Dynamic *k*-Times Anonymous Authentication

Lan Nguyen and Rei Safavi-Naini

School of Information Technology and Computer Science
University of Wollongong, Wollongong 2522, Australia
`{ldn01,rei}@uow.edu.au`

Abstract. *k-times anonymous authentication* (k-TAA) schemes allow members of a group to be anonymously authenticated by application providers for a bounded number of times. k-TAA has application in e-voting, e-cash, electronic coupons and anonymous trial browsing of content. In this paper, we extend k-TAA model to *dynamic* k-TAA in which application providers can independently grant or revoke users from their own groups and so have the required control on their clients. We give a formal model for dynamic k-TAA, propose a dynamic k-times anonymous authentication scheme from bilinear pairing, and prove its security. We also construct an ordinary k-TAA from the dynamic scheme and show communication efficiency of the schemes compared to the previously proposed schemes.

1 Introduction

In many scenarios, it is required that authenticated users can anonymously access applications while application providers can decide the number of times users can access their applications. Teranisi et al. [15] proposed k-times anonymous authentication as a solution to this problem. In a k-TAA system, participants are a group manager (GM), a number of application providers (AP) and a group of users. The GM registers users into the group and each AP independently announces the number of times a user can access his application. A registered user can then be anonymously authenticated by APs within their allowed numbers of times and without the need to contact the GM. Dishonest users can be traced by anyone while no one, even the GM or APs, can identify honest users or link two authentication executions performed by the same user. Finally no one, even the GM, is able to successfully impersonate an honest user to an AP.

Applications of k-TAA to e-voting, e-cash, electronic coupons and trial browsing of content have been shown in [15]. A particularly interesting application is trial browsing of content, where each provider allows members of a designated group to anonymously and freely browse content (e.g. movies or music on trial) while he also wants to limit the number of times that users can access the service on trial. Users who try to go over the prescribed quota will be identified and removed. It is shown that none of the known related primitives such as identity escrow/group signature [1, 2, 13, 12], blind signature [7], multiple-show cash [5] and electronic coupon [11], can provide all required properties listed above.

However, k-TAA schemes are inflexible in the sense that the GM decides on the group membership and APs do not have any control over giving users access permission to their services. APs are passive and their role is limited to announcing the number of times a user can access their applications. This requires a lot of trust to be put on the GM and all group members to share all applications offered by all APs. In practice, APs want to select their user groups and grant or revoke access to users independently. For example, in the case of trial browsing, the AP may prefer to give access to users with good profile, or he may require some small fee to be included in his group. Another case is when the AP needs to put also an expiry date on the trial access. We introduce *dynamic* k-times anonymous authentication to provide these properties. In dynamic k-TAA, APs have more control over granting and revoking access to their services, and less trust and computation from the GM is required. Dynamic k-TAA allows APs to restrict access to their services based on not only the number of times but also other factors such as expiry date and so can be used in much wider range of realistic scenarios.

Our contribution
We extend the formal model of k-TAA in [15] to a formal model of *dynamic* k-TAA schemes and construct a dynamic k-TAA scheme from bilinear pairings. We also construct a new k-TAA scheme, and prove security of both schemes under the Strong Diffie-Hellman (SDH) and the Decisional Bilinear Diffie-Hellman (DBDH) assumptions. Dynamic k-TAAs have two new procedures, i.e. granting access and revoking access, that allow APs to grant and revoke users' access, respectively. Security requirements of dynamic systems are similar to the original k-TAAs but are more complex to accommodate the dynamic property.

We propose a new assumption, (l, m, n)-DBDH, and prove that it is implied by the DBDH assumption. And we show that our schemes have lower communication costs than the TFS04 scheme. For example, for k-times authentication with a comparable level of security (1024 bit composite modulus for TFS04) the communication costs of our two schemes are $60k+224$ bytes and $60k+304$ bytes, respectively, while the cost of the TFS04 scheme is $60k + 1617$ bytes. The interactive protocols in our schemes achieve perfect zero-knowledge whereas those in the TFS04 scheme only provide statistical zero-knowledge. (We note that in all cases honest verifier model is used.) Adapting a revocation method proposed in [6] to our system, the revocation costs become independent of the group size and the number of revoked users.

The organization of the paper is as follows. We give the background in section 2 and present the model of dynamic k-TAA schemes in section 3. Section 4 gives descriptions of our dynamic and ordinary k-TAA schemes with their security proofs, and provides efficiency analysis of the schemes and comparison of communication costs with the TFS04 scheme.

2 Preliminaries

Notation. A function $f : \mathbb{N} \rightarrow \mathbb{R}^+$ is called *negligible*, if for every positive number α, there exists a positive integer κ_0 such that for every integer $\kappa > \kappa_0$, it holds that $f(\kappa) < \kappa^{-\alpha}$. Let PT denote polynomial-time, PPT denote probabilistic PT and DPT denote deterministic PT. For a PT algorithm $\mathcal{A}(\cdot)$, "$x \leftarrow A(\cdot)$" denotes an output from the algorithm. For a set $\mathbf{X}$, "$x \leftarrow \mathbf{X}$" denotes an element uniformly chosen from $\mathbf{X}$, and $\#\mathbf{X}$ denotes the number of elements in $\mathbf{X}$. Let "$\mathsf{Pr}[Procedures|Predicate]$" denote the probability that $Predicate$ is true after executing the $Procedures$, $\mathcal{H}_{\mathbf{X}}$ denote a hash function from the set of all finite binary strings $\{0,1\}^*$ onto the set $\mathbf{X}$, and $PK\{x : R(x)\}$ denote a proof of knowledge of x that satisfies the relation $R(x)$.

2.1 Bilinear Groups

Let $\mathbb{G}_1, \mathbb{G}_2$ be additive cyclic groups generated by P_1 and P_2, respectively, whose orders are a prime p, and $\mathbb{G}_T$ be a cyclic multiplicative group with the same order p. Suppose there is an isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ such that $\psi(P_2) = P_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing with the following properties:

1. **Bilinearity:** $e(aP, bQ) = e(P,Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$
2. **Non-degeneracy:** $e(P_1, P_2) \neq 1$
3. **Computability:** There is an efficient algorithm to compute $e(P,Q)$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$

For simplicity, hereafter, we set $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$ but the proposed schemes can be easily modified for the general case when $\mathbb{G}_1 \neq \mathbb{G}_2$.

We define a Bilinear Pairing Instance Generator as a PPT algorithm $\mathcal{G}$ that takes as input a security parameter 1^κ and returns a uniformly random tuple $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_T, e, P)$ of bilinear pairing parameters, including a prime number p of size κ, a cyclic additive group $\mathbb{G}_1$ of order p, a multiplicative group $\mathbb{G}_T$ of order p, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ and a generator P of $\mathbb{G}_1$.

2.2 Complexity Assumptions

q-Strong Diffie-Hellman (q-SDH) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\mathsf{q\text{-}SDH}}(\kappa)$ is negligible.*

$$Adv_{\mathcal{A}}^{\mathsf{q\text{-}SDH}}(\kappa) = \mathsf{Pr}[(\mathcal{A}(\mathbf{t}, P, sP, \ldots, s^qP) = (c, \frac{1}{s+c}P)) \wedge (c \in \mathbb{Z}_p)]$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_T, e, P) \leftarrow \mathcal{G}(1^\kappa)$ *and* $s \leftarrow \mathbb{Z}_p^*$.

