# Constant-Size Dynamic $k$-TAA* **

Man Ho Au[1], Willy Susilo[1], and Yi Mu[1]

Center for Information Security Research
School of Information Technology and Computer Science
University of Wollongong, Wollongong 2522, Australia
{mhaa456,wsusilo,ymu}@uow.edu.au

**Abstract.** $k$-times anonymous authentication ($k$-TAA) schemes allow members of a group to be authenticated anonymously by application providers for a bounded number of times. Dynamic $k$-TAA allows application providers to independently grant or revoke users from their own access group so as to provide better control over their clients. In terms of time and space complexity, existing dynamic $k$-TAA schemes are of complexities $O(k)$, where $k$ is the allowed number of authentication. In this paper, we construct a dynamic $k$-TAA scheme with space and time complexities of $O(log(k))$. We also outline how to construct dynamic $k$-TAA scheme with a constant proving effort. Public key size of this variant, however, is $O(k)$.
We then describe a trade-off between efficiency and setup freeness of AP, in which AP does not need to hold any secret while maintaining control over their clients.
To build our system, we modify the short group signature scheme into a signature scheme and provide efficient protocols that allow one to prove in zero-knowledge the knowledge of a signature and to obtain a signature on a committed block of messages. We prove that the signature scheme is secure in the standard model under the $q$-SDH assumption.
Finally, we show that our dynamic $k$-TAA scheme, constructed from bilinear pairing, is secure in the random oracle model.

**Keywords:** $k$-TAA, dynamic $k$-TAA

## 1 Introduction

Teranisi *et al.* [19] proposed $k$-times anonymous authentication ($k$-TAA) so that users of a group can access applications anonymously while application providers (AP) can decide the number of times users can access their applications. In $k$-TAA, there are three entities, namely, group manager (GM), application providers (AP) and users. Users first register to GM and each AP announce independently the allowable number of access to its application. A registered user can then authenticate himself to the AP's anonymously, up to the allowed

* This work is partially supported by ARC Linkage Project Grant LP0667899.
** This paper is the extended version of the paper that appear in SCN '06 under the same title [1].

number of times. Anyone can trace a dishonest user who tries to access an application for more than the allowable number of times.

In $k$-TAA, AP's have no control over the group of users accessing their applications. In actual scenarios, AP's may wish to select their own group of users. Dynamic $k$-TAA, proposed by Nguyen *et al.* [16], has added this flexibility over ordinary $k$-TAA systems. In a dynamic $k$-TAA, the role of AP's is more active and they can select their user groups, granting and revoking access of registered users independently.

Many existing $k$-TAA schemes (and dynamic $k$-TAA schemes) [19, 16] are quite efficient, with time and space complexities independent of the total number of users. However, size of the public key of AP's, together with the communication cost between users and AP's, are both of order $O(k)$. The computational cost of the user for an authentication protocol is also of order $O(k)$. In this paper, we construct dynamic $k$-TAA scheme with complexity of $O(log(k))$. We also outline how to reduce the proving cost to $O(1)$ at the cost of public key size of AP.

In constructing our scheme, we modify the short group signature from Boneh *et al.* [3] into a signature scheme, which we shall referred to as BBS+ signature, with two protocols, similar to [9, 11] (referred to as CL, CL+ respectively hereafter). We do not claim originality of this modification as it has been outlined in [11]. However, we supply the details of the modification, together with the protocols and analyze its security. In particular, the protocol of showing possession of a signature is different from [3] in which the modified protocol achieve perfect zero-knowledge while the original protocol is computational. We prove that BBS+ signature is secure in the standard model under the $q$-SDH assumption. This BBS+ signature could be used as building blocks for other cryptographic systems. It has similar properties to CL (based on Strong RSA) and CL+ signatures (based on LRSW). To sign a block of messages, the signature scheme outperforms the existings schemes in the literature (signature size of CL+ is linear to number of messages in the block to be signed, CL is 1346 bits while BBS+ is only 511 bits).

The recently proposed group signature from [6] can also be modified into signature scheme with efficient protocol secured in the stand model. However, the signing of a message have to be done in a bit-by-bit manner.

### 1.1 Related Works

Very recently, Teranishi and Sako [20] proposed an ordinary $k$-TAA scheme with constant proving cost. We shall refer to it as TS06 hereafter. Our construction can be thought of as an extension of TS06 to dynamic $k$-TAA to give AP more control over their clients. This is achieved by the use of dynamic accumulator and the idea of using dynamic accumulator for access control was introduced in [10]. Finally, as pointed out in [20], $k$-TAA shares certain similarities with compact e-cash schemes, introduced in [8]. The main difference being in $k$-TAA schemes, each provider may chooses its only $k$ and a user could authenticated himself $k_1$ times to provider-1, $k_2$ times to provider-2, etc., while in a compact

e-cash scheme, the user can only spend his wallet a total of $k$ times to all the shops combined. Nevertheless, the techniques used in our scheme is very similar to the compact e-cash scheme [8]. The main difference being we show how to incorporate the provider's name into the pseudo-random function such that authentication to different providers cannot be linked together.

Finally, the BBS+ signature we analyzed can be regarded as an extension of a digital signature scheme very recently proposed by [17] to support signing of block of committed messages.

**Our Contributions.**

- we construct efficient dynamic $k$-TAA scheme.
- we reduce the security of our scheme to well-known intractable assumptions in the random oracle model.
- we analyze an modification of the BBS group signature, provide efficient protocols, and show its security in the standard model.

**Organization.** The rest of the paper is as follows. Preliminaries are presented in Section 2. We then briefly review the security notions in section 3. Our construction is shown in Section 4, followed by its variants in Section 5. Complexity and security analysis are given in Section 6. Finally, we conclude in Section 7.

## 2 Preliminaries

### 2.1 Notations

Let $e$ be a bilinear map such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic multiplicative groups of prime order $p$.
- each element of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ has unique binary representation.
- $g_0$, $h_0$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(h_0) = g_0$.
- (Bilinear) $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and $a$, $b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$.
- (Non-degenerate)$e(g_0, h_0) \neq 1$.

$\mathbb{G}_1$ and $\mathbb{G}_2$ can be same or different groups. We say that two groups $(\mathbb{G}_1, \mathbb{G}_2)$ are a bilinear group pair if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$, the isomorphism $\psi$ and the bilinear mapping $e$ are all efficiently computable.

