

University of Wollongong

Research Online

Faculty of Engineering and Information
Sciences - Papers: Part A

Faculty of Engineering and Information
Sciences

1-1-2013

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

Follow this and additional works at: <https://ro.uow.edu.au/eispapers>



Part of the [Engineering Commons](#), and the [Science and Technology Studies Commons](#)

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

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

Au, M. Ho., Susilo, W., Mu, Y. & Chow, S. S. M. (2013). Constant-size dynamic k-times anonymous authentication. IEEE Systems Journal, 7 (2), 249-261.

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 TSF04 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.

M. H. Au, W. Susilo and Y. Mu are with Centre for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, NSW, Australia.

S. Chow is with Centre for Applied Cryptographic Research / Department of Combinatorics and Optimization, University of Waterloo, Ontario, Canada.

A preliminary version of this work appeared in SCN '06 [1]. This work is partially supported by ARC Linkage Project Grant LP0667899.

Received Date: Oct 02, 2011

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 n_1 times to one AP and n_2 times to another AP, provided that n_1 and n_2 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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear pairing if

- \mathbb{G}_1 and \mathbb{G}_2 are cyclic multiplicative groups of prime order p .
- g, h 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 $\psi(h) = g$.
- (Bilinear) $\forall x \in \mathbb{G}_1, y \in \mathbb{G}_2$ 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 $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T has unique binary representation.

\mathbb{G}_1 and \mathbb{G}_2 can be the same group 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 ψ 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 $\mathbb{G} = \langle g \rangle$ 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 -Strong Diffie-Hellman): The q -Strong Diffie-Hellman (q -SDH) problem in a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with trace map ψ is defined as follow: On input a $(q+2)$ -tuple $(g, h, h^\gamma, h^{\gamma^2}, \dots, h^{\gamma^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$ such that there exists a trace map ψ from \mathbb{G}_2 to \mathbb{G}_1 with $g = \psi(h)$, output a pair (B, e) such that $B^{(\gamma+e)} = g$ where $e \in \mathbb{Z}_p^*$. We say that the q -SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no PPT algorithm has non-negligible advantage 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 [27] even when $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair with trace map.

Definition 3 (q -Decisional Diffie-Hellman Inversion): The q -Decisional Diffie-Hellman Inversion problem (q -DDHI) in prime order group $\mathbb{G} = \langle g \rangle$ is defined as follow: On input a $(q+2)$ -tuple $g, g^x, g^{x^2}, \dots, g^{x^q}, g^c \in \mathbb{G}^{q+2}$, output 1 if $c = 1/x$ and 0 otherwise. We say that the q -DDHI assumption holds in \mathbb{G} if no PPT algorithm has non-negligible advantage over random guessing in solving the q -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 -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 $Sig(m)$ 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, \dots, m_L) from the signer while the signer only gets a commitment of a multi-block message (m_1, \dots, 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 message 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 , $\text{PRF}_{g,s}(x)$ is defined as $\text{PRF}_{g,s}(x) : x \mapsto g^{\frac{1}{s+x+1}}$. 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 -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 -SDH assumption, see [33] for details.

III. FRAMEWORK

A. Syntax

We briefly review the model of dynamic k -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^λ , 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_i to join the group and obtain a membership public/secret key pair (mpk_i, msk_i) from GM. GM also adds U_i '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 generate the public and private key (apk_j, ask_j) of AP_j .
- **GrantAccess.** Each AP_j manages its own access group \mathcal{L}_j which is initially empty. This procedure allows AP_j to add a user U_i to its access group \mathcal{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 \mathcal{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 \mathcal{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 λ be the security parameter. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair with computable isomorphism ψ as discussed in Section II-A such that $\mathbb{G}_1 = \langle g \rangle$, $\mathbb{G}_2 = \langle h \rangle$ and $|g| = |h| = p$ for some prime p of λ bits. Assume $\mathbb{G}_p = \langle u \rangle$ is a cyclic group of order p such that the DDH assumption holds. Let $g_0, g_1, g_2, \dots, g_L, g_L$ be additional random elements in \mathbb{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 $\mu \in_R \mathbb{Z}_p^*$ and compute $Z = h^\mu$. The secret key is μ and the public key is Z .