The q-SDH assumption is proposed by Boneh and Boyen [3] and is originated from an earlier (and weaker) assumption introduced by Mitsunari et. al. [10]. The assumption informally means that there is no PPT algorithm that can compute a pair $(c, \frac{1}{s+c}P)$, where $c \in \mathbb{Z}_p$, from a tuple $(P, sP, \ldots, s^qP)$, where $s \leftarrow \mathbb{Z}_p^*$.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\mathsf{DBDH}}(\kappa)$ is negligible.*

$$Adv_{\mathcal{A}}^{\mathsf{DBDH}}(\kappa) = |\mathsf{Pr}[\mathcal{A}(\mathbf{t}, aP, bP, cP, e(P,P)^{abc}) = 1] \\ -\mathsf{Pr}[\mathcal{A}(\mathbf{t}, aP, bP, cP, \Gamma) = 1]|$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_T, e, P) \leftarrow \mathcal{G}(1^\kappa)$*,* $\Gamma \leftarrow \mathbb{G}_T^*$ *and* $a, b, c \leftarrow \mathbb{Z}_p^*$*.*

Informally, the DBDH assumption states that there is no PPT algorithm that can distinguish between a tuple $(aP, bP, cP, e(P,P)^{abc})$ and a tuple (aP, bP, cP, Γ), where $\Gamma \leftarrow \mathbb{G}_T^*$ and $a, b, c \leftarrow \mathbb{Z}_p^*$.

The *Discrete Log (DL)* assumption can be found in the full version [14] and it is easy to prove that either SDH or DBDH implies DL.

3 A Model for Dynamic k-TAA schemes

We propose a formal model for dynamic k-TAA and underline the differences between this model and the TFS04 model for ordinary k-TAA in [15].

3.1 Entities and Procedures

Entities in the model are the GM, APs and users; the procedures are setup, joining, bound announcement, granting access, revoking access, authentication and public tracing. In the setup procedure (SETUP), the GM obtains a group public key/group secret key pair, and each AP $\mathcal{V}$ obtains a pair of public and secret keys $(apk_{\mathcal{V}}, ask_{\mathcal{V}})$. AP $\mathcal{V}$ also has an *access group* $AG_{\mathcal{V}}$ which is the set user identities who can access his application, and also some other *public information* $PI_{\mathcal{V}}$. $AG_{\mathcal{V}}$ is initially empty.

The joining procedure ($\mathcal{U}_{JOIN-GM}$, $\mathcal{U}_{JOIN-U}$) allows a user i to join the group by obtaining a member public key/member secret key pair (mpk_i, msk_i) and the GM adds the user's identification and public key to an *identification list* LIST. In the bound announcement procedure (BD-ANN), an AP announces the number of times a group member can access his application by publishing his identity ID, and the upper bound k. An AP $\mathcal{V}$ uses the granting access procedure (GRAN-AP) to give selected group members permission to access the his application. He includes the new member in his access group $AG_{\mathcal{V}}$ and updates his public information $PI_{\mathcal{V}}$. Similarly, in the revoking access procedure (REVO-AP), AP $\mathcal{V}$ can stop a group member from accessing his application by excluding the member from his access group and updating his public information. The authentication procedure ($\mathcal{U}_{AUTH-AP}$,$\mathcal{U}_{AUTH-U}$), between a user i and AP $\mathcal{V}$ succeeds if and only if user i has been granted access and his access has not been revoked, and the number of accesses has not reached the allowed number. AP $\mathcal{V}$ records the transcripts of authentication executions in the authentication log LOG. Tracing procedure TRACE can be executed by anyone using the public information and the authentication log. Possible outputs of the procedure are user i's identity, GM, or NO-ONE which mean "user i tries to access more than the prescribed limit", "the GM published information is not correct", and "the public tracing procedure cannot find any malicious entity", respectively.

The main difference between dynamic k-TAA and the TFS04 model with regard to procedures is that an AP has a pair of public and secret keys and maintains his own access group using two new procedures, granting access and revoking access; and authentication procedure succeeds only if the user has been granted access, the access has not been revoked, and he has not accessed the application over the allowed number of times.

3.2 Oracles

The adversary has access to a number of oracles and can query them according to the description below, to learn about the system and increase his success chance in the attacks.

List oracle model We will use a *list* oracle $\mathcal{O}_{LIST}$ first defined in [15]. The oracle is used to ensure correct correspondence between the identity of a group member and his public key. In *list oracle model* there is a $\mathcal{O}_{LIST}$ oracle which manages the LIST that contains identities and the corresponding public key list. The response of the oracle to a query to view a group member's public key is the member's public key. The oracle allows an entity to choose and write a user's pair of identity and public key to the LIST only if the entity is the user or a colluder with the user. The oracle allows an entity to delete data from LIST only if the entity is the GM or a colluder with the GM.

Other oracles Other oracles in our model are the *join GM* oracle $\mathcal{O}_{JOIN-GM}$, the *join user* oracle $\mathcal{O}_{JOIN-U}$, the *authentication AP* oracle $\mathcal{O}_{AUTH-AP}$, the *authentication user* oracle $\mathcal{O}_{AUTH-U}$, the *query* oracle $\mathcal{O}_{QUERY}$, the *granting user* oracle $\mathcal{O}_{GRAN-AP}$, the *revoking user* oracle $\mathcal{O}_{REVO-AP}$ and the *corrupting AP* oracle $\mathcal{O}_{CORR-AP}$. The $\mathcal{O}_{JOIN-GM}$ oracle, given a user specified by the adversary, performs joining procedure as executed by the honest GM on the user. The $\mathcal{O}_{JOIN-U}$ oracle, given an honest user specified by the adversary, performs joining procedure between the GM and the user. The $\mathcal{O}_{AUTH-AP}$ oracle, given an honest AP and a user from the adversary, makes the AP to perform an authentication procedure with the user. The $\mathcal{O}_{AUTH-U}$ oracle, given an honest user and an AP, makes the user to perform an authentication procedure with the AP. Oracles $\mathcal{O}_{LIST}$, $\mathcal{O}_{JOIN-U}$, $\mathcal{O}_{JOIN-GM}$ and $\mathcal{O}_{AUTH-AP}$ are the same as in TFS04 model and their formal definition can be found in [15].

The $\mathcal{O}_{QUERY}$ oracle gives the adversary the challenged authentication transcript in the D-Anonymity definition and more details can be found in this definition. The $\mathcal{O}_{QUERY}$ oracle first checks if input identities i_1 and i_2 are current members of the input AP with identity ID; if not, it outputs CHEAT. It then proceeds, as defined in [15], by randomly choosing one of the two identities and executing the authentication procedure between the chosen identity and the input AP. The $\mathcal{O}_{AUTH-U}$ oracle performs as defined in [15], but it takes one more input (ID, k) to indicate the AP. The formal definitions of $\mathcal{O}_{QUERY}$ and $\mathcal{O}_{AUTH-U}$ are presented in the full version [14].