### 2.2 Mathematical Assumptions

**Definition 1 (Decisional Diffie-Hellman).** *The Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}$ is defined as follow: On input a quadruple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, output 1 if $c = ab$ and 0 otherwise. We say that the $(t, \epsilon)$-DDH assumption holds in $\mathbb{G}$ if no $t$-time algorithm has advantage at least $\epsilon$ over random guessing in solving the DDH problem in $\mathbb{G}$.*

**Definition 2 ($q$-Strong Diffie-Hellman).** *The $q$-Strong Diffie-Hellman ($q$-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follow: On input a $(q+2)$-tuple $(g_0, h_0, h_0^x, h_0^{x^2}, \cdots, h_0^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, output a pair $(A, c)$ such that $A^{(x+c)} = g_0$ where $c \in \mathbb{Z}_p^*$. We say that the $(q, t, \epsilon)$-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no $t$-time algorithm has advantage at least $\epsilon$ in solving the $q$-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.*

The $q$-SDH assumption is shown to be true in the generic group model [2].

**Definition 3 ($y$-Decisional Diffie-Hellman Inversion Assumption).** *The $y$-Decisional Diffie-Hellman Inversion problem ($y$-DDHI) in prime order group $\mathbb{G}$ is defined as follow: On input a $(y+2)$-tuple $g, g^x, g^{x^2}, \cdots, g^{x^y}, g^c \in \mathbb{G}^{y+2}$, output 1 if $c = 1/x$ and 0 otherwise. We say that the $(y, t, \epsilon)$-DDHI assumption holds in $\mathbb{G}$ if no $t$-time algorithm has advantage at least $\epsilon$ over random guessing in solving the $y$-DDHI problem in $\mathbb{G}$.*

### 2.3 Building Blocks

**Verifiable Random Function.** Our constant-size dynamic k-TAA make use of verifiable random function (VRF), introduced in [14]. Informally speaking, a VRF is a pseudo-random function with non-interactive proof of correctness of its output. The VRF used in our paper is due to Dodis *et al.* [12] and is described as follows. The pseudo-random function $f$ is defined by a tuple $(\mathbb{G}_p, p, g, s)$, where $\mathbb{G}_T$ is a cyclic group of prime order $p$, $g$ a generator of $\mathbb{G}_p$ and $s$ is a seed in $\mathbb{Z}_p$. On input $x$, $f_{\mathbb{G}_p,p,g,s}(x) = g^{\frac{1}{s+x+1}}$. Efficient proof such that the output is correctly formed (with respect to $s$ and $x$ in some commitment scheme such as Pedersen Commitment [18]) exists and the output of $f$ is indistinguishable from random elements in $\mathbb{G}_p$ if the $y$-DDHI assumption in $\mathbb{G}_p$ holds. The verifiable random function in [12] uses a stronger bilinear version of the y-DDHI assumption, see [8] for details.

**Accumulator.** Our construction is built based on the accumulator with one-way domain due to [15]. Roughly speaking, an accumulator is an algorithm to combine a large set of values ($\{x_i\}$) into a short value $v$. For each value $x_j \in \{x_i\}$, a witness $w_j$ exists and with $w_j$, it can be proved that $x$ is indeed accumulated into $v$. An accumulator is dynamic if it allows values to be added or deleted dynamically.

**Signature Scheme with Efficient Protocols.** In this paper, a signature scheme with efficient protocols refers to signature scheme with two protocols: (1) a protocol between a user and a signer with keys $(pk, sk)$. Both user and signer agreed on a commitment scheme such as Pedersen commitment. The user input is a block of messages $(m_1, \cdots, m_L)$ and a random value $r$ such that $C$=PedersenCommit$(m_1, \cdots, m_L, r)$. After executing the protocol, user obtains a signature on $(m_1, \cdots, m_L)$ from the signer while the signer learns nothing about the block of messages. (2) a protocol to proof the knowledge of a signature. This allows the user to prove to a verifier that he is in possession of a signature. Examples include CL signature, CL+ signature [9, 11]. In this paper,

we analyze another signature scheme with efficient protocols which is a modification of the short group signature from Boneh *et al.*[3], that we referred to as BBS+ signature.

## 3 Security Model

### 3.1 Syntax

We follow the model of dynamic $k$-TAA in [16] and briefly review them. A dynamic $k$-times anonymous authentication is a tuple (GMSetup, Join, APSetup, GrantingAccess, RevokeingAccess, Authentication, PublicTracing) of six polynomial time algorithms between three entities GM, APs, users. The following enumerates the syntax.

- GMSetup On input a unary string $1^\lambda$, where $\lambda$ is a security parameter, the algorithm outputs GM secret key $gsk$ and group public key $gpk$. All algorithms below have implicitly $gpk$ as one of their inputs.
- Join Protocol. This protocol allows a user to join the group and obtain a member public/secret key pair $(mpk, msk)$ from GM. The GM also add the user's identification and member public key to an identification list.
- APSetup An AP publishes its identity $ID$ and announces the number of times $k$ a group member can access its application. It may also generate certain public and private key for the AP.
- GrantingAccess Each AP manages its own access group $AG$ which is initially empty. This procedure allows the AP to give selected group members the permission to access his application.
- RevokingAccess It allows the AP to remove a member from his access group and stop a member from accessing his application.
- Authentication Protocol. The user authenticated himself to an AP under this protocol. The user is authenticated only if it is in the access group of the AP and the number of accesses have not exceeded the allowed number $k$. AP records the transcripts of authentication in an authentication log.
- PublicTracing Anyone can execute this procedure using public information and the authentication log. The outputs are user $i$'s identity, GM or NO-ONE which indicates "user $i$ tries to access more than $k$ times", "the GM cheated" and "there is no malicious entity in this authentication log" respectively.

A dynamic $k$-times anonymous authentication must possess *Correctness* which means that an honest member who is in the access group of an honest AP, and has not authenticate himself for more than the allowed number of times, must be authenticated by the AP.

### 3.2 Security Notions

We briefly recall security requirements, for formal definition please refer to [16, 19].

- *D-Detectability.* Roughly speaking, it means that a subset of colluded users cannot perform the authentication procedure with the same honest AP for more than the allowed number of times, or they must be detected by the PublicTracing algorithm.
- *D-Anonymity.* It is required that no collusion of AP, users and GM can distinguish between authentication executions of two honest group members who are in the access group of the AP.
- *D-Exculpability.* It is required that an honest user cannot be accused of having performed the authentication procedure with the same honest AP for more than the allowed number of time. It is also required that the PublicTracing algorithm shall not output GM if the GM is honest even though the AP and the users colludes.

## 4 Our Construction

Our dynamic $k$-TAA is built from the $q$-SDH based accumulator due to Nguyen [15] and a modification of the BBS group signature [3], that we call BBS+ signature, which is a signature scheme with efficient protocols. BBS+ signature is unforgeable against adaptive chosen message attack in the standard model under the $q$-SDH assumption and we also propose two protocols:(1) for issuing a signature on a committed value (so the signer has no information about the signed value), and (2) for proving knowledge of a signature on a committed value. We first describe the global common parameters, followed by descriptions of BBS+ signature and finally our dynamic $k$-TAA scheme.

