A Selectable k-Times Relaxed Anonymous Authentication Scheme

Keita Emura[1], Atsuko Miyaji[1], and Kazumasa Omote[1]

School of Information Science, Japan Advanced Institute of Science and Technology,
1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan
{k-emura, miyaji, omote}@jaist.ac.jp

Abstract. In a k-Times Anonymous Authentication (k-TAA) scheme, Application Providers (APs) authenticates each group member in k times. A user can preserve his/her own privacy and an AP can restrict the number of used services to just k-times, since users are identified if they access an AP more than k times. In all previous schemes, each AP assumes the same k for all users. Added to this, total anonymity of unlinkability requires large computation amount such as pairing computations compared with non-anonymous schemes. In this paper, we propose a selectable k-TAA scheme with relaxed anonymity. Relaxed anonymity is an intermediate level of privacy required between total anonymity and linkability, where authentication executions for the same AP are linkable, but an authentication execution with an AP v_0 and an authentication execution with an AP v_1 ($v_0 \neq v_1$) are unlinkable. Our authentication algorithm is efficient than existing ones thanks to this relaxed notion.

1 Introduction

In the recent information society, there are many types of applications. SaaS (Software as a Service) [20], a kind of providing service, has become popular to save cost of time such as an install and expense to use software. In the SaaS environment, a program (which provides the service) runs on the server of the Application Provider (AP), and the user is provided the service from the AP using an on-line network. Recently, there are many SaaS-type services such as the Oracle Database SaaS Platform [16], the IBM LotusLive [11], and so on. A user is forced to manage many assumed names together with passwords since each service requires separate account. Recently, OpenID has been introduced (by Google, Microsoft, IBM, Verisign and so on) to reduce such a managing cost. In OpenID, a user with only manages one account can be authenticated from many APs. However, a new issue that access to services (these indicate a user's tastes and habits) is exposed to different APs by the same OpenID. Therefore, anonymous authentication schemes based on group signature schemes [2, 5] are required, where users are authenticated whether they are members of the group or not. Group signatures are unlinkable, namely anybody cannot decide whether an authenticated user is the same authenticated user or not. However, a simple group signature scheme is not suitable in the SaaS environment, in which the

AP cannot demand a service fee by detecting each user. This is why k-Times Anonymous Authentication (k-TAA) schemes [1, 13, 14, 17, 18] are attractive for the SaaS environment, where each group member is anonymously authenticated by APs in k times. A user can preserve his/her own privacy and an AP can restrict the number of used services to just k-times, since users are identified if they access an AP more than k times.

Previous k-TAA Schemes : The first k-TAA scheme has been proposed in [17]. There are three entities in a k-TAA scheme, a Group Manager (GM), users and APs. The GM manages a group, and issues membership certificates for users. The first k-TAA scheme is constructed by a group signature scheme with the *tracing tag mechanism*: an AP opens k tag bases. A user makes a tag by using his/her secret key and one of the tag bases, which has not been used before. The user makes an authentication proof including the tag. The AP stores the authentication execution transcript. If the user attempts authentication more than k times, then he/her can be identified the same tag base. This tracing tag mechanism is used by other k-TAA schemes [1, 13, 14, 18]. In all previous k-TAA schemes, each AP assumes the same k for all users. Therefore, an AP cannot decide the number of access of services for each user. Total anonymity of unlinkability requires large computation amount such as pairing computations compared with non-anonymous schemes. Furthermore, even if APs want to obtain the number of accesses by each user to improve the application providing strategy, e.g., long tail marketing [21], APs cannot obtain the number of accesses. For example, assume that any user accesses less than k-times and AP knows the total number of accesses is 10000, then AP cannot distinguish whether 10000 users access only once or 100 users access 100 times. On the other hands, by using a linkable authentication, access to services (these indicate a user's tastes and habits) is exposed among different APs. Therefore, an intermediate level of privacy required between total anonymity and linkability is necessary to preserve privacy and to satisfy information utilization, simultaneously.

Our Contribution : In this paper, we propose a selectable k-Times Relaxed Anonymous Authentication (k-TRAA) scheme that enables an allowable number to be assigned for each user. There are some suitable scenarios, where different allowable numbers are decided for each user. For example, a user who spends much money has to be given the right to more accesses to AP compared to other users. This is a natural requirement in the SaaS environment. We introduce a relaxed security notion called relaxed anonymity, which satisfies two kinds of anonymity: (1) two authentication executions with the same AP are linkable, but the AP cannot identify a user from the group of users, and (2) an authentication execution with an AP v_0 and an authentication execution with an AP v_1 ($v_0 \neq v_1$) are unlinkable. Relaxed anonymity is an intermediate level of privacy required between total anonymity and linkability. Under relaxed anonymity, a user's taste is not exposed among different APs. Otherwise, an AP can obtain the number of accesses of own application from each user. Our authentication algorithm is more efficient than previous k-TAA schemes such that no pairing computation is required and computation is independent to the allowable num-

ber k in authentication phase, where these large computations are concentrated in grant phase. Moreover, in our authentication algorithm, full off-line computation is available. We insist that these efficiencies are due to relaxed anonymity offering a tradeoff between privacy preservation and efficiency. To realize these efficiencies, our scheme is made from both a group signature scheme [6] and the newly introduced *sequence-of-zero-knowledge-proof mechanism*.
Organization : Some definitions are given in Section 2. Our scheme is presented in Section 3. The efficiency comparison is discussed in Section 4. Security analyses are presented in Section 5.

2 Definitions

In this section, we define complexity assumptions, the model of selectable k-TRAA and security requirements. Note that $x \in_R S$ means x is randomly chosen for a set S.

2.1 Bilinear Groups and Complexity Assumptions

Definition 1. (**Bilinear Groups**)

1. $\mathbb{G}_1$, $\mathbb{G}_2$ *and* $\mathbb{G}_3$ *are cyclic groups of prime order* p.
2. g_1 *and* g_2 *are generators of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$, *respectively.*
3. ψ *is an efficiently computable isomorphism* $\mathbb{G}_2 \to \mathbb{G}_1$ *with* $\psi(g_2) = g_1$.
4. e *is an efficiently computable bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ *with the following properties.*
   - *Bilinearity : for all* $u, u' \in \mathbb{G}_1$ *and* $v, v' \in \mathbb{G}_2$, $e(uu', v) = e(u, v)e(u', v)$ *and* $e(u, vv') = e(u, v)e(u, v')$.
   - *Non-degeneracy :* $e(g_1, g_2) \neq 1_{\mathbb{G}_3}$ ($1_{\mathbb{G}_3}$ *is the* $\mathbb{G}_3$*'s unit*).

Our scheme is based on the q-strong Diffie-Hellman (q-SDH) [3], k-Power Computational Diffie-Hellman (k-PDDH) [10], and k-Power Decisional Diffie-Hellman (k-PDDH) [10] assumptions. For the security parameter λ, let $\epsilon = \epsilon(\lambda)$ be a negligible function, namely for every polynomial $poly(\cdot)$ and for sufficiently large λ, $\epsilon(\lambda) < 1/poly(\lambda)$.

