Almost-Fully Secured Fully Dynamic Group Signatures with Efficient Verifier-local Revocation and Time-bound keys

Maharage Nisansala Sevwandi Perera[1] and Takeshi Koshiba[2]

[1] Graduate School of Science and Engineering
Saitama University, Saitama, Japan
perera.m.n.s.119@ms.saitama-u.ac.jp,
[2] Faculty of Education and Integrated Arts and Sciences
Waseda University, Tokyo, Japan
tkoshiba@waseda.jp

Abstract. One of the prominent requirements in group signature schemes is revoking group members who are misbehaved or resigned. Among the revocation approaches Verifier-local revocation (VLR) is more convenient than others because VLR requires updating only the verifiers with revocation messages. Accordingly, at the signature verification, the verifiers check whether the signer is not in the given revocation detail list. However, the cost of the revocation check increases linearly with the size of the revocation details. Moreover, original VLR group signature schemes rely on a weaker security notion. Achieving both efficient member revocation and reliably strong security for a group signature scheme is technically a challenge. This paper suggests a fully dynamic group signature scheme that performs an efficient member revocation with VLR and which is much more secure than the original VLR schemes.

Keywords: group signatures, verifier-local revocation, member registration, almost-full anonymity, dynamical-almost-full anonymity, time-bound keys

1 Introduction

Group signatures, introduced by Chaum and van Heyst [4] grant group members to sign messages on behalf of the group such that the resulting signature will not reveal the identity of the signer (anonymity). However, anonymity may open paths to offenses. Thus in case of dispute, the tracing authority can identify the misbehaved members (traceability). These two key features of the group signatures attracted real-life applications such as e-commerce systems, digital right management, and key-card access.

In many settings, it is desirable to offer flexibility for members to join the group or leave the group as they wish. There are several models for revoking member's signing capability. When a member is removed, the group manager

can issue a new group public key and give each member a new signing key, except to revoked member. This approach does not suit in practice well because it requires generating new keys and updating each member and verifier for each member revocation. Another method is when a member is revoked, distributing a message to all the existing signers. Thus signers have to prove his validity at the time of signing. However, this approach also cannot count as a suitable revocation method in real-life applications since each existing members have to keep track of revocation messages. Verifier-local revocation (VLR) suggested by Brickell [3] and formalized by Boneh et al. [2], also used in the schemes [1], [5], [6], and [8] seems to be the most flexible revocation approach because VLR requires to update only the verifiers when a member is revoked. Since the number of the verifiers is less than the number of members in a group, VLR is convenient than any other revocation approach.

In VLR group signature schemes, an additional argument called the revocation list (RL) is given to the algorithm Verify. Each member has another secret key called 'revocation token' other than the secret signing key. When a member is revoked, his revocation token is added to RL. At the verification stage, the verifier authenticates the signer against the latest RL. Thus, the algorithm Verfiy in the group signatures with VLR consists of two steps; Signature-check and Revocation-check. The signature-check verifies the signature is generated by a group member and the signature is generated on the given message. The revocation-check verifies whether the signer has not been revoked.

In some applications of group signatures with VLR, like roaming authenticity of a telecommunication company and visitors authentication in a hotel the size of the revocation list (RL) may increase in a short time. Moreover, any system that provides members to join the group for a short period leads to increase the size of RL, and this may decrease the efficiency of the revocation-check in Verify. In 2012, Chu et al. [5] suggested time-bound keys to group signature schemes with VLR to obtain more efficient group signature schemes. In their scheme, they have categorized the member revocation into two, namely, "natural-revocation" and "premature-revocation". If a member is revoking because of his expiration date (retirement / leaving date) is passed, then it is a natural revocation. If any member is revoking before the expiration date, then it is a premature-revocation. For instance, members who are retiring because their contract period is finished are natural-revoking members, and members who are eliminating because of their misbehavior are premature-revoking members. The technique in [5] is to get rid of the natural-revocation members to shorten the size of RL. Even though the scheme in [5] provides techniques to reduce the size of RL and speed up the revocation-check in signature verification, still it relies on a weaker security notion as most of the VLR group signature schemes.

The original VLR group signature schemes rely on *selfless-anonymity* which is a weaker security notion. Scheme in [8] suggested a security notion called, *almost-full anonymity* for VLR group signature schemes, which is much more stronger than the selfless-anonymity. Again another strong security notion called, *dynamical-almost-full anonymity* was proposed with a lattice-based fully dy-

namic group signature scheme with VLR [7]. While the almost-full anonymity is for group signature schemes only with member revocation with VLR, the dynamical-almost-full anonymity is for fully dynamic group signature schemes which satisfy both member registration and member revocation with VLR. However, cost of the verification checks in both of the schemes increases when we apply them to rapidly changing groups, where the members are joining the group for a short time.

Since all the schemes discussed above failed to provide a solution to minimize the verification cost of VLR fully dynamic group signature schemes while relying on strong security, this paper focuses on delivering a scheme with solutions for that matter.

Our Contribution

In this paper, we address two weaknesses of VLR group signature schemes. One problem is that the most of the existing VLR schemes are relying on the weak security notion, the selfless-anonymity. The next issue is when members are joining a group for a short time, the size of the revoked member detail list (RL) increases fast. Even some solutions are suggested separately for these problems in previous schemes they have not discussed the outcome when those results are added together. Thus, in this work, we improve the scheme given in [8] by proposing time-bound keys to reduce the size of RL, and we apply the security notion, the dynamical-almost-full anonymity to make our new scheme's security strong. We modify the methods of the scheme given in [8] to deliver a much more efficient fully dynamic VLR group signature scheme which supports both member registration and revocation and which relies on stronger security than original VLR schemes.

