frameability of $\pi_{SLAnon}(\mathcal{GS})$.

4.3.1 Proof of Theorem 4.2

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

Hybrid experiment $\mathbf{H}_0$: This experiment denotes $\mathsf{Exp}^{anon-b}_{\pi_{SLAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$, and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_0)$ is the same as $\mathsf{Adv}^{anon}_{\pi_{SLAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$.

Hybrid experiment $\mathbf{H}_j$ for $1 \le j \le n$: The computation of $\mathsf{GSig}'(gpk, gsk[i]', m; (r'_1, \ldots, r'_n))$ in Ch_b 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 $\mathcal{D}$ to distinguish $\tilde{r}_j$ from $r_j = F_j(r'_j, k_j)$ by assuming an adversary $\mathcal{A}$ distinguish $\mathbf{H}_j$ from $\mathbf{H}_{j-1}$. $\mathcal{D}$ performs as follows:

Setup. Generate $(gpk', ik, ok) \leftarrow \mathsf{GKg}'(1^\kappa)$, and set $\mathsf{CU} \leftarrow \emptyset$, $\mathsf{HU} \leftarrow \emptyset$, $\mathsf{Gset} \leftarrow \emptyset$ and $\sigma\mathsf{set} \leftarrow \emptyset$. Give gpk' and ik to $\mathcal{A}$ as input.

Simulation.

- $\mathsf{Op}(m, \sigma)$**:** If $(m, \sigma) \in \mathsf{Gset}$, then return $\perp$. Otherwise, return $\mathsf{Open}'(gpk', ok, reg, m, \sigma)$.
- $\mathsf{SndToU}(i, M_{in})$**:** If $i \notin \mathsf{HU}$ then $\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$. Generate $(upk[i], usk[i]') \leftarrow \mathsf{UKg}'(1^\kappa)$, and set $gsk[i]' \leftarrow \perp$ and $St^i_{jn} \leftarrow (gpk', upk[i], usk[i]')$. Generate $(St^i_{jn}, M_{out}, dec^i) \leftarrow \mathsf{Join}'(St^i_{jn}, M_{in})$. If $dec^i = accept$, then $gsk[i]' \leftarrow St^i_{jn}$. Otherwise, return (M_{out}, dec^i).
- $\mathsf{WReg}(i, \rho)$**:** Set $reg[i] \leftarrow \rho$.
- $\mathsf{USK}(i)$**:** Return $(gsk[i]', usk[i])$.
- $\mathsf{CrptU}(i, upk)$**:** If $i \in \mathsf{HU} \cup \mathsf{CU}$, then return $\perp$. Set $\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$, $upk[i] \leftarrow upk$, $dec^i \leftarrow cont$, and $St^i_{iss} \leftarrow (gpk', ik, i, upk[i])$.
- $\mathsf{Sig}(i, m)$**:** If $i \notin \mathsf{HU}$, then return $\perp$. If $gsk[i]' = \perp$, then return $\perp$. Choose $r'_1, \ldots, r'_n$ from Dom of PRFs and $\tilde{r}_1, \ldots, \tilde{r}_{j-1}$ from Rng of PRFs, pose r'_j to the PRF oracle (i.e., F_j or RF), and obtain r_j. Generate $\sigma \leftarrow \mathsf{GSig}(gpk, gsk[i_b], m; (\tilde{r}_1, \ldots, \tilde{r}_{j-1}, r_j, F_{j+1}(r'_{j+1}, k_{j+1}), \ldots, F_n(r'_n, k_n)))$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Ch}_b(i_0, i_1, m)$**:** If $i_0 \notin \mathsf{HU}$ or $i_1 \notin \mathsf{HU}$, then return $\perp$. If $gsk[i_0]' = \perp$ or $gsk[i_1]' = \perp$, then return $\perp$. Choose $r'_1, \ldots, r'_n$ from Dom of PRFs and $\tilde{r}_1, \ldots, \tilde{r}_{j-1}$ from Rng of PRFs, pose r'_j to the PRF oracle (i.e., F_j or RF), and obtain r_j. Generate $\sigma \leftarrow \mathsf{GSig}(gpk, gsk[i_b], m; (\tilde{r}_1, \ldots, \tilde{r}_{j-1}, r_j, F_{j+1}(r'_{j+1}, k_{j+1}, \ldots, F_n(r'_n, k_n))$. Set $\mathsf{Gset} \leftarrow \mathsf{Gset} \cup (m, \sigma)$ and $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (i_b, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Rev}(m, \sigma)$**:** If $(i, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. Otherwise, return r.