Definition 2. (**q-SDH assumption** *[3]*) *The q-SDH problem in* $(\mathbb{G}_1, \mathbb{G}_2)$ *is a problem, for input of a* $(q+2)$ *tuple* $(g, g', (g')^{\xi}, \cdots, (g')^{\xi^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, *where* $g = \psi(g')$, *to compute a tuple* $(x, g^{1/(\xi+x)})$. *An algorithm* $\mathcal{A}$ *has an advantage* ϵ *in solving the q-SDH problem in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if* $\Pr[\mathcal{A}(g, g', (g')^{\xi}, \cdots, (g')^{\xi^q}) = (x, g^{1/(\xi+x)})] \geq \epsilon$. *We say that the q-SDH assumption holds in* $(\mathbb{G}_1, \mathbb{G}_2)$ *if no PPT algorithm has an advantage of at least* ϵ *in solving the q-SDH problem in* $(\mathbb{G}_1, \mathbb{G}_2)$.

Definition 3. (**k-PCDH** *[10]*) *The k-PCDH problem in* $\mathbb{G}_1$ *is a problem, for input a tuple* $(h', h, h^y, h^{y^2}, \ldots, h^{y^{k-1}}) \in \mathbb{G}_2 \times \mathbb{G}_1^k$, *where* $h = \psi(h')$, *to compute* h^{y^k}. *An algorithm* $\mathcal{A}$ *has an advantage* ϵ *in solving the k-PCDH problem in* $\mathbb{G}_1$

if $\Pr[\mathcal{A}(h', h, h^y, \ldots, h^{y^{k-1}}) = h^{y^k}] \geq \epsilon$. *We say that the k-PCDH assumption holds in* $\mathbb{G}_1$ *if no PPT algorithm has an advantage of at least* ϵ *in solving the k-PCDH problem in* $\mathbb{G}_1$.

A decisional version of k-PCDH assumption (k-PDDH assumption) is simply defined such that $|\Pr[\mathcal{A}(h', h, h^y, \ldots, h^{y^k}) = 0] - \Pr[\mathcal{A}(h', h, h_1, \ldots, h_k) = 0]|$ is negligible, where $h_1, \ldots, h_k \in \mathbb{G}_1$. Note that a 3-PDDH problem instance $(h, h^y, h^{y^2}, h^{y^3})$ is a DDH (Decisional Diffie-Hellman[1]) tuple $(h, h^{y^2}, h^y, (h^{y^2})^y) = (h, h'', h^y, (h'')^y)$, where $h'' := h^{y^2}$. This means that if the DDH problem is easy to solve, then the k-PDDH problem ($k \geq 3$) is also easy to solve. Therefore, to hold the k-PDDH assumption in $\mathbb{G}_1$, we require that the k-PDDH assumption holds in $\mathbb{G}_1$. It is a stronger assumption than the eXternal Diffie-Hellman (XDH) assumption [4] which only requires that the DDH assumption holds in $\mathbb{G}_1$. We can use MNT curves [12], where there is no efficient isomorphism between $\mathbb{G}_1$ to $\mathbb{G}_2$.

2.2 Model of Selectable k-Times Relaxed Anonymous Authentication

Let λ be the security parameter, GM the group manager, AP the application provider, gpk the group public key, gsk the group secret key which is used for issuing a membership certificate, (mpk_i, msk_i) the member public/secret key of U_i $(i = 1, 2, \ldots, n)$, LIST an identification list for tracing, Log_v the log list for keeping logs, and ID_v an identity of AP v. To simplify, we describe an allowable number k_i, although at times we describe $k_{v,i}$ for an AP v and a user U_i.

Definition 4. *System operations of a Selectable k-TRAA*

- $\mathsf{GM\text{-}Setup}(1^\lambda)$*: This algorithm takes as input* λ *and returns gpk, gsk and* LIST.
- $\mathsf{AP\text{-}Setup}(ID_v)$*: This algorithm takes as input* ID_v *and returns* Log_v.
- $\mathsf{Join}(\mathsf{Join\text{-}GM}\langle gpk, gsk, \mathsf{LIST}\rangle, \mathsf{Join\text{-}U}\langle gpk\rangle)$*: This algorithm takes as input gpk,* gsk, LIST *and* upk_i *from* GM*, and gpk,* upk_i *and* usk_i *from* U_i*, and returns* (mpk_i, msk_i)*, and appends* mpk_i *and the user's identity i to* LIST *making the updated* LIST.
- $\mathsf{Grant}(\mathsf{Grant\text{-}AP}\langle gpk, Log_v\rangle, \mathsf{Grant\text{-}U}\langle gpk, msk_i, k_i\rangle)$*: This algorithm takes as input gpk and* Log_v *from AP, and gpk,* msk_i *and* k_i *from* U_i*. The transcript of grants are recorded in* $reg_d \in Log_v$*. Note that* $d \in \mathbb{Z}_{>0}$ *is just an indexed number in the log, namely, d is independent of the user's identity i.*
- $\mathtt{Auth}(\mathsf{Verify}\langle gpk, Log_v\rangle, \mathsf{Proof}\langle gpk, msk_i\rangle)$*: Let* U_i *be assigned with the indexed number d. This algorithm takes as input gpk and* Log_v *from AP, and gpk and* msk_i *from* U_i*, and outputs* accept *when conditions (1) and (2) are satisfied: (1) the anonymous user is a member of the group, and (2) the anonymous user has already been authenticated less than k times, otherwise,*

[1] Note that the DDH problem is a problem, for input a tuple $(g, g', g^u, (g')^v)$, where $g, g' \in \mathbb{G}_1$ and $u, v \in \mathbb{Z}_p^*$, to decide $u = v$ or not.

outputs reject. *When the output of this algorithm is* accept, *this algorithm records the transcript of authentications in* $reg_d \in Log_v$.

– `Trace`(gpk, Log_v)*: This algorithm takes as input* gpk *and* Log_v*, and computes a user's public key* mpk_i *for a user who has already been authenticated more than* k_i *times. If the entry* (i, mpk_i) *is included in* LIST*, then it outputs* i*. Otherwise, the algorithm verifies all proofs included in* Log_v*. If all proofs are valid, then it outputs "*GM*". Otherwise, if a proof is invalid, then it outputs "*AP*".*