### 4.1 Global Common parameters

Let $\lambda$ be the security parameter. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair with computable isomorphism $\psi$ as discussed such that $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime $p$ of $\lambda$ bits. Assume $\mathbb{G}_p$ be a group of order $p$ where DDH is intractable. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $H_{evt}\{0,1\}^* \rightarrow \mathbb{G}_p$ be cryptographic hash functions. Let $g_0, g_1, g_2, g_3$ be generators of $\mathbb{G}_1$, $h_0, h_1, h_2, h_3$ be generators of group $\mathbb{G}_2$ such that $\psi(h_i) = g_i$ and $u_0, u_1, u_2, u_3$ be generators of $\mathbb{G}_p$ such that relative discrete logarithm of the generators are unknown. One possible way is to make use of some hash functions $f : \{0,1\}^* \rightarrow \mathbb{G}_1$, $g : \{0,1\}^* \rightarrow \mathbb{G}_p$ and set $h_i = f(seed, i)$, $g_i = \psi(h_i)$, $u_i = g(seed, i)$ for some publicly known $seed$.

*Remarks: the generation of this common parameters can be done by GM or some trusted third parties.*

### 4.2 BBS+ Signature

The idea of modifying the BBS group signature into a signature with efficient protocols is stated in [11]. We supply the details, provide efficient protocols and prove its security.

KenGen. Randomly chooses $\gamma \in_R \mathbb{Z}_p^*$ and computes $w = {h_0}^{\gamma}$. The secret key is $\gamma$ and the public key is $w$.

Signing block of messages. On input $(m_1, \cdots, m_L) \in \mathbb{Z}_p^L$, chooses $e$ and a random number $s$. Computes $A = [g_0 g_1^s g_2^{m_1} g_3^{m_2} \cdots g_{L+1}^{m_L}]^{\frac{1}{e+\gamma}}$. Signature on $(m_1, \cdots, m_L)$ is $(A, e, s)$.

Signature Verification. To verify a signature $(A, e, s)$ on $(m_1, \cdots, m_L)$ , check if $e(A, wh_0^e) = e(g_0 g_1^s g_2^{m_1} g_3^{m_2} \cdots g_{L+1}^{m_L}, h_0)$.

Regarding security of BBS+ signature, we have the following theorem and its proof is shown in appendix B.

**Theorem 1.** *BBS+ signature is unforgeable against adaptively chosen message attack under the q-SDH assumption.*

Protocol for Signing Committed Block of Messages. The user computes a Pedersen Commitment on the block of messages to be signed by $C_m = g_1^{s'} g_2^{m_1} g_3^{m_2} \cdots g_{L+1}^{m_L}$. The user also need to prove to the signer that $C_m$ is correctly formed by the following PK: $PK\{(s', m_1, \cdots, m_L) : C_m = g_1^{s'} g_2^{m_1} g_3^{m_2} \cdots g_{L+1}^{m_L}\}$. The signer then chooses $s'', e$, computes $A = [g_0 g_1^{s''} C_m]^{\frac{1}{e+\gamma}}$ and sends back $(A, e, s'')$ back to the user. The user computes $s = s' + s''$ and the signature on the block of messages is then $(A, e, s)$. For whatever block of messages $(m_1, \cdots, m_L)$, there exists an $s'$ such that $C_m = g_1^{s'} g_2^{m_1} g_3^{m_2} \cdots g_{L+1}^{m_L}$ and $s'$ completely hides the information about the block of messages. Thus, the signer learns nothing about the block of messages to be signed.

Proof of Knowledge of A Signature. We give a zero-knowledge proof of knowledge protocol for showing possession of a signature. Using any protocol for proving relations among components of a discrete-logarithm representations of a group element [7], it can be used to demonstrate relations among components of a signed block of messages. A user possessing a signature $(A, e, s)$ on the block of message $(m_1, \cdots, m_L)$ can compute $SPK\{(A, e, s, m_1, \cdots, m_L) : A^{e+\gamma} = g_0 g_1^s g_2^{m_1} g_3^{m_2} \cdots g_{L+1}^{m_L}\}(M)$ by first computing the following quantities: $A_1 = g_1^{r_1} g_2^{r_2}$, $A_2 = A g_2^{r_1}$ for some randomly generated $r_1, r_2 \in_R \mathbb{Z}_p^*$. Then it computes the following SPK $\Pi_5$.

$$\Pi_5 : SPK\Big\{(r_1, r_2, e, \delta_1, \delta_2, e, s, m_1, \cdots, m_L) : A_1 = g_1^{r_1} g_2^{r_2} \ \wedge \ A_1^e = g_1^{\delta_1} g_2^{\delta_2} \ \wedge \ \frac{e(A_2, w)}{e(g_0, h_0)} = e(A_2, h_0)^{-e} e(g_2, w)^{r_1} e(g_2, h_0)^{\delta_1} e(g_1, h_0)^s e(g_2, h_0)^{m_1} \cdots e(g_{L+1}, h_0)^{m_L}\Big\}(M)$$

where $\delta_1 = r_1 e$ and $\delta_2 = r_2 e$.

Regarding SPK $\Pi_5$, we have the following theorem which is straight forward and the proof is thus omitted.

**Theorem 2.** *$\Pi_5$ is an non-interactive honest-verifier zero-knowledge proof-of-knowledge protocol with special soundness.*

*Remarks: this protocol is a different from the protocol in [3], where the HVZK is computational (under the DLDH assumption) while $\Pi_5$ is perfect. One possible reason is that the SDH protocol in [3] is used for group signature scheme where certain user information in 'verifiably encrypted' within the protocol for GM to revoke identity of the signer.*

### 4.3 Overview of Our Construction

Join. The GM is in possession of the public/secret key pair of BBS+ signature. User randomly generates $x \in \mathbb{Z}_p^*$ and $u_0^x$ is the identity of the user. A membership certificate of a user is a BBS+ signature (of the form $(A, e)$ ) on the set of values $(s, t, x)$, where $s$ and $t$ are also random elements in $\mathbb{Z}_p*$. Finally, $(u_0^x, e)$ are placed on an identification list.

GrantingAccess/RevokeAccess. Each AP generates its own accumulator due to Nguyen [15]. It accumulates the value $e$ into the accumulator and gives the witness $w_{AP}$ to the user. To revoke access, the AP removes the value $e$ from the accumulator.

In the variant of our scheme (to be shown in the next section), AP only publishes the access group and let the users work with the accumulator itself. This make it possible to remove the interactive granting access/revoke access protocol. The cost to pay is that user have to perform $O(|AccessGroup|)$ operations to obtain his own witness.