We introduce three new oracles, $\mathcal{O}_{GRAN-AP}$, $\mathcal{O}_{REVO-AP}$ and $\mathcal{O}_{CORR-AP}$. The $\mathcal{O}_{GRAN-AP}$ oracle takes as input an honest AP and a user and executes the granting access procedure GRAN-AP by the AP to grant access to the user. The $\mathcal{O}_{REVO-AP}$ oracle takes as input an honest AP and a member of the AP's access group and executes the revoking access procedure REVO-AP by the AP to revoke access from the user. The $\mathcal{O}_{CORR-AP}$ oracle corrupts an AP specified by the adversary and maintains the set $S_{CORR-AP}$ of corrupted APs. The formal definitions and explanation of these oracles are presented in the full version [14].

3.3 Security Requirements

Security requirements of dynamic k-TAA are D-Correctness, D-Anonymity, D-Detectability, D-Exculpability for users and D-Exculpability for the GM. These are similar to requirements Correctness, Anonymity, Detectability, Exculpability for users and Exculpability for the GM defined in the TFS04 model.

In the following we give an informal definition of these requirements. D-Correctness requires that an honest member who is in the access group of an honest AP and has not performed the authentication procedure for more than the allowed number of times, be successfully authenticated by the AP. D-Anonymity means that it is computationally hard for the adversary to distinguish between authentication executions of two honest group members i_1 and i_2 who are in the access group of an AP, and have not performed authentication with the AP for more than the limited number of times, even if the GM, all APs, and all users except i_1 and i_2 are corrupted. D-Detectability means that if a subgroup of corrupted members have performed the authentication procedure with the same honest AP for more than the allowed number of times, then the public tracing procedure using the AP's authentication log outputs NO-ONE with negligible probability. D-Exculpability for users means that the tracing procedure does not output the identity of an honest user even if other users, the GM and all APs are corrupted. D-Exculpability for the GM means that the tracing procedure does not output the honest GM even if all users and all APs are corrupted. The difference between D-Detectability and Detectability is more significant and so the formal definition of D-Detectability is given below. The formal definitions of other requirements can be found in the full version [14].

D-Detectability. The adversary $\mathcal{A}$ is allowed to corrupt all members and the experiment has two stages. In the first stage, the adversary can query the three new oracles ($\mathcal{O}_{GRAN-AP}$, $\mathcal{O}_{REVO-AP}$ and $\mathcal{O}_{CORR-AP}$), $\mathcal{O}_{LIST}$, $\mathcal{O}_{JOIN-GM}$ and $\mathcal{O}_{AUTH-AP}$. After that, all authentication logs of all APs are emptied. Then the adversary continues the experiments, but without access to the revoking oracle $\mathcal{O}_{REVO-AP}$. The adversary wins if he can be successfully authenticated by an honest AP with identity ID and access bound k for more than $k \times \#AG_{ID}$, where $\#AG_{ID}$ is the number of members in the AP's access group. The set $S_{AUTH-AP}$ contains all APs' information used by the $\mathcal{O}_{AUTH-AP}$, and LOG_{ID}

is the authentication log produced by $\mathcal{O}_{AUTH-AP}$ using information of the AP (ID, k). The formula of the experiment is as follows.

```
Experiment $\mathsf{Exp}^{d-decis}_{\mathcal{A},\mathcal{H}}(\kappa)$
$((gpk, gsk), \{(apk, ask)\}) \leftarrow SETUP(1^\kappa)$.
$St \leftarrow \mathcal{A}^{ORACLES}(1^\kappa)$ where $ORACLES = \{\mathcal{O}_{LIST}(\{GM\}^c, \cdot)$,
    $\mathcal{O}_{JOIN-GM}(gpk, gsk, \cdot)$, $\mathcal{O}_{AUTH-AP}(gpk, \cdot, \cdot)$, $\mathcal{O}_{GRAN-AP}(gpk, \cdot, \cdot)$,
    $\mathcal{O}_{REVO-AP}(gpk, \cdot, \cdot)$, $\mathcal{O}_{CORR-AP}(\cdot)\}$.
Empty all LOGs.
$\mathcal{A}^{ORACLES \setminus \{\mathcal{O}_{REVO-AP}(gpk,\cdot,\cdot)\}}(St)$.
If $(\exists (ID, k) \in S_{AUTH-AP} \setminus S_{CORR-AP}$ s.t. $\#LOG_{ID} > k \times \#AG_{ID})$
    Return $TRACE^{\mathcal{O}_{LIST}(\emptyset,\cdot)}(gpk, apk_{ID}, LOG_{ID})$.
Return $\perp$.
```

A dynamic k-TAA scheme provides D-Detectability if the following function $Adv^{d-decis}_{\mathcal{A}}(\kappa)$ is negligible.

$$Adv^{d-decis}_{\mathcal{A}}(\kappa) = \mathsf{Pr}[\mathsf{Exp}^{d-decis}_{\mathcal{A},\mathcal{H}}(\kappa) = \mathsf{NO\text{-}ONE}]$$

Similar to arguments for group signatures [2], these requirements for dynamic k-TAA also imply *unforgeability*, *coalition resistance* and *traceability* that can be informally defined as follows. Unforgeability means that any adversary, who is not in the access group of an AP, can not be authenticated by the AP without colluding with some group members, or both of the GM and the AP. Coalition resistance means that a colluding subset of group members can not produce a new member public key/secret key pair which has not been generated in the joining procedure. Traceability means that if a user has accessed an AP for more than the bound number of times, then that user can be traced from the public information and the AP's authentication log.

4 A Dynamic k-TAA scheme

4.1 Overview

Our proposed scheme is constructed in cyclic groups with bilinear mapping. For simplicity, we present the scheme when the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are the same but the scheme can be easily modified for the general case when $\mathbb{G}_1 \neq \mathbb{G}_2$.

Suppose a bilinear pairing tuple $(p, \mathbb{G}_1, \mathbb{G}_T, e, P)$ is given, the GM's group secret key is $\gamma \leftarrow \mathbb{Z}_p^*$ and group public key is $(P, P_{pub} = \gamma P, P_0 \leftarrow \mathbb{G}_1, H, G_1, G_2 \leftarrow \mathbb{G}_1, \Delta = e(P, P))$. In the joining procedure, a user obtains a membership public key/secret key pair $((a, S), x)$ from the GM such that $S = \frac{1}{\gamma + a}(xP + P_0)$, where P and P_0 are in the GM's public key, γ is the GM's secret key and x is randomly generated by both the user and the GM but is only known to the user. The user can be anonymously authenticated as a group member by proving the knowledge of (a, S, x) such that $e(S, aP + P_{pub}) = e(xP + P_0, P)$.

