2013

Constant-size dynamic k-times anonymous authentication

Man Ho Au  
*University of Wollongong, aau@uow.edu.au*

Willy Susilo  
*University of Wollongong, wsusilo@uow.edu.au*

Yi Mu  
*University of Wollongong, ymu@uow.edu.au*

Sherman S. M Chow  
*University of Hong Kong*

---

**Publication Details**

Constant-size dynamic k-times anonymous authentication

Abstract
Dynamic 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, where application providers can independently and dynamically grant or revoke access right to members in their own group. In this paper, we construct a dynamic k-TAA scheme with space and time complexities of $O(\log(k))$ and a variant, in which the authentication protocol only requires constant time and space complexities at the cost of $O(k)$-sized public key. We also describe some tradeoff issues between different system characteristics. We detail all the zero-knowledge proof-of-knowledge protocols involved and show that our construction is secure in the random oracle model under the $q$-strong Diffie-Hellman assumption and $q$-decisional Diffie-Hellman inversion assumption. We provide a proof-of-concept implementation, experiment on its performance, and show that our scheme is practical.

Keywords
size, dynamic, constant, authentication, k, anonymous, times

Disciplines
Engineering | Science and Technology Studies

Publication Details

This journal article is available at Research Online: http://ro.uow.edu.au/eispapers/2391
Constant-Size Dynamic 
k-Times Anonymous Authentication

Man Ho Au, Member, IEEE, Willy Susilo, Senior Member, IEEE, Yi Mu, Senior Member, IEEE, Sherman S.M. Chow Member, IEEE,

Abstract—Dynamic 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, where application providers can independently and dynamically grant or revoke access right to members in their own group. In this paper, we construct a dynamic k-TAA scheme with space and time complexities of O(log(k)) and a variant in which the authentication protocol only requires constant time and space complexities at the cost of O(k-sized public key. We also describe some trade-off issues between different system characteristics.

We detail all the zero-knowledge proof-of-knowledge protocols involved and show that our construction is secure in the random oracle model under the q-Strong Diffie-Hellman assumption and q-Decisional Diffie-Hellman Inversion assumption. We provide a proof-of-concept implementation, experiment on its performance and show that our scheme is practical.

Index Terms—anonymity, applied cryptography, authentication, implementation, pairings

I. INTRODUCTION

Anonymous authentication allows users to show their membership of a particular group without revealing their exact identities. Teranishi, Furukawa and Sako [2] proposed a k-times anonymous authentication (k-TAA) scheme (TFS04) so that users of a group can access applications anonymously while application providers (AP’s) can decide the number of times users can access their applications. To do so, users first register to the group manager (GM) and each AP announces 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 number of times. Anyone can trace a dishonest user who tries to access an application for more than the allowable number of times.

For higher flexibility, AP’s may wish to select their own group of users. However, there is no control over who can access which applications in k-TAA. In dynamic k-TAA, proposed by Nguyen and Safavi-Naini [3] (NS05), the role of AP’s is more active and they can select their user groups, granting and revoking access of registered users independently.

Previous k-TAA schemes such as TFS04 and NS05 are quite efficient in the sense that both time and space complexities are independent of the total number of users. However, the size of AP’s public key and the communication cost between users and AP’s are both O(k). The computational cost of the user for an authentication protocol is also O(k).

A. Our Contributions

In this paper, we construct a dynamic k-TAA scheme with complexity of O(log(k)). We then propose a variant of our scheme with cost O(1) at the cost of O(k-sized public key. The security of our scheme is proven in the random oracle model under the q-Strong Diffie-Hellman (q-SDH) assumption and q-Decisional Diffie-Hellman Inversion assumption.

Our construction requires a signature scheme with efficient protocols such as CL signature [4]. As outlined in [5], a short group signature scheme due to Boneh, Boyen and Shacham [6] can be modified as a q-SDH variant of CL signature. We supply the details of the modification with the protocols, and get a new signature scheme which we call BBS+ signature. We prove that BBS+ signature is strongly-unforgeable in the standard model under the q-SDH assumption. Besides, the protocol of showing possession of a signature is different from [6] in which the modified protocol achieves perfect zero-knowledge, while the original protocol is only computational due to the identity-escrow feature. This BBS+ signature is a building block which can be employed in other settings and may be of independent interest.

Compared with the conference version [1], we gave the details of all zero-knowledge proof-of-knowledge (ZKPoK) used in our system. BBS+ and the ZKPoK we built has been adopted as a building block for a range of applications.

Most importantly, with these details, we can provide a formal proof of security of our system, which was not available before.

Finally, we provide a sample implementation of our construction and experiment on its performance. The result confirms our theoretical complexities analysis that our protocol is practical. The result of this part is done after the publication of [1].

B. Subsequent Work

The BBS+ signature described in this paper has been used in various subsequent work on anonymous authentication system and privacy-enhancing technologies. For example, [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. BBS signature [6] has also been extended or modified in other work. For example, Yang et al. [12] proposed a re-randomizable (and hence cannot be strongly unforgeable by definition) variant.
Some of the conceptual building blocks used by our scheme have also been improved or generalized. For example, Camenisch, Chaabouni and Shelat [17] generalized the signature-based range proof used by our system. Another pairing-based accumulator has been proposed in [18]. ZKPoK used by our systems are all instantiated by using Fiat-Shamir heuristics. A common-reference string approach of ZKPoK for pairing-product equations has been proposed by Groth and Sahai [19], which provides stronger formal security guarantee at the cost of run-time performance.

Some design principles used in our system have also influenced the design of other kinds of anonymous authentication systems, such as the notion of real traceable signature [10]. This notion is put forth by Chow [10] which gives an efficient cryptographic building block for various applications, namely, transforming an anonymous system to one with “fair privacy”, a mix-net application where originators of messages can be opened, and open-bid auctions [20]. The mechanism for tracing the signatures of malicious users is very efficient.

C. Related Work

Teranishi and Sako [21] (TS06) proposed a k-TAA scheme with constant proving cost. Our construction can be seen as an extension of TS06 to dynamic k-TAA in which AP can control their access group dynamically. This is achieved by the use of dynamic accumulator as in [3] due to the idea in [22]. Instead of pseudorandom function (PRF), TS06 uses a weakened version which they termed as partial pseudorandom function (PPRF). Nevertheless, our choice of PRF is more efficient than their PPRF.

As pointed out in [21], k-TAA schemes share certain similarities with compact e-cash schemes, introduced in [23]. In the latter, a user can only spend (c.f. authenticate) up to k times to all shops (c.f. application providers) combined; while for k-TAA, each application provider may choose its own allowable number of access and the number of accesses to different applications by a user are not related. For instance, a user could authenticate himself n1 times to one AP and n2 times to another AP, provided that n1 and n2 are less than the limit imposed by the respective AP. Despite these differences, similar techniques can be used to build k-TAA and compact e-cash.

One may view identity-based ring signatures as group signatures with no anonymity management mechanism, by treating the key generation center as the group manager. However, many schemes such as [24] produce signature of size linear in the number of group members, although the verification procedure may be efficient. Some schemes such as [25] do feature constant-size signature sizes. On top of that, linkability is considered which can be seen as a kind of anonymity management mechanism. However, the standard notion of linkable identity-based ring signatures just leads to 2-times anonymous authentication.

Another closely related notion is event-oriented k-times revocable iff-linked group signatures (k-RiffLGS), introduced by Au, Susilo and Yiu [26]. k-RiffLGS is a group signature scheme such that every user can sign on behalf of the group anonymously for up to k times per event, represented by a bit string. No one, not even the group manager, can revoke the identity of the signer. Signatures of the same user for different event cannot be linked. If the user signs for more than k times for any event, his identity can be revoked by everyone. Thus, every legitimate user can sign on behalf of the group for up to k times per event and there is no limit for the number of events. In fact, as mentioned in [26], k-RiffLGS can be viewed as a non-interactive version of k-TAA. On the other hand, dynamic k-TAA can be viewed as an interactive version of k-RiffLGS with revocation.

Other signature schemes with efficient protocol, such as CL signature [4] and CL+ signature [5], could also be used for our construction. BBS+ is good in our case for two reasons.
1) Signature size of BBS+ is shorter than CL or CL+ signature for multi-block messages. Specifically, signature size of CL+ is linear to number of message blocks to be signed; while CL and BBS+ are constant size, of length 1346 and 511 bits respectively, for security comparable to 1024-bit standard RSA signature.
2) The accumulator we will use is secure based on the same assumption for which security of BBS+ signature also relies on. On the other hand, the security of CL signature and CL+ signature is based on Strong RSA assumption and LRSW assumption respectively.