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

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

Bounding $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_n)$: In $\mathbf{H}_n$, all computations of GSig' in Ch_b oracle use random $(\tilde{r}_1 \ldots, \tilde{r}_n)$ instead of $(F_1(r'_1, k_1), \ldots, F_n(r'_n, k_n))$. Thus, outputs of Rev oracle $(r'_1, \ldots, r'_n)$ are independent from σ.

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

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

Simulation.

- $\mathsf{Op}(m, \sigma)$**:** Pose (m, σ) to Op, and return the output of Op.
- $\mathsf{SndToU}(i, M_{in})$**:** Pose (i, M_{in}) to SndToU, and return the output of SndToU.
- $\mathsf{WReg}(i, \rho)$**:** Pose (i, ρ) to WReg.
- $\mathsf{USK}(i)$**:** 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 $(upk[i], usk[i]')$.
- $\mathsf{CrptU}(i, upk)$**:** Pose (i, upk) to CrptU, and return the output of CrptU.
- $\mathsf{Sig}(i, m)$**:** Pose (i, m) to Sig, obtain σ, and choose $(r'_1, \ldots, r'_n)$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Ch}_b(i_0, i_1, m)$**:** Pose (i_0, i_1, m) to Ch_b, obtain σ, and choose $(r'_1, \ldots, r'_n)$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (*, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Rev}(m, \sigma)$**:** If $(*, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. 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 $\mathbf{H}_n$. Thus, if the advantage of $\mathcal{B}$ is negligible, then $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_n)$ is negligible. □

4.3.2 Proof of Theorem 4.3

The proof is trivial. In $\mathsf{Exp}^{trace}_{\pi_{SLAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$, $\mathcal{A}$ does not have Sig and Ch_b oracles. Thus, there is no record in $\sigma\mathsf{set}$, and Rev oracle always returns $\perp$. It means that Rev oracle is not any help of $\mathcal{A}$. Therefore, even if $\mathcal{GS}$ is traceable without Rev oracle, $\pi_{SLAnon}(\mathcal{GS})$ is also traceable. □

4.3.3 Proof of Theorem 4.4

We change the interface of oracle queries in the experiment of $\mathsf{Exp}^{nf}_{\pi_{SLAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, non-frameability of $\pi_{SLAnon}(\mathcal{GS})$ is guaranteed from non-frameability of $\mathcal{GS}$ without Rev oracle. We denote these hybrid experiments by $\mathbf{H}_0, \ldots, \mathbf{H}_{n+1}$, and the advantage of the adversary $\mathcal{A}$ when participating in experiment $\mathbf{H}_i$ by $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_i)$.

Hybrid experiment $\mathbf{H}_0$: This experiment denotes $\mathsf{Exp}^{nf}_{\pi_{SLAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$, and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_0)$ is the same as $\mathsf{Adv}^{nf}_{\pi_{SLAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$.

Hybrid experiment $\mathbf{H}_1$: The experiment fixes a user i^* which is chosen from $[1, \ldots, N]$, and if $\mathcal{A}$ 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, $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_1) \geq 1/N \cdot \mathsf{Adv}(\mathcal{A}, \mathbf{H}_0)$.

Hybrid experiment $\mathbf{H}_j$ for $2 \leq j \leq n+1$: The computation of $\mathsf{GSig}'(gpk, gsk[i]', m; (r'_1, \ldots, r'_n))$ in Sig oracle is changed. Instead of computing $r_{j-1} = F_{j-1}(r'_{j-1}, k_{j-1})$, it is changed as obtaining $\tilde{r}_{j-1}$ from a random function RF.