An AP publishes k *tag bases* to be used for up to k times user access to the AP's service. A tag base is a pair $(\Theta_i, \check{\Theta}_i)$ of $\mathbb{G}_T$'s elements. In an authentication execution, a group member interacts with the AP and constructs a tag $(\Gamma, \check{\Gamma}) = (\Theta_i^x, (\Delta^l \check{\Theta}_i)^x)$ to be sent to the AP, where $\Delta = e(P, P)$, and the AP has randomly selected l. The group member also proves the knowledge of (i, x) satisfying the above equation. If the member uses the same tag base to compute another tag $(\Gamma', \check{\Gamma}') = (\Theta_i^x, (\Delta^{l'} \check{\Theta}_i)^x)$, anyone can find these from the AP's authentication log (since $\Gamma = \Gamma'$) and use it to compute $(\check{\Gamma}/\check{\Gamma}')^{1/(l-l')} = \Delta^x$, which is published in the joining procedure. However, if the member does not use the same tag base twice, based on the DBDH assumption his anonymity is protected. Similar to [6], we will use dynamic accumulators to provide the dynamic property. However we will use a new accumulator scheme based on the SDH assumption. Each AP has a public key/secret key pair $((Q, Q_{pub}), s)$, where $Q_{pub} = sQ$. To grant access to a member with a public key (a, S), the AP accumulates the value a of the public key into an *accumulated value* $V \leftarrow (s + a)V$, and the member obtains the old accumulated value as the witness W. The member shows that the AP has granted access to him by proving the knowledge of (a, W) such that $e(W, aQ + Q_{pub}) = e(V, Q)$. To revoke access from a member, the AP computes a new *accumulated value* $V \leftarrow 1/(s + a)V$.

4.2 Description

Setup.
For GM: On input a security parameter 1^κ, the Bilinear Pairing Instance Generator generates a tuple $(p, \mathbb{G}_1, \mathbb{G}_T, e, P)$ as in Section 2.2. GM selects $P_0, H, G_1, G_2 \leftarrow \mathbb{G}_1$, $\gamma \leftarrow \mathbb{Z}_p^*$, and sets $P_{pub} = \gamma P$ and $\Delta = e(P, P)$. The group public and secret keys are $gpk = (P, P_{pub}, P_0, H, G_1, G_2, \Delta)$ and $gsk = \gamma$, respectively. The identification list LIST of group members is initially empty.
For APs: AP $\mathcal{V}$ selects $Q \leftarrow \mathbb{G}_1$, $s \leftarrow \mathbb{Z}_p^*$, $\Lambda, \Upsilon \leftarrow \mathbb{G}_T$, and sets $Q_{pub} = sQ$. The public and secret keys for the AP are $apk = (Q, Q_{pub}, \Lambda, \Upsilon)$ and $ask = s$, respectively. AP maintains an authentication log LOG, an *accumulated value*, which is published and updated after granting or revoking a member, and a public archive ARC (as the other public information PI in the formal model), which is a list of 3-tuples. The first component of the tuple is an element in the public key of a member, who was granted or revoked from accessing the AP. The second component is a single bit indicating whether the member was granted (1) or revoked (0). The third component is the accumulated value after granting or revoking the member. Initially, the accumulated value is set to $V_0 \leftarrow \mathbb{G}_1$ and LOG and ARC are empty.

Joining.
A user U_i can join the group as follows.

1. User U_i selects $x', r \leftarrow \mathbb{Z}_p^*$, and sends a commitment $C' = x'P + rH$ of x' to the GM.
2. The GM sends $y, y' \leftarrow \mathbb{Z}_p^*$ to U_i.
3. User U_i computes $x = y + x'y'$ and $(C, \beta) = (xP, \Delta^x)$, then adds new data (i, β) to the identification list LIST. Next, U_i sends (C, β) to the GM with a

standard proof $Proof_1 = PK\{(x, r') : C = xP \wedge yP + y'C' - C = r'H\}$ to show that C is correctly computed from C', y, y' and U_i knows x satisfying $C = xP$.

4. The GM verifies that (i, β) is an element of the LIST, $\beta = e(C, P)$ and the proof is valid. Then, the GM generates $a \leftarrow \mathbb{Z}_p^*$ different from all corresponding previously generated values, computes $S = \frac{1}{\gamma + a}(C + P_0)$, and sends (S, a) to user U_i.
5. User U_i confirms that equation $e(S, aP + P_{pub}) = e(C + P_0, P)$ is satisfied. The new member U_i's secret key is $msk = x$, and his public key is $mpk = (a, S, C, \beta)$.

Bound announcement.
An AP publishes his identity ID and a number k as the bound. Let $(\Theta_j, \check{\Theta}_j) = \mathcal{H}_{\mathbb{G}_T \times \mathbb{G}_T}(ID, k, j)$ for $j = 1, ..., k$. We call $(\Theta_j, \check{\Theta}_j)$ the j^{th} *tag base* of the AP.

Granting access.
An AP grants access to a user U_i with public key $mpk = (a, \cdot, \cdot, \cdot)$ as follows. Suppose there are j tuples in the AP's ARC and the AP's current accumulated value is V_j. The AP computes a new accumulated value $V_{j+1} = (s + a)V_j$, adds $(a, 1, V_{j+1})$ to his ARC. The user U_i can form his access key $mak = (j + 1, W)$, where $W = V_j$. The user keeps a counter ι, which is initially set to 0.

Revoking access.
An AP revokes access from a user U_i with public key $mpk = (a, \cdot, \cdot, \cdot)$ as follows. Suppose there are j tuples in the AP's ARC and the AP's current accumulated value is V_j. The AP computes a new accumulated value $V_{j+1} = 1/(s + a)V_j$, and adds $(a, 0, V_{j+1})$ to ARC.

Authentication.
An AP (ID, k), whose public key and current accumulated value are $apk = (Q, Q_{pub}, \Lambda, \Upsilon)$ and V respectively, authenticates a member M with public and secret keys $mpk = (a, S, C, \beta)$ and $msk = x$, respectively, as follows.

1. Member M increases counter ι. If value $\iota > k$, then M sends $\perp$ to the AP and stops. Otherwise, M runs the algorithm Update (see Section 4.3 for this algorithm) to update his access key $mak = (j, W)$.
2. The AP sends a random integer $l \leftarrow \mathbb{Z}_p^*$ to M.
3. Member M computes tag $(\Gamma, \check{\Gamma}) = (\Theta_\iota^x, (\Delta^l \check{\Theta}_\iota)^x)$ using the ι^{th} tag base $(\Theta_\iota, \check{\Theta}_\iota)$, and sends $(\Gamma, \check{\Gamma})$ to the AP with $Proof_2 = PK\{(\iota, a, S, x, W) : \Gamma = \Theta_\iota^x \wedge \check{\Gamma} = (\Delta^l \check{\Theta}_\iota)^x \wedge e(S, aP + P_{pub}) = e(xP + P_0, P) \wedge e(W, aQ + Q_{pub}) = e(V, Q)\}$.
4. If the proof is valid and if Γ is different from all corresponding tags in the AP's LOG, the AP adds tuple $(\Gamma, \check{\Gamma}, l)$ and the proof to the LOG, and outputs accept. If the proof is valid and Γ is already written in the LOG, the AP adds tuple $(\Gamma, \check{\Gamma}, l)$ and the proof to the LOG, outputs (detect,LOG) and stops. If the proof is invalid, the AP outputs reject and stops.

Public tracing.
The identity of a malicious user can be traced from an AP's LOG as follows.