Organization. The rest of the paper is organized as follows. Preliminaries are presented in Section II. The framework and the security notions of dynamic k-TAA are reviewed in Section III. We present our construction in Section IV, followed by its variants in Section V. We analyze the security and complexity of our systems in Sections VII and VIII. Details of our sample implementation and efficiency analysis are given in Section IX. We conclude our paper and discuss some future research directions in Section X.

II. Preliminaries

A. Bilinear Pairing

We review the notion of bilinear pairing here. A mapping $\hat{e} : G_1 \times G_2 \rightarrow G_T$ is a bilinear pairing if

- $G_1$ and $G_2$ are cyclic multiplicative groups of prime order $p$.
- $g, h$ are generators of $G_1$ and $G_2$ respectively.
- $\psi : G_2 \rightarrow G_1$ is a computable isomorphism from $G_2$ to $G_1$, with $\psi(h) = g$.
- (Bilinear) $\forall x \in G_1, y \in G_2$ and $a, b \in \mathbb{Z}_p$, $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$.
- (Non-degenerate) $\hat{e}(g, h) \neq 1$.
- (Unique Representation) each element of $G_1, G_2$ and $G_T$ has unique binary representation.

$G_1$ and $G_2$ can be the same group or different groups. We say that two groups ($G_1, G_2$) are a bilinear group pair if the group action in $G_1, G_2$, the isomorphism $\psi$ and the bilinear mapping $\hat{e}$ are all efficiently computable.

B. Mathematical Assumptions

Definition 1 (Decisional Diffie-Hellman): The Decisional Diffie-Hellman (DDH) problem in $G = \langle g \rangle$ 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 DDH assumption holds in \(\mathbb{G}\) if no PPT algorithm has non-negligible advantage over random guessing in solving the DDH problem in \(\mathbb{G}\).

Definition 2 \((q\text{-Strong Diffie-Hellman})\): The \(q\text{-Strong Diffie-Hellman} (q\text{-SDH})\) problem in a bilinear group pair \(\left(\mathbb{G}_1, \mathbb{G}_2\right)\) with trace map \(\psi\) is defined as follows: On input a \((q + 2)\)-tuple \((g, h, h^\gamma, h^{\gamma^2}, \ldots, h^{\gamma^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}\) such that there exists a trace map \(\psi\) from \(\mathbb{G}_2\) to \(\mathbb{G}_1\) with \(g = \psi(h)\), output a pair \((B, c)\) such that \(B^{\gamma^{c+1}} = g\) where \(c \in \mathbb{Z}_q^*\). We say that the \(q\text{-SDH} assumption holds in \(\left(\mathbb{G}_1, \mathbb{G}_2\right)\) if no PPT algorithm has non-negligible advantage in solving the \(q\text{-SDH}\) problem in \(\left(\mathbb{G}_1, \mathbb{G}_2\right)\).

The \(q\text{-SDH} assumption is shown to be true in the generic group model \([27]\) even when \(\left(\mathbb{G}_1, \mathbb{G}_2\right)\) is a bilinear group pair with trace map.

Definition 3 \((q\text{-Decisional Diffie-Hellman Inversion})\): The \(q\text{-Decisional Diffie-Hellman Inversion (q-DDHI)}\) in prime order group \(\mathbb{G} = \langle g \rangle\) is defined as follows: On input a \((q + 2)\)-tuple \(g, g^x, g^{x^2}, \ldots, g^{x^q}, g^c \in \mathbb{G}^{q+2}\), output 1 if \(c = 1/x\) and 0 otherwise. We say that the \(q\text{-DDHI} assumption holds in \(\mathbb{G}\) if no PPT algorithm has non-negligible advantage over random guessing in solving the \(q\text{-DDHI}\) problem in \(\mathbb{G}\).

C. Building Blocks

Signature Scheme. Signature scheme is a basic cryptographic primitive for message authentication. There are various security notions for unforgeability of signatures. A weak notion is that the set of messages to be signed is known in advance, even before the generation of the public key. Scheme that is existentially unforgeable under this setting is called weakly-secure. On the other hand, strong unforgeability guarantees that it is difficult to come up with a new signature on a message \(m\) even if the adversary can adaptively get many signatures on the messages he wants, including one on \(m\). In a variant of our dynamic \(k\text{-TAA}\) scheme with constant proving effort, we employed a weakly-secure short signature by Boneh and Boyen \([27]\) as in \([21]\). For the BBS+ signature we are going to describe, we will show that it is strongly-unforgeable. We use the notation \(\text{Sig}(m)\) to denote a signature on the message \(m\).

Zero-Knowledge Proof of Knowledge. In zero-knowledge proof of knowledge \([28]\), a prover proves to a verifier that a statement is true without revealing anything other than the veracity of the statement. Our construction involves statements related to knowledge of discrete logarithms constructed over a cyclic group \(\mathbb{G}\) of prime order \(p\). These proofs can also be done non-interactively by incorporating the Fiat-Shamir heuristic \([29]\). The non-interactive counterpart is referred to as signature proof of knowledge, or SPK for short. They are secure in the random oracle model. Following the notation introduced by Camenisch and Stadler \([30]\), \(\text{PoK}\{x : y = g^x\}\) denotes a zero-knowledge proof of knowledge protocol between a prover and a verifier such that the prover knows some \(x \in \mathbb{Z}_p\) where \(y = g^x \in \mathbb{G}\). The corresponding non-interactive signature proof of knowledge on a message \(m\) shall be denoted as \(\text{SPK}\{x : y = g^x\}(m)\). One can view this as a signature on the message \(m\) signed by a discrete-logarithm based key pair \((g^x, x)\).

Signature with Efficient Protocols. In this paper, a signature scheme with efficient protocols refers to a signature scheme with two protocols: (1) a protocol between a signature requester and a signer in which the requester obtains a signature on \((m_1, \ldots, m_L)\) from the signer while the signer only gets a commitment of a multi-block message \((m_1, \ldots, m_L)\) but learns nothing about all these messages; (2) a protocol for the proof of the knowledge of a signature with respect to some multi-block messages without revealing any information about the signature nor the messages. For signature scheme with efficient protocols, the security notion allows the adversary to get signatures through the signature generation protocol or “directly” (by supplying the messages in clear and obtaining only the final output of the protocol). Details can be found in \([4]\). In our construction, we employed BBS+ signature to be described.

Pseudorandom Function. Another building block of our construction is a pseudorandom function with efficient proof of correctness of its output. We employed a particular construction of PRF due to Dodis and Yampolskiy \([31]\) (DY-PRF). DY-PRF is defined by a tuple \((\mathbb{G}_p, p, g, s)\), where \(\mathbb{G}_p = \langle g \rangle\) is a cyclic group of prime order \(p\) and \(s\) is an element in \(\mathbb{Z}_p\). On input \(x\), \(\text{PRF}_{g,s}(x) = g^{x^s}\). Efficient proof for correctly formed output (with respect to \(s\) and \(x\) in some commitments such as Pedersen commitment \([32]\) exists and the output of \(\text{PRF}_{g,s}\) is indistinguishable from random elements in \(\mathbb{G}_p\), provided that the \(q\text{-DDHI} assumption holds, see \([23]\) for details.

Accumulator. The dynamic feature of our construction is built from the dynamic accumulator with one-way domain due to Nguyen \([33]\) (N-Acc). Roughly speaking, an accumulator is an algorithm that combines a large set of values \(\{x_i\}\) into a short accumulator \(V\). For each value \(x_j \in \{x_i\}\), there exists a witness \(w_j\) which can prove \(x\) is indeed accumulated in accumulator \(V\). An accumulator is dynamic if it allows values to be added or deleted dynamically. SPK of a witness is also described in \([33]\). N-Acc is a secure dynamic accumulator under the \(q\text{-SDH} assumption, see \([33]\) for details.

III. Framework

A. Syntax

We briefly review the model of dynamic \(k\text{-TAA}\) in \([3]\). A dynamic \(k\)-times anonymous authentication involves three kinds of entities, namely, group manager (GM), application providers (AP\(_j\)) and users (U\(_i\)). It consists of seven polynomial time algorithms or protocols (GMSetup, Join, APSetup, GrantAccess, RevokeAccess, Authentication, PublicTracing). The following enumerates the syntax.