Signing Multi-block Messages. On input $(m_1, \dots, m_L) \in \mathbb{Z}_p^L$, choose $e, s \in_R \mathbb{Z}_p^*$. Compute $\varsigma = [gg_0^s g_1^{m_1} \dots g_L^{m_L}]^{\frac{1}{e+\mu}}$. Signature on (m_1, \dots, m_L) is $(\varsigma, e, s) \in (\mathbb{G}_1 \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*)$.

Signature Verification. To verify a signature (ς, e, s) on a multi-block message (m_1, \dots, m_L) , check if $\hat{e}(\varsigma, Zh^e) = \hat{e}(gg_0^s g_1^{m_1} \dots g_L^{m_L}, h)$.

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 $s' \in_R \mathbb{Z}_p^*$ and computes $C_m = g_0^{s'} g_1^{m_1} \dots g_L^{m_L}$. He sends C_m to the signer, along with the following proof

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

After verifying PoK_0 , the signer chooses $s'', e \in_R \mathbb{Z}_p^*$, computes $\varsigma = [gg_0^{s''} C_m]^{\frac{1}{e+\mu}}$ and sends (ς, e, s'') back to the user. The user computes $s = s' + s''$. The signature on the multi-block messages is (ς, e, s) . For any block of messages (m_1, \dots, m_L) , there exists an s' such that $C_m = g_0^{s'} g_1^{m_1} \dots g_L^{m_L}$ 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 $\mathfrak{C}_m = g_0^r g_1^{m_1} \dots g_L^{m_L}$ be a commitment of a multi-block message (m_1, \dots, m_L) with randomness r , a user who is in possession of a signature (ς, e, s) can conduct the following zero-knowledge proof-of-knowledge protocol with any verifier.

$$\text{PoK}_1 \left\{ \begin{array}{l} (\varsigma, e, s, m_1, \dots, m_L, r) : \\ \hat{e}(\varsigma, Zh^e) = \hat{e}(gg_0^s g_1^{m_1} \dots g_L^{m_L}, h) \quad \wedge \\ \mathfrak{C}_m = g_0^r g_1^{m_1} \dots g_L^{m_L} \end{array} \right\}$$

Instantiation of PoK_1 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 $x, t \in_R \mathbb{Z}_p^*$. A membership public key is $(y = u^x, e)$ while the membership secret key is (x, ς, s, t) such that (ς, e, s) is a BBS+ signature from the GM on “message” (x, t) . We stress that x, t is unknown to the GM due to signature generation protocol of BBS+. The membership public key (y, e) is placed on the membership public key archive.

GrantAccess and RevokeAccess. To grant access to a user with membership public key (y, e) , the AP adds e into his accumulator N-Acc and gives the witness w to the user. To revoke access of the user, AP removes e from the accumulator. For every GrantAccess, RevokeAccess, all users included in the access group of that AP have to update their own witness using the update algorithm from N-Acc.

In the variant of our scheme (to be discussed in the next section), AP only publishes the access group which is a list of e ’s and let users work with the accumulator themselves. Consequently, the interactive GrantAccess or RevokeAccess protocols are not required. The drawback is that users have to perform $O(|\mathcal{L}|)$ operations, where $|\mathcal{L}|$ is the size of the list, to compute his own witness.

Authentication. The idea is to have the user to demonstrate to AP_j that he is in possession of a BBS+ signature (ς, e, s) from the GM on values (t, x) , and that e is inside the accumulator of AP_j . To restrict the user from authenticating himself for more than k times, we borrow the idea of double spender revocation in e-cash system. Specifically, let $u_j = H(\text{AP}_j)$ be some random element of \mathbb{G}_p for some cryptographic hash function H and n_j be the number of times the user has authenticated with AP_j . The user computes a value $S = \text{PRF}_{u_j, s}(n_j)$. S is called a “one-time pass” corresponding to that authentication. In our construction, $S = u_j^{1/(s+n_j+1)}$. The user is required to prove that S is correctly formed by demonstrating he is in possession of a BBS+ signature component (ς, e, s) and $0 \leq n_j < k$. For any AP, a user can only generate k valid one-time pass.

The user is also required to compute $T = u^x \text{PRF}_{u_j, t}(n_j)^R$, where R is a random challenge issued by AP_j . T is called

a tracing tag. Specifically, $T = u^x (u_j^{1/(t+n_j+1)})^R$ and the user is required to prove that T is correctly formed. Each authentication is accompanied by a pair of one-time pass and tracing tag.

In case a user attempts to authenticate more than k times, he will have to use repeated one-time pass and will thus be detected. Since the R values are different, the two tracing tags accompanying those authentication attempts shall be different. With different tracing tags, identity of the user who attempt to authenticate more than k times can be computed.