1. Look for two entries $(\mathit{\Gamma}, \check{\mathit{\Gamma}}, l, Proof)$ and $(\mathit{\Gamma}', \check{\mathit{\Gamma}}', l', Proof')$ in the LOG, such that $\mathit{\Gamma} = \mathit{\Gamma}'$ and $l \neq l'$, and that $Proof$ and $Proof'$ are valid. If no such entry can be found, output NO-ONE.
2. Compute $\beta = (\check{\mathit{\Gamma}}/\check{\mathit{\Gamma}}')^{1/(l-l')} = ((\mathit{\Delta}^l\check{\mathit{\Gamma}})^x/(\mathit{\Delta}^{l'}\check{\mathit{\Gamma}}')^x)^{1/(l-l')} = \mathit{\Delta}^x$, and look for a pair (i, β) from the LOG. Output member identity i, or if no such (i, β) can be found conclude that the GM has deleted some data from the LOG, and output GM.

4.3 Details

Proof$_2$.
The member M computes the proof as follows:

1. Select $v \leftarrow \mathbb{Z}_p^*$, and compute the perfect commitment $\mathit{\Omega} = \mathit{\Lambda}^x\mathit{\Upsilon}^v$ of x.
2. Publish $\mathit{\Omega}$ and proofs of knowledge of the following:
   - $Proof_{2a} = PK\{(\iota, x, v) : \mathit{\Gamma} = \mathit{\Theta}_\iota^x \wedge \check{\mathit{\Gamma}} = (\mathit{\Delta}^l\check{\mathit{\Theta}}_\iota)^x \wedge \mathit{\Omega} = \mathit{\Lambda}^x\mathit{\Upsilon}^v\}$.
   - $Proof_{2b} = PK\{(a, S, x, W, v) : e(S, aP + P_{pub}) = e(xP + P_0, P) \wedge e(W, aQ + Q_{pub}) = e(V, Q) \wedge \mathit{\Omega} = \mathit{\Lambda}^x\mathit{\Upsilon}^v\}$.

The $Proof_{2a}$ can be constructed the same as proof 1 in [15] using the standard techniques. We describe $Proof_{2b}$ as follows. Most of the pairing operations can be pre-computed.

1. Generate $r_1, r_2, r_3, k_0, ..., k_8 \leftarrow \mathbb{Z}_p$ and compute $U_1 = S + r_1H$; $U_2 = W + r_2H$; $R = r_1G_1 + r_2G_2 + r_3H$; $T_1 = k_1G_1 + k_2G_2 + k_3H$; $T_2 = k_4G_1 + k_5G_2 + k_6H - k_7R$; $\mathit{\Pi}_1 = e(P,P)^{k_0}e(U_1,P)^{-k_7}e(H,P)^{k_4}e(H,P_{pub})^{k_1}$; $\mathit{\Pi}_2 = e(U_2,Q)^{-k_7}e(H,Q)^{k_5}e(H,Q_{pub})^{k_2}$; $\mathit{\Pi}_3 = \mathit{\Lambda}^{k_0}\mathit{\Upsilon}^{k_8}$
2. Compute $c = \mathcal{H}_{\mathbb{Z}_p}(P||P_{pub}||P_0||H||G_1||G_2||\mathit{\Delta}||Q||Q_{pub}||\mathit{\Lambda}||\mathit{\Upsilon}||\mathit{\Omega}||ID||k||l||V$ $||U_1||U_2||R||T_1||T_2||\mathit{\Pi}_1||\mathit{\Pi}_2||\mathit{\Pi}_3)$
3. Compute in $\mathbb{Z}_p$: $s_0 = k_0 + cx$; $s_1 = k_1 + cr_1$; $s_2 = k_2 + cr_2$; $s_3 = k_3 + cr_3$; $s_4 = k_4 + cr_1a$; $s_5 = k_5 + cr_2a$; $s_6 = k_6 + cr_3a$; $s_7 = k_7 + ca$; $s_8 = k_8 + cv$
4. Output $(U_1, U_2, R, c, s_0, ..., s_8)$

Verification of $Proof_{2b}$. Checking the following equation $c \overset{?}{=} \mathcal{H}_{\mathbb{Z}_p}(P||P_{pub}||P_0||H$ $||G_1||G_2||\mathit{\Delta}||Q||Q_{pub}||\mathit{\Lambda}||\mathit{\Upsilon}||\mathit{\Omega}||ID||k||l||V||U_1||U_2||R||s_1G_1 + s_2G_2 + s_3H - cR||$ $s_4G_1 + s_5G_2 + s_6H - s_7R||e(P,P)^{s_0}e(U_1,P)^{-s_7}e(H,P)^{s_4}e(H,P_{pub})^{s_1}e(P_0,P)^c$ $e(U_1,P_{pub})^{-c}||e(U_2,Q)^{-s_7}e(H,Q)^{s_5}e(H,Q_{pub})^{s_2}e(V,Q)^ce(U_2,Q_{pub})^{-c}||\mathit{\Lambda}^{s_0}\mathit{\Upsilon}^{s_8}$ $\mathit{\Omega}^{-c})$.

Update.
Suppose the AP's ARC currently has n tuples, the member M with the public key $(a, \cdot, \cdot, \cdot)$ and the access key (j, W_j) computes a new access key as follows.

```
for $(k = j + 1; k + +; k \leq n)$ do
    retrieve from ARC the $k^{th}$ tuple $(u, b, V_k)$;
    if $b = 1$, then $W_k = V_{k-1} + (u - a)W_{k-1}$
    else $W_k = (1/(u - a))(W_{k-1} - V_k)$ end if;
end for;
return $(n, W_n)$;
```

Public Inspection.
Any party can run this algorithm to assure the correctness of an AP's public archive ARC. With such an algorithm, we can assume that ARC is always updated correctly. Any party, after a change on ARC, can retrieve the new tuple (u, b, V_k). If $(b = 1)$ then he checks if $e(V_{k-1}, aQ + Q_{pub}) \overset{?}{=} e(V_k, Q)$; otherwise, he checks if $e(V_k, aQ + Q_{pub}) \overset{?}{=} e(V_{k-1}, Q)$;

4.4 Correctness and Security

Correctness and security of our scheme is stated in Theorem 1, whose proof can be found in the full version [14].

Theorem 1. *In the random oracle model and the list oracle model, our dynamic k-TAA scheme provides (i) Correctness; (ii) D-Anonymity under the Decisional Bilinear Diffie-Hellman assumption; (iii) D-Detectability under the q-Strong Diffie-Hellman assumption, where q is the upper bound of the group size; (iv) D-Exculpability for users under the Discrete Log assumption on* $\mathbb{G}_1$*; (v) D-Exculpability for the GM under the q-Strong Diffie-Hellman assumption, where q is the upper bound of the group size.*

Theorem 1's proof is based on the following lemmas, definition and theorem. Lemma 1's proof can be found in the full version [14].