2 Preliminaries

In this section, we provide some notations that we use in this paper and the primitives with which we use in our scheme. Thus, we describe the time-bound keys and dynamical-almost-full anonymity. Then we define the three building blocks that used in [1] and [8] and which we use to construct the scheme.

2.1 Notations

We denote by λ the security parameter of the scheme and let $\mathbb{N} = \{1, 2, 3, ...\}$ be the set of *positive integers*. For any $k \geq 1 \in \mathbb{N}$, we denote by $[k]$ the set of integers $\{1, ..., k\}$. An empty string is denoted by ε . If s is a string, then $|s|$ indicates the length of the string and if $\mathcal{S}$ is a set, then $|\mathcal{S}|$ denotes the size of the set. If $\mathcal{S}$ is a finite set, $b \xleftarrow{\$} \mathcal{S}$ denotes that b is chosen uniformly at random from $\mathcal{S}$. We denote experiments by **Exp**.

2.2 Time-bound keys

Chu et al. [5] proposed time-bound keys as a solution for reducing the cost of revocation check. In their scheme, the group member's key has an expiration date. When a new user joins the group, the group manager selects an expiration date for the new user. Thus only the members having non-expired keys can create signatures. It leads to cut down the cost of checking the naturally revoked members (members with expired keys) at the signature verification. Since naturally revoked members cannot sign messages, the group manager does not need to add their details to the revocation list RL. Thus, the verifiers check whether the signers are being revoked prematurely. The technique used in [5] shows significant reduction of the revocation check because the number of naturally revoked members is higher than the number of prematurely revoked members in most of the real settings. However, the group member keys have an additional attribute, the expiration date. Thus, other than the signature generation, the signers should prove that their keys are not expired.

To efficiently compare two dates, Chu et al. [5] have used the date format as "YYMMDD" in integer form. For instance, the date 2018 June 20th is indicated as "180620". Accordingly, $t_1 > t_2$ represents that t_1 date is later than the date t_2. According to their scheme, the group manager selects the key expiration date t_r. Moreover, at the time of signing the signer can select a signature expiration date t_s which should satisfy $t_r > t_s$ and $t_s > t_v$, where t_v is the verification date. The requirement of the valid signature is satisfying $t_r > t_s > t_v$.

A VLR group signature scheme with time-bound keys is as follows.

- $\mathsf{KeyGen}(1^\lambda, \ell)$: This key generation algorithm takes as inputs the security parameter λ and the maximum length ℓ of the defined date format, and it outputs a group public key $\mathbf{gpk}$ and a group master key $\mathbf{gmsk}$.
- Join: This is the interactive protocol between the group manager GM and the new users who want to join the group. The user i takes as inputs $\mathbf{gpk}$ while GM takes as inputs $\mathbf{gpk}$, $\mathbf{gmsk}$, and t_{r_i}, where t_{r_i} is the key expiration date for the user i. Finally, GM outputs a revocation token $\mathbf{grt}[i]$ for the user i, and the user i outputs a secret signing key $\mathbf{gsk}[i]$.
- $\mathsf{Sign}(\mathbf{gpk}, t_s, \mathbf{gsk}[i], M)$: This algorithm takes as inputs the group public key $\mathbf{gpk}$, the signature expiration date t_s, the secret signing key $\mathbf{gsk}[i]$ of the user i, and a message M. It creates and outputs a signature Σ on M if $t_s < t_r$ (t_r is included in $\mathbf{gsk}[i]$). It outputs $\perp$ if this condition fails.
- $\mathsf{Verify}(\mathbf{gpk}, t_v, RL, M, \Sigma)$: This algorithm takes as inputs $\mathbf{gpk}$, the current date t_v, the revocation list RL, and a message-signature pair (M, Σ). It outputs valid if Σ is a valid signature on M and the signer is not a revoked member. It outputs $\mathsf{invalid}$ otherwise.
- $\mathsf{Open}(\mathbf{gpk}, \Sigma, M, \{\mathbf{grt}_i\})$: This algorithm takes as inputs the group public key $\mathbf{gpk}$, a signature Σ, a message M, and the revocation tokens $\{\mathbf{grt}_i\}$ for all users, and it outputs the signer's index of Σ or $\perp$.

2.3 Dynamical-almost-full anonymity

The security notion, the dynamical-almost-full anonymity is for fully dynamic VLR group signature schemes that serves both member registration and revocation. In an anonymity game between an adversary A and a challenger C, the dynamical-almost-full anonymity allows the adversary to add new members to the group. If the new user details are valid, then the challenger stores new user's index in a list called, **HUL**. **HUL** consists of indices of the members that A added. Even the revocation token is generated for the new user, the revocation token is not given to A at the time of member registering. The adversary can request revocation tokens of any member using the revocation query. If the indices used for requesting revocation tokens are not used for generating the challenging signature, then C returns revocation tokens. At the challenging phase, the signature is only generated for the indices that are in **HUL** and that are not used to request revocation tokens. Moreover, A can access opening oracle with any message-signature pair except one used in the challenging phase.