Authentication. The idea is to have the users prove to the AP that it is in possession of a BBS+ signature $(A, e, s)$ from the GM on the values $(t, x)$, and that $e$ is inside the accumulator of the AP. To restrict the user from authenticating himself for more than $k$ times, pseudo-random function(PRF) due to Dodis and Yampolskiy [12] is used as follow. Let $u_{AP}$ be a random element in a cyclic group equal to hash of identity of the AP. The user computes $S = u_{AP}^{\frac{1}{s+J_{AP}+1}}$ and proves that $S$ is correctly formed with respect to the BBS+ signature component $s$. Also, user needs to prove that $1 \leq J_{AP} \leq k$. In this way, for a particular $AP$, the user can only generate $k$ valid $S$, which we called serial number. If he attempts to authenticate himself for more than $k$ times, duplicated serial number has to be used and can thus be detected.

Finally, to allow revocation of identity of user attempting to authenticate himself for more than $k$ times, another component $T = u_0^x u_{AP}^{\frac{R}{t+J_{AP}+1}}$ is added, where $R$ is a random nonce chosen by the AP during each authentication attempt. User needs to prove that $T$ is correctly formed. In case the user attempts to use the same serial number to authenticate twice, due to $R$ being different, the two $T$'s shall be different. With different $T$'s, identity of the cheater, $u_0^x$, can be computed.

*Remarks: it is obvious that other signature schemes with efficient protocol such as CL, CL+ could also be used for our scheme. However, in our case, BBS+ is most suitable for two reasons: (1) it is most efficient in our context*

*and (2) the accumulator we used is based on the q-SDH assumption for which security of BBS+ signature also relies on.*

### 4.4 Details of Our Construction

GMSetup. The GM randomly selects $\gamma \in_R \mathbb{Z}_p^*$ and computes $w = h_0{}^{\gamma}$. The GM also manages an identification list which is a tuple $(i, U_i, e_i)$ where $i$ refers to user $i$ and $U_i$ is an entry for identification of user and $e_i$ is called the membership public key of user $i$. See Join for a more detailed description of this item.

APSetup. Each AP publishes his identity $ID$ and a number $k$, much smaller than $2^{\lambda}$. In addition, each AP selects $h_{AP} \in \mathbb{G}_2$, $q_{AP} \in \mathbb{Z}_p^*$. The public and secret keys for the AP are $h_{AP}, p_{AP} = h_{AP}^{q_{AP}}$ and $q_{AP}$ respectively. The AP maintains an authentication log, an accumulated value, which is published and updated after granting or revoking access of a member, and a public archive $ARC$ which is describe as follows. The $ARC$ is a 3-tuple ($arc_1$, $arc_2$, $arc_3$) where $arc_1$ is a component of the membership public key of a user, $arc_2$ is a single bit 0/1 indicating if the member was granted (1) or revoked (0). Finally, $arc_3$ is the accumulated value after granting or revoking the member. Initially, the authentication log and $ARC$ are empty while the accumulated value is set to $h_{AP}$.

Join. User $i$ obtains his membership secret key from GM through the following interactive protocol.

1. User $i$ randomly selects $s', t, x \in_R \mathbb{Z}_p^*$ and sends $C' = g_1^{s'} g_2^t g_3^x$, along with the proof $\Pi_0 = PK\{(s', t, x) : C' = g_1^{s'} g_2^t g_3^x\}$ to GM.
2. GM verifies that $\Pi_0$ is valid and randomly selects $s'' \in_R \mathbb{Z}_p^*$. It sends $s''$ to the user.
3. User computes $s = s' + s''$ and add an entry $(i, U_i) = (i, u_0^x)$ to the identification list and send a proof $\Pi_1 = PK\{(s, t, x) : U_i = u_0^x \wedge C = g_1^s g_2^t g_3^x \wedge C = C' g_1^{s''}\}$.
4. GM computes $C = C' g_1^{s''}$, check that $\Pi_1$ is valid , and selects $e \in_R \mathbb{Z}_p^*$. It then computes $A = (g_0 C)^{\frac{1}{e+\gamma}}$ and sends $(A, e, s'')$ to the user. GM also appends $e$ to the entry $(i, U_i)$ to make it $(i, U_i, e)$
5. User checks if $e(A, w h_0^e) = e(g_0 g_1^s g_2^t g_3^x, h_0)$. It then stores $(A, e, s, t, x)$. User's membership public key is $e$ and membership secret key is $(A, s, t, x)$.

GrantingAccess. An AP grants access to user $i$ with membership public key $e$ and secret key $(A, s, t, x)$ as follows. Suppose there are $j$ tuples in the AP's $ARC$ and the current accumulated value is $v_j$. The AP computes a new accumulated value $v_{j+1} = v_j^{e+q_{AP}}$. Then the AP adds $(e, 1, v_{j+1})$ to the ARC. The user keeps $w = v_j$ as his witness that his public key has been accumulated in the accumulated value. Existing members in the access group update their own witness by the information of $ARC$ as follows. User with membership key $e_k$ and witness $w_e$ such that $w_e^{e_k+q_{AP}} = v_j$ computes $w_{new} = v_j w_e^{e-e_k}$. In this case $w_{new}^{e_k+q_{AP}} = v_{j+1}$ and $w_{new}$ serves as a new witness for user $e_k$.

RevokingAccess. An AP revokes access from user $i$ with membership public key $e$, such that $(e, 1, v)$ is a tuple in the $ARC$, as follows. Suppose there are $j$ tuples in the AP's $ARC$ and the current accumulated value is $v_j$. The AP computes $v_{j+1} = v_j^{\frac{1}{e+q_{AP}}}$. It then adds $(e, 0, v_{j+1})$ to $ARC$. Similar to the case of GrantingAccess, existing members in the access group update their own witness by the information of $ARC$, which is shown as follows. Suppose user $e_k$ possesses witness such that $w_e^{e_k+q_{AP}} = v_j$, it computes $w_{new} = (w_e/v_{j+1})^{\frac{1}{e-e_k}}$ such that $w_{new}^{e_k+q_{AP}} = [\frac{v_j}{v_{j+1}^{q_{AP}+e_k}}]^{\frac{1}{e-e_k}} = [\frac{v_{j+1}^{q_{AP}+e}}{v_{j+1}^{q_{AP}+e_k}}]^{\frac{1}{e-e_k}} = v_{j+1}$.

Authentication. The user manages a set of counters, one for each AP , $J_{AP}$, such that it did not attempt to sign more than $k$ times for each AP. User with membership public key $e$ and secret key $(A, s, t, x)$, having granted access from the AP and thus possesses a witness $w_{AP}$ such that $w_{AP}^{e+q_{AP}} = v_{AP}$ where $v_{AP}$ is the current accumulated value of the AP authenticates himself by the following interactive protocol. For simplicity we drop the subscript $AP$ for $q_{AP}$ and $v_{AP}$.