Lemma 1. *Under the Discrete Log assumption on* $\mathbb{G}_1$*, the interactive protocol corresponding to* $Proof_2$ *by the Fiat-Shamir heuristic [8] is an honest verifier perfect zero-knowledge proof.*

Lemma 2. *Suppose a PPT adversary can corrupt all APs and all users and can query the oracle* $\mathcal{O}_{JOIN-GM}$*. Let* $S = \{((a_i, S_i, \cdot, \cdot), x_i)\}_{i=1}^q$ *be the set of public key/secret key pairs of all member which are obtained by the adversary using* $\mathcal{O}_{JOIN-GM}$*. If the adversary can output a new valid member public key/secret key pair* $((a^*, S^*, \cdot, \cdot), x^*) \notin S$*, then the q-SDH assumption does not hold.*

Proof. Suppose there is a PPT adversary $\mathcal{A}$ such that from set $S = \{((a_i, S_i, \cdot, \cdot), x_i)\}_{i=1}^q$ of public key/secret key pairs of all members, obtained by $\mathcal{O}_{JOIN-GM}$, $\mathcal{A}$ can generate the public key/secret key pair $((a^*, S^*, \cdot, \cdot), x^*) \notin S$ of a new valid member. We show a construction of a PPT adversary $\mathcal{B}$ that can break the q-SDH assumption. Suppose a tuple *challenge* $= (Q, zQ, \ldots, z^qQ)$ is given, where $z \leftarrow \mathbb{Z}_p^*$, we show that $\mathcal{B}$ can compute $(c, 1/(z + c)Q)$, where $c \in \mathbb{Z}_p$ with non-negligible probability. We consider two cases.
Case 1: This is a trivial case, where $\mathcal{A}$ outputs $S^* \in \{S_1, ..., S_q\}$ with non-negligible probability. In this case, $\mathcal{B}$ chooses $\gamma \leftarrow \mathbb{Z}_p^*$ and $H, G_1, G_2 \leftarrow \mathbb{G}_1$, gives

$\mathcal{A}$ the group public key $(P = Q, P_{pub} = \gamma P, P_0 = zQ, H, G_1, G_2, \Delta = e(P, P))$, simulates a GM and a set of possible users, and simulates a set of possible APs with their public/secret key pairs. Then $\mathcal{B}$ can simulate the oracle $\mathcal{O}_{JOIN-GM}$ that $\mathcal{A}$ needs to access. Suppose a set of keys $S = \{((a_i, S_i, \cdot, \cdot), x_i)\}_{i=1}^{q}$ is generated and $\mathcal{A}$ outputs a new $((a^*, S^*, \cdot, \cdot), x^*)$ with non-negligible probability such that $S^* \in \{S_1, ..., S_q\}$. Suppose $S^* = S_j$, where $j \in \{1, ..., q\}$, then $\frac{1}{a^*+\gamma}(x^* P + P_0) = \frac{1}{a_j+\gamma}(x_j P + P_0)$, so $(a_j - a^*)P_0 = (a^* x_j - a_j x^* + x_j\gamma - x^*\gamma)P$. Therefore, z is computable by $\mathcal{B}$ from this, and so is $(c, 1/(z + c)Q)$, for any $c \in \mathbb{Z}_p$.

Case 2: This is when the first case does not hold. That means $\mathcal{A}$ outputs $S^* \notin \{S_1, ..., S_q\}$ with non-negligible probability. Then $\mathcal{B}$ plays the following game:

1. Generate $\alpha, a_i, x_i \leftarrow \mathbb{Z}_p^*$, $i = 1, ..., q$, where a_is are different from one another, and choose $m \leftarrow \{1, ..., q\}$.
2. Suppose $\gamma = z - a_m$ ($\mathcal{B}$ does not know γ). Then $\mathcal{B}$ can compute the following P, P_{pub}, P_0 from the tuple *challenge*.

$$P = \prod_{i=1, i\neq m}^{q} (z + a_i - a_m)Q$$

$$P_{pub} = \gamma P = (z - a_m) \prod_{i=1, i\neq m}^{q} (z + a_i - a_m)Q$$

$$P_0 = \alpha \prod_{i=1}^{q} (z + a_i - a_m)Q - x_m \prod_{i=1, i\neq m}^{q} (z + a_i - a_m)Q$$

3. Generate $H, G_1, G_2 \leftarrow \mathbb{G}_1$ and give $\mathcal{A}$ the group public key $(P, P_{pub}, P_0, H, G_1, G_2, \Delta = e(P, P))$. Simulate a GM, a set of possible users and a set of possible APs with their public/secret key pairs.
4. $\mathcal{B}$ can simulate the oracle $\mathcal{O}_{JOIN-GM}$ that $\mathcal{A}$ needs to access as follows. Suppose $\mathcal{A}$ wants an execution of the joining procedure between the GM (controlled by $\mathcal{B}$) and a user (controlled by $\mathcal{A}$). As being able to extract information from $\mathcal{A}$, after receiving the commitment C' from $\mathcal{A}$, $\mathcal{B}$ can find x', r and generate y, y', a in the joining procedure so that the prepared a_i, x_i above are computed in the protocol to be the corresponding parts of the user i's keys. $\mathcal{B}$ can compute S_i as follows:
   - If $i = m$, then

$$S_m = \frac{1}{a_m + \gamma}(x_m P + P_0) = \alpha \prod_{i=1, i\neq m}^{q} (z + a_i - a_m)Q$$

   This is computable from the tuple *challenge*.
   - If $i \neq m$, then

$$S_i = \frac{1}{a_i + \gamma}(x_i P + P_0) =$$

$$(x_i - x_m) \prod_{j=1, j\neq m,i}^{q} (z + a_j - a_m)Q + \alpha \prod_{j=1, j\neq i}^{q} (z + a_j - a_m)Q$$

This is computable from the tuple *challenge*.

5. Get the output $((a^*, S^*, \cdot, \cdot), x^*)$ from $\mathcal{A}$, where

$$S^* = \frac{1}{a^*+\gamma}(x^*P + P_0) = \frac{1}{z+a^*-a_m}(\alpha z + x^* - x_m)\prod_{i=1, i\neq m}^{q}(z+a_i-a_m)Q$$

We can see that the case $\alpha z + x^* - x_m = \alpha(z+a^*-a_m)$ happens with negligible probability, as it results in $S^* = S_m$. So the case $\alpha z + x^* - x_m \neq \alpha(z+a^*-a_m)$ happens with non-negligible probability ϵ_1. Suppose in this case, the probability that $a^* \in \{a_1, ..., a_q\}$ is ϵ_2. Then the probability that $a^* \notin \{a_1, ..., a_q\}\backslash\{a_m\}$ is $\epsilon_1 - \frac{q-1}{q}\epsilon_2$ (as $m \leftarrow \{1, ..., q\}$), which is also non-negligible if q is bound by a polynomial of l. If $\alpha z + x^* - x_m \neq \alpha(z+a^*-a_m)$ and $a^* \notin \{a_1, ..., a_q\}\backslash\{a_m\}$, then $\frac{1}{z+a^*-a_m}Q$ is computable from the tuple *challenge* and S^* and so $\mathcal{B}$ can compute $(c, \frac{1}{z+c}Q)$, where $c = a^* - a_m$.