The dynamical-almost-full anonymity game between a challenger and an adversary is as follow.

- **Initial Phase:** The challenger C executes the key generation algorithm KeyGen to obtain a group public key **gpk**, authorities' secret keys (**ik**,**ok**). Then C gives **gpk** and existing group members' secret signing keys **gsk** to the adversary A, and creates a new list **HUL**.
- **Query Phase:** The adversary A can add new users, request revocation tokens of the user, and he can ask to open signatures. If A adds new valid users to the group, then the challenger C adds the new user index to **HUL** and responses with a success message without delivering the revocation tokens. If A requests to reveal revocation tokens of a user, then C returns the revocation tokens. If C accesses the opening oracle with a valid message-signature pair, then A returns the index of the signer of the signature.
- **Challenge Phase:** The adversary A outputs a message M^* and two distinct identities i_0, i_1. If i_0, i_1 are added by the adversary ($i_0, i_1 \in$ **HUL**) and if i_0, i_1 are not used to request revocation tokens, then the challenger C selects a bit $b \xleftarrow{\$} \{0,1\}$, generates $\Sigma^* = \mathsf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i_b], M^*)$, and sends Σ^* to A. A still can query the opening oracle except for Σ^*, and A can query revocation tokens except using i_0, i_1. However, A can add new users to the group without any restrictions.
- **Guessing Phase:** Finally, A outputs a bit b', the guess of b. If $b' = b$, then A wins.

2.4 Digital Signature Schemes

A digital signature scheme $\mathsf{DS}=(\mathsf{K}_s, \mathsf{Sig}, \mathsf{Vf})$ consists of key generation K_s, signing Sig, and verification Vf algorithms. DS should satisfy the standard notion of unforgeability under chosen message attack. For an adversary A, consider an

experiment $\mathbf{Exp}^{unforg\text{-}cma}_{\mathsf{DS},A}(\lambda)$. First obtain a pair of a public key and a corresponding secret key as $(\mathbf{pk}, \mathbf{sk}) \xleftarrow{\$} \mathsf{K}_s(1^\lambda)$. Then give $\mathbf{pk}$ to A and A can access $\mathsf{Sig}(\mathbf{sk}, \cdot)$ for any number of messages. Finally, A outputs a forgery message-signature pair (M, Σ). He wins if Σ is a valid signature on M and M is not queried so far. We let $\mathbf{Adv}^{unforg\text{-}cma}_{\mathsf{DS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{unforg\text{-}cma}_{\mathsf{DS},\mathcal{A}}(\lambda) = 1]$.

A digital signature scheme DS is secure against forgeries under chose message attack if $\mathbf{Adv}^{unforg\text{-}cma}_{\mathsf{DS},\mathcal{A}}(\lambda)$ is negligible in λ for any polynomial-time A.

2.5 Encryption Scheme

An encryption scheme $\mathsf{E}=(\mathsf{K}_e, \mathsf{Enc}, \mathsf{Dec})$ consists of key generation K_e, encryption Enc, and decryption Dec algorithms. E should satisfy the standard notion of indistinguishability under adaptive chosen-ciphertext attack. For an adversary A, consider an experiment $\mathbf{Exp}^{ind\text{-}cca\text{-}b}_{\mathsf{E},A}(\lambda)$. First obtain a pair of a public key and a corresponding secret key as $(\mathbf{pk}, \mathbf{sk}) \xleftarrow{\$} \mathsf{K}_e(1^\lambda, r_e)$ where r_e is a randomness string (the length of r_e is bounded by some fixed polynomial $r(\lambda)$). Let $\mathsf{LR}(M_0, M_1, b)$ a function which returns M_b for a bit b and messages M_0, M_1. We assume A never queries $\mathsf{Dec}(\mathbf{sk}, \cdot)$ on a ciphertext previously returned by $\mathsf{Enc}(\mathbf{pk}, \mathsf{LR}(\cdot, \cdot, b))$. We let $\mathbf{Adv}^{ind\text{-}cca}_{\mathsf{E},A}(\lambda) = |\Pr[\mathbf{Exp}^{ind\text{-}cca\text{-}1}_{\mathsf{E},A}(\lambda) = 1] - \Pr[\mathbf{Exp}^{ind\text{-}cca\text{-}0}_{\mathsf{E},A}(\lambda) = 1]|$.

An encryption scheme E is IND-CCA secure if $\mathbf{Adv}^{ind\text{-}cca}_{E,\mathcal{A}}(\lambda)$ is negligible in λ for any polynomial-time adversary $\mathcal{A}$.

2.6 Simulation-sound Non-interactive zero knowledge proof system

An NP-*relation over domain* $\mathsf{Dom} \subseteq \{0,1\}^*$ is a subset ρ of $\{0,1\}^* \times \{0,1\}^*$ and x is a *theorem* and w is a *proof* of x if $(x, w) \in \rho$. The membership of $(x, w) \in \rho$ is decidable in polynomial time in the length of the first argument for all x in Dom. Fix an NP relation ρ over Dom and take a pair of polynomial time algorithms (P, V), where P is randomized, and V is deterministic. Both P and V have access to a *common reference string* R and (P, V) is a non-interactive proof system for ρ over Dom if the following two conditions are satisfied for polynomials p and ℓ.