2.3 Security Definitions

In this subsection, we define relaxed anonymity, which is an intermediate level of privacy required between total anonymity and linkability. $\mathcal{A}$ is admitted to collude with users, GM and APs, respectively. Then $\mathcal{A}$ can play the role of these entities although accessible oracles are restricted. We define oracles as follows:
Oracles : We use oracles $\mathcal{O}_{\mathsf{List}}$, $\mathcal{O}_{\mathsf{Join\text{-}GM}}$, $\mathcal{O}_{\mathsf{Join\text{-}U}}$, $\mathcal{O}_{\mathsf{AP\text{-}Setup}}$, $\mathcal{O}_{\mathsf{Proof}}$, $\mathcal{O}_{\mathsf{Verify}}$, $\mathcal{O}_{\mathsf{Grant\text{-}AP}}$, $\mathcal{O}_{\mathsf{Grant\text{-}U}}$ and $\mathcal{O}_{\mathsf{Query}}$. Each definition is as follows: the list oracle $\mathcal{O}_{\mathsf{List}}$ [17, 18] manages LIST. An adversary $\mathcal{A}$ is allowed to read LIST to call $\mathcal{O}_{\mathsf{List}}$. If $\mathcal{A}$ colludes with some users, then $\mathcal{A}$ can write corresponding entries of LIST. In addition, if $\mathcal{A}$ colludes with the GM, then $\mathcal{A}$ can delete entries of LIST. $\mathcal{O}_{\mathsf{Join\text{-}GM}}$ is the oracle which runs the Join-GM algorithm honestly. $\mathcal{O}_{\mathsf{Join\text{-}U}}$ is the oracle which runs the Join-U algorithm on behalf of honest users. $\mathcal{O}_{\mathsf{AP\text{-}Setup}}$ is the oracle which runs the AP-Setup algorithm honestly. $\mathcal{O}_{\mathsf{Proof}}$ is the oracle which runs the Proof algorithm on behalf of honest user. $\mathcal{O}_{\mathsf{Verify}}$ is the oracle which runs the Verify algorithm on behalf of honest APs. $\mathcal{O}_{\mathsf{Grant\text{-}AP}}$ is the oracle which runs the Grant-AP algorithm on behalf of honest APs. $\mathcal{O}_{\mathsf{Grant\text{-}U}}$ is the oracle which runs the Grant-U algorithm on behalf of honest users. $\mathcal{O}_{\mathsf{Query}}$ is the challenge oracle which is defined in Definition 5. We describe situations when $\mathcal{A}$ is allowed to access the oracles as follows: $\mathcal{A}$ is always allowed to access $\mathcal{O}_{\mathsf{List}}$, to access $\mathcal{O}_{\mathsf{Join\text{-}GM}}$ if $\mathcal{A}$ does not collude with the GM, to access $\mathcal{O}_{\mathsf{Join\text{-}U}}$, $\mathcal{O}_{\mathsf{Grant\text{-}U}}$ and $\mathcal{O}_{\mathsf{Proof}}$ if $\mathcal{A}$ does not collude with the user, and to access $\mathcal{O}_{\mathsf{Grant\text{-}AP}}$ and $\mathcal{O}_{\mathsf{Verify}}$ if $\mathcal{A}$ does not collude with the AP.

Next, we define the relaxed anonymity game. We refer to the L-anonymity game for linkable ring signature [19], namely, target users U_{i_0} and U_{i_1} are not input in $\mathcal{O}_{\mathsf{Proof}}$ by an adversary.

Definition 5. `Relaxed anonymity` *: Relaxed anonymity requires that for all PPT* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the following game be negligible.*

An adversary $\mathcal{A}$ *is allowed to collude with the* GM*, all APs, and all users except target users* U_{i_0} *and* U_{i_1}*, and to query oracles* $\mathcal{O}_{\mathsf{List}}$*,* $\mathcal{O}_{\mathsf{Join\text{-}U}}$*,* $\mathcal{O}_{\mathsf{Proof}}$*,* $\mathcal{O}_{\mathsf{Grant\text{-}U}}$ *and* $\mathcal{O}_{\mathsf{Query}}$*.* U_{i_0} *and* U_{i_1} *are not input in* $\mathcal{O}_{\mathsf{Proof}}$ *by* $\mathcal{A}$*, and can be input in* $\mathcal{O}_{\mathsf{Join\text{-}U}}$*, and have not been granted by an AP* v_0 *and an AP* v_1*. Let* k_{v_0,i_0} *and* k_{v_0,i_1} *(resp.* k_{v_1,i_0} *and* k_{v_1,i_1}*) be allowable numbers of* U_{i_0} *and* U_{i_1} *for the AP* v_0 *(resp. the AP* v_1*). When* $\mathcal{O}_{\mathsf{Query}}$ *is called by* $\mathcal{A}$ *for the first time,* $\mathcal{O}_{\mathsf{Query}}$ *randomly chooses* $b \in \{0, 1\}$*, executes* $\mathcal{O}_{\mathsf{Grant\text{-}U}}(U_{i_0})$ *with the AP* v_0 *and* $\mathcal{O}_{\mathsf{Grant\text{-}U}}(U_{i_b})$ *with the AP* v_1*, and outputs the* Log_{v_0} *and* Log_{v_1}*. From the second*

call, $\mathcal{O}_{\mathsf{Query}}$ executes both $\mathcal{O}_{\mathsf{Proof}}(U_{i_0})$ with the AP v_0 and $\mathcal{O}_{\mathsf{Proof}}(U_{i_b})$ with the AP v_1, and outputs the Log_{v_0}, Log_{v_1} and the transcript of the authentication protocol. $\mathcal{O}_{\mathsf{Query}}$ can be used less than k times, where $k = Min\{k_{v_0,i_0}, k_{v_0,i_1}, k_{v_1,i_0}, k_{v_1,i_1}\}$. $\mathcal{A}$ outputs a bit b', and wins if $b' = b$. The advantage of $\mathcal{A}$ is defined as $Adv^{R\text{-}anon}(\mathcal{A}) = |\Pr(b = b') - \frac{1}{2}|$.

If $Adv^{R\text{-}anon}(\mathcal{A})$ is negligible, then $\mathcal{A}$ cannot distinguish authentications of U_{i_0} (namely $\mathcal{O}_{\mathsf{Proof}}(U_{i_0})$) with the AP v_0 from authentications of U_{i_b} (namely $\mathcal{O}_{\mathsf{Proof}}(U_{i_b})$) with the different AP v_1. This means $\mathcal{A}$ cannot determine whether U_{i_b} is U_{i_0} or not. Note that the definition of relaxed anonymity does not guarantee that two authentications with the same AP are unlinkable. Therefore, definitions of relaxed anonymity captures the two properties (1) two authentications with the same AP are linkable, but AP v cannot determine a user from the group of users, and (2) an authentication with an AP v_0 and an authentication with an AP v_1 are unlinkable.

Next, we define the detectability game. This definition captures the fact that all users cannot execute Auth algorithm more than the allowable number of times, or they have to be detected by the Trace algorithm.

Definition 6. `Detectability` *: Detectability requires that for all PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game be negligible.*

An adversary $\mathcal{A}$ is allowed to collude with all users. Let N be the number of users who colluded with $\mathcal{A}$, k_i be an allowable number of a user U_i ($i = 1, 2, \ldots, N$), and $K = \sum_{i=1}^{N} k_i$ be the total of allowable numbers. $\mathcal{A}$ wins if $\mathcal{A}$ can be accepted more than K times. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

The difference between our detectability game and the previous detectability games (defined in [1, 13, 14, 18]) is that the previous game uses $K = Nk$, since all users are forced to use the same allowable number k.

Next, we define the exculpability games for users, GM and AP. These definitions are the same as in [18].