4.5 Relationship between DBDH and (l, m, n)-DBDH assumptions

We now present the (l, m, n)-Decisional Bilinear Diffie-Hellman assumption and show that it is weaker than the DBDH assumption in Theorem 2.

(l, m, n)**-Decisional Bilinear Diffie-Hellman Assumption.** *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{\textsf{(l,m,n)-DBDH}}(\kappa)$ is negligible.*

$$Adv_{\mathcal{A}}^{\textsf{(l,m,n)-DBDH}}(\kappa) = |Pr[\mathcal{A}(\mathbf{t}, \{x_uP\}_{u=0}^{l}, \{e(P,P)^{x_u y_v z_w}\}_{(u,v,w)=(0,1,1)}^{(l,m,n)}) = 1]$$
$$-Pr[\mathcal{A}(\mathbf{t}, \{P_u\}_{u=0}^{l}, \{\Gamma_{uvw}\}_{(u,v,w)=(0,1,1)}^{(l,m,n)}) = 1]|$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_T, e, P) \leftarrow \mathcal{G}(1^\kappa)$ *and* $x_u, y_v, z_w \leftarrow \mathbb{Z}_p^*$, $P_u \leftarrow \mathbb{G}_1$, $\Gamma_{u,v,w} \leftarrow \mathbb{G}_T$, *for* $(u, v, w) = (0, 1, 1)...(l, m, n)$.

Theorem 2. *If the DBDH assumption holds then the (l, m, n)-DBDH assumption also holds.*

Proof. We first prove that if the DBDH assumption holds, then the $(1, 1, 1)$-DBDH assumption holds. We show that if a PPT algorithm $\mathcal{A}$ has non-negligible $Adv_{\mathcal{A}}^{\textsf{(1,1,1)-DBDH}}(\kappa)$ (i.e. the $(1, 1, 1)$-DBDH assumption does not hold), then we can build an algorithm $\mathcal{B}$ that has non-negligible $Adv_{\mathcal{B}}^{\textsf{DBDH}}(\kappa)$ (i.e. the DBDH assumption does not hold). Suppose $a, b, c \in \mathbb{Z}_p^*$ and $\Gamma \in \mathbb{G}_T^*$, we observe that if a and b are uniformly distributed in $\mathbb{Z}_p^*$, then $x = ab$ is also uniformly distributed in $\mathbb{Z}_p^*$ and if Γ is uniformly distributed in $\mathbb{G}_T^*$, then s is also uniformly distributed in $\mathbb{Z}_p^*$, where $\Gamma = e(P, P)^s$. So to distinguish between $(aP, bP, cP, e(P, P)^{abc})$ and (aP, bP, cP, Γ), the algorithm $\mathcal{B}$ can choose an uniformly random $d \in \mathbb{Z}_p^*$ and simply return the output by $\mathcal{A}$ when it takes as input $(\mathbf{t}, \{dP, dcP\}, \{e(aP, bP)^d, e(P, P)^{abcd}\})$ or $(\mathbf{t}, \{dP, dcP\}, \{e(aP, bP)^d, \Gamma^d\})$.

We now prove that if the (l, m, n)-DBDH assumption and the $(1, 1, 1)$-DBDH assumption hold, then the $(l + 1, m, n)$-DBDH assumption holds. We show that if a PPT algorithm $\mathcal{A}$ has non-negligible $Adv_{\mathcal{A}}^{\mathsf{(l+1,m,n)\text{-}DBDH}}(\kappa)$ (i.e. the $(l+1, m, n)$-DBDH assumption does not hold), then we can build a PPT algorithm $\mathcal{B}$ that has non-negligible $Adv_{\mathcal{B}}^{\mathsf{(l,m,n)\text{-}DBDH}}(\kappa)$ (i.e. the (l, m, n)-DBDH assumption does not hold) or has non-negligible $Adv_{\mathcal{B}}^{\mathsf{(1,1,1)\text{-}DBDH}}(\kappa)$ (i.e. the $(1, 1, 1)$-DBDH assumption does not hold). We define the following sets

$$\begin{aligned}
\mathbf{S}_1 &= \{(\{x_u P\}_{u=0}^{l+1}, \{e(P,P)^{x_u y_v z_w}\}_{(u,v,w)=(0,1,1)}^{(l+1,m,n)}) \mid x_u, y_v, z_w \leftarrow \mathbb{Z}_p^*\} \\
\mathbf{S}_2 &= \{(\{P_u\}_{u=0}^{l+1}, \{\Gamma_{uvw}\}_{(u,v,w)=(0,1,1)}^{(l+1,m,n)}) \mid P_u \leftarrow \mathbb{G}_1;\ \Gamma_{u,v,w} \leftarrow \mathbb{G}_T;\ r \leftarrow \mathbb{Z}_p^*; \\
&\qquad P_{l+1} = rP_l;\ \Gamma_{(l+1)vw} = \Gamma_{lvw}^r\} \\
\mathbf{S}_3 &= \{(\{P_u\}_{u=0}^{l+1}, \{\Gamma_{uvw}\}_{(u,v,w)=(0,1,1)}^{(l+1,m,n)}) \mid P_u \leftarrow \mathbb{G}_1;\ \Gamma_{u,v,w} \leftarrow \mathbb{G}_T\}
\end{aligned}$$

A non-negligible $Adv_{\mathcal{A}}^{\mathsf{(l+1,m,n)\text{-}DBDH}}(\kappa)$ means that $\mathcal{A}$ can distinguish between a random element of $\mathbf{S}_1$ and a random element of $\mathbf{S}_3$. It means $\mathcal{A}$ can distinguish either between a random element of $\mathbf{S}_1$ and a random element of $\mathbf{S}_2$ or between a random element of $\mathbf{S}_2$ and a random element of $\mathbf{S}_3$. We consider these 2 cases:

- We show that if $\mathcal{A}$ can distinguish between a random element of $\mathbf{S}_1$ and a random element of $\mathbf{S}_2$, then we can build an algorithm $\mathcal{B}$ that has non-negligible $Adv_{\mathcal{B}}^{\mathsf{(l,m,n)\text{-}DBDH}}(\kappa)$. Suppose $\mathcal{B}$ is given an input $(\mathbf{t}, \{P'_u\}_{u=0}^{l}, \{\Gamma'_{uvw}\}_{(u,v,w)=(0,1,1)}^{(l,m,n)})$, $\mathcal{B}$ chooses an uniformly random x and computes $P'_{l+1} = xP'_l$ and $\Gamma'_{(l+1)vw} = \Gamma'^{x}_{lvw}$, for $v = 1, ..., m$ and $w = 1, ..., n$. $\mathcal{B}$ then return the output from $\mathcal{A}$ when it takes as input $(\mathbf{t}, \{P'_u\}_{u=0}^{l+1}, \{\Gamma'_{uvw}\}_{(u,v,w)=(0,1,1)}^{(l+1,m,n)})$.
- We show that if $\mathcal{A}$ can distinguish between a random element of $\mathbf{S}_2$ and a random element of $\mathbf{S}_3$, then we can build an algorithm $\mathcal{B}$ that has non-negligible $Adv_{\mathcal{B}}^{\mathsf{(1,1,1)\text{-}DBDH}}(\kappa)$. Suppose $\mathcal{B}$ is given an input$(\mathbf{t}, \{P'_l, P'_{l+1}\}, \{\Gamma'_{lmn}, \Gamma'_{(l+1)mn}\})$. $\mathcal{B}$ chooses $P'_u \leftarrow \mathbb{G}_1$, $\Gamma'_{u,v,w} \leftarrow \mathbb{G}_T$, for $(u, v, w) = (0, 1, 1) \cdots (l-1, m, n)$. $\mathcal{B}$ then return the output from $\mathcal{A}$ when it takes as input $(\mathbf{t}, \{P'_u\}_{u=0}^{l+1}, \{\Gamma'_{uvw}\}_{(u,v,w)=(0,1,1)}^{(l+1,m,n)})$.