- *Completeness*: $\forall\lambda \in \mathbb{N}, \forall(x, w) \in \rho$ with $|x| \leq \ell(\lambda)$ and $x \in \mathsf{Dom}$:
  $\Pr\left[R \xleftarrow{\$} \{0,1\}^{p(\lambda)};\ \pi \xleftarrow{\$} P(1^\lambda, x, w, R) : V(1^\lambda, x, \pi, R) = 1\right] = 1$.
- *Soundness*: $\forall\lambda \in \mathbb{N}, \forall\hat{P}$ and $x \in \mathsf{Dom}$ such that $x \notin L_\rho$:
  $\Pr[R \xleftarrow{\$} \{0,1\}^{p(\lambda)};\ \pi \xleftarrow{\$} \hat{P}(1^\lambda, x, R) : V(1^\lambda, x, \pi, R) = 1] \leq 2^{-\lambda}$.

3 Our Scheme

We use the scheme given in [8] as the underlying scheme and change the techniques given in [8] to manage time-bound keys and to secure in the dynamical-almost-full anonymity. Using time-bound keys is not given in the scheme in [8].

Thus we change the joining-protocol, Sign, Verify, Open, and Judge algorithms given in [8] to suits with the time-bound keys in our scheme. At the joining-protocol, if a new user shows up with valid keys and if the keys are not being used before by previous members, then the group manager selects and sends a revocation token and a key expiration date. For instance, in a real-life application, the contract ending date is the key expiration date. Thus, after the contract period, the user cannot use those keys. At the end of the joining-protocol, the joining user (new member) creates his secret signing key with the key expiration date. At the time of signing, first, the signer should pass the key-expiration checking. A dishonest member can cheat on this validation step by giving a fake expiration date. To remove such kind of disputes, we allow the group manager to generate the member-certification with the key expiration date at the joining-protocol. Thus, at the time of signing, even the cheating member passes the key-expiration checking and produces the signature, his signature will not pass the validation process in signature verification. Moreover, the signer's signature includes the expiration date. This date should be later date than the time of validation. Otherwise, it will not pass the verification process. For instance, in the real-life application, within the contract period if a member issues a signature to a business certificate which should be validated within a certain period and should not be valid after that period, then the signature should have an expiration date. On the other hand, this ensures that the signature expires before the contract period.

3.1 Description of the Scheme

We denote the key expiration date as t_r, the signature expiration date as t_s, and the current time (verification time) as t_v. Our scheme uses date as "YYMMDD". The new fully dynamic group signature scheme consists of two authorities, namely, the issuer (the group manager GM) and the opener (the tracing manager). The new scheme is a tuple $FDGS$=(GKg, UKg, Join, Issue, Revoke, Sign, Verify, Open, Judge), and we maintain a table called, registration table *reg* to track the registered member details. We depict *group joining protocol* of the scheme which executes Join and Issue in Figure 1, and other algorithms in Figure 2. Each algorithm function as described in below.

- $\mathsf{GKg}(1^\lambda)$: The trusted party executes *group-key generation* algorithm GKg at the setup stage on input 1^λ to produce a group public key **gpk** and authorities keys. Then passes authorities' secret keys, **ik** to the group manager and **ok** to the tracing manager.
- $\mathsf{UKg}(1^\lambda)$: Every new user before interacting with *group-joining protocol* executes *user-key generation* algorithm UKg. UKg takes as input 1^λ and outputs a long-term personal public and private key pair (**upk**$[i]$, **usk**$[i]$) for user i. We assume that **upk**$[i]$ is publicly available.
- Join, Issue: *Group-joining protocol* is an *interactive protocol* between the group manager and a new user. Any new user i who is expecting to be a new member and having a personal key pair (**upk**$[i]$, **usk**$[i]$) can join the

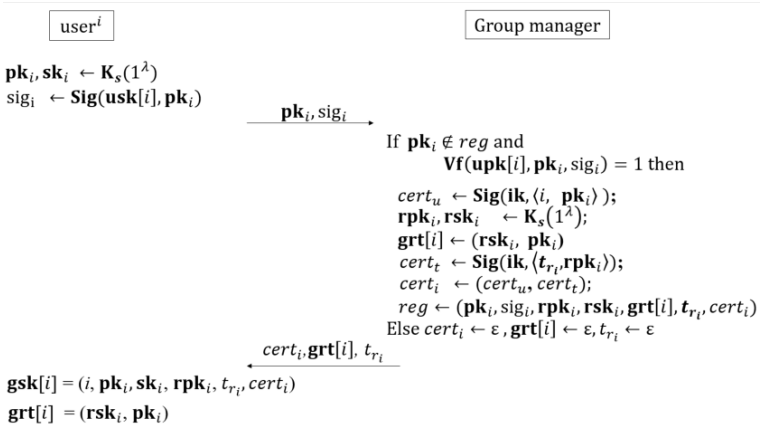

Fig. 1. Group joining protocol

group via *group-joining protocol*. First, the new user generates his public and secret key pair $\mathbf{pk}_i$ and $\mathbf{sk}_i$. Then i produces a signature sig_i on $\mathbf{pk}_i$ using $\mathbf{usk}[i]$ and interacts with the group manager by sending sig_i and $\mathbf{pk}_i$. The group manager checks whether $\mathbf{pk}_i$ is used by the previous members and whether sig_i is generated on $\mathbf{pk}_i$. If those conditions are satisfied then GM produces a revocation token $\mathbf{grt}[i]$ and chooses a key expiration date t_{r_i}. Next GM creates member certification $cert_i$, saves the new member details in *reg* with the status (st) as 1 and sends $cert_i, \mathbf{grt}[i]$, and t_{r_i}. Finally, the new user i produces the secret signing key $\mathbf{gsk} = (i, \mathbf{pk}_i, \mathbf{rpk}_i, t_{r_i}, cert_i)$.