We construct a distinguisher $\mathcal{D}$ to distinguish $\tilde{r}_{j-1}$ from $r_{j-1} = F_{j-1}(r'_{j-1}, k_{j-1})$ by assuming an adversary $\mathcal{A}$ distinguish $\mathbf{H}_j$ from $\mathbf{H}_{j-1}$. $\mathcal{D}$ performs as follows:

Setup. Generate $(gpk', ik, ok) \leftarrow \mathsf{GKg}'(1^\kappa)$, and set $\mathsf{CU} \leftarrow \emptyset$ and $\mathsf{HU} \leftarrow \emptyset$, $\mathsf{Gset} \leftarrow \emptyset$ and $\sigma\mathsf{set} \leftarrow \emptyset$. Give (gpk', ik, ok) to $\mathcal{A}$ as input.

Simulation.

- $\mathsf{SndToU}(i, M_{in})$**:** If $i \notin \mathsf{HU}$ then $\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$. Generate $(upk[i], usk[i]') \leftarrow \mathsf{UKg}'(1^\kappa)$, and set $gsk[i]' \leftarrow \perp$ and $St^i_{jn} \leftarrow (gpk', upk[i], usk[i]')$. Generate $(St^i_{jn}, M_{out}, dec^i) \leftarrow \mathsf{Join}'(St^i_{jn}, M_{in})$. If $dec^i = accept$, then $gsk[i]' \leftarrow St^i_{jn}$. Otherwise, return (M_{out}, dec^i).
- $\mathsf{WReg}(i, \rho)$**:** Set $reg[i] \leftarrow \rho$.
- $\mathsf{USK}(i)$**:** Return $(gsk[i]', usk[i])$.
- $\mathsf{CrptU}(i, upk)$**:** If $i \in \mathsf{HU} \cup \mathsf{CU}$, then return $\perp$. Set $\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$, $upk[i] \leftarrow upk$, $dec^i \leftarrow cont$, and $St^i_{iss} \leftarrow (gpk', ik, i, upk[i])$.
- $\mathsf{Sig}(i, m)$**:** If $i \notin \mathsf{HU}$, then return $\perp$. If $gsk[i]' = \perp$, then return $\perp$. If $i = i^*$, then choose $r'_1, \ldots, r'_n$ from Dom of PRFs and $\tilde{r}_1, \ldots, \tilde{r}_{j-2}$ 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 $\sigma \leftarrow \mathsf{GSig}(gpk', gsk[i], m; (\tilde{r}_1, \ldots, \tilde{r}_{j-2}, r_{j-1}, F_j(r'_j, k_j), \ldots, F_n(r'_n, k_n)))$. Otherwise, then choose $r'_1, \ldots, r'_n$, and generate $\sigma \leftarrow \mathsf{GSig}'(gpk', gsk[i]', m; (r'_1, \ldots, r'_n))$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Rev}(m, \sigma)$**:** If $(i, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. Otherwise, return r.

Output. If $\mathcal{A}$ outputs $b' = b$, then $\mathcal{D}$ outputs 1. Otherwise, $\mathcal{D}$ 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 $\mathsf{USK}(i^*)$ nor $\mathsf{CrptU}(i^*)$, $\mathcal{A}$ cannot know salts of i^*. For $\mathcal{A}$, the simulation by $\mathcal{D}$ is same as the experiment $\mathbf{H}_{j-1}$ if the PRF oracle is the PRF F_{j-1}. Otherwise, the simulation by $\mathcal{D}$

is same as the experiment $\mathbf{H}_j$. Thus, if the advantage of $\mathcal{D}$ is negligible, then $|\mathsf{Adv}(\mathcal{A}, \mathbf{H}_j) - \mathsf{Adv}(\mathcal{A}, \mathbf{H}_{j-1})|$ is negligible.

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

We construct an adversary $\mathcal{B}$ for $\mathcal{GS}$ by assuming an adversary $\mathcal{A}$ for $\mathbf{H}_{n+1}$. $\mathcal{B}$ 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 $\sigma\mathsf{set} \leftarrow \emptyset$. Give (gpk', ik, ok) to $\mathcal{A}$ as input.

Simulation.