- AP sends a random $seed \in \{0,1\}^*$ to user. In practice, $seed$ can be some random number or information about the current session. Both parties compute $R = H(seed)$ locally.
- User computes $u_{AP} = H_{evt}(ID_{AP})$ where $ID_{AD}$ is the identity of the AP. User then computes $S = u_{AP}^{\frac{1}{J_{AP}+s+1}}$, $T = u_0^x u_{AP}^{\frac{R}{J_{AP}+t+1}}$ and proves in zero-knowledge manner (1) - (5):
  1. $A^{e+\gamma} = g_0 g_1^s g_2^t g_3^x$.
  2. $w_{AP}^{e+q} = v$.
  3. $S = u_{AP}^{\frac{1}{J_{AP}+s+1}}$.
  4. $T = u_0^x u_{AP}^{\frac{R}{J_{AP}+t+1}}$.
  5. $1 \leq J_{AP} \leq k$
- The above can be abstracted as

$$\Pi_2 : SPK\Big\{(A, e, s, t, x, w, J_{AP}) : A^{e+\gamma} = g_0 g_1^s g_2^t g_3^x \ \wedge \ w^{e+q} = v \ \wedge \ S = u_{AP}^{\frac{1}{J_{AP}+s+1}} \ \wedge \ T = u_0^x u_{AP}^{\frac{R}{J_{AP}+t+1}} \ \wedge \ 1 \leq J_{AP} \leq k\Big\}(M)$$

- AP then verifies that the SPK is correct. If yes, then accept and saves $S$, $T$, $R$ into database.
- User then increases its counter, $J_{AP}$, by one.

**Instantiation of $\Pi_2$.** Upon receiving $seed$, the user computes the following quantities: $A_1 = g_1^{r_1} g_2^{r_2} g_3^{r_3}$, $A_2 = A g_2^{r_1}$, $A_3 = w_{AP} g_3^{r_2}$, $A_4 = g_1^{J_{AP}} g_2^t g_3^{r_4}$, $S = u_{AP}^{\frac{1}{J_{AP}+s+1}}$,$T = u_0^x u_{AP}^{\frac{R}{J_{AP}+t+1}}$, $R = H(seed)$ and computes the following SPK $\Pi_3$.

$$\Pi_3 : SPK\Big\{(r_1, r_2, r_3, r_4, \delta_1, \delta_2, \delta_3, \delta_4, \delta_J, \delta_t, e, s, t, x, J_{AP}) :$$

$$
\begin{aligned}
A_1 = g_1^{r_1} g_2^{r_2} g_3^{r_3} \ \wedge \ A_1^e = g_1^{\delta_1} g_2^{\delta_2} g_3^{\delta_3} \ \wedge \\
\tfrac{e(A_3, P_{AP})}{e(v, h_{AP})} = e(g_3, h_{AP})^{\delta_2} e(g_3, p_{AP})^{r_2} e(A_3, h_{AP})^{-e} \ \wedge \\
\tfrac{e(A_2, w)}{e(g_0, h_0)} = e(g_1, h_0)^s e(g_2, h_0)^t e(g_3, h_0)^x e(g_3, h_0)^{\delta_1} e(g_3, w)^{r_1} e(A_2, h_0)^{-e} \ \wedge \\
\tfrac{u_{AP}}{S} = S^{J_{AP}} S^s \ \wedge \ A_4 = g_1^{J_{AP}} g_2^t g_3^{r_4} \ \wedge \ A_4^x = g_1^{\delta_J} g_2^{\delta_t} g_3^{\delta_4} \ \wedge \\
\left. \tfrac{u_{AP}^R}{T} = T^{J_{AP}} T^t u_0^{-\delta_J} u_0^{-\delta_t} u_0^x \ \wedge \ 1 \le J_{AP} \le k \right\} (M)
\end{aligned}
$$

where $\delta_1 = r_1 e, \delta_2 = r_2 e, \delta_3 = r_3 e, \delta_J = J_{AP} x, \delta_t = tx, \delta_4 = r_4 x$.

For a more detail protocol for the range check of $J_{AP}$, please refer to the appendix A.

PublicTracing. For two entries $(SPK, S, T, R)$ and $(SPK', S', T', R')$, if $S \neq S'$, then the underlying user of both authentications has not exceed its prescribed usage $k$ or they are from different user.

If $S = S'$, then everyone can compute $u_0^x = (\frac{T^{R'}}{T'^R})^{((R'-R)^{-1})}$. From $u_0^x$ and the identification list, output $i$ as the cheating user. Now if $u_0^x$ does exists, then it can be concluded that GM has deleted some data from the identification list and output GM.

## 5 Variants of Our Scheme

### 5.1 Trading computation efficiency for setup-freeness

We propose a variant of our scheme where the AP enjoys a high degree of setup-freeness. That is, the AP only needs to publish its access group, identity $ID$ and bound $k$. In this new scheme, interactive GrantingAccess and RevokingAccess are no longer needed and there is no need for the AP to keep the $ARC$, too.

We highlight the changes as follow. In the init phase, a common accumulator is initialized for all AP's by randomly selecting $q \in_R \mathbb{Z}_p^*$ and computing $q_i = {h_0}^{q^i}$ for $i = 1, \cdots, t_{max}$, where $t_{max}$ is the maximum number of users in an access group. This procedure can be done by the GM or a trusted third party.

In APSetup, the AP only need to publish is identity and bound $k$. It also needs to maintain a list of users allowed to access its application. Interactive grating access and revoking access are removed. The AP simply needs to change the content of the list of users in its access group.

Finally, users in the access group have to compute its own witness as follow. Retrieve the list of membership public key $\{e_j\}$ of the AP's access group. User with membership public key $e_i \in \{e_j\}$ first accumulates the set $\{e_j\}$ into a value $v$ by computing $v = h_0^{\prod_{k=1}^{k=|\{e_j\}|}(e_k+q)}$. This quantity could be computed without knowledge of $q$ using the $q_i$. Note that both user and AP can compute $v$ locally. The user also computes the witness $w$ by $h_0^{\prod_{k=1,k\neq i}^{k=|\{e_j\}|}(e_k+q)}$ such that $v_w^{(q+e_i)} = v$.

The rest of the protocol follows the original scheme, and same SPK $\Pi_3$ is used.

### 5.2 Trading Key-Size for Constant Proving Effort

Motivated by [20], we outline how to a construct dynamic $k$-TAA with constant proving effort. Each AP has to publish $k$ signatures $Sig(1), \cdots, Sig(k)$. In the proof, instead of proving $1 \leq J_{AP} \leq k$ (which has complexity $O(logk)$), the user proves possession of signature on $J_{AP}$ (which has complexity $O(1)$). This indirectly proves that $J_{AP}$ is within the range. The price to pay is that, the public key size of the AP is now linear in $k$, and user colluding with AP can be untraceable (since the malicious AP can issue several $Sig(J_{AP})$ for the user. BBS+ signature is a natural candidate for the signature scheme used by the AP.