- GMSetup. On input a security parameter \(1^\lambda\), the algorithm outputs GM secret key \(gsk\) and group public key \(gpk\). For simplicity of the framework, we assume the GM is also responsible for the generation of the system parameter, which is included in \(gpk\). All algorithms below have \(gpk\) as one of their implicit inputs.
• Join. This protocol allows a user $U_j$ to join the group and obtain a membership public/secret key pair $(mpk_j, msk_j)$ from GM. GM also adds $U_j$’s identification and membership public key to the membership public key archive. A user is called a group member if its identification and membership public key is in the membership public key archive.

• APSetup. An application provider $AP_j$ publishes its identity and announces the number of times $k_j$ that a user can access its application. The algorithm may also generates the public and private key $(apk_j, ask_j)$ of $AP_j$.

• GrantAccess. Each $AP_j$ manages its own access group $L_j$ which is initially empty. This procedure allows $AP_j$ to add a user $U_i$ to its access group $L_j$ and thus grant him the permission to use its application.

• RevokeAccess. It allows $AP_j$ to remove a member from his access group $L_j$ and stop this group member from accessing his application.

• Authentication. User $U_i$ authenticates himself to an application provider $AP_j$ through this protocol. $U_i$ is authenticated only if he is in the access group $L_j$ and the number of accesses have not exceeded the allowed number $k_j$. 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 the membership public key of a user $U_i$, GM or NIL which indicates user “$U_i$ tries to access more than the allowed number of times”, “GM cheated” and “there is no malicious party in this authentication log” respectively.

For correctness, an honest member who is in the access group and has not authenticated himself for more than the allowed number of times, must be authenticated by an honest AP.

B. Security Requirements

We briefly recall security requirements here, for formal definition please refer to [3], [2].

• D-Detectability. A subset of colluded users cannot perform the authentication procedure with the same honest application provider for more than the allowed number of times, or they must be detected by the PublicTracing algorithm.

• D-Anonymity. No collusion of application providers, users and group manager can distinguish between authentication executions of two honest group members who are in the access group of that application provider.

• D-Exculpability. An honest user cannot be proven to have performed the authentication procedure with the same honest AP for more than the allowed number of times. It is also required that the PublicTracing algorithm shall not output GM if the group manager is honest even though all application providers and users collude.

IV. OUR CONSTRUCTION

Our dynamic k-TAA is built from the $q$-SDH based accumulator due to Nguyen [33] (N-Acc), the PRF due to Dodis and Yampolskiy [31] (DY-PRF) and BBS+ signature described below.

A. Global Common Parameters

Let $\lambda$ be the security parameter. Let $(G_1, G_2)$ be a bilinear group pair with computable isomorphism $\psi$ as discussed in Section II-A such that $G_1 = \langle g \rangle$, $G_2 = \langle h \rangle$ and $|g| = |h| = p$ for some prime $p$ of $\lambda$ bits. Assume $G_\mu = \langle u \rangle$ is a cyclic group of order $p$ such that the DDH assumption holds. Let $g_0, g_1, g_2, \ldots, g_L, g_l$ be additional random elements in $G_1$.

They are required for the construction of the zero-knowledge proof-of-knowledge protocols. $L$ is the number of messages in BBS+ signature. In our dynamic k-TAA scheme, $L = 2$.

The generation of this common parameter can be done by the GM or by a trusted third party.

B. BBS+ Signature

KeyGen. Choose $\mu \in R Z_p^*$ and compute $Z = h^{\mu}$. The secret key is $\mu$ and the public key is $Z$.

Signing Multi-block Messages. On input $(m_1, \ldots, m_L) \in Z_p^L$, choose $e, s \in R Z_p^*$. Compute $\varsigma = \frac{gg_0^{m_1} \cdots g_L^{m_L}}{Z^s}$. Signature on $(m_1, \ldots, m_L)$ is $(\varsigma, e, s) \in \langle G_1 \times G_p \times Z_p^* \rangle$.

Signature Verification. To verify a signature $(\varsigma, e, s)$ on a multi-block message $(m_1, \ldots, m_L)$, check if $\hat{\epsilon}(\varsigma, Z^{h^e}) = \hat{\epsilon}(gg_0^{m_1} \cdots g_L^{m_L}, h)$.

Protocol for Signing Committed Messages. The protocol is also known as the signature generation protocol. The user first computes a Pedersen commitment on the multi-block message to be signed. That is, the user randomly generate $s' \in R Z_p^*$ and computes $C_m = g_0^{s'} g_1^{m_1} \cdots g_L^{m_L}$. He sends $C_m$ to the signer, along with the following proof

$$\text{PoK}_0\left\{ (s', m_1, \ldots, m_L) : C_m = g_0^{s'} g_1^{m_1} \cdots g_L^{m_L} \right\}$$

After verifying $\text{PoK}_0$, the signer chooses $s'', e' \in R Z_p^*$, computes $\varsigma' = \frac{gg_0^{s''} g_m}{Z^{s''}}$ and sends $(\varsigma, e, s'')$ back to the user. The user computes $s = s' + s''$. The signature on the multi-block messages is $(\varsigma, e, s)$. For any block of messages $(m_1, \ldots, m_L)$, there exists an $s'$ such that $C_m = g_0^{s'} g_1^{m_1} \cdots g_L^{m_L}$ and thus $C_m$ reveals no information about the multi-block message signed.

Proof of Knowledge of A Signature on Committed Messages. We give a zero-knowledge proof of knowledge protocol for showing possession of a message-signature pair. Using any protocol for proving relations amongst components of a discrete-logarithm representation of a group element [34], it can be used to demonstrate relations among components of the multi-block message signed. Specifically, let $\varsigma_m = g_0 g_1^{m_1} \cdots g_L^{m_L}$ be a commitment of a multi-block message $(m_1, \ldots, m_L)$ with randomness $r$, a user who is in possession of a signature $(\varsigma, e, s)$ can conduct the following zero-knowledge proof-of-knowledge protocol with any verifier.

$$\text{PoK}_1\left\{ (\varsigma, e, s, m_1, \ldots, m_L, r) : \begin{align*}
\hat{\epsilon}(\varsigma, Z^{h^e}) &= \hat{\epsilon}(gg_0^{m_1} \cdots g_L^{m_L}, h) \\
\varsigma_m &= g_0 g_1^{m_1} \cdots g_L^{m_L}
\end{align*} \right\}$$
Instantiation of PoK₁ is shown in Section VI-A. We would like to remark that our instantiation is different from the protocol in [6] in which the later is computational zero-knowledge (under the Decision-Linear Diffie-Hellman assumption) while ours is perfect zero-knowledge.

Security Analysis. Unforgeability of BBS+ signature is asserted by the following theorem.

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

Proof is deferred to Section VII-A.

C. Overview of Our Construction

We give a high-level description of our construction here.

GMSetup and APSetup. The GM generates the key pair of BBS+ signature. Each AP publishes a bound k together with the public parameters for the accumulator N-Acc and those for the pseudorandom function DY-PRF.

Join. To join the group, user randomly generates x, t ∈ R Zp*, where k only generate component (by demonstrating he is in possession of a BBS+ signature S asserted by the following theorem.

Security Analysis. Unforgeability of BBS+ signature is shown in Section VI-A. We would like to remark that our instantiation is different from the protocol in [6] in which the later is computational zero-knowledge (under the Decision-Linear Diffie-Hellman assumption) while ours is perfect zero-knowledge.

D. Details of Our Construction

GMSetup. The GM randomly selects µ ∈ R Zp* and computes Z = hµ. The GM also manages a membership public key archive which is a list of 3-tuple (U_i, y_i, e_i). The list is initially empty.

APSetup. AP_j publishes his identity (denoted by AP_j) and a number k_j which is much smaller than 2^A. In addition, AP_j selects h_j ∈ R G_2, γ_j ∈ R Zp* and computes u_j = H(AP_j) ∈ G_p, for some hash function H and Y_j = h_jγ_j ∈ G_2. The public key and the secret key of AP_j are (AP_j, k_j, h_j, u_j, γ_j) and γ_j respectively. AP_j maintains an authentication log and an access group list L_j of 3-tuple. The list is initialized to (∩, ∩, ∩, ψ(h_j)).

Join. User U_i obtains his membership public/secret key pair from GM through the following interactive protocol.

1) U_i randomly selects s_i, t_i, x_i ∈ R Zp* and sends C’ = g_0^{s_i} g_1^{t_i} g_2^{x_i} to GM, along with proof
   PoK_2( (s_i, t_i, x_i) : C’ = g_0^{s_i} g_1^{t_i} g_2^{x_i} )

2) GM verifies the proof and randomly selects s''_i ∈ R Zp*.
   He sends s''_i to the user.