Therefore, if the PPT algorithm $\mathcal{A}$ has non-negligible $Adv_{\mathcal{A}}^{\mathsf{(l+1,m,n)\text{-}DBDH}}(\kappa)$, then the algorithm $\mathcal{B}$ has either non-negligible $Adv_{\mathcal{B}}^{\mathsf{(l,m,n)\text{-}DBDH}}(\kappa)$ or non-negligible $Adv_{\mathcal{B}}^{\mathsf{(1,1,1)\text{-}DBDH}}(\kappa)$.

It can be similarly proved that if the (l, m, n)-DBDH assumption and the $(1, 1, 1)$-DBDH assumption hold, then the $(l, m + 1, n)$-DBDH assumption and the $(l, m, n + 1)$-DBDH assumption hold. Therefore, by induction, Theorem 2 has been proved.

4.6 A new k-TAA scheme

A dynamic k-TAA scheme can be converted into an ordinary k-TAA as follows. The setup procedure remains the same, except that the APs do not obtain any key, do not maintain any access group and other public information. The joining, the bound announcement and the public tracing procedures remain the same. The granting access and revoking access procedures are removed. In the authentication procedure for dynamic k-TAA, a user needs to prove to an AP three conditions: (i) he has been registered as a group member; (ii) he is in the access group of the AP; and (iii) he has not accessed the AP more than the allowable number of times. For ordinary k-TAA, the user do not have to prove condition (ii) and just needs to prove two conditions (i) and (iii).

Using the above approach we can construct a k-TAA scheme from the proposed dynamic scheme. The k-TAA scheme has the following differences with the dynamc one. In the setup procedure, the for APs part is removed and only the part for GM is performed. The joining, the bound announcement and the public tracing procedures remain the same. There is no granting access or revoking access. In the authentication procedure, a different $Proof_{2b}$ is used and there is no Update or Public Inspection algorithm. Security of the ordinary scheme is stated in Theorem 3. The authentication procedure and Theorem 3's proof are provided in the full version [14].

Theorem 3. *In the random oracle model and the list oracle model, our k-TAA scheme provides (i) Correctness; (ii) Anonymity under the Decisional Bilinear Diffie-Hellman assumption; (iii) Detectability under the q-Strong Diffie-Hellman assumption, where q is the upper bound of the group size; (iv) Exculpability for users under the Discrete Logarithm assumption on $\mathbb{G}_1$; (v) Exculpability for the GM under the q-Strong Diffie-Hellman assumption, where q is the upper bound of the group size.*

4.7 Efficiency

Our schemes have the same desirable features of the TFS04 scheme. The size of a group member's keys does not depend on the group size and the GM can add new members to the group without modifying the public key or secret key of group members. After being registered into the group, a user does not need to contact the GM. Each AP can independently determine his bound. In the dynamic scheme, each AP can independently decide which members are allowed to access his services. Without considering the Update algorithm the computational cost of authentication depends only on the bound of the AP.

Our schemes have higher communication efficiency compared to the TFS04 scheme. For instance, assume the scheme is implemented by an elliptic curve or hyperelliptic curve over a finite field. p is a 160-bit prime, $\mathbb{G}_1$ is a subgroup of an elliptic curve group or a Jacobian of a hyperelliptic curve over a finite field of order p. $\mathbb{G}_T$ is a subgroup of a finite field of size approximately 2^{1024}. Techniques in [9] can be used to compress elements of $\mathbb{G}_T$ by a factor of three. A possible choice for the parameters can be from Boneh *et al.* [4]: $\mathbb{G}_1$ is derived from the curve $E/GF(3^\iota)$ defined by $y^2 = x^3 - x + 1$. In addition, we assume that system parameters in the TFS04 scheme are $\nu = 1024$, $\varepsilon = \mu = \kappa = 160$. We summarize the comparison of communication costs, which are measured by the number of bytes sent, for authentication procedures in the following table.

	Bytes sent by AP	Bytes sent by User	Dynamic
The TFS04 scheme	40	60 k+ 1617	No
Our ordinary scheme	20	60 k+ 224	No
Our dynamic scheme	20	60 k+ 304	Yes

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. CRYPTO 2000, Springer-Verlag, LNCS 1880, pp. 255-270.
2. M. Bellare, H. Shi, and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. Cryptology ePrint Archive: Report 2004/077.
3. D. Boneh, and X. Boyen. Short Signatures Without Random Oracles. EUROCRYPT 2004, Springer-Verlag, LNCS 3027, pp. 56-73.
4. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. ASIACRYPT 2001, Springer-Verlag, LNCS 2248, pp. 514-532.
5. S. Brands. An Efficient Off-line Electronic Cash System Based On The Representation Problem. Technical Report CS-R9323, Centrum voor Wiskunde en Informatica.
6. J. Camenisch, and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. CRYPTO 2002, Springer-Verlag, LNCS 2442, pp. 61-76.
7. D. Chaum. Blind signature system. CRYPTO 1983, Plenum Press, pp. 153-153.
8. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. CRYPTO 1986, Springer-Verlag, LNCS 263, pp. 186-194.
9. R. Granger, D. Page, and M. Stam. A Comparison of CEILIDH and XTR. Algorithmic Number Theory, 6th International Symposium, ANTS-VI, pages 235-249. Springer, June 2004.
10. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. IEICE Trans. Vol. E85-A, No.2, pp.481-484, 2002. [29]
11. T. Nakanishi, N. Haruna, and Y. Sugiyama. Unlinkable Electronic Coupon Protocol with Anonymity Control. ISW 1999, Springer-Verlag, LNCS 1729, pp. 37-46.
12. L. Nguyen. *Accumulators from Bilinear Pairings and Applications*. RSA Conference 2005, Cryptographers' Track (CT-RSA), Springer-Verlag, LNCS 3376, pp. 275-292, 2005.
13. L. Nguyen and R. Safavi-Naini. *Efficient and Provably Secure Trapdoor-free Group Signature Schemes from Bilinear Pairings*. ASIACRYPT 2004, Springer-Verlag, LNCS 3329, pp. 372-386, 2004.
14. L. Nguyen and R. Safavi-Naini. *Dynamic k-Times Anonymous Authentication*. Full version.
15. I. Teranisi, J. Furukawa, and K. Sako. k-Times Anonymous Authentication. ASIACRYPT 2004, Springer-Verlag, LNCS 3329, pp. 308-322, 2004.