## 6 Security and Efficiency Analysis

### 6.1 Efficiency Analysis

Following the parameters suggested by Boneh *et al.*[4, 3], we can take $p = 170$ bits and each group element in $\mathbb{G}_1$, $\mathbb{G}_2$ can be represented by 171 bits. The authentication protocol will then consists of AP sending a 160-bit *seed* to the prover, while SPK $\Pi_3$ consists of 16 elements in $\mathbb{Z}_p^*$, 4 elements in $\mathbb{G}_1$ and 2 elements in $\mathbb{G}_p$. Assume elements in $\mathbb{G}_p$ is represented by 171 bits (using another elliptic curve group where pairing is not available[13]). Range proof of $J_{AP}$ could be efficiently done if we set $k = 2^\kappa$ for some integer $\kappa$, using the protocol in appendix A, bits transmitted is $((5 + 3\kappa) * 170 + 171\kappa)$.

Then $\Pi_3$ consists of $574.5+85logk$ bytes. On the other hand, if we implement the tradeoff described in Section 5, the range proof is replaced by the possession of a BBS+ signature, which is of size 213 bytes. The following table summarizes the communication cost of most (dynamic) k-TAA schemes in the literature. Security parameters of all schemes are set such that they have comparable security with standard 1024-bit RSA signature (though it should be noted that, the parameters are in slight favor towards NS05 [16], since they use group of orders of a 160-bit prime which result in a slightly weaker security than the 1024-bit RSA signature). The first 3 entries of the table are taken from [16]. Note that $k$ is the allowable number of authentication.

| | Bytes sent by AP | Bytes sent by User | Dynamic |
|---|---|---|---|
| TFS04 scheme | 40 | $60k + 1617$ | No |
| NS05 ordinary | 20 | $60k + 224$ | No |
| NS05 dynamic | 20 | $60k + 304$ | Yes |
| Our dynamic scheme | 20 | 700 or $574 + 85\log(k)$ | Yes |
| TS06 scheme* | 20 | 500 or $300 + 85\log(k)$ | No |

* In TS06[20], full details of the proof of knowledge protocol is not given and thus the figure is just an estimation. We assume same proof of knowledge on range is used. TS06 make use of a group signature scheme [13](referred to as FI scheme hereafter) as we use the BBS+ signature scheme for the join protocol. Assume TS06 uses the signature protocol of the FI scheme for proving knowledge

of a membership certificate, which is 1711 bits(very similar to BBS+). A point to note is that if used this way, the zero-knowledge of the protocol is computational.

### 6.2 Security Analysis

Regarding the security of our dynamic $k$-TAA, we have the following theorem whose proof is shown in is shown in appendix C.

**Theorem 3.** *Our scheme possesses D-Detectability, D-Anonymity and D-Exculpability under the y-DDHI assumptions in the random oracle model.*

## 7 Conclusion

We constructed a constant-size dynamic $k$-TAA scheme and proved its security. We also analyzed the efficiency of our system and compare it with existing (dynamic) $k$-TAA schemes. Our scheme outperforms any existing dynamic $k$-TAA schemes in the literature. Finally, the BBS+ signature we analyze could be useful for other cryptographic systems.

## References

1. M. H. Au, W. Susilo, and Y. Mu. Constant-size dynamic -taa. In *SCN*, pages 111–125, 2006.
2. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT*, pages 56–73, 2004.
3. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *CRYPTO*, pages 41–55, 2004.
4. D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *ASIACRYPT*, pages 514–532, 2001.
5. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT*, pages 431–444, 2000.
6. X. Boyen and B. Waters. Compact group signatures without random oracles. In *EUROCRYPT*, 2006. Available at http://www.cs.stanford.edu/ xb/eurocrypt06/.
7. J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem. *PhD Thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.*, 1998.
8. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In *EUROCRYPT*, pages 302–321, 2005.
9. J. Camenisch and A. Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN*, pages 268–289, 2002.
10. J. Camenisch and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *CRYPTO*, pages 61–76, 2002.
11. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO*, pages 56–72, 2004.
12. Y. Dodis and A. Yampolskiy. A verifiable random function with short proofs and keys. In *PKC*, pages 416 – 431, 2005.

13. J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. In *ACISP*, pages 455–467, 2005.
14. S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.
15. L. Nguyen. Accumulators from Bilinear Pairings and Applications. In *CT-RSA*, pages 275–292, 2005.
16. L. Nguyen and R. Safavi-Naini. Dynamic k-times anonymous authentication. In *ACNS*, pages 318–333, 2005.
17. T. Okamoto. Efficient blind and partially blind signatures without random oracles. In *TCC*, pages 80–99, 2006.
18. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
19. I. Teranishi, J. Furukawa, and K. Sako. k-Times Anonymous Authentication (Extended Abstract). In *ASIACRYPT*, pages 308–322, 2004.
20. I. Teranishi and K. Sako. -times anonymous authentication with a constant proving cost. In *PKC*, pages 525–542, 2006.

## A Range Proof for $J_{AP}$

Secure and efficient exact proof of range is possible in groups of unknown order under factorization assumption [5]. Here, we make use of the fact that if we set $k = 2^t$ for some integer $t$, efficient range check for $J_{AP}$ could be achieved as follows.

Let $g$, $h$ be two generators of a cyclic group $\mathbb{G}$ of order $p$ whose relative discrete logarithm is unknown. To prove knowledge of a number $J$ such that $0 < J \leq k$ in a commitment $C_J = g^J h^r$, let $J_i$ be the $i$-th bit of $J$ for $i = 1, \cdots t$. Compute $C_i = g^{J_i} h^{r_i}$ for some $r_i \in_R \mathbb{Z}_p^*$ for $i = 1, \cdots, t$. Compute the following SPK.

$$\Pi_{range} : SPK\Big\{(J, a, b, r, r_i) : C_J = g^J h^r \ \wedge \ C_J/g = g^a h^r \ \wedge \ \prod_{j=1}^{t} (C_j)^{2^j} = g^J h^b \ \wedge \ [C_i = h^{r_i} \vee C_i/g = h^{r_i}]_{i=1}^{i=t}\Big\}(M)$$

where $a = J - 1$, $b = \prod_{j=1}^{t} r_j 2^j$.

The total protocol consists of $4+t$ elements in $\mathbb{Z}_p$, $2t+1$ challenges also in $\mathbb{Z}_p$ and $t$ $C_i$'s in $\mathbb{G}$. In our protocol, total size of the range proof is $(5+3t)*170+t*171$ bits.

## B Security Analysis of BBS+ Signature

*Proof (theorem 1, unforgeability against adaptively chosen message attack).* Assume there exists a forger $\mathcal{F}$ which could forge a BBS+ signature under adaptively chosen message attack. Suppose it makes $q$ signature queries. We construct a simulator $\mathcal{S}$ which solves the $q$-SDH problem.