Definition 7. `Exculpability for users` *: Exculpability for users requires that for all PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game be negligible.*

An adversary $\mathcal{A}$ is allowed to collude with all entities except a target user U^*. $\mathcal{A}$ can run $\mathcal{O}_{\mathsf{Proof}}$ less than k^* times, where k^* is an allowable number of U^*. $\mathcal{A}$ wins if $\mathcal{A}$ can compute the authentication log, which is the input of the tracing algorithm that outputs the identification of U^*. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Definition 8. `Exculpability for GM` *: Exculpability for GM requires that for all PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game be negligible.*

An adversary $\mathcal{A}$ is allowed to collude with all entities except the GM. $\mathcal{A}$ wins if $\mathcal{A}$ can compute the authentication log which is the input of the tracing algorithm that outputs GM. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

Definition 9. `Exculpability for AP` *: Exculpability for AP requires that for all PPT $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game be negligible.*

An adversary $\mathcal{A}$ is allowed to collude with all entities except a target AP v^*. $\mathcal{A}$ wins if $\mathcal{A}$ can compute the authentication log which is the input of tracing algorithm that outputs AP v^*. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.

2.4 Proving Relations on Representations

By using the Fiat-Shamir heuristic [8], a digital signature scheme is constructed from zero-knowledge proofs of knowledge (ZK). These signatures are called SPKs. For a message M and secret values $(x_1, \ldots, x_n)$, we use the notation $SPK\{(x_1, \ldots, x_n) : R(x_1, \ldots, x_n)\}(M)$, which is a signature of M signed by a signer who has $(x_1, \ldots, x_n)$ satisfying the relation $R(x_1, \ldots, x_n)$. SPKs can be simulated without the knowledge of $(x_1, \ldots, x_n)$ in the random oracle model.

3 Proposed Scheme

In this section, we first introduce the underlying idea of our construction, and then propose a selectable k-TRAA scheme.

3.1 A primitive selectable k-TAA scheme

Here we give a simple construction of a selectable k-TAA scheme based on the previous k-TAA scheme: The AP makes k_i tag bases, where k_i is an allowable number of a user U_i, and issues these bases to U_i regarding his/her secret keys for exclusive use. In this simple scheme, the number of the secret key of U_i depends on k_i. As another construction, the AP makes k_i tag bases by using a pseudo-random number for U_i, and opens these bases to U_i regarding his/her public keys for exclusive use. In this primitive scheme, the number of the public key depends on $N \sum_{i=1}^{N} k_i$, where N be the number of application group members. Our purpose is to achieve that the number of both the number of the secret key of each U_i and the number of the public key is independent of k_i and N.

3.2 Underlying Idea of Our Construction

In this subsection, we introduce a higher-level description of the key ideas called the sequence-of-zero-knowledge-proof mechanism. This idea is similar to a source authentication using hash chaining proposed in GR97 [9]. For the sake of clarity, we summarize the GR97 scheme as follows: Let H be a one-way hash function and Sign be a signing algorithm of a digital signature scheme: The sender splits a data to be signed to m blocks $Data_1, Data_2, \ldots, Data_m$, computes the hash chain $h_m := H(Data_m)$, $h_{m-1} := H(Data_{m-1}||h_m)$, $\ldots$, $h_1 := H(Data_1||h_2)$, and makes a signature of h_1, $\sigma := \mathsf{Sign}(h_1)$, and sends σ, h_1, $Data_1||h_2$, $\ldots$, $Data_{m-1}||h_m$, $Data_m$. The receiver verifies σ, and checks $H(Data_1||h_2) \stackrel{?}{=} h_1$,

$H(Data_2||h_3) \stackrel{?}{=} h_2 \ldots$, $H(Data_{m-1}||h_m) \stackrel{?}{=} h_{m-1}$ and $H(Data_m) \stackrel{?}{=} h_m$. Only one signature verification is performed and only one hash is computed for every data block.

The concept of our scheme is the same as the GR97 scheme. Specifically, in the grant phase, a user proves that both (1) the user has a valid membership certificate by using a group signature (which requires pairing computations) and (2) a commitment is computed by using a part of the membership certificate (which requires the computations depend on the allowable number). In each authentication phase, the user only has to prove that a commitment is computed by using the same secret key which has already been used in the previous authentication phase.

3.3 Proposed Scheme

Let $NIZK$ be a Non-Interactive Zero-Knowledge proof and SPK be a Signatures based on Proofs of Knowledge. To detect a user U_i after k_i times authentication, a polynomial with degree k_i+2 is applied. Let $f_{k_i}(X) = \prod_{j=1}^{k_i+2}(X+j) = \sum_{j=0}^{k_i+2} a_j X^j \in \mathbb{Z}_p[X]$[2]. Let $M \in \{0,1\}^*$ be a message for deciding an allowable number k_i. It is assumed that the user cannot be identified by AP from M and k_i. The concurrently secure Join algorithm proposed in DP06 [6] is used in our Join algorithm, where all proofs are non-interactive using $NIZK$ and a signature scheme $DSig$. The verifying/signing key (upk_i, usk_i) of the signature scheme $DSig$ is used in Join.

- GM-Setup(1^λ)
  1. The GM selects cyclic groups of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ with λ-bits of prime order p, an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, and a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$.
  2. The GM chooses $\tilde{g} \in_R \mathbb{G}_1$ and a generator $g_2 \in \mathbb{G}_2$, and sets $g_1 = \psi(g_2)$.
  3. The GM chooses $\gamma \in_R \mathbb{Z}_p$, and computes $\omega = g_2^\gamma$.
  4. The GM outputs $gpk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, e, \mathcal{H}, g_1, g_2, \tilde{g}, \omega)$ and $gsk = (\gamma)$.
- AP-Setup(ID_v)
  1. AP v outputs $Log_v = \emptyset$.
- Join(Join-GM$\langle gpk, gsk, \mathsf{LIST}, upk_i\rangle$, Join-U$\langle gpk, upk_i, usk_i\rangle$)
  1. U_i chooses $y_i \in_R \mathbb{Z}_p$, and computes $F_i = \tilde{g}^{y_i}$ and $\pi_1 = NIZK\{y_i : F_i = \tilde{g}^{y_i}\}$.
  2. U_i sends F_i and π_1 to the GM.
  3. The GM checks π_1. If π_1 is not valid, then aborts.
  4. The GM chooses $x_i \in_R \mathbb{Z}_p$, and computes $A_i = (g_1 F_i)^{1/(\gamma+x_i)}$, $B_i = e(g_1 F_i, g_2)/e(A_i, \omega)$, $E_i = e(A_i, g_2)$ and $\pi_2 = NIZK\{x_i : B_i = E_i^{x_i}\}$.
  5. The GM sends A_i, B_i, E_i and π_2 to U_i.
  6. U_i checks π_2. If π_2 is not valid, then aborts.
  7. U_i makes $S_{i,A_i} = DSig_{usk_i}(A_i)$, and sends S_{i,A_i} to the GM.

[2] If k_i is known, then coefficients $\{a_j\}_{j=0}^{k_i+2}$ can be computed easily.