- $\mathsf{SndToU}(i, M_{in})$**:** Pose (i, M_{in}) to SndToU, and return the output of SndToU.
- $\mathsf{WReg}(i, \rho)$**:** Pose (i, ρ) to WReg.
- $\mathsf{USK}(i)$**:** 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 $(upk[i], usk[i]')$.
- $\mathsf{CrptU}(i, upk)$**:** Pose (i, upk) to CrptU, and return the output of CrptU.
- $\mathsf{Sig}(i, m)$**:** Pose (i, m) to Sig, obtain σ, and choose $(r'_1, \ldots, r'_n)$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Rev}(m, \sigma)$**:** If $(*, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. 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 $\mathbf{H}_{n+1}$. Thus, if the advantage of $\mathcal{B}$ is negligible, then $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_{n+1})$ 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 π_{FAnon} which provides FAnon-RE to a FAnon GS scheme. For a GS scheme $\mathcal{GS}$ = $(\mathsf{GKg}, \mathsf{UKg}, \mathsf{Join}, \mathsf{Iss}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$, we denote the transformed GS scheme by $\pi_{FAnon}(\mathcal{GS})$ = $(\mathsf{GKg}', \mathsf{UKg}', \mathsf{Join}', \mathsf{Iss}', \mathsf{GSig}', \mathsf{GVf}', \mathsf{Open}', \mathsf{Judge}')$. We denote randomness used in GSig by $(r_1, \ldots, r_n)$ (i.e., GSig internally generates n random values). π_{FAnon} uses PRFs $(F_1, \ldots, F_n)$ and the SC SC, where Rng of F_j depend on the space of r_j[†]. The protocol of π_{FAnon} is as follows:

[†]If some randomness are chosen from the common space, then we can use the common PRF for such randomness.

$\mathsf{GKg}'(1^\kappa)$**:** Choose PRFs $(F_1, \ldots, F_n)$, generate $(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^\kappa)$, and set $gpk' = (gpk, F_1, \ldots, F_n)$. Output (gpk', ik, ok).

$\mathsf{UKg}'(1^\kappa)$**:** Choose salts $(k_1, \ldots, k_n)$, generate $(upk[i], usk[i]) \leftarrow \mathsf{UKg}(1^\kappa)$, and set $usk[i]' = (usk[i], k_1, \ldots, k_n)$. Output $(upk[i], usk[i]')$.

$\mathsf{Join}'(St^i_{jn}, M_{jn})$ **and** $\mathsf{Iss}'(St^i_{iss}, M_{iss})$**:** Iss' is the same as Iss. For Join', execute $(St^i_{iss}, M_{jn}, dec^i) \leftarrow \mathsf{Iss}(St^i_{iss}, M_{iss})$ and $(St^i_{jn}, M_{iss}, dec^i) \leftarrow \mathsf{Join}(St^i_{jn}, M_{jn})$ until $dec^i = accept$, store $(k_1, \ldots, k_n)$ to SC, and set $gsk[i]' = (gsk[i], SC)$. Output $gsk[i]'$ for Join' and $reg[i]$ for Iss'.

$\mathsf{GSig}'(gpk', gsk[i]', m; (r'_1, \ldots, r'_n))$**:** Parse gpk' into $(gpk, F_1, \ldots, F_n)$ and $gsk[i]'$ into $(gsk[i], SC)$, extract $(k_1, \ldots, k_n)$ from SC, compute $\{r_j\} = \{F_j(r'_j, k_j)\}$ for $1 \leq j \leq n$, and generate $\sigma \leftarrow \mathsf{GSig}(gpk, gsk[i], m; (r_1, \ldots, r_n))$. Output σ.

$\mathsf{GVf}'(gpk', m, \sigma)$**,** $\mathsf{Open}'(gpk', ok, reg, m, \sigma)$ **and** $\mathsf{Judge}'(gpk', i, upk[i], m, \sigma, \tau)$: GVf', Open' and Judge' are the same as GVf, Open and Judge, respectively.

5.1 Security

We show security statements of the generic transformation π_{FAnon}.

Theorem 5.1 (Correctness of π_{FAnon}): If $\mathcal{GS}$ satisfies correctness, then $\pi_{FAnon}(\mathcal{GS})$ does as well.

Correctness is obvious from the protocol.

Theorem 5.2 (FAnon-RE of π_{FAnon}): If $\mathcal{GS}$ is FAnon and $(F_1, \ldots, F_n)$ are PRFs, then $\pi_{FAnon}(\mathcal{GS})$ is FAnon-RE.

Theorem 5.3 (Traceability of π_{FAnon}): If $\mathcal{GS}$ is traceable and $(F_1, \ldots, F_n)$ are PRFs, then $\pi_{FAnon}(\mathcal{GS})$ is traceable against randomness exposure.