3) User computes and sends s_i = s''_i mod p, y_i = u^{x_i} to GM (recall that u is the generator of G_p), along with proof
   PoK_3 ( (s_i, t_i, x_i) : y_i = u^{x_i} ∧ C’ = g_0^{s_i} g_1^{t_i} g_2^{x_i} )

4) GM verifies the proof computes C = C’ g_0^{s_i} and selects e_i ∈ R Zp*. He then computes s_i = (C’ g_0^{s_i})^{e_i} and sends (s_i, e_i) to the user. The GM also adds the entry (U_i, y_i, e_i) to its membership public key archive.

5) U_i checks if ĉ(s_i, Z^{h_i}) = ĉ(g^{s_i} g_0^{t_i} g_2^{x_i}, h). He stores his membership public key and membership secret key as (y_i, e_i) and (s_i, t_i, x_i).

GrantAccess. AP_j grants access to user U_i who has a public key (y_i, e_i) in this protocol. Suppose the last entry in L_j is (s_i, y_i, V_j). AP_j computes V_j’ = V_j e_iγ_j and sets w_{i,j} = V_j’. He returns w_{i,j} to U_i and appends (e_i, ADD, V_j’) to L_j. w_{i,j} is called a membership witness of U_i for AP_j. U_i can check its correctness by checking if ĉ(w_{i,j}, Y_j h_i) = ĉ(V_j’, h).

The following describe how other users update his witness when the list L_j is updated. This is simply a rephrasing of the update algorithm of N-Acc. Specifically, suppose user U_i such that e_i is in L_j is required to update their membership witness
PublicTracing. For two entries \((w_{i,j}, Y_jh_j^{c_i}) = \hat{e}(V_j, h_j)\). \(U_i\) just computes \(w'_{i,j} = V_jw^{(c_i-e_i)}\) and sets \(w_{i,j}' = w'_{i,j}\).

RevokeAccess. This protocol allows \(AP_j\) to remove the access right of user \(U_i\). Suppose the last entry of \(L_j\) is \((*,*,v_j)\). \(AP_j\) wishes to remove the access right of \(U_i\) implies there exists an entry \((e_i, \text{ADD}, V_j)\) in \(L_j\) and there is no entry of the form \((e_i, \text{REMOVE}, *)\) in the list after that. \(AP_j\) computes \(V'_j = (V_j)^{t_{i,j}^{-1}}\) and appends \((e_i, \text{REMOVE}, V'_j)\) to \(L_j\).

User \(U_i\) (except \(U_i\)) such that \(e_i\) is in \(L_j\) is required to update their membership witness \(w_{i,j}\) using the update algorithm of N-Acc. Suppose \(\hat{e}(w'_{i,j}, Y_jh_j^{c_i}) = \hat{e}(V_j, h_j)\). \(U_i\) computes \(w'_{i,j} = \left(\frac{v'_j}{w'_{i,j}}\right)^{t_{i,j}^{-1}}\) and sets \(w_{i,j} = w'_{i,j}\).

Authentication. User \(U_i\) manages a set of counters \(\{n_{i,j}\}\), such that \(n_{i,j}\) is the number of times \(U_i\) has authenticated himself to \(AP_j\). The protocol below shows how \(U_i\) authenticates himself to \(AP_j\). We assume the membership witness \(w_{i,j}\) of \(U_i\) for \(AP_j\) is update, that is, if the last entry of \(L_j\) is \((*,*,V_j)\), \(\hat{e}(w_{i,j}, Y_jh_j^{c_i}) = \hat{e}(V_j, h_j)\).

- \(AP_j\) sends a random challenge \(m\) to \(U_i\). Denote \(R = H(m||AP_j)\) such that \(H\) is a cryptographic hash function which maps to \(\mathbb{Z}_p^*\). Both parties compute \(R\) locally. In practice, \(m\) can also be a random number together with some information about the current session.
- \(U_i\) computes one-time pass \(S = u_j^{\frac{1}{t_{i,j}+n_{i,j}}}\), tracing tag \(T = u_j^{x_i}u_j^{\frac{1}{t_{i,j}+n_{i,j}}}\) and proves in non-interactive zero-knowledge manner (1) - (5):
  1. \(U_i\) is in possession of a BBS+ signature \((s_i, e_i, s_i)\) from \(GM\) on \((t_i, x_i)\).
  2. \(e_i\) is in the accumulator of \(AP_j\), that is, \(\hat{e}(w_{i,j}, Y_jh_j^{c_i}) = \hat{e}(V_j, h_j)\).
  3. \(S\) is \(PRF_{u_j,s_i}(n_{i,j})\), that is, \(S = u_j^{\frac{1}{t_{i,j}+n_{i,j}}}\).
  4. \(T\) is \(\hat{e}(w_{i,j}, Y_jh_j^{c_i})\), that is, \(T = u_j^{x_i}u_j^{\frac{1}{t_{i,j}+n_{i,j}}}\).
  5. \(0 \leq n_{i,j} < k_j\)

- The above can be abstracted as \(\text{SPK}_4\) whose instantiation is shown in Section VI-D.

\[
\begin{align*}
\text{SPK}_4 & = \left\{ 
\begin{array}{l}
\{ (s_i, e_i, s_i, t_i, x_i, n_{i,j}, w_{i,j}) : \\
\quad \hat{e}(s_i, Zh^{c_i}) = \hat{e}(g g_0^i g_1^i g_2^i, h) \land \\
\quad S = u_j^{\frac{1}{t_{i,j}+n_{i,j}}} \land \\
\quad T = u_j^{x_i}u_j^{\frac{1}{t_{i,j}+n_{i,j}}} \land \\
\quad \hat{e}(V_j, h_j) = \hat{e}(w_{i,j}, Y_jh_j^{c_i}) \land \\
\phantom{S = u_j^{\frac{1}{t_{i,j}+n_{i,j}}} \land \\
\phantom{T = u_j^{x_i}u_j^{\frac{1}{t_{i,j}+n_{i,j}}} \land \\
\phantom{\hat{e}(V_j, h_j) = \hat{e}(w_{i,j}, Y_jh_j^{c_i}) \land \\
\phantom{0 \leq n_{i,j} < k_j} \\
\end{array}
\right\}
\end{align*}
\]

\(AP_j\) verifies \(\text{SPK}_4\) is correct. If yes, \(AP_j\) accepts and adds \((\text{SPK}_4, S, T, R, m)\) to its authentication log.

- \(U_i\) increases its counter, \(n_{i,j}\), by one.

PublicTracing. For two entries \((\text{SPK}, S, T, R)\) and \((\text{SPK}', S', T', R')\), if \(S \neq S'\), the underlying user of these authentications has not exceed its prescribed usage \(k_j\) or they are from different user.

If \(S = S'\), everyone can compute \(y_i := u_j^{x_i} = (\frac{R'^{t_i}}{R})^{(t_i^k-1)}\) and outputs the corresponding \(U_i\) as the cheating user by looking up for the entry indexed by \(y_i\) in the membership public key archive. If \(y_i\) does not exist, it can be concluded that \(GM\) has deleted some data from the list and this algorithm outputs \(GM\).

Security Analysis. Regarding the security of our dynamic \(k\)-TAA, we have the following theorem whose proof is shown in Section VII-B.

**Theorem 2:** Our scheme possesses D-Detectability, D-Anonymity and D-Exculpability under the \(k\)-DDHI assumptions in the random oracle model.

V. VARIANTS OF OUR CONSTRUCTION

A. Local Witness Update

We propose a variant of our scheme where the APs do not need to interact with the users via GrantAccess and RevokeAccess when the group of users are changed dynamically.

We highlight the changes. In the initialization phase, a common accumulator is initialized for all AP's by randomly selecting \(q \in_R \mathbb{Z}_p^*\) and computing \(q_i = h_j^q\) 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 \text{APSetup}, the AP only needs to publish its identity and bound \(k_i\). It also needs to maintain a list of users' membership public key of users allowed to access its application. Granting access and revoking access simply means that the AP change such a list of users.

Finally, user in the access group have to compute their own witness as follows. A user with membership public key \(e_i\) first retrieves the list of membership public key \(\{e_j\}\) of the AP's access group. If \(e_i \in \{e_j\}\), the user accumulates the set \(\{e_j\}\) into a value \(v\) by computing \(v = h_j^{\prod_{k=1}^{t_i} e_j(k+a)}\). This quantity could be locally computed by both user and AP without knowledge of \(q\). The user also computes the witness \(w\) by \(h_0^{\prod_{k=1}^{t_i} e_j(k+a)}\) such that \(w^{(q+a)} = v\).