8. The GM verifies S_{i,A_i} with respect to upk_i and A_i. If S_{i,A_i} is valid, then the GM sends x_i to U_i, and adds $(i, mpk_i = g_1^{x_i})$ to LIST.
9. U_i checks $e(A_i, g_2)^{x_i} e(A_i, w) e(\tilde{g}, g_2)^{-y_i} \stackrel{?}{=} e(g_1, g_2)$ to verify the relation $A_i^{(x_i+\gamma)} = g_1 \tilde{g}^{y_i}$.

– $\mathsf{Grant}(\mathsf{Grant\text{-}AP}\langle gpk, Log_v\rangle, \mathsf{Grant\text{-}U}\langle gpk, msk_i, k_i\rangle)$

1. U_i chooses $\alpha, \beta, \gamma \in_R \mathbb{Z}_p$, and computes $C_1 = A_i \tilde{g}^{\alpha}$, $C_2 = mpk_i (g_1^{f_{k_i}(y_i)})^{\beta} = g_1^{x_i + \beta \sum_{j=0}^{k_i+2} a_j y_i^j}$, $C_{3,1} = g_1^{\beta}$, $C_{3,2} = C_{3,1}^{y_i}$, $C_{4,1} = g_1^{\gamma}$, $C_{4,2} = C_{4,1}^{y_i}$, $C_{4,3} = C_{4,2}^{y_j}, \ldots, C_{4,k_i+2} = C_{4,k_i+1}^{y_j}$ and $h_j = g_1^{a_j \beta}$ $(j \in [0, k_i + 2])$.
2. U_i sets $\tau = \alpha x_i$, $z_{i,j} = y_i^j$ $(j \in [0, k_i{+}2])$, and computes $\pi_c = SPK\{(\alpha, \beta, x_i, y_i, \tau, z_{i,1}, z_{i,2}, \ldots, z_{i,k_i+2}) : \frac{e(C_1,\omega)}{e(g_1,g_2)} = \frac{e(\tilde{g},g_2)^{\tau} \cdot e(\tilde{g},g_2)^{y_i} \cdot e(\tilde{g},\omega)^{\alpha}}{e(C_1,g_2)^{x_i}} \wedge C_2 = g_1^{x_i} \prod_{j=0}^{k_i+2} h_j^{z_{i,j}} \wedge C_{3,1} = g_1^{\beta} \wedge C_{3,2} = C_{3,1}^{y_i} \wedge C_{4,1} = g_1^{\gamma} \wedge C_{4,2} = C_{4,1}^{y_i} = C_{4,1}^{z_{i,1}} \wedge \ldots \wedge C_{4,k_i+2} = C_{4,k_i+1}^{y_i} = C_{4,1}^{z_{i,k_i+2}} \wedge h_0 = g_1^{a_0 \beta} \wedge \ldots \wedge h_{k_i+2} = g_1^{a_{k_i+2} \beta}\}(ID_v, k_i, M)$.
   Concretely, U_i computes π_c as follows:
   (a) U_i chooses $r_\alpha, r_\beta, r_\gamma, r_{x_i}, r_{y_i}, r_\tau, r_{z_{i,2}}, \ldots, r_{z_{i,k_i+2}} \in_R \mathbb{Z}_p$. Note that $r_{z_{i,0}}$ and $r_{z_{i,1}}$ are not necessary since $z_{i,0} = 1$ and $z_{i,1} = y_i$.
   (b) U_i computes $R_1 = \frac{e(\tilde{g},g_2)^{r_\tau} e(\tilde{g},g_2)^{r_{y_i}} e(\tilde{g},\omega)^{r_\alpha}}{e(C_1,g_2)^{r_{x_i}}}$, $R_2 = g_1^{r_{x_i}} h_0 h_1^{r_{y_i}} \prod_{j=2}^{k_i+2} h_j^{r_{z_{i,j}}}$, $R_{3,1} = g_1^{r_\beta}$, $R_{3,2} = C_{3,1}^{r_{y_i}}$, $R_{4,1} = g_1^{r_\gamma}$, $R_{4,2} = C_{4,1}^{r_{y_i}}$, $R_{4,3} = C_{4,2}^{r_{y_i}}$, $R'_{4,3} = C_{4,1}^{r_{z_{i,2}}}$, $R_{4,4} = C_{4,3}^{r_{y_i}}$, $R'_{4,4} = C_{4,1}^{r_{z_{i,3}}}$, $\ldots$, $R_{4,k_i+2} = C_{4,k_i+1}^{r_{y_i}}$, $R'_{4,k_i+2} = C_{4,1}^{r_{z_{i,k_i+2}}}$, $R_{5,0} = h_0^{r_\beta}$, $R_{5,1} = h_1^{r_\beta}$, $\ldots$, $R_{5,k_i+1} = h_{k_i+1}^{r_\beta}$ and $R_{5,k_i+2} = h_{k_i+2}^{r_\beta}$.
   (c) U_i computes $c = \mathcal{H}(gpk, ID_v, k_i, M, C_1, C_2, C_{3,1}, C_{3,2}, C_{4,1}, \ldots, C_{4,k_i+2}, R_1, R_2, R_{3,1}, R_{3,2}, R_{4,1}, \ldots, R_{4,k_i+2}, R'_{4,3}, \ldots, R'_{4,k_i+2}, R_{5,1}, \ldots, R_{5,k_i+2}, h_0, \ldots, h_{k_i+2})$.
   (d) U_i computes $s_\alpha = r_\alpha + c\alpha$, $s_\beta = r_\beta + c\beta$, $s_\gamma = r_\gamma + c\gamma$, $s_{x_i} = r_{x_i} + cx_i$, $s_{y_i} = r_{y_i} + cy_i$, $s_\tau = r_\tau + c\tau$, $s_{z_{i,2}} = r_{z_{i,2}} + cz_{i,2}$, $\ldots$, $s_{z_{i,k_i+2}} = r_{z_{i,k_i+2}} + cz_{i,k_i+2}$.
3. U_i sends $(k_i, M, C_1, C_2, C_{3,1}, C_{3,2}, C_{4,1}, \ldots, C_{4,k_i+2}, h_0, \ldots, h_{k_i+2}, \pi_c = (c, s_\alpha, s_\beta, s_\gamma, s_{x_i}, s_{y_i}, s_\tau, s_{z_{i,2}}, \ldots, s_{z_{i,k_i+2}}))$ to AP v.
4. If AP v cannot accept k_i and M, then it outputs reject. Otherwise, AP v checks π_c as follows:
   (a) AP v computes $\tilde{R}_1 = \frac{e(\tilde{g},g_2)^{s_\tau} \cdot e(\tilde{g},g_2)^{s_{y_i}} \cdot e(\tilde{g},\omega)^{s_\alpha}}{e(C_1,g_2)^{s_{x_i}}} \left(\frac{e(g_1,g_2)}{e(C_1,\omega)}\right)^{c}$, $\tilde{R}_2 = g_1^{s_{x_i}} h_0 h_1^{r_{y_i}} \prod_{j=2}^{k_i+2} h_j^{r_{z_{i,j}}} C_2^{-c}$, $\tilde{R}_{3,1} = g_1^{s_\beta} C_{3,1}^{-c}$, $\tilde{R}_{3,2} = C_{3,1}^{s_{y_i}} C_{3,2}^{-c}$, $\tilde{R}_{4,1} = g_1^{s_\gamma} C_{4,1}^{-c}$, $\tilde{R}_{4,2} = C_{4,1}^{s_{y_i}} C_{4,1}^{-c}$, $\tilde{R}_{4,3} = C_{4,2}^{s_{y_i}} C_{4,2}^{-c}$, $\tilde{R}'_{4,3} = C_{4,1}^{s_{z_{i,2}}} C_{4,1}^{-c}$, $\ldots$, $\tilde{R}_{4,k_i+2} = C_{4,k_i+1}^{s_{y_i}} C_{4,k_i+1}^{-c}$, $\tilde{R}'_{4,k_i+2} = C_{4,1}^{s_{z_{i,k_i+2}}} C_{4,1}^{-c}$, $\tilde{R}_{5,0} = h_0^{s_\beta} h_0^{-c}$, $\ldots$, $\tilde{R}_{5,k_i+2} = h_{k_i+2}^{s_\beta} h_{k_i+2}^{-c}$.
   (b) AP v checks $c \stackrel{?}{=} \mathcal{H}(gpk, ID_v, k_i, M, C_1, C_2, C_{3,1}, C_{3,2}, C_{4,1}, \ldots, C_{4,k_i+2}, \tilde{R}_1, \tilde{R}_2, \tilde{R}_{3,1}, \tilde{R}_{3,2}, \tilde{R}_{4,1}, \ldots, \tilde{R}_{4,k_i+2}, \tilde{R}'_{4,3}, \ldots, \tilde{R}'_{4,3}, \tilde{R}_{5,0}, \ldots, \tilde{R}_{5,k_i+2}, h_0, \ldots, h_{k_i+2})$.