Theorem 5.4 (Non-Frameability of π_{FAnon}): If $\mathcal{GS}$ is non-frameable and $(F_1, \ldots, F_n)$ are PRFs, then $\pi_{FAnon}(\mathcal{GS})$ is non-frameable against randomness exposure.

Proofs of Theorem 5.2, 5.3 and 5.4 are shown in Appendix A. Here, we give sketches of proofs.

First, we show the intuition of the proof of Theorem 5.2. The difference between $\mathcal{GS}$ and $\pi_{FAnon}(\mathcal{GS})$ is how to generate a signature. In $\mathcal{GS}$ randomness is directly used, and in $\pi_{FAnon}(\mathcal{GS})$ 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 obtain salts $(k_1, \ldots, k_n)$ from SC even when USK is posed. Thus, outputs of PRFs in $\pi_{FAnon}(\mathcal{GS})$ can be replaced with random values generated from random functions. In this situation, Rev oracle does not help the adversary because outputs of Rev are independent to random values used in $\pi_{FAnon}(\mathcal{GS})$. Therefore, FAnon of $\mathcal{GS}$ implies FAnon-RE of $\pi_{FAnon}(\mathcal{GS})$.

Next, we show the intuition of the proof of Theorem 5.3. In the experiment, an adversary does not have the access to Sig and Ch_b. Thus, Rev oracle never returns other than $\perp$, and is not any help of the adversary. Therefore, traceability of $\mathcal{GS}$ without Rev oracle naturally implies traceability of $\pi_{FAnon}(\mathcal{GS})$.

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 $\pi_{FAnon}(\mathcal{GS})$ can be replaced with random values generated from random functions. Therefore, non-frameability of $\mathcal{GS}$ without Rev oracle implies non-frameability of $\pi_{FAnon}(\mathcal{GS})$.

References

[1] D. Chaum and E. van Heyst, "Group Signatures," EUROCRYPT1991, vol.547, pp.257–265, 1991.

[2] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions," EUROCRYPT2003, vol.2656, pp.614–629, 2003.

[3] M. Bellare, H. Shi, and C. Zhang, "Foundations of Group Signatures: The Case of Dynamic Groups," CT-RSA 2005, vol.3376, pp.136–153, 2005.

[4] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," ACM Conference on Computer and Communications Security 2004, pp.168–177, 2004.

[5] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures," CRYPTO 2004, vol.3152, pp.41–55, 2004.

[6] X. Boyen and B. Waters, "Compact Group Signatures Without Random Oracles," EUROCRYPT 2006, vol.4004, pp.427–444, 2006.

[7] X. Boyen and B. Waters, "Full-Domain Subgroup Hiding and Constant-Size Group Signatures," Public Key Cryptography 2007, pp.1–15, 2007.

[8] L. Nguyen and R. Safavi-Naini, "Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings," ASIACRYPT 2004, vol.3329, pp.372–386, 2004.

[9] C. Delerablée and D. Pointcheval, "Dynamic Fully Anonymous Short Group Signatures," VIETCRYPT 2006, vol.4341, pp.193–210, 2006.

[10] J. Groth, "Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures," ASIACRYPT 2006, vol.4284, pp.444–459, 2006.

[11] J. Groth, "Fully Anonymous Group Signatures Without Random Oracles," ASIACRYPT 2007, pp.164–180, 2007.

[12] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo, "Structure-Preserving Signatures and Commitments to Group Elements," CRYPTO 2010, vol.6223, pp.209–236, 2010.

[13] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo, "Constant-Size Structure-Preserving Signatures: Generic Constructions and Simple Assumptions," ASIACRYPT 2012, vol.7658, pp.4–24, 2012.

[14] T. Nakanishi and N. Funabiki, "Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps," ASIACRYPT 2005, vol.3788, pp.533–548, 2005.

[15] T. Nakanishi and N. Funabiki, "A Short Verifier-Local Revocation Group Signature Scheme with Backward Unlinkability," IWSEC 2006, vol.4266, pp.17–32, 2006.

[16] B. Libert and D. Vergnaud, "Group Signatures with Verifier-Local Revocation and Backward Unlinkability in the Standard Model," CANS 2009, vol.5888, pp.498–517, 2009.