The rest of the protocol follows the original scheme, and the same \(\text{SPK}\) (\(\text{SPK}_4\)) is used.

B. Key-Size versus Proving Effort

Motivated by [21], we can construct dynamic \(k\)-TAA with constant proving effort by requiring each AP to publish \(k\) signatures \(\text{Sig}(1), \ldots, \text{Sig}(k)\). In the authentication, instead of proving \(0 < n \leq k\) (which is the only part with complexity \(O(\log k)\)), the user proves possession of a signature on \(n\) (which can be done in \(O(1)\)). This indirectly proves that \(n\) is within the range. The trade-off is that public key size of the \(AP\) is now linear in \(k\), and user colluding with \(AP\) can be untraceable, says the malicious \(AP\) can issue \(\text{Sig}(n^*)\) for some user where \(n^* > k\). However, we can trust the AP would not issue \(\text{Sig}(n)\) for users because it is against the interest of the AP. The weakly-secure short signature from Boneh and Boyen [27] is sufficient for our purpose since
the choice of message is fixed (1 to k). That is, there
are a fixed and polynomial number of messages to be signed
and the security of weakly-secure Boneh-Boyen short signature
guarantees that under this condition the scheme is unforgeable.
The advantage is that the signature is extremely short, only
1 single group element. Our implementation employs this
approach and details can be found in Section VI-E.

VI. DETAILS OF THE ZERO-KNOWLEDGE
PROOF-OF-KNOWLEDGE PROTOCOLS

A. Details of PoK1

To conduct PoK1, the prover first computes χ1 = g₁₁, g₂₁, χ₂ = g₀₁ for some randomly generated r₁, r₂ ∈ Zₚₙ. Then
he conducts the following protocol with the verifier.

• (Commitment.) The prover randomly generates \( r₁, r₂, r₃, r₄, r₅, \ldots, rₘ, rₘ₊₁, \ldots, rₚ \), \( \rho₁, \rho₂, \rho₃, \rho₄, \rho₅ \), \( r₁, r₂, r₃, r₄, r₅, \ldots, rₚ \) ∈ Zₚₙ, computes \( z₁ = g₁₁, g₂₁, g₃₁, g₄₁, \ldots, gₚ₁ \) ∈ G₁ and \( z₂ = g₁₁, g₂₁, g₃₁, g₄₁, \ldots, gₚ₁ \) ∈ G₂. The prover sends \( (χ₁, χ₂, μ₁, μ₂) \) to the verifier.

• (Challenge.) The verifier chooses a random challenge \( c ∈ Zₚₙ \) and sends \( c \) to the prover.

• (Response.) The prover computes \( z₁ = r₁ - cr₁, z₂ = r₂ - cr₂, \ldots, zₚ = rₚ - crₚ \) and sends \( z₁, z₂, \ldots, zₚ \) to the verifier.

• (Verify.) The verifier outputs 1 if \( χ₁ = g₁₁, g₂₁, g₃₁, g₄₁, \ldots, gₚ₁ \) and 0 otherwise.

Regarding PoK1, we have the following theorem which is
straightforward and the proof is thus omitted.

**Theorem 3:** PoK1 is an interactive honest-verifier zero-knowledge proof-of-knowledge protocol with special soundness.

B. Details of PoK2

PoK2 can be done using standard proof of representation of
discrete logarithms.

• (Commitment.) U₁ randomly generates \( r₁, r₂, r₃, r₄, r₅, \ldots, rₖ \), \( r₁, r₂, r₃, r₄, r₅, \ldots, rₖ \) ∈ Zₚₙ, computes \( T₁ = g₀₁, g₁₁, g₂₁, \ldots, gₚ₁ \) ∈ G₁ and sends \( T₁ \) to GM.

• (Challenge.) GM chooses a random challenge \( c ∈ Zₚₙ \) and sends \( c \) to U₁.

• (Response.) U₁ computes \( z₁ = r₁ - cs₁, z₂ = r₂ - cs₂, \ldots, zₖ = rₖ - csₖ \) and sends \( z₁, z₂, \ldots, zₖ \) to GM.

• (Verify.) GM outputs 1 if \( χ₁ = g₁₁, g₂₁, g₃₁, g₄₁, \ldots, gₚ₁ \) and 0 otherwise.

C. Details of PoK3

PoK3 can be done using standard proof of representation of
of discrete logarithms with equality of discrete logarithms.

• (Commitment.) U₁ randomly generates \( r₁, r₂, r₃, r₄, r₅, \ldots, rₖ \), \( r₁, r₂, r₃, r₄, r₅, \ldots, rₖ \) ∈ Zₚₙ, computes \( T₁ = g₀₁, g₁₁, g₂₁, \ldots, gₚ₁ \) and \( T₂ = g₀₁, g₁₁, g₂₁, \ldots, gₚ₁ \) ∈ G₂ and sends \( (T₁, T₂) \) to GM.

• (Challenge.) GM chooses a random challenge \( c ∈ Zₚₙ \) and sends \( c \) to U₁.

• (Response.) U₁ computes \( z₁ = r₁ - cs₁, z₂ = r₂ - cs₂, \ldots, zₖ = rₖ - csₖ \) and sends \( z₁, z₂, \ldots, zₖ \) to GM.

• (Verify.) GM outputs 1 if \( χ₁ = g₁₁, g₂₁, g₃₁, g₄₁, \ldots, gₚ₁ \) and 0 otherwise.

D. Details of SPK₁

To conduct SPK₁, the prover computes \( A₁ = g₁₁, g₂₁, A₂ = g₁₁, g₂₁ \), \( A₃ = g₁₁, g₂₁ \), \( A₄ = w₁₁, g₁₁, A₅ = g₁₁, g₂₁ \) for some randomly generated \( r₁, r₂, r₃, r₄, r₅, \ldots, rₖ \), \( r₁, r₂, r₃, r₄, r₅, \ldots, rₖ \) ∈ Zₚₙ. Next, U₁ sends \( A₁, A₂, A₃, A₄, A₅ \) to \( A₁ \) and computes the following two
SPK’s.

\[
\begin{pmatrix}
    \left( e₁, s₁, t₁, t₄, m, s₄, s₅, s₆, s₇, \ldots, sₙ \right) \\
    \left( e₂, e₃, e₄, e₅, e₆, \ldots, eₙ \right) \\
    \left( e₅, e₆, e₇, e₈, e₉, \ldots, eₙ \right) \\
    \left( \frac{\epsilon(g₁, h)}{\epsilon(g₂, h)} \right) \\
    \left( \frac{\epsilon(g₁, h)}{\epsilon(g₂, h)} \right) \\
    \left( \frac{\epsilon(g₁, h)}{\epsilon(g₂, h)} \right) \\
    \left( \frac{\epsilon(g₁, h)}{\epsilon(g₂, h)} \right) \\
    \left( \frac{\epsilon(g₁, h)}{\epsilon(g₂, h)} \right) \\
\end{pmatrix}
\]

SPK₁₄ \( \left( n₁, n₂, n₃, n₄, n₅, n₆, n₇, n₈, n₉ \right) \) : \( n₅ = g₁₁, g₂₁ \wedge 0 ≤ n₁ ≤ k \) \( m \) otherwise.

\[
\begin{pmatrix}
    \left( u₁, u₂, u₃, u₄, u₅, u₆, \ldots, uₙ \right) \\
    \left( v₁, v₂, v₃, v₄, v₅, v₆, \ldots, vₙ \right) \\
\end{pmatrix}
\]

SPK₁₄ \( \left( n₁, n₂, n₃, n₄, n₅, n₆, n₇, n₈, n₉ \right) \) : \( n₅ = g₁₁, g₂₁ \wedge 0 ≤ n₁ ≤ k \) \( m \) otherwise.

Instantiation of SPK₁₄ is a simple range proof and is discussed in Section VI-E. Below we show how to instantiate
SPK₁₄.

• (Commitment.) U₁ randomly generates \( r₁, r₂, r₃, r₄, r₅, \ldots, rₖ, r₁, r₂, r₃, r₄, r₅, \ldots, rₖ \) ∈ Zₚₙ, computes \( T₁ = g₀₁, g₁₁, g₂₁, \ldots, gₚ₁ \) and \( T₂ = g₀₁, g₁₁, g₂₁, \ldots, gₚ₁ \) ∈ G₂ and sends \( (T₁, T₂) \) to GM.

• (Challenge.) GM chooses a random challenge \( c ∈ Zₚₙ \) and sends \( c \) to U₁.