- Revoke(i, $\mathbf{grt}[i]$, $\mathbf{ik}$, RL, reg): The group manager executes *member revoking* algorithm to remove disputed members from the group (premature revocation). Revoke takes, an index i of the revoking member, the group manager's secret key $\mathbf{ik}$, RL, and reg as inputs. First, GM queries reg with the index i to obtain the details of the user stored and checks whether the user i is active and the queries are equal to the data collected by parsing $\mathbf{grt}[i]$. If the data are identical to the queries and if the user i is active, then GM adds $\mathbf{grt}[i] = (\mathbf{rsk}_i, \mathbf{pk}_i)$ to RL and updates reg to inactive status 0.
- Sign($\mathbf{gpk}$, $\mathbf{gsk}[i]$, $\mathbf{grt}[i]$, M, t_s): This randomized *group signing* algorithm creates a signature Σ on a given message M. First, Sign confirms that the given signature expiration date t_s is not later than the key expiration date t_r included in the secret signing key $\mathbf{gsk}[i]$.Then it generates Σ.
- Verify($\mathbf{gpk}$, M, Σ, RL, t_v): This deterministic *group signature verification* algorithm allows the verifiers in possession of $\mathbf{gpk}$ to verify the given signature Σ is generated on the given message M. First, the algorithm verifies whether the signature expiration date t_s is later than the current date. Then it validates Σ and confirms the signer is not being revoked using RL. This algorithm outputs 1 if both the conditions are valid. Otherwise, it returns 0.

$\underline{\mathsf{GKg}(1^\lambda) \to (\mathbf{gpk}, \mathbf{ok}, \mathbf{ik})}$
$R_1 \xleftarrow{\$} \{0,1\}^{P1(\lambda)}$;
$R_2 \xleftarrow{\$} \{0,1\}^{P2(\lambda)}$;
$r_e \xleftarrow{\$} \{0,1\}^{r(\lambda)}$;
$(opk, osk) \leftarrow K_e(1^\lambda; r_e)$;
$(ipk, isk) \xleftarrow{\$} K_s(1^\lambda)$;
$\mathbf{gpk} \leftarrow (1^\lambda, R_1, R_2, \mathbf{opk}, \mathbf{ipk})$;
$\mathbf{ok} \leftarrow (osk, r_e)$;
$\mathbf{ik} \leftarrow isk$;
Return ($\mathbf{gpk}$,$\mathbf{ok}$,$\mathbf{ik}$).

$\underline{\mathsf{UKg}(1^\lambda) \to (\mathbf{upk}, \mathbf{usk})}$
$(\mathbf{upk}, \mathbf{usk}) \xleftarrow{\$} K_s(1^\lambda)$;
Return ($\mathbf{upk}$, $\mathbf{usk}$).

$\underline{\mathsf{Revoke}(i, \mathbf{grt}[i], \mathbf{ik}, RL, reg) \to (RL, reg)}$
Query $\mathbf{grt}[i] \to (i, \mathbf{grt}[i]', st)$;
If ($st \neq 0$ and $\mathbf{grt}[i] = \mathbf{grt}[i]'$)
then $RL \leftarrow RL \cup (\mathbf{grt}[i])$;
update $reg[i]$ to inactive;
Return RL, reg.

$\underline{\mathsf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i], \mathbf{grt}[i], M, t_s) \to \Sigma}$
Parse $\mathbf{gpk}$ as $(1^\lambda, R_1, R_2, \mathbf{opk}, \mathbf{ipk})$;
Parse $\mathbf{gsk}[i]$ as $(i, \mathbf{pk}_i, \mathbf{sk}_i, \mathbf{rpk}_i, t_{r_i}, cert_i)$;
Parse $\mathbf{grt}[i]$ as $(\mathbf{rsk}_i, \mathbf{pk}_i)$;
Parse $cert_i$ as $(cert_u, cert_t)$;
If $t_s > t_{r_i}$ then return ε.
$s \leftarrow \mathsf{Sig}(\mathbf{sk}_i, M)$; $r \xleftarrow{\$} \{0,1\}^\lambda$;
$C_{rt} \leftarrow \mathsf{Enc}(\mathbf{rpk}_i, s; r)$;
$C \leftarrow \mathsf{Enc}(\mathbf{opk}, \langle i, \mathbf{pk}_i cert_u, s\rangle; r)$;
$\pi_1 \xleftarrow{\$} P_1(1^\lambda, (\mathbf{opk}, \mathbf{ipk}, M, C),$
$(i, \mathbf{pk}_i, cert_u, t_{r_i}, \mathbf{rpk}_i, cert_t, s, r), R_1)$;
$\Sigma \leftarrow (C, \pi_1, C_{rt}, cert_t, t_s, t_r, \mathbf{rpk}_i)$;
Return Σ.