[17] P. Bichsel, J. Camenisch, G. Neven, N.P. Smart, and B. Warinschi, "Get Shorty via Group Signatures without Encryption," SCN 2010, vol.6280, pp.381–398, 2010.

[18] A. Langlois, S. Ling, K. Nguyen, and H. Wang, "Lattice-Based Group Signature Scheme with Verifier-Local Revocation," Public Key Cryptography 2014, vol.8383, pp.345–361, 2014.

[19] "Debian – Security Information – DSA-1571-1 openssl." https://www.debian.org/security/2008/dsa-1571.

[20] S. Faust, E. Kiltz, K. Pietrzak, and G. Rothblum, "Leakage-Resilient Signatures," TCC 2010, 2010. http://eprint.iacr.org/2009/142/.

[21] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek, "Hedged Public-Key Encryption: How to Protect against Bad Randomness," ASIACRYPT, vol.5912, pp.232–249, 2009.

[22] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger Security of Authenticated Key Exchange," ProvSec 2007, pp.1–16, 2007.

Appendix A: Security Proofs for π_{FAnon}

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 $\mathsf{Exp}^{anon-b}_{\pi_{FAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, FAnon-RE of $\pi_{FAnon}(\mathcal{GS})$ is guaranteed from FAnon of $\mathcal{GS}$. We denote these hybrid experiments by $\mathbf{H}_0, \ldots, \mathbf{H}_n$, and the advantage of the adversary $\mathcal{A}$ when participating in experiment $\mathbf{H}_i$ by $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_i)$.

Hybrid experiment $\mathbf{H}_0$: This experiment denotes $\mathsf{Exp}^{anon-b}_{\pi_{FAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$, and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_0)$ is the same as $\mathsf{Adv}^{anon}_{\pi_{FAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$.

Hybrid experiment $\mathbf{H}_j$ for $1 \leq j \leq n$: The computation of $\mathsf{GSig}'(gpk, gsk[i]', m; (r'_1, \ldots, r'_n))$ in Ch_b 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 $\mathcal{D}$ to distinguish $\tilde{r}_j$ from $r_j = F_j(r'_j, k_j)$ by assuming an adversary $\mathcal{A}$ distinguish $\mathbf{H}_j$ from $\mathbf{H}_{j-1}$. $\mathcal{D}$ performs as follows:

Setup. Generate $(gpk', ik, ok) \leftarrow \mathsf{GKg}'(1^\kappa)$, and set $\mathsf{CU} \leftarrow \emptyset$, $\mathsf{HU} \leftarrow \emptyset$, $\mathsf{Gset} \leftarrow \emptyset$ and $\sigma\mathsf{set} \leftarrow \emptyset$. Give gpk' and ik to $\mathcal{A}$ as input.

Simulation.

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

- $\mathsf{CrptU}(i, upk)$**:** If $i \in \mathsf{HU} \cup \mathsf{CU}$, then return $\perp$. Set $\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$, $upk[i] \leftarrow upk$, $dec^i \leftarrow cont$, and $St^i_{iss} \leftarrow (gpk', ik, i, upk[i])$.

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

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

- $\mathsf{Rev}(m, \sigma)$**:** If $(i, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. Otherwise, return r.

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

Analysis. If $\mathcal{A}$ can check if randomness used in Ch_b oracle is outputs of PRFs, then $\mathcal{A}$ can distinguish the simulation from the real experiment. However, since salts of i_0 and i_1 are stored in the SC, $\mathcal{A}$ cannot know salts even if $\mathsf{USK}(i_0)$ and $\mathsf{USK}(i_1)$ are posed. For $\mathcal{A}$, the simulation by $\mathcal{D}$ is same as the experiment $\mathbf{H}_{j-1}$ if the PRF oracle is the PRF F_j. Otherwise, the simulation by $\mathcal{D}$ is same as the experiment $\mathbf{H}_j$. Thus, if the advantage of $\mathcal{D}$ is negligible, then $|\mathsf{Adv}(\mathcal{A}, \mathbf{H}_j) - \mathsf{Adv}(\mathcal{A}, \mathbf{H}_{j-1})|$ is negligible.