• (Response.) U₁ computes \( z₁ = r₁ - cs₁, z₂ = r₂ - cs₂, \ldots, zₖ = rₖ - csₖ \) and sends \( z₁, z₂, \ldots, zₖ \) to GM.

• (Verify.) GM outputs 1 if \( χ₁ = g₁₁, g₂₁, g₃₁, g₄₁, \ldots, gₚ₁ \) and 0 otherwise.
The (Challenge.) $U_i$ computes challenge $c = H(m||Ap_j||\Sigma)$ using a cryptographic hash function $H$.

The (Response.) $U_i$ computes, in $\mathbb{Z}_p$, $z_{c_i} = r_{c_i} - c e_i$, $z_{s_i} = r_{s_i} - c s_i$, $z_{t_i} = r_{t_i} - c t_i$, $z_{x_i} = r_{x_i} - c x_i$, $z_{n_{i,j}} = r_{n_{i,j}} - c n_{i,j}$, $z_{p_1} = r_{p_1} - c p_1$, $z_{p_2} = r_{p_2} - c p_2$, $z_{p_3} = r_{p_3} - c p_3$, $z_{p_4} = r_{p_4} - c p_4$, $z_{p_5} = r_{p_5} - c p_5$, $z_{\beta_1} = r_{\beta_1} - c p_3 e_i$, $z_{\beta_2} = r_{\beta_2} - c p_4 p_1$, $z_{\beta_3} = r_{\beta_3} - c p_4 e_i$, $z_{\beta_4} = r_{\beta_4} - c p_4 p_1$, $z_{\beta_5} = r_{\beta_5} - c x_i n_{i,j}$, $z_{\beta_6} = r_{\beta_6} - c p_3 n_{i,j}$, $U_i$ sets $j = (z_{c_i}, z_{s_i}, z_{t_i}, z_{x_i}, z_{n_{i,j}}, z_{p_1}, z_{p_2}, z_{p_3}, z_{p_4}, z_{p_5}, z_{\beta_1}, z_{\beta_2}, z_{\beta_3}, z_{\beta_4}, z_{\beta_5}, z_{\beta_6}, (z_{\beta_6}, z_{\beta_6}))$.

The (Output.) $U_i$ outputs $(c, j)$ as $SPK_{4\lambda}$.

The (Verify.) Upon receiving $SPK_{4\lambda} := (c, j)$, $AP_j$ computes the following.

$T_1' = z_{c_1} z_{s_1} z_{t_1}$

$T_2' = z_{c_2} z_{s_2} z_{t_2}$

$T_3' = z_{c_3} z_{s_3} z_{t_3}$

$T_4' = z_{c_4} z_{s_4} z_{t_4}$

$T_5' = z_{c_5} z_{s_5} z_{t_5}$

$T_6' = z_{c_6} z_{s_6} z_{t_6}$

$T_7' = z_{c_7} z_{s_7} z_{t_7}$

$T_8' = z_{c_8} z_{s_8} z_{t_8}$

$T_9' = z_{c_9} z_{s_9} z_{t_9}$

$T_{10}' = z_{c_{10}} z_{s_{10}} z_{t_{10}}$

$T_{11}' = z_{c_{11}} z_{s_{11}} z_{t_{11}}$

Set $T' = (T_1', \ldots, T_{11}')$. Output 1 if $c \not\in H(m||\Sigma)$ and 0 otherwise.

E. Range Proof of $SPK_4$

Exact Range Proof. Secure and efficient exact proof of range is possible in groups of unknown order under factorization assumption [35]. However, observe that the range proof required in our authentication protocol is always of the form of $0 \leq n < k$. If we set $k = 2^c$, a simple range proof of order $O(k)$ can be constructed easily.

For the ease of presentation, we let $c_n = g_0^{n} g_1^r$ be a commitment of $n$ and the goal is the following zero-knowledge proof-of-knowledge.

$$\text{PoK}_{\text{Range}}\{ (n, r) : c_n = g_0^n g_1^r \land 0 \leq n < 2^c \}$$

We show the non-interactive version here for two reasons. Firstly, it is more space-efficient. Secondly, it is compatible with our protocol in which the non-interactive version (signature of knowledge) is used.

The prover do the following. Let $n[\ell]$ be the $\ell$-th bit of $n$ such that $\ell$ starts from 0. For $\ell = 0$ to $\ell = 1 - 1$, compute $\mathcal{C}_\ell = g_1^n g_1^r$ such that $r_\ell \in \mathbb{Z}_p$.

Conduct the following two $SPK$'s.

$SPK_{5A} \{ (n, r, \alpha) : c_n = g_0^n g_1^r \land \prod_{\ell=0}^{\alpha-1} \mathcal{E}^{\ell}_{c_\ell} = g_0^n g_1^r \} (m)$

$SPK_{5B} \{ (r_0, \ldots, r_{\ell-1}) : \prod_{\ell=0}^{\ell-1} \mathcal{C}_\ell = g_0^{r_\ell} \lor \mathcal{E}_{c_\ell} = g_0^{r_\ell} \} (m)$

We describe $SPK_{5A}$ first, which can be done using standard proof of representation of discrete logarithms together with equality of discrete logarithms.

(Commitment.) The prover generates $\rho_n, \rho_1, \rho_2 \in \mathbb{Z}_p$ and computes $T_1 = g_0^n g_1^r, T_2 = g_1^n g_2^r$. Set $X = (T_1, T_2)$.

(Challenge.) The prover computes challenge $c = H(m||\Sigma)$ using a cryptographic hash function $H$.

(Response.) The prover computes $z_{n} = \rho_n - cn, z_{r} = \rho_r - cr$ and $z_{\alpha} = \rho_\alpha - c \sum_{j=0}^{\alpha-1} 2^j r_j \in \mathbb{Z}_p$ and sets $j = (z_n, z_r, z_\alpha)$.

(Output.) The prover outputs $(c, j)$ as $SPK_{5A}$.

(Verify.) The verifier computes $T_4' = c_n^x g_0^y g_1^{-z} \land T_2' = (\prod_{\ell=0}^{\ell-1} \mathcal{E}^{\ell}_{c_\ell})^c g_0^y g_1^{-z}$. The verifier sets $T' = (T_1', T_2')$ and outputs 1 if $c \not\in H(m||\Sigma)$ and 0 otherwise.

$SPK_{5B}$ is constructed using techniques of conjunction of disjunction of discrete logarithms.

(Commitment.) For $\ell = 0$ to $\ell - 1$, randomly picks $c_{\ell,1-n}[\ell], z_{\ell,1-1}[\ell], \rho_{\ell,1-1}[\ell] \in \mathbb{Z}_p$ and computes $T_{x,1-n}[\ell] = (\mathcal{E}_{c_{\ell,1-n}[\ell]} v_\ell g_0^{z_{\ell,1-n}[\ell]})^c g_0^y = g_2^{z_{\ell,1-n}[\ell]}$. For $\ell = 0$ to $\ell - 1$, set $T_{x} = (T_{x,0}, T_{x,1})$.

(Challenge.) For $\ell = 0$ to $\ell - 1$, the prover computes challenge $c_{\ell}[\ell] = H(m||\Sigma)$ using a cryptographic hash function $H$.

(Response.) For $\ell = 0$ to $\ell - 1$, the prover computes $c_{\ell,1-n}[\ell] = H(m||\Sigma)$ using a cryptographic hash function $H$.

(Verify.) For $\ell = 0$ to $\ell - 1$, the prover computes $T_{x,1}[\ell] = g_0^{z_{\ell,1}} g_1^{-z_{\ell,1} z_{\ell,1}} T_{x,1} T_{x,1}$. Set $T_{x} = (T_{x,0}, T_{x,1})$. Output 1 if $c_{\ell,0} + c_{\ell,1} = H(m||\Sigma)$ for all $\ell = 0$ to $\ell - 1$.

The two $SPK$'s consists of $(4k + 4)$ elements in $\mathbb{Z}_p$, and $\kappa$ $\mathcal{C}_\ell$'s in $\mathbb{G}_1$. In our protocol, total size of the range proof is $(4 + 4k) * 170 + k * 171$ bits.