5. If π_c is a valid proof, then AP v adds $reg_{d+1} = \{k_i, (C_1, C_2), (C_{3,1}, C_{3,2}), (C_{4,1}, \ldots, C_{4,k_i+2}, h_0, \ldots, h_{k_i+2}), \pi_c\}$ to Log_v. Otherwise, it outputs reject.

U_i makes an NIZK proof that the ciphertext C_2 is a ciphertext against the valid identification mpk_i by using a part of membership certificate y_i. For $\ell = 0, 1, \ldots, k_i + 1$, the discrete logarithm of $C_{4,\ell+1}$ based on $C_{4,\ell}$ is y_i and the discrete logarithm of $C_{4,\ell+1}$ based on $C_{4,1}$ is $z_{i,\ell}$ ($C_{4,\ell+1} = C_{4,\ell}^{y_i} = C_{4,1}^{z_{i,\ell}}$). $z_{i,1} = y_i$ holds since $C_{4,2} = C_{4,1}^{z_{i,1}} = C_{4,1}^{y_i}$, and $z_{i,2} = y_i^2$ holds since $C_{4,3} = C_{4,1}^{z_{i,1}} = C_{4,2}^{y_i} = (C_{4,1}^{y_i})^{y_i} = C_{4,1}^{y_i^2}$. To repeat the above procedure, $z_{i,\ell} = y_i^\ell$ ($\ell = 1, \ldots, k_i + 2$) holds. Therefore, $C_2 = g_1^{x_i} \prod_{j=0}^{k_i+2} h_j^{z_{i,j}} = g_1^{x_i + \beta \sum_{j=0}^{k_i+2} a_j y_i^j} = mpk_i (g_1^{f_{k_i}(y_i)})^\beta$ hold.

– $\mathsf{Auth}(\mathsf{Verify}\langle gpk, ask_v, Log_v\rangle, \mathsf{Proof}\langle gpk, msk_i\rangle)$

On the ℓ-th authentication ($1 \le \ell \le k_i + 1$), U_i and AP execute Auth as follows: Note that $C_{3,\ell} = (g_1^{y_i^{\ell-1}})^\beta$ and $C_{3,\ell+1} = (g_1^{y_i^\ell})^\beta$ have already been computed in the $(\ell-1)$-th authentication. Let Grant algorithm be the 0-th authentication.

1. U_i computes $C_{3,\ell+2} = C_{3,\ell+1}^{y_i} (= (g_1^{y_i^\ell})^\beta)$.
2. U_i computes $\pi_\ell = SPK\{(y_i) : C_{3,\ell+1} = C_{3,\ell}^{y_i} \wedge C_{3,\ell+2} = C_{3,\ell+1}^{y_i}\}(ID_v)$. Concretely, U_i computes π_ℓ as follows:
   (a) U_i chooses $r_{y_i} \in_R \mathbb{Z}_p$.
   (b) U_i computes $R_1 = C_{3,\ell}^{r_{y_i}}$, $R_2 = C_{3,\ell+1}^{r_{y_i}}$, $c = \mathcal{H}(gpk, \ell, ID_v, R_1, R_2)$ and $s_{y_i} = r_{y_i} + c y_i$.
3. U_i sends $\pi_\ell = (\ell, c, s_{y_i}, C_{3,\ell}, C_{3,\ell+1}, C_{3,\ell+2})$ to AP v.
4. AP v searches $(C_{3,\ell}, C_{3,\ell+1})$ from Log_v. There are three cases as follows:

   Case-1 : If there exist $C_{3,\ell}$ and $C_{3,\ell+1}$ on reg_d and $\ell \le k_i$, then AP v computes $\tilde{R}_1 = C_{3,\ell}^{s_{y_i}} C_{3,\ell+1}^{-c}$ and $\tilde{R}_2 = C_{3,\ell+1}^{s_{y_i}} C_{3,\ell+2}^{-c}$, and checks $c \stackrel{?}{=} \mathcal{H}(gpk, \ell, ID_v, \tilde{R}_1, \tilde{R}_2)$. If the checking condition c holds, then AP v adds $C_{3,\ell+2}$ and π_ℓ to $reg_d \in Log_v$, and outputs accept.

   Case-2 : If there exist $C_{3,\ell}$ and $C_{3,\ell+1}$ on reg_d and $\ell = k_i + 1$, then AP v outputs reject, and executes Trace.

   Case-3 : Otherwise, neither $C_{3,\ell}$ and $C_{3,\ell+1}$ exist nor the proof is valid, then AP v outputs reject.

In the ℓ-th authentication, U_i only computes $C_{3,\ell+2} = C_{3,\ell+1}^{y_i}$, where $C_{3,\ell+1} = (g_1^\beta)^{y_i^\ell}$, and an NIZK proof π_ℓ that the discrete logarithm of $C_{3,\ell+2}$ based on $C_{3,\ell+1}$ is the same as the discrete logarithm of $C_{3,\ell+1}$ based on $C_{3,\ell}$. We explain the relation between grant messages and the 1st authentication in Fig. 1. In Grant, the knowledge of the discrete logarithm of $C_{3,2}$ based on $C_{3,1}$ has already been proven. Therefore, if $\pi 1$ is valid, then the discrete logarithm of $C_{3,3}$ based on $C_{3,2}$ is y_i, which is a part of a valid membership

certificate. This is called the sequence-of-zero-knowledge-proof mechanism whose concept is the same as the GR97 scheme.