$\underline{\mathsf{Verify}(\mathbf{gpk}, M, \Sigma, RL, t_v) \to 1/0}$
Parse $\mathbf{gpk}$ as $(1^\lambda, R_1, R_2, \mathbf{opk}, \mathbf{ipk})$;
Parse Σ as $(C, \pi_1, C_{rt}, cert_t, t_s, t_r, \mathbf{rpk}_i)$;
If $t_v > t_s$ or $t_s > t_r$ then return ε.
If $\mathsf{Vf}(\mathbf{ipk}, \langle t_r, \mathbf{rpk}_i\rangle, cert_t) = 0$
then return ε.
If $V_1(1^\lambda, (\mathbf{opk}, \mathbf{ipk}, M, C), \pi_1, R_1) = 0$
then return 0
For $(\mathbf{grt} = (\mathbf{rsk}, \mathbf{pk})) \in RL$:
$\mathsf{Vf}(\mathbf{pk}, M, \mathsf{Dec}(\mathbf{rsk}, C_{rt})) = 1$
then return 0;
Return 1.

$\underline{\mathsf{Open}(\mathbf{gpk}, \mathbf{ok}, reg, M, \Sigma) \to (i, \tau, st)}$
Parse $\mathbf{gpk}$ as $(1^\lambda, R_1, R_2, \mathbf{opk}, \mathbf{ipk})$;
Parse $\mathbf{ok}$ as (osk, r_e);
Parse Σ as $(C, \pi_1, C_{rt}, cert_t, t_s, t_r, \mathbf{rpk}_i)$;
$M_s \leftarrow Dec(osk, C)$;
Parse M_s as $\langle i, \mathbf{pk}, cert_u, s\rangle$;
If $reg[i] \neq \varepsilon$ then Parse $reg[i]$ as
$(i, \mathbf{pk}_i, sig_i, cert_i, \mathbf{grt}[i], status)$;
Else $\mathbf{pk}_i \leftarrow \varepsilon$; $sig_i \leftarrow \varepsilon$; $st \leftarrow \varepsilon$;
$\pi_2 \leftarrow P_2(1^\lambda, (\mathbf{opk}, C, i, \mathbf{pk}, cert_u, s),$
$(osk, r_e), R_2)$;
If $V_1(1^\lambda, (\mathbf{opk}, \mathbf{ipk}, M, C), \pi_1, R_1) = 0$
then return (0, ε, 0);
If $\mathbf{pk} \neq \mathbf{pk}_i$ or $reg[i] = \varepsilon$ or status = 0
then return (0, ε, 0);
$\tau \leftarrow (\mathbf{pk}_i, sig_i, i, \mathbf{pk}, cert_u, s, \pi_2)$;
Return (i, τ, st).

$\underline{\mathsf{Judge}(\mathbf{gpk}, i, \mathbf{upk}[i], M, \Sigma, \tau) \to 1/0}$
Parse $\mathbf{gpk}$ as $(1^\lambda, R_1, R_2, \mathbf{opk}, \mathbf{ipk})$;
Parse Σ as $(C, \pi_1, C_{rt}, cert_t, t_s, t_r, \mathbf{rpk}_i)$;
If (i, τ, st) = (0, ε, 0) then return
$V_1(1^\lambda, (\mathbf{opk}, \mathbf{ipk}, M, C), \pi_1, R_1) = 0$.
Parse τ as $(\overline{\mathbf{pk}}, \overline{sig}, i', \mathbf{pk}, cert_u, s, \pi_2)$;
If $V_2(1^\lambda, (C, i', \mathbf{pk}, cert_u, s), \pi_2, R_2) = 0$
then return 0
If all of the followings are true then return
1 else return 0.
-$i = i'$.
-$\mathsf{Vf}(\mathbf{upk}[i], \overline{\mathbf{pk}}, \overline{sig})$.
-$\overline{\mathbf{pk}} = \mathbf{pk}$.

Fig. 2. Algorithms of the new Scheme

- Open(**gpk**, **ok**, *reg*, M, Σ): This deterministic verifiable *opening* algorithm traces the signers by taking **gpk**, the opener's secret key **ok**, *reg*, the message-signature pair (M, Σ) as inputs. It returns the index of the signer i, the proof of the claim τ and the status of the signer *st* in *reg*. If the algorithm fails to trace the signature to a particular group member, then it returns (0, ε, 0).
- Judge(**gpk**, i, **upk**$[i]$, M, Σ, τ): This deterministic *judge* algorithm outputs either 1 or 0 depending on the validity of the proof τ on Σ. This takes **gpk**, the member index i, the tracing proof τ, the member verification key **upk**$[i]$, the message M and the signature Σ as inputs and outputs 1 if τ can prove that i produced Σ. Otherwise, it returns 0.

4 Security Analysis of the Scheme

To define the security requirements of the scheme we use a set of experiments as given in Figure 3 and global variables, honest user list **HUL**, corrupted user list **CUL**, token revealed user list **TUL**, signing key revealed user list **SUL**, signatures queried users set SS, challenged signatures set CS, and revoked members set RS.

4.1 Applying Dynamical-almost-full anonymity

When applying the security notion, the dynamical-almost-full anonymity to our scheme, we have to check whether the adversary can get advantage using the time bound keys to win the anonymity game.