Signature-Based Proof. The idea has been described in Section V. While not exactly a range proof, it suffices for all our purpose. Let $i \in \mathbb{Z}_p$ be a secret value of an AP, and $I = h^i$ be a public value. Specifically, a weakly-secure Boneh-Boyen short signature[27] $\text{Sig}(n)$ on $n$ is $\text{Sig}(n) = g_0^r n$. Note that each $(\text{Sig}(n), n)$ pair satisfies $c(\text{Sig}(n), h^i) = c(g, h)$. The AP also publishes $\text{Sig}(1), \ldots, \text{Sig}(k)$ as public parameter. Then, $SPK_{1\lambda B}$ can be instantiated as $SPK_{5C}$.
Let \( A \) be a forger which could forge a BBS+ signature under adaptively chosen queries. 

\[
\begin{align*}
\mathbb{A}_5 &= \mathbb{A}_6 \oplus \mathbb{A}_7 \\
\mathbb{A}_6 &= \mathbb{A}_6' \oplus \mathbb{A}_7 \\
\mathbb{A}_7 &= \mathbb{A}_7' \\
\mathbb{A}_7' &= \mathbb{A}_6' \oplus \mathbb{A}_7 \\
\mathbb{A}_6' &= \mathbb{A}_6. 
\end{align*}
\]

For the \( i \)-th query, denote the multi-block message to be signed as \((m_1, \ldots, m_{t+1})\) such that \( t \leq L \). For each query, \( S \) computes \( M_i = \sum_{j=1}^{t+1} m_{i+1} \). 

Out of these \( q \) queries, \( S \) randomly chooses one, called query \( \ast \) which shall be handled differently. For the other \( q-1 \) queries, \( S \) randomly picks \( s_i \in \mathbb{Z}_p \), computes \( s_i = s_i + M_i \) and \( \varsigma_i = (g_9 s_i)^{\frac{1}{t+1}}. \) Note that 

\[
\varsigma_i = g_9 s_i^{\frac{1}{t+1}} = (g_{\ast}^{\frac{t}{t+1}})^{s_i} = \left(g_{\ast}^{\frac{t}{t+1}}\right)^{s_i}, \]

and is computable by \( S \) even though \( \mu \) is unknown since \((e_1 + \mu) \) divides \( f(\mu) \) and \((e_1 + \mu) \) is a degree \( q \) polynomial. \( S \) returns \((\varsigma_i, e_1, s_i)\) as the answer to the \( i \)-th signature query. 

For query \( \ast \), \( S \) computes \( s^* = a^* - M_s \) and returns \((g^{k*}, e^*, s^*)\) as the answer. Note that 

\[
\begin{align*}
(g^{k*})^{e^*+\mu} &= g_9^{s^*} \\
&= g_9^{s^*} g_{\ast}^{M_s} \\
&= g_9^{s^*} g_{\ast}^{1} \cdots g_{\ast}^{m_{t+1}},
\end{align*}
\]

and thus \((g^{k*}, e^*, s^*)\) is a valid signature. 

Finally, \( \mathcal{F} \) outputs \( q+1 \) message-signature pairs. At least one of them is different from the \( q \) message-signature pairs obtained during the signing query phase. 

Let this signature-message pair be \((\varsigma', e', s')\), \((m_1', \ldots, m_{t+1}', \mathbf{m}_p')\). Denote \( S' = S + \sum_{i=1}^{t+1} m_{i+1} \). 

There are three possibilities. 

- Case I \([e' \notin \{e_1, e^*\}]:\) 
  
  \[
  \begin{align*}
  \varsigma'^{e'+\mu} &= g_9^{s'} \\
  \varsigma^e \varsigma'^{e'+\mu} &= g_{\ast}^{s'+s'} \left( g_{\ast}^{(e_1+\mu)k^*} \right)^{s'}. 
  \end{align*}
  \]

  Since \( e' \notin \{e_1, e^*\} \), \((e^* + \mu) \) does not divide \( f(\mu) \) (respectively for \( f(\mu)(e_1 + \mu) \)). \( S \) computes a \( q-2 \) (respectively \((q-1)\)) degree polynomial \( Q(\mu) \) (respectively \( Q^*(\mu) \)) and constant \( Q \) (respectively \( Q^* \)) such that \( f(\mu) = Q(\mu)(\mathbf{m}_p') \) and \((e^* + \mu) \) is degree \( q \) polynomial. \( S \) returns \((\varsigma', e^*, \mathbf{m}_p')\). Thus, 

  \[
  \varsigma' = \left(g_{\ast}^{s' - s} \right)^Q(\mathbf{m}_p') = g_{\ast}^{s' - s} \left( g_{\ast}^{(e_1+\mu)k^*} \right)^{Q(\mu)} + \frac{Q^*}{\mathbf{m}_p'}
  \]

Thus, \((g_{\ast}^{s' - s}, e^* \mathbf{m}_p')\) is the solution to \( q \)-SDH problem. 

- Case II: \([e' = e_1\text{ and } e' = e^*]:\) This happens with negligible probability unless \( \mathcal{F} \) solves the relative discrete logarithm amongst two of the \( h_i \)'s.
Case III: \( \{e' \in \{e_i, e^*\} \text{ and } c' \neq c_i\} \): With probability, 1/q, \( e' = e^* \).

\[
\begin{align*}
\varsigma c' + \mu & = g_{90}^{c'} \\
\varsigma c^* + \mu & = g_{90}^{(c^* + k^*)g^* + a^* - S'} \\
\varsigma' & = g^{k^*g^*} g^{(c^* + \mu)g^*} \\
\varsigma' & = g^{k^*g^*} g^{(c^* + \mu)g^*} \\
\end{align*}
\]

Since \( (e^* + \mu) \) does not divide \( f(\mu) \), \( S \) returns \( q - 2 \) degree polynomial \( Q(\mu) \) and constant \( Q \) such that \( f(\mu) = (e^* + \mu)Q(\mu) + Q \). Thus, \( \varsigma' = g^{k^*g^*} g^{(c^* + \mu)g^*} \).

Thus, \( (g^{k^*g^*}, e') \) is the solution to the \( q \)-SDH problem. If the success probability of \( F \) is \( \epsilon \), then in the worst case, success probability of \( S \) is \( \epsilon/q \).

B. Analysis of Our \( k \)-TAA

D-Detectability. Let \( A \) be an adversary who executes \( f \) Join protocol with \( S \) acting as the GM. Let \( \mathcal{E}_{\text{PoK}_2} \), \( \mathcal{E}_{\text{PoK}_3} \) be extractors of the PoK2 and PoK3 respectively. For each join request, \( S \) acts exactly as an honest GM would, except during step 3, where \( S \) runs the extractor \( \mathcal{E}_{\text{PoK}_3} \) to extract the values \( \{s, t, x\} \). From the value \( s, \) \( S \) compute the tuple \( u_i = (S_{s_i}, \ldots, S_{s_k}, s, t, x) \) such that \( S = u_i^{1/(J_{AP} + s + 1)} \) for \( i = 1, \ldots, k \). Let \( A_j = \{S_{s_j} \mid 1 \leq i \leq f, 1 \leq j \leq k\} \) after \( f \) executions of the Join protocol.

Since the protocol is done non-interactively, \( S \) is given control over the random oracle in addition to the black-box access to \( A \). Moreover, such extraction requires rewind simulation and thus the Join protocol cannot be executed concurrently.

For each \( AP \), let \( u = H(A \_P_j) = u_i^{r_j} \) for some randomly chosen \( r_j \). Due to the soundness of the underlying SPK, \( AP \), together with the set \( \{r_j\} \) contains all valid one-time pass \( S \) that \( A \) can produce, except with negligible probability. For \( A \) to break D-Detectability, one of the following two happens. (1) \( A \) convinces an honest AP to accept a one-time pass \( S \) for which it cannot generate an honest proof of validity with some non-negligible probability. (2) \( A \) uses duplicated \( S \) but public tracing does not output the identity of the user within the \( f \) Join protocol.

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

1. \( \varsigma c + \mu = g_{90}^{c} g_{91}^{d} g_{92}^{d} \)
2. \( u_j^{c + \gamma} = V_j \)
3. \( S = u_j^{1/(n_j + s + 1)} \)
4. \( T = u_j^{x_t/R(n_j + t + 1)} \)
5. \( 0 \leq n_j \leq k \)

Item 1 happens with negligible probability under the \( q \)-SDH assumption, as violating 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[33] (also relied on \( q \)-SDH assumption). Item 3,4 happen with negligible probability under the DL assumption (which will be subsumed by the \( k \)-DDHI assumption in the theorem). Item 5 happens with negligible probability if the proof-of-knowledge of committed number lies in an exact interval exists. For instance, [35] proposes such a zero-knowledge proof under the factorization assumption. If technique of [35] is used, in the setup procedure, a group of unknown order have to be chosen during set up. On the other hand, if we set \( k = 2^e \) then we can use the protocol outlined in Section VI-E. We omit the details of exact range proof in the protocol for simplicity. To conclude, the total success probability of \( A \) in case 1 is negligible.