Bounding $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_n)$**:** In $\mathbf{H}_n$, all computations of GSig' in Ch_b oracle use random $(\tilde{r}_1 \ldots, \tilde{r}_n)$ instead of $(F_1(r'_1, k_1), \ldots, F_n(r'_n, k_n))$. Thus, outputs of Rev oracle $(r'_1, \ldots, r'_n)$ are independent from σ.

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

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

Simulation.

- $\mathsf{Op}(m, \sigma)$**:** Pose (m, σ) to Op, and return the output of Op.

- $\mathsf{SndToU}(i, M_{in})$**:** Pose (i, M_{in}) to SndToU, and return the output of SndToU.

- $\mathsf{WReg}(i, \rho)$**:** Pose (i, ρ) to WReg.

- $\mathsf{USK}(i)$**:** 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 $(upk[i], usk[i]')$.

- $\mathsf{CrptU}(i, upk)$**:** Pose (i, upk) to CrptU, and return the output of CrptU.

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

- $\mathsf{Ch}_b(i_0, i_1, m)$**:** Pose (i_0, i_1, m) to Ch_b, obtain σ, and choose $(r'_1, \ldots, r'_n)$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (*, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.

- $\mathsf{Rev}(m, \sigma)$**:** If $(*, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. 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 $\mathbf{H}_n$. Thus, if the advantage of $\mathcal{B}$ is negligible, then $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_n)$ is negligible. □

A.2 Proof of Theorem 5.3

The proof is trivial. In $\mathsf{Exp}^{trace}_{\pi_{FAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$, $\mathcal{A}$ does not have Sig and Ch_b oracles. Thus, there is no record in $\sigma\mathsf{set}$, and Rev oracle always returns $\perp$. It means that Rev oracle is not any help of $\mathcal{A}$. Therefore, even if $\mathcal{GS}$ is traceable without Rev oracle, $\pi_{FAnon}(\mathcal{GS})$ is also traceable. □

A.3 Proof of Theorem 5.4

We change the interface of oracle queries in the experiment of $\mathsf{Exp}^{nf}_{\pi_{FAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$. These instances are gradually changed over hybrid experiments, depending on specific sub-cases. In the last hybrid experiment, non-frameability of $\pi_{FAnon}(\mathcal{GS})$ is guaranteed from non-frameability of $\mathcal{GS}$ without Rev oracle. We denote these hybrid experiments by $\mathbf{H}_0, \ldots, \mathbf{H}_n$, and the advantage of the adversary $\mathcal{A}$ when participating in experiment $\mathbf{H}_i$ by $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_i)$.

Hybrid experiment $\mathbf{H}_0$: This experiment denotes $\mathsf{Exp}^{nf}_{\pi_{FAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$, and in this experiment the environment for $\mathcal{A}$ is as defined in the protocol. Thus, $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_0)$ is the same as $\mathsf{Adv}^{nf}_{\pi_{FAnon}(\mathcal{GS}),\mathcal{A}}(\kappa)$.

Hybrid experiment $\mathbf{H}_j$ for $1 \le j \le n$: The computation of $\mathsf{GSig}'(gpk, gsk[i]', m; (r'_1, \ldots, r'_n))$ in 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 $\mathcal{D}$ to distinguish $\tilde{r}_j$ from $r_j = F_j(r'_j, k_j)$ by assuming an adversary $\mathcal{A}$ distinguish $\mathbf{H}_j$ from $\mathbf{H}_{j-1}$. $\mathcal{D}$ performs as follows:

Setup. Generate $(gpk', ik, ok) \leftarrow \mathsf{GKg}'(1^\kappa)$, and set $\mathsf{CU} \leftarrow$

$\emptyset$ and $\mathsf{HU} \leftarrow \emptyset$, $\mathsf{Gset} \leftarrow \emptyset$ and $\sigma\mathsf{set} \leftarrow \emptyset$. Give (gpk', ik, ok) to $\mathcal{A}$ as input.

Simulation.