Consider an anonymity game between an adversary and a challenger. The adversary may send two indices with one index having a later expiration date and another having a short expiration date which will expire before the validation. Then the adversary can identify the owner of the signature using that difference. For instance, if i_0 key expiration date is 180601, i_1 key expiration date is 180801, and the signature is generating with signature expiration date 180701, then the challenger generates the signature definitely using i_1. This is a very easy catch for the adversary. Thus, at the challenging phase the challenger should confirm both the challenging indices have later key expiration dates that satisfy $t_r > t_s$. This is already true because Sign generates signatures if only $t_r > t_s$. Next consider a scenario that the adversary changes the current date of Verify. For instance, consider a scenario that the key expiration dates of i_0 and i_1 are 180801 and 181001 respectively. The signature is generated with signature expiration date 180501. The real verification date is 180301. Thus, the challenging signature is a valid signature. Assume that the adversary changes the verification date to 180701 and tries to identify the signature is generated using i_0 or i_1. If the adversary executes Verify with current date (verification date) as 180701 then Verify returns invalid because $t_s < t_v$. Hence, the adversary cannot identify the signer. The adversary may try to generate a signature for i_0 or i_1 using the secret signing keys. But since the adversary does not know the revocation token of i_0 or i_1, he fails on generating signatures for i_0, i_1.

$\underline{\mathbf{Exp}^{corr}_{FDGS,A}(\lambda)}$
$(\mathbf{gpk}, \mathbf{ok}, \mathbf{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathbf{HUL} \leftarrow \emptyset$;
$(i, M) \leftarrow A(\mathbf{gpk}; \mathsf{AddU}, \mathsf{ReadReg}, \mathsf{Revoke})$;
If $i \notin \mathbf{HUL}$ or $\mathbf{gsk}[i] = \varepsilon$ or $\mathbf{grt}[i] = \varepsilon$ or $\mathbf{grt}[i] \in RS$ or
$\mathsf{IsActive}(i, reg) = 0$ or $t_i = \varepsilon$ then return 0.
$\Sigma \leftarrow \mathsf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i], \mathbf{grt}[i], M, t_s)$;
If $\mathsf{Verify}(\mathbf{gpk}, M, \Sigma, RS, t_v) = 0$ then return 1.
$(i', \tau, st) \leftarrow \mathsf{Open}(\mathbf{gpk}, \mathbf{ok}, reg, M, \Sigma)$;
If $i \neq i'$ then return 1.
If $\mathsf{Judge}(\mathbf{gpk}, i, \mathbf{upk}[i], M, \Sigma, \tau) = 0$ then return 1 else return 0.

$\underline{\mathbf{Exp}^{anon\text{-}b}_{FDGS,A}(\lambda)}$;
$(\mathbf{gpk}, \mathbf{ok}, \mathbf{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{HUL}, \mathbf{CUL}, \mathbf{TUL}, \mathbf{SUL}, SS, CS, RS \leftarrow \emptyset$;
$b^* \leftarrow A(\mathbf{gpk}$;
$\mathsf{AddU}, \mathsf{CrptU}, \mathsf{SendToUser}, \mathsf{RevealSk}, \mathsf{RevealRt}, \mathsf{Open}, \mathsf{ModifyReg}, \mathsf{Revoke}, \mathsf{Chal}_b)$;
Return b^*;

$\underline{\mathbf{Exp}^{trace}_{FDGS,A}(\lambda)}$
$(\mathbf{gpk}, \mathbf{ok}, \mathbf{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{HUL}, \mathbf{CUL}, \mathbf{TUL}, \mathbf{SUL}, SS, CS, RS \leftarrow \emptyset$;
$(M, \Sigma) \leftarrow A(\mathbf{gpk}, \mathbf{ok}$;
$\mathsf{AddU}, \mathsf{CrptU}, \mathsf{SendToIssuer}, \mathsf{RevealSk}, \mathsf{RevealRt}, \mathsf{Sign}, \mathsf{Revoke})$;
If $\mathsf{Verify}(\mathbf{gpk}, M, \Sigma, RS) = 0$ then return 0.
$(i, \tau) \leftarrow \mathsf{Open}(\mathbf{gpk}, \mathbf{ok}, reg, M, \Sigma)$;
$\mathsf{IsActive}(i, reg) = 0$ then return 0.
If $i = 0$ or $\mathsf{Judge}(\mathbf{gpk}, i, \mathbf{upk}[i], M, \Sigma, \tau) = 0$ then return 1 else return 0.

$\underline{\mathbf{Exp}^{non\text{-}fram}_{FDGS,A}(\lambda)}$
$(\mathbf{gpk}, \mathbf{ok}, \mathbf{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$
$\mathbf{HUL}, \mathbf{CUL}, \mathbf{TUL}, \mathbf{SUL}, SS, CS, RS \leftarrow \emptyset$;
$(M, \Sigma, i, \tau) \leftarrow A(\mathbf{gpk}, \mathbf{ik}, \mathbf{ok}$;
$\mathsf{CrptU}, \mathsf{SendToUser}, \mathsf{RevealSk}, \mathsf{RevealRt}, \mathsf{Sign}, \mathsf{ModifyReg}, \mathsf{Revoke})$;
If $i \notin \mathbf{HUL}$ then return 0.
If $\mathbf{gsk}[i] = \varepsilon$ then return 0.
If $\mathsf{Verify}(\mathbf{gpk}, M, \Sigma, RS) = 0$ then return 0.
If $\mathsf{Judge}(\mathbf{gpk}, i, \mathbf{upk}[i], M, \Sigma, \tau) = 0$ then return 0.
If $(i, M, \Sigma) \in \mathbf{SL}$ then return 0.
If $i \in \mathbf{BUL}$ or $i \in \mathbf{SUL}$ then return 0.
else 1.