$\mathcal{S}$ is given an instance of the $q$-SDH problem $(g_0, h_0, h_0^{\gamma}, \cdots, h_0^{\gamma^q})$. By applying the technique of Boneh and Boyen in lemma 3.2 in [2], it obtains generator $g_0$, $h_0$, $w = h_0^{\gamma}$ and $q$ SDH pairs $(B_i, e_i)$ such that $e(B_i, wh_0^{e_i}) = e(g_0, h_0)$ for each $i$. Due to the technique of the same lemma, any new SDH pairs $(B, e)$ besides these $q-1$ pairs lead to the solution of the original $q$-SDH problem.

$\mathcal{S}$ then randomly select $e^*, a^*, k^* \in_R \mathbb{Z}_p^*$, computes $h_1 = [(wh_0^{e^*})^{k^*} h_0^{-1}]^{1/a^*}$. Note that $h_1 = h_0^{\frac{(e^*+\gamma)k^*-1}{a^*}}$. For $j = 2 \cdots L+1$, where $L$ is the maximum length of the block of messages, randomly select $\mu_j \in_R \mathbb{Z}_p^*$ and compute $h_j = h_1^{\mu_j}$. $(w, g_0, h_0, h_1, \cdots, h_{L+1})$ is then given to $\mathcal{F}$ as the system parameter. Denote $g_i = \psi(h_i)$ for $i = 0 \cdots L+1$.

Out of the $q$ signature queries, $\mathcal{S}$ randomly chooses one, which we shall called query $*$. Suppose the message block to be signed for the $i$-th query is $(m_{1,i}, \cdots, m_{\ell_i,i})$ such that $\ell_i \leq L$. Compute $t_i = m_{1,i}\mu_2 + \cdots + m_{\ell_i,i}\mu_{\ell_i+1}$. Note that $g_2^{m_{1,i}} \cdots g_{\ell_i+1}^{m_{\ell_i,i}} = g_1^{t_i}$.

For the $q-1$ signature queries other than query $*$, $\mathcal{S}$ answers by using the $q-1$ SDH pairs $(B_i, e_i)$ as follows. Randomly chooses $s_i$ and computes $a_i = s_i + t_i$. $\mathcal{S}$ is then to compute $A_i = [g_0 g_1^{a_i}]^{\frac{1}{e_i+\gamma}}$ and return the signature for the $i$-th query as $(A_i, e_i, s_i)$.

$$\begin{aligned} A_i = [g_0 g_1^{a_i}]^{\frac{1}{e_i+\gamma}} &= B_i g_1^{\frac{a_i}{\gamma+e_i}} \\ &= B_i [g_0^{\frac{a_i k^*(e^*+\gamma)-a_i}{(e_i+\gamma)a^*}}] \\ &= (B_i^{(1-\frac{a_i}{a^*})})([g_0^{\frac{a_i k^*}{a^*}}]^{(1-\frac{e_i-e^*}{e_i+\gamma})}) \\ &= (B_i^{(1-\frac{a_i}{a^*}-\frac{(e_i-e^*)a_i k^*}{a^*})})(g_0^{\frac{a_i k^*}{a^*}}) \end{aligned}$$

$\mathcal{S}$ then returns $(A_i, e_i, s_i)$ as the signature.

For query $*$, $\mathcal{S}$ chooses $s^*$ such that $s^* + t_i = a^*$. Compute $A^* = g_0^{k^*}$. Return $(A^*, e^*, s^*)$ as the signature.

Finally, $\mathcal{F}$ output a forged signature $(A', e', s')$ on message $(m'_1, \cdots, m'_{\ell'})$. There are three possibilities.

- Case I $[e' \notin \{e_i, e^*\}]$: Denote $y = s' + m'_1\mu_2 + \cdots + m_{\ell'}\mu_{\ell'+1}$.

$$\begin{aligned} A'^{e'+\gamma} &= g_0 g_1^y \\ A'^{e'+\gamma} &= g_0^{(\frac{k^* y(e^*+\gamma)-y}{a^*})} \\ A' &= (g_0^{\frac{a^*-y}{a^*(e'+\gamma)}})[(g_0^{\frac{k^* y}{a^*}})^{(1-\frac{e'-e^*}{e'+\gamma})}] \\ B' = g_0^{\frac{1}{e'+\gamma}} &= [A' g_0^{\frac{-k^* y}{a^*}}]^{\frac{a^*}{a^*-y-k^* y(e'-e^*)}} \end{aligned}$$

  Uses this new SDH pair $(B', e')$, $\mathcal{S}$ can solve the original $q$-SDH problem.
- Case II: $[e' = e_i$ and $A' = A_i$ or $e' = e^*$ and $A' = A^*]$: This happens with negligible probability unless $\mathcal{F}$ solves the relative discrete logarithm amongst two of the $h_i$'s.

– Case III: [$e' \in \{e_i, e^*\}$ and $A' \neq A_i$ or $A' \neq A^*$]: With probability, $1/q$, $e' = e^*$. Define $y$ as in case 1.

$$A'^{e^*+\gamma} = g_0 g_1^y$$
$$A' = (g_0^{\frac{a^*-y}{a^*(e^*+\gamma)}})(g_0^{\frac{k^*y}{a^*}})$$
$$B^* = g_0^{\frac{1}{e^*+\gamma}} = [A' g_0^{\frac{-k^*y}{a^*}}]^{\frac{a^*}{a^*-y}}$$

$\mathcal{S}$ uses this new SDH pair to solve the original $q$-SDH problem.

If the success probability of $\mathcal{F}$ is $\epsilon$, then in the worst case, success probability of $\mathcal{S}$ is $\epsilon/q$.

## C Security Analysis of Our Dynamic $k$-TAA

*Proof (Sketch, theorem 3).*
**D-Detectability.** Let $\mathcal{A}$ be an adversary who executes $f$ join protocol with simulator $\mathcal{S}$ acting as the GM. Let $\mathcal{E}_{\Pi_0}, \mathcal{E}_{\Pi_1}$ be extractors of the SPK $\Pi_0, \Pi_1$ respectively. For each join request, $\mathcal{S}$ acts exactly as an honest GM would, except during step 3, where $\mathcal{S}$ runs the simulator $\mathcal{E}_{\Pi_1}$ to extract the values $(s, t, x)$. From the value $s$, $\mathcal{S}$ compute the tuple $w_l = (S_1, \cdots, S_k, s, t, x)$ such that $S_i = S = u_0^{\frac{1}{J_{AP}+s+1}}$ for $i = 1, \cdots, k$. Let $A_f = \{S_{i,j} | 1 \leq i \leq f, 1 \leq j \leq k\}$ after $f$ executions of the join protocol.