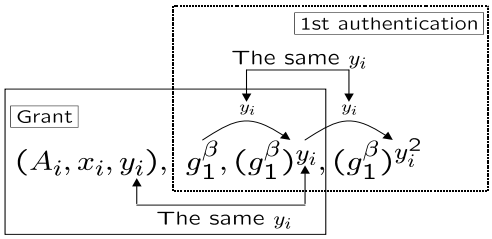

Fig. 1. The relation between Grant and the 1st authentication

– $\mathsf{Trace}(gpk, Log_v)$
Let $reg_d = \{k_i, (C_1, C_2), (C_{3,1}, \ldots, C_{3,k_i+3}), (C_{4,1}, \ldots, C_{4,k_i+2}, h_0, \ldots, h_{k_i+2}), \pi_c, (\pi_1, \ldots, \pi_{k_i+1})\}$ be the log in the (k_i+1)-th authentication.

1. AP v computes $C_2 / \prod_{j=1}^{k_i+3} C_{3,j}^{a_{j-1}} = g_1^{x_i}$.
2. If there exists $(i, g_1^{x_i}) \in \mathsf{LIST}$, then output i. Otherwise, if $g_1^{x_i}$ is not included in LIST, then AP v verifies π_c and all π_j $(j = 1, 2, \ldots, k_i+1)$. If there is an invalid proof, then AP v outputs "AP". Otherwise, all proofs are valid, then AP v outputs "GM".

3.4 Application of Our scheme

Our scheme is characterized by (1) the allowable number selectability with constant size secret keys, (2) relaxed anonymity, and (3) an efficient authentication algorithm Auth. As a natural requirement in the SaaS environment, a user who spends much money has to be given the right to more accesses to AP compared to other users. It can be achieved by the property (1). To improve the application providing strategy, e.g., long tail marketing [21], APs want to obtain the number of accesses by each user. However, from the viewpoint of users, accesses to services do not want to be linked to preserve one's own tastes and habits. The property (2) can achieve these different requirements, simultaneously. There is a situation in which each authentication requires small computations although the first certificate issuing requires large computations. We assume that a power-restricted device, e.g., a smart card, a cell phone, and so on, is used for authentication. First, a membership certificate is embedded in this power-restricted device. When users want to be granted a service, they go to a service counter, insert the power-restricted device into a high-spec machine, and execute the Grant algorithm by using the computational power of the high-spec machine. This is a natural situation, e.g., prepaid rail pass cards with built-in IC chips, cell phones equipped with an electronic money function, and so on. For example, SUICA (Super Urban Intelligent CArd) [7] is known as prepaid rail pass cards in Japan. These cards can be refilled using more money at card vending machines in train stations, and can be read by a card reader to enter the station. In this

kind of services, each authentication requires small computations. Therefore, our scheme (with the property (3)) is suitable to realize these services. In Grant, we assume that a user U_i cannot be identified by AP from M and k_i. For example, a user who spends much money has to be given the right to more accesses to AP compared to other users. Then, each AP needs both the total amount of money and whether the user is a group member or not. In the above situation, M can be considered as the total amount of money.

4 Comparison

In this section, we compare our scheme with previous k-TAA schemes. In our scheme, the size of Log_v is $O(K)$, where $K = \sum_{i=1}^{N} k_i$ and N is the number of application group members. The log size of the previous k-TAA schemes [1, 13, 14, 17, 18] is $O(Nk)$, namely, the size of Log_v is similar to that of the previous schemes. The TFS04 [17] and the NS05 [14] scheme do not enable the constant proving cost. Although the TS06 scheme and the ASM06 scheme enable the constant proving cost, each AP has to publish k signatures. The Nguyen06 [13] scheme enable the constant proving cost without $O(k)$ public keys. Note that the number of public keys of the Nguyen06 scheme is $O(|AG|)$ where $|AG|$ is the number of users in application group managed by an AP, since this scheme enable the dynamic property using an accumulator [4]. If the dynamic property is deleted from the Nguyen06 scheme (namely update algorithm is deleted), then this scheme enable the constant proving cost with constant size public key. However, these schemes assume the same k for all users. In the selectable allowable number setting, the number of the secret key of U_i depends on k_i under the simple construction (See Section 3.1). Our selectable scheme enables the constant proving cost with constant size public key and secret key without increasing the number of secret and public key. In addition, our scheme enables a higher efficiency compared with previous k-TAA schemes. Concretely, in our Auth algorithm, a user requires 3 exponentiations and 1 hash function, and an AP requires 4 exponentiations and 2 scalar multiplications, whereas, the Nguyen06 scheme (without update algorithm), a user requires 22 exponentiations, 27 scalar multiplications and 1 hash function, and an AP requires 21 exponentiations, 20 scalar multiplications, 6 pairings and 1 hash function. (See Section 4.3 in [13] for details). In the previous schemes constructed by the tracing tag mechanism, the AP sends a challenge number for each authentication execution to extract a user's public key stored in LIST. Therefore, the user will be able to compute a part of the proof after she receives the challenge number. Our scheme enables full off-line computation that ALL proofs are pre-computable before executing Auth. In [1, 13], the length of the proof of knowledge sent by the user is 500 Bytes, and it is 700 Bytes in [18]. On the other hands, in our scheme, the length of the proof of knowledge sent by the user is only 100 Bytes. We insist that these efficiencies are due to relaxed anonymity offering a tradeoff between privacy preservation and efficiency. This result due to the fact that the efficiency of

group signature schemes can be drastically improved when unlinkability is not taken into account [15].

5 Security

In this section, we show the sketch of security proofs.

Theorem 1. *The proposed scheme satisfies relaxed anonymity under the* $(k+2)$*-PDDH assumption and the q-SDH assumption in the random oracle model.*