Fig. 3. Security Experiments for the scheme

This proves that we can secure our scheme with time bound keys using dynamical-almost- full anonymity.

4.2 Correctness

A signature is verified as valid only if it is generated by a non-revoked member with a non-expired key.

Theorem 1. *For all* $t_r, t_s, t_v, RL, M \in \{0,1\}^*, (\boldsymbol{gpk}, \boldsymbol{ok}, \boldsymbol{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$ *and* $(\boldsymbol{gsk}[i], \boldsymbol{grt}[i]) \leftarrow \mathit{Join}$,

$\mathit{Verify}(\boldsymbol{gpk}, M, \mathit{Sign}(\boldsymbol{gpk}, \boldsymbol{gsk}[i], \boldsymbol{grt}[i], M, t_s), RS, t_v) = \mathit{valid} \iff \boldsymbol{grt}[i] \notin RL$ *and* $t_r > t_s \geq t_v$ *and*

$\mathit{Open}(\boldsymbol{gpk}, \boldsymbol{ok}, reg, M, \mathit{Sign}(\boldsymbol{gpk}, \boldsymbol{gsk}[i], \boldsymbol{grt}[i], M, t_s)) = i, \tau$ *and* $\mathit{Judge}(\boldsymbol{gpk}, i, \boldsymbol{upk}[i], M, \mathit{Sign}(\boldsymbol{gpk}, \boldsymbol{gsk}[i], \boldsymbol{grt}[i], M, t_s), \tau) = 1$.

4.3 Dynamical-almost-full anonymity

Theorem 2 ([1]). *If E is an IND-CCA secure encryption scheme, (P_1, V_1) and (P_2, V_2) are simulation sound, computational zero-knowledge proof systems for ρ_1 over Dom_1 and ρ_2 over Dom_2 respectively, then fully dynamic group signature scheme FDGS is anonymous.*

As per the discussion in 4.1 and using the proof given in [1] the proposed scheme is dynamical-almost-full anonymous.

4.4 Traceability

Theorem 3 ([1]). *If D is secure against forgery under chosen-message attack, (P_1, V_1) and (P_2, V_2) are simulation sound, computational zero-knowledge proof systems for ρ_1 over Dom_1 and ρ_2 over Dom_2 respectively, then fully dynamic group signature scheme FDGS is traceable.*

If an adversary A can break the traceability of our scheme with non-negligible probability, then we construct another polynomial-time algorithm B that can break the unforgeability of the digital signature scheme D. According to the proof provided in [1], we claim that the proposed scheme is traceable.

4.5 Non-Frameability

Theorem 4 ([1]). *If D is secure against forgery under chosen-message attack, (P_1, V_1) and (P_2, V_2) are simulation sound, computational zero-knowledge proof systems for ρ_1 over Dom_1 and ρ_2 over Dom_2 respectively, then fully dynamic group signature scheme FDGS is non-frameable.*

Similar to the proof of traceability, our scheme is non-frameable based on the non-frameability of the digital signature scheme D.

5 Evaluation of the scheme and Conclusion

When generating signatures the signer has to proof that his secret key has not expired and he has to pass the expiration dates of the secret key and the signature. Thus, this makes additional work to the signer and it makes longer signature size comparing to the existing scheme. At the verification stage the verifier has to check the expiration dates. Even it is an additional work for verifiers, it is less than the cost of revocation check in previous schemes. At the time of joining, the group manager has to pick the expiration date for each new user. This is also not in the previous scheme [8]. However, all the additional cost in the proposed scheme is less when there are large number of revoked members. Thus, efficiency of the revocation check in our scheme is far better than the schemes like [1] and [8] when applying to a large group.

Since the proposed scheme intends to reduce the cost of revocation cost while being secured, we have achieved the goal with a reasonable solution. The proposed scheme can me improve to a system like a customer managing system where the customers are joining the group (system) for a short period.

Acknowledgments. This work is supported in part by JSPS Grant-in-Aids for Scientic Research (A) JP16H01705 and for Scientic Research (B) JP17H01695.

References

1. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: CT-RSA 2005. vol. 3376, pp. 136–153. LNCS (2005)
2. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM-CCS 2004. pp. 168–177. ACM (2004)
3. Brickell, E.: An efficient protocol for anonymously providing assurance of the container of the private key. *Submitted to the Trusted Comp. Group (April 2003)* (2003)
4. Chaum, D., Van Heyst, E.: Group signatures. In: EUROCRYPT 1991, LNCS. vol. 547, pp. 257–265. Springer (1991)
5. Chu, C.K., Liu, J.K., Huang, X., Zhou, J.: Verifier-local revocation group signatures with time-bound keys. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security. pp. 26–27. ACM (2012)
6. Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-based group signature scheme with verifier-local revocation. In: PKC 2014, LNCS. vol. 8383, pp. 345–361. Springer (2014)
7. Perera, M.N.S., Koshiba, T.: Achieving almost-full security for lattice-based fully dynamic group signatures with verifier-local revocation. In: ISPEC 2018, LNCS (to appear)
8. Perera, M.N.S., Koshiba, T.: Fully dynamic group signature scheme with member registration and verifier-local revocation. In: ICMC 2018, Mathematics and Computing (to appear)