Since the protocol is done non-interactively, $\mathcal{S}$ is given control over the random oracle in addition to the black-box access to $\mathcal{A}$. Moreoever, such extraction requires rewind simulation and thus the join protocol cannot be executed concurrently.

For each AP, let $u_{AP} = H_{evt}(ID_{AP}) = u_0^{r_{AP}}$ for some randomly chosen $r_{AP}$. Due to the soundness of the underlying SPK, $A_f$, together with the set $\{r_{AP}\}$ contains all *valid* $S$ that $\mathcal{A}$ can produce, except with negligible probability. For $\mathcal{A}$ to break D-Detectability, one of the following two happens. (1) $\mathcal{A}$ convince an honest AP to accept an $S$ for which it cannot generate an honest proof of validity with some non-negligible probability. (2) $\mathcal{A}$ uses duplicated $S$ but public tracing does not output the identity of the user within the $f$ join protocol.

Consider case (1) such that $\mathcal{A}$ convinces an honest AP to accept an invalid $S$ during the authentication protocol. Then $\mathcal{A}$ must have conducted a *false* proof as part of the signature of knowledge such that one of the following is fake:

1. $A^{e+\gamma} = g_0 g_1^s g_2^t g_3^x$.
2. $w^{e+q} = v$.
3. $S = u_{AP}^{\frac{1}{J_{AP}+s+1}}$.
4. $T = u_0^x u_{AP}^{\frac{R}{J_{AP}+t+1}}$.
5. $0 \leq J_{AP} \leq k$

Item 1 happens with negligible probability under the $q$-SDH assumption, as forging item 1 implies breaking the unforgeability of the BBS+ signature discussed above. Item 2 happens with negligible probability under the assumption that the accumulator is secure[15] (also reduced to $q$-SDH assumption). Item 3,4 happens with negligible probability under the DL assumption (which will be subsumed by the $y$-DDHI assumption in the theorem). Item 5 happens with negligible probability if ZKPOK of committed number lies in an exact interval exists. For instance, [5] propose such an ZKPOK under the factorization assumption. If technique of [5] is used, in the setup procedure, a group of unknown order have to be set up. On the other hand, if we set $k = 2^t$, then we can use the protocol outlined in A. We omit the details of exact range proof in the protocol for simplicity. Thus, the total success probability of $\mathcal{A}$ in case 1 is negligible.

Consider case (2). It have already been proved in case (1) that $\mathcal{A}$ cannot have an honest AP to accept an invalid $S$ with non-negligible probability. Since $\mathcal{A}$ must use valid $S$, to authenticated more than $k$ times for the same AP, it must uses dulplicated $S$. Let the honest AP accept two authentication $(S, SPK_1)$ and $(S, SPK_2)$ for some valid $S$ such that $S^{r_{AP}} \in A_f$. Since the AP is honest, $R_1 \neq R_2$ with high probability. We are to show that $T_1 = u_0^x u_{AP}^{\frac{R_1}{J_{AP}+t+1}}$ and $T_2 = u_0^x u_{AP}^{\frac{R_2}{J_{AP}+t+1}}$ so that identity of the cheater could be recovered from the PublicTracing algorithm.

Since $R_1$, $R_2$ are chosen by the honest AP, this uniquely fixes $T_1$, $T_2$ as the only valid T's to accompany the dulplicated $S$ in these two authentications. To deviate from these $S$ and $T$, $\mathcal{A}$ must conduct fake proof of validity of the authentication protocol which we already shown to happen with only negligible probability. Thus, PublicTracing output the identity of the cheater without overhelming probability.

During the course of the running of $\mathcal{A}$, it is allowed to query several oracles. We outline how $\mathcal{S}$ simulate these oracles. The join oracle is simulated by invoking the signing oracle of the BBS+ signature (protocol for signing committed block of messages, which involve extracted the committed block of messages in the commitment. Again, this requirement make the join procedure non-concurrent). Authentication oracle is simulated by randomly generate $s, t, x, J_{AP}$, backpatch the random oracle and simulate the signature of knowledge of the authentication procedure.

**D-Anonymity.** The adversary $\mathcal{A}$, colluding with GM and all AP's, create the global system parameters. Finally, $\mathcal{A}$ will be asked to engage in a legal number of authentication protocol with some real user $j$ or simulator $\mathcal{S}$.

For each authentication procedure, $\mathcal{S}$ simulate as follow.

- $\mathcal{S}$ is given *seed* and compute $R = H(seed)$.
- $\mathcal{S}$ random chooses $s, t, x$ and a random $J_{AP} \in_R \{1, \cdots, k\}$ and compute $S = u_{AP}^{\frac{1}{J_{AP}+s+1}}$, $T = u_0^x u_{AP}^{\frac{R}{J_{AP}+t+1}}$.
- $\mathcal{S}$ simulated a proof of $A, w, e, s, t, x, J_{AP}$ such that $(1)A^{e+\gamma} = g_0 g_1^s g_2^t g_3^x$, $(2)w^{e+q} = v$, $(3)\ S = u_{AP}^{\frac{1}{J_{AP}+s+1}}$, $(4)\ T = u_0^x u_{AP}^{\frac{R}{J_{AP}+t+1}}$, $(5)0 \leq J_{AP} \leq k$.

– The proof of item (3), (4), (5) are real while (1) is handled by the simulator for a proof of knowledge of a BBS+ signature and (2) is handled by the simulator for a proof of knowledge of the accumulator.

We now explain why the output of $\mathcal{S}$ is computationally indistinguishable from the output of a real user. The idea is that during the join protocol $\mathcal{A}$ learn nothing about the set of secrets of $(s, t, x)$ of the real user, due to the security of the BBS+ signature. Thus, the values $(s, t, x)$ chosen by $\mathcal{S}$ is indistinguishable from those chosen by real users. Due to the security of the PRF[12], $S, T$ are indistinguishable from randomly elements under the $y$-DDHI assumption. Thus, $\mathcal{A}$ can distinguish a real user and a simulator only if it could distinguish a real proof or a simulated proof of the BBS+ signature, or it could break the security of the PRF($y$-DDHI assumption). The probability is negligible under $y$-DDHI assumption.

**D-Exculpability.** D-Exculpability for GM is quite straight forward. Suppose the GM is honest but two authentication trascript $(S, R_1, SPK_1)$, $(S, R_2, SPK_2)$ pointed to an entry not exists in the identification list, then someone have been able to fake the proof of knowledge either in the join protocol or in the authetntication protocol, which happen with negligible probability. Proof of D-Exculpability for honest user is also quite straight forward. The proof of knowledge of $T$ in the authentication protocol involve the user secret $x$. To slander an honest user, adversary without knowledge of user secret $x$ have to fake the knowledge of $T$ which involve knowledge of $x$ to base $u_0$. This happens with negligible probability.