Proof. In our scheme, C_1 is a group signature [6] using a SDH-tuple member certificate. If $\mathcal{A}$ can guess $b \in \{0,1\}$, then $\mathcal{A}$ can also break the anonymity of the group signature scheme [6]. Next, we construct an algorithm $\mathcal{B}$ to break the $(k+2)$-PDDH problem by using an adversary $\mathcal{A}$ which breaks relaxed anonymity. Let $(h'', h', h'_1, \ldots, h'_{k+2})$ be a $(k+2)$-PDDH instance, where $k = Min\{k_{v_0,i_0}, k_{v_0,i_1}, k_{v_1,i_0}, k_{v_1,i_1}\}$ which is defined in the relaxed anonymity game. $\mathcal{B}$ sets $g_2 := h''$ and $g_1 = h'$, and chooses other secret keys, the same as in the real scheme. For U_{i_0}, $\mathcal{B}$ sets $y_{i_0} := \log_{h'} h'_1$. For U_{i_1}, $\mathcal{B}$ chooses $y_{i_1} \in_R \mathbb{Z}_p^*$. For $\mathcal{O}_{\mathsf{Join\text{-}U}}(U_{i_0})$ and $\mathcal{O}_{\mathsf{Grant\text{-}U}}(U_{i_0})$ with the AP v_0 (which is executed in $\mathcal{O}_{query}$), $\mathcal{B}$ computes the simulated NIZK proof which includes the backpatch of the hash function without knowing y_{i_0}. Note that $C_{3,1} = g_1^\beta = (h')^\beta$ and $C_{3,2} := (h'_1)^\beta$ where $\beta \in_R \mathbb{Z}_p^*$. If $b = 0$, then the above simulation is executed twice for $\mathcal{O}_{\mathsf{Grant\text{-}U}}(U_{i_0})$ with the AP v_1. If $b = 1$, then $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Join\text{-}U}}(U_{i_1})$ and $\mathcal{O}_{\mathsf{Grant\text{-}U}}(U_{i_1})$ with the AP v_1 the same as the real scheme. For $\mathcal{O}_{\mathsf{Proof}}(U_{i_0})$ in the ℓ-th authentication, $\mathcal{B}$ computes the simulated NIZK proof which includes the backpatch of the hash function without knowing y_{i_0}, and sets $C_{3,\ell+1} := (h'_\ell)^\beta$ and $C_{3,\ell+2} := (h'_{\ell+1})^\beta$. For $1 \le \ell \le k$, if $h'_\ell = (h')^{(\log_{h'} h'_1)^\ell} = (h')^{y_{i_0}^\ell}$ and $h'_{\ell+1} = (h')^{y_{i_0}^{\ell+1}}$, then this simulation is the same as the real scheme. Therefore $\mathcal{A}$ has an advantage. Otherwise, if h'_ℓ and $h'_{\ell+1}$ are random values, then $\mathcal{A}$ has no advantage. Therefore, if $\mathcal{A}$ outputs b' and $b' = b$, then $\mathcal{B}$ outputs 1. Otherwise, $\mathcal{B}$ outputs 0. □

Theorem 2. *The proposed scheme satisfies detectability under the q-SDH assumption and the* $(k+1)$*-PCDH assumption, where* $k \in \mathbb{Z}_{>0}$ *is the largest allowable number of users.*

Proof. There are three cases, as follows: (1) U_i who is assigned in $reg_d \in Log_v$ executes the authentication protocol more than k_i times. This case does not happen because the number of accesses is restricted by only k_i times in Grant. (2) U_i who is assigned in $reg_d \in Log_v$ executes the authentication protocol more than k_i times by using a transcript of the authentication execution of U_j $(i \neq j)$. In this case, U_i has to compute $C_{3,\ell+2} = C_{3,\ell+1}^{y_j}$ from $(C_{3,1}, C_{3,2}, \ldots, C_{3,\ell+1}) = (g_1^\beta, (g_1^\beta)^{y_j}, \ldots, (g_1^\beta)^{y_j^\ell})$ where a random number $\beta \in_R \mathbb{Z}_p^*$ chosen by U_j and $y_j \in \mathbb{Z}_p^*$ which is a membership certificate of U_j. Then the $(\ell+1)$-PCDH assumption does not hold. (3) U_i with a valid membership certificate (A_i, x_i, y_i) executes the Grant protocol by using (A, x, y) which is a forged membership certificate. If U_i can forge a membership certificate, then the SDH assumption does not hold. □

Theorem 3. *The proposed scheme satisfies exculpability under the q-SDH assumption and the $(k+1)$-PCDH assumption, where $k \in \mathbb{Z}_{>0}$ is the largest allowable number of users.*

Proof. **Exculpability for users** : If $\mathcal{A}$ can output the authentication log for the identification of non-corrupted user U^*, then at least one commitment $C_{3,\ell}$ (which has not appeared in the outputs of $\mathcal{O}_{\mathsf{Proof}}(U^*)$) is included in the log. Then the $(k+1)$-PCDH assumption does not hold since $\ell \leq k+2$.
Exculpability for AP : This is clearly satisfied.
Exculpability for GM : If a public key g_1^x, where the result of the decryption of $C_2 = g_1^{x+\beta\sum_{j=0}^{k+2} a_j y^j}$, is not included in LIST, then there exists a forged membership certificate $A = (g_1\tilde{g}^y)^{\frac{1}{x+\gamma}}$, where γ is the group secret key, since $C_1 = (g_1\tilde{g}^y)^{\frac{1}{x+\gamma}}\tilde{g}^\alpha$ and $C_2 = g_1^{x+\beta\sum_{j=0}^{k+2} a_j y^j}$ have to be computed for the same values x and y on Grant. Then the SDH assumption does not hold since a new valid membership certificate A can be computed without being issued from the GM. □

6 Conclusion

In this paper, we propose for the first time a selectable k-TRAA that enables an allowable number to be assigned for each user and the number of the secret key of each U_i does not depend on k_i. We introduce relaxed anonymity, and our scheme enables a higher efficiency compared with previous k-TAA schemes. This result due to the fact that the efficiency of group signature schemes can be drastically improved when unlinkability is not taken into account [15]. Under relaxed anonymity, user information, such as access to services, is not exposed among different APs. Otherwise, an AP can obtain the number of accesses of own application from each user. This information is important to improve application-providing strategy. This means that relaxed anonymity is useful security requirement to preserve privacy and to satisfy information utilization, simultaneously.

References

1. Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic -TAA. In *SCN*, pages 111–125, 2006.
2. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, pages 614–629, 2003.
3. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.
4. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.
5. David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

6. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT*, pages 193–210, 2006.
7. East Japan Railway Company Homepage. About Suica. `http://www.jreast.co.jp/e/suica-nex/suica.html`.
8. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
9. Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *CRYPTO*, pages 180–197, 1997.
10. Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. Cryptographic primitives enforcing communication and storage complexity. In *Financial Cryptography*, pages 120–135, 2002.
11. IBM. Lotuslive. `https://www.lotuslive.com/`.
12. Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions*, 84-A(5):1234–1243, 2001.
13. Lan Nguyen. Efficient dynamic -times anonymous authentication. In *VIETCRYPT*, pages 81–98, 2006.
14. Lan Nguyen and Reihaneh Safavi-Naini. Dynamic k-times anonymous authentication. In *ACNS*, pages 318–333, 2005.
15. Go Ohtake, Arisa Fujii, Goichiro Hanaoka, and Kazuto Ogawa. On the theoretical gap between group signatures with and without unlinkability. In *AFRICACRYPT*, pages 149–166, 2009.
16. Oracle. SaaS Platform. `http://www.oracle.com/technologies/saas/index.html`.
17. Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-times anonymous authentication (extended abstract). In *ASIACRYPT*, pages 308–322, 2004.
18. Isamu Teranishi and Kazue Sako. k-times anonymous authentication with a constant proving cost. In *Public Key Cryptography*, pages 525–542, 2006.
19. Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *ISPEC*, pages 48–60, 2005.
20. Wikipedia. Software as a service. `http://en.wikipedia.org/wiki/Software_as_a_service`.
21. Wikipedia. The Long Tail. `http://en.wikipedia.org/wiki/The_Long_Tail`.