- $\mathsf{SndToU}(i, M_{in})$**:** If $i \notin \mathsf{HU}$ then $\mathsf{HU} \leftarrow \mathsf{HU} \cup \{i\}$. Generate $(upk[i], usk[i]') \leftarrow \mathsf{UKg}'(1^\kappa)$, and set $gsk[i]' \leftarrow \perp$ and $St^i_{jn} \leftarrow (gpk', upk[i], usk[i]')$. Generate $(St^i_{jn}, M_{out}, dec^i) \leftarrow \mathsf{Join}'(St^i_{jn}, M_{in})$. If $dec^i = accept$, then $gsk[i]' \leftarrow St^i_{jn}$. Otherwise, return (M_{out}, dec^i).
- $\mathsf{WReg}(i, \rho)$**:** Set $reg[i] \leftarrow \rho$.
- $\mathsf{USK}(i)$**:** Return $(gsk[i]', usk[i])$.
- $\mathsf{CrptU}(i, upk)$**:** If $i \in \mathsf{HU} \cup \mathsf{CU}$, then return $\perp$. Set $\mathsf{CU} \leftarrow \mathsf{CU} \cup \{i\}$, $upk[i] \leftarrow upk$, $dec^i \leftarrow cont$, and $St^i_{iss} \leftarrow (gpk', ik, i, upk[i])$.
- $\mathsf{Sig}(i, m)$**:** If $i \notin \mathsf{HU}$, then return $\perp$. If $gsk[i]' = \perp$, then return $\perp$. 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 or RF), obtain r_j, and generate $\sigma \leftarrow \mathsf{GSig}(gpk, gsk[i], m; (\tilde{r}_1, \ldots, \tilde{r}_{j-1}, r_j, F_{j+1}(r'_{j+1}, k_{j+1}), \ldots, F_n(r'_n, k_n)))$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Rev}(m, \sigma)$**:** If $(i, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. Otherwise, return r.

Output. If $\mathcal{A}$ outputs $b' = b$, then $\mathcal{D}$ outputs 1. Otherwise, $\mathcal{D}$ 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 salts of users are stored in their SCs, $\mathcal{A}$ cannot know salts even if $\mathsf{USK}(i)$ is posed. For $\mathcal{A}$, the simulation by $\mathcal{D}$ is same as the experiment $\mathbf{H}_{j-1}$ if the PRF oracle is the PRF F_j. Otherwise, the simulation by $\mathcal{D}$ is same as the experiment $\mathbf{H}_j$. Thus, if the advantage of $\mathcal{D}$ is negligible, then $|\mathsf{Adv}(\mathcal{A}, \mathbf{H}_j) - \mathsf{Adv}(\mathcal{A}, \mathbf{H}_{j-1})|$ is negligible.

Bounding $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_n)$**:** In $\mathbf{H}_n$, all computations of GSig' in Sig oracle use random $(\tilde{r}_1 \ldots, \tilde{r}_n)$ instead of $(F_1(r'_1, k_1), \ldots, F_n(r'_n, k_n))$. Thus, outputs of Rev oracle $(r'_1, \ldots, r'_n)$ are independent from σ.

We construct an adversary $\mathcal{B}$ for $\mathcal{GS}$ by assuming an adversary $\mathcal{A}$ for $\mathbf{H}_n$. $\mathcal{B}$ 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 $\sigma\mathsf{set} \leftarrow \emptyset$. Give (gpk', ik, ok) to $\mathcal{A}$ as input.

Simulation.

- $\mathsf{SndToU}(i, M_{in})$**:** Pose (i, M_{in}) to SndToU, and return the output of SndToU.
- $\mathsf{WReg}(i, \rho)$**:** Pose (i, ρ) to WReg.
- $\mathsf{USK}(i)$**:** 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 $(upk[i], usk[i]')$.
- $\mathsf{CrptU}(i, upk)$**:** Pose (i, upk) to CrptU, and return the output of CrptU.
- $\mathsf{Sig}(i, m)$**:** Pose (i, m) to Sig, obtain σ, and choose $(r'_1, \ldots, r'_n)$. Set $\sigma\mathsf{set} \leftarrow \sigma\mathsf{set} \cup (i, m, \sigma, (r'_1, \ldots, r'_n))$. Return σ.
- $\mathsf{Rev}(m, \sigma)$**:** If $(i, m, \sigma, r) \notin \sigma\mathsf{set}$, then return $\perp$. 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 $\mathbf{H}_n$. Thus, if the advantage of $\mathcal{B}$ is negligible, then $\mathsf{Adv}(\mathcal{A}, \mathbf{H}_n)$ 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.