Consider case (2), it has already been proved in case (1) that \( A \) cannot have an honest AP to accept an invalid \( S \) with non-negligible probability. Since \( A \) must use valid one-time pass \( S \), to authenticated more than \( k \) times for the same \( AP \), it must uses duplicated \( S \). Let \( (S, SPK_1) \) and \( (S, SPK_2) \) be the transcript of authentication for which an honest \( AP \) (\( AP_j \)) accept a duplicated one-time pass \( S \) such that \( S^{1/\alpha_j} \in A \). Since \( A \) is honest, \( R_1 \neq R_2 \) with high probability. We are to show that \( T_1 = u_0^{x_t} u_j^{R(n_j + s + 1)} \) and \( T_2 = u_0^{x_t} u_j^{R(n_j + s + 1)} \) 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 tracing tags to accompany the duplicated one-time pass \( S \) in these two authentications. To deviate from these \( S \) and \( T \), \( A \) 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 overwhelming probability.

During the course of the running of \( A \), it is allowed to query several oracles. We outline how \( S \) simulates these oracles. The join oracle is simulated by invoking the signing oracle of the BBS+ signature (\( S \) needs to extract the multi-block message from the commitment and this makes the join procedure non-concurrent). Authentication oracle is simulated by randomly generating \( s, t, x, n_j \), backpatching the random oracle and simulating the signature of knowledge of the authentication procedure.

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

For each authentication procedure, \( S \) simulates as follow.

- \( S \) is given \( seed \) and compute \( R = H(seed) \).
- \( S \) random chooses \( s, t, x \) and a random \( n_j \in R \{1, \ldots, k\} \) and compute \( S = u_j^{1/((n_j + s + 1))} \). \( T = u_0^{x_t} u_j^{R(n_j + s + 1)} \).
- \( S \) simulates a proof of \( c, w, e, s, t, x, n_j \) such that \( \varsigma c + \mu = g_{90}^{c} g_{91}^{d} g_{92}^{d} \). \( 2^w + \gamma = V_j \), \( S = u_j^{1/(n_j + s + 1)} \).
- \( T = u_0^{x_t} u_j^{R(n_j + s + 1)} \), \( n_j \leq k \).
- 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 $S$ is computationally indistinguishable from the output of a real user. The idea is that during the Join protocol $A$ 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 $S$ is indistinguishable from those chosen by real users. Due to the security of the PRF [31], $S$ and $T$ are indistinguishable from randomly elements under the $k$-DDHI assumption. Thus, $A$ 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. The probability is negligible under $k$-DDHI assumption.

**D-Exculpability.** D-Exculpability for the GM is quite straightforward to show. Suppose the GM is honest but PublicTracing on input two authentication transcripts $(S, R_1, SPK_1)$, $(S, R_2, SPK_2)$ outputs 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 authentication protocol, which happen with negligible probability. Proof of D-Exculpability for honest user is also quite straightforward. 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.

**VIII. Complexity Analysis**

We analyze the efficiency of our system in terms of both time and communication complexities. Both complexities are not dependent on the number of users and APs but on the number of allowable accesses $k$ (in different manner, according to how range proof in authentication is implemented). In the system with constant size public key, both time and communication complexities are logarithmic in $k \log (k))$. For the variant described in Section V-B, the complexities are constant ($O(1)$); however, the space complexity of an AP’s public key is $O(k)$ (and the time complexity is also $O(k)$).

In the analysis here, we focus on the variant of our system described in Section V-B since it is more efficient in terms of communication cost. In particular, an authentication protocol consists of a one-time pass $S$, a tracing tag $T$, a random challenge $m$ and a proof-of-correctness $SPK_1$. Assume $m$ is also an element in $Z_p^*$, the total communication cost consists of 2 elements in $G_p$, 7 elements in $G_1$ and 24 elements in $Z_p$.

For time complexity, we count the number of multi-exponentiations (multi-EXP’s) in various groups and the number of pairings. Operations such as hashing, element negation or addition are neglected as they take insignificant time compared with multi-EXP or pairing. A multi-EXP computes the product of exponentiations faster than performing the exponentiations separately. For our usage, we assume one multi-EXP operation multiplies up to 3 exponentiations. In fact, our experimental result confirms that a multi-EXP with 3 different bases is almost as fast as a single exponentiation.

For computation, user is required to compute 16 multi-EXP’s in $G_1$, 2 multi-EXP’s in $G_p$, 4 multi-EXP’s in $G_T$ and 3 pairings. We would like to stress that a large part of the user’s work can be pre-computed. In fact, out of the above operations, only 2 multi-EXP in $G_p$ should be computed online. We shall see in the experiment result that online computation for user is very efficient.

On the other hand, AP is required to compute 9 multi-EXP’s in $G_1$, 2 multi-EXP’s in $G_2$, 2 multi-EXP’s in $G_p$, 7 multi-EXP’s in $G_T$ and 3 pairings. However, they cannot be pre-computed, since they are dependent on the choice of the user.

**IX. Experimental Results**

The test machine is a Dell GX620 with an Intel Pentium-4 3.0 GHz CPU and 2GB RAM running Windows XP Professional SP2 as the host. We used Sun xVM VirtualBox 2.0.0 to emulate a guest machine of 1GB RAM running Ubuntu 7.04. Our implementation is written in $C$ and relies on the Pairing-Based Cryptography (PBC) library (version 0.4.18) for the underlying elliptic-curve group and pairing operations.

We chose the type D pairing bundled with the PBC library. Specifically, $p$ is of size 181-bit. An element in $Z_p$, $G_1$ can be represented by 24 and 25 bytes respectively. As for $G_p$, we have two choices. We could take $G_p$ as $G_1$ if we assume DDH problem is difficult in $G_1$. Such a assumption is formally called the external Diffie-Hellman (XDH) assumption, which implies there is no efficient mapping from $G_1$ to $G_2$. On the other hand, we could take $G_p$ as $G_T$ and in this case no extra assumption is needed. We implemented both solutions. In both cases, experimental results show that 1 pairing operation takes roughly 6 ms on our test machine. A single base exponentiation (respectively 3-base exponentiation) in $G_1$ takes 2.32ms (resp. 2.36ms).

When we take $G_p$ to be $G_1$, it takes 89 ms and 4 ms for the user to complete the offline and online computation respectively. It takes 72 ms for the application provider to complete the protocol. Timing figures are similar when we do not make the XDH assumption. For instance, it takes 93 ms and 4 ms for the user to complete the offline and online computation respectively. For the application provider, it takes 70 ms to complete the protocol.

The main difference between the two implementations is the bandwidth requirement. With XDH assumption made, the total bandwidth required is 777 bytes; while it becomes 1015 bytes without such an assumption. The reason is that 144 bytes is required to represent an element in $G_T$.

The public key size of the APs is the only quantity that is linear in $k$ in our system, which involves $k$ weakly-secure Boneh-Boyen short signature, 25 bytes each. In addition, it takes roughly 3 ms for the AP to generate one such signature. Thus, for moderate $k$, the public key size of the AP and the parameter generation time, while linear in $k$, are more acceptable than one may expect.

**X. Conclusion and Research Directions**

It has been suggested that anonymous authentication systems like $k$-TAA systems [3], [2], [21] are suitable cryptographic primitives for secure applications with privacy concern like e-voting [36]. In this paper, we constructed a constant-size dynamic $k$-TAA scheme, which solved the open problem left
in [37]. We also provided a proof-of-concept implementation and analyzed its efficiency. Finally, BBS+ signature derived as a building block of our system could be useful for other cryptographic systems.

Our dynamic $k$-TAA system requires constant proving effort but linear-size AP public keys. One technical challenge remains to be solved is to design a dynamic $k$-TAA scheme with constant complexities for all parameters.

The dynamic feature of our scheme relies on the use of dynamic accumulator, but in order to prove useful relationship between the accumulated value, the signature, and the pseudo-random function, etc., we take an accumulator which requires either a secret or a long public parameter (increases linearly with the size of the membership) for the dynamic update. A real dynamic accumulator which we can prove things about, without the need of a secret or a long parameter, will be a useful cryptographic construct.

Our construction is built from specific building blocks. It will be nice to have a generic construction of dynamic $k$-TAA, perhaps borrowing the generic design of traceable signatures in [38].

Similar to the discussions in [26] and [25], authentication schemes can be classified according to their revocability level (unrevocable, revocable-iff-linked, revocable) and anonymity level (fully anonymous, escrowed-linkable [25], publicly linkable). It is also interesting is to identify practical application for authentication scheme with a “nice” level of privacy.

REFERENCES