D. Details of Our Construction

GMSetup. The GM randomly selects $\mu \in_R \mathbb{Z}_p^*$ and computes $Z = h^\mu$. 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^λ . In addition, AP_j selects $h_j \in_R \mathbb{G}_2$, $\gamma_j \in_R \mathbb{Z}_p^*$ and computes $u_j = H(\text{AP}_j) \in \mathbb{G}_p$ for some hash function H and $Y_j = h_j^{\gamma_j} \in \mathbb{G}_2$. The public key and the secret key of AP_j are $(\text{AP}_j, k_j, h_j, u_j, Y_j)$ and γ_j respectively. AP_j maintains an authentication log and an access group list \mathcal{L}_j of 3-tuple. The list is initialized to $(\perp, \perp, \psi(h_j))$.

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

- 1) U_i randomly selects $s'_i, t_i, x_i \in_R \mathbb{Z}_p^*$ and sends $C' = g_0^{s'_i} g_1^{t_i} g_2^{x_i}$ to GM, along with proof

$$\text{PoK}_2\{(s'_i, t_i, x_i) : C' = g_0^{s'_i} g_1^{t_i} g_2^{x_i}\}$$

- 2) GM verifies the proof and randomly selects $s''_i \in_R \mathbb{Z}_p^*$. He sends s''_i to the user.
- 3) User computes and sends $s_i = s'_i + s''_i \bmod p$, $y_i = u^{x_i}$ to GM (recall that u is the generator of \mathbb{G}_p), along with proof

$$\text{PoK}_3\{(s_i, t_i, x_i) : y_i = u^{x_i} \wedge C' g_0^{s''_i} = g_0^{s_i} g_1^{t_i} g_2^{x_i}\}$$

- 4) GM verifies the proof computes $C = C' g_0^{s''_i}$ and selects $e_i \in_R \mathbb{Z}_p^*$. He then computes $\varsigma_i = (gC)^{\frac{1}{e_i + \mu}}$ and sends (ς_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 $\hat{e}(\varsigma_i, Zh^{e_i}) = \hat{e}(g g_0^{s_i} g_1^{t_i} g_2^{x_i}, h)$. He stores his membership public key and membership secret key as (y_i, e_i) and $(\varsigma_i, s_i, t_i, x_i)$.

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 \mathcal{L}_j is $(*, *, V_j)$. AP_j computes V'_j as $V_j^{e_i + \gamma_j}$ and sets $w_{i,j} = V'_j$. He returns $w_{i,j}$ to U_i and appends (e_j, ADD, V'_j) to \mathcal{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 $\hat{e}(w_{i,j}, Y_j h_j^{e_i}) = \hat{e}(V'_j, h)$.

The following describe how other users update his witness when the list \mathcal{L}_j is updated. This is simply a rephrase of the update algorithm of N-Acc. Specifically, suppose user U_i such that e_i is in \mathcal{L}_j is required to update their membership witness

$w_{i,j}$ and $\hat{e}(w_{i,j}, Y_j h_j^{e_i}) = \hat{e}(V_j, h_j)$. U_i just computes $w'_{i,j} = V_j w_{i,j}^{(e_i - e_i')}$ and sets $w_{i,j}$ as $w'_{i,j}$.

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

User U_i (except U_i) such that e_i is in \mathcal{L}_j is required to update their membership witness $w_{i,j}$ using the update algorithm of N-Acc. Suppose $\hat{e}(w_{i,j}, Y_j h_j^{e_i}) = \hat{e}(V_j, h_j)$, user U_i computes $w'_{i,j} = (\frac{V'_j}{w_{i,j}})^{\frac{1}{e_i - e_i'}}$ and sets $w_{i,j}$ as $w'_{i,j}$.

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 \mathcal{L}_j is $(*, *, V_j)$, $\hat{e}(w_{i,j}, Y_j h_j^{e_i}) = \hat{e}(V_j, h_j)$.

- 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 $S = u_j^{\frac{R}{s_i + n_{i,j} + 1}}$, tracing tag $T = u^{x_i} u_j^{\frac{R}{t_i + n_{i,j} + 1}}$ 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, $\hat{e}(w_{i,j}, Y_j h_j^{e_i}) = \hat{e}(V_j, h_j)$.
 - 3) S is $PRF_{u_j, s_i}(n_{i,j})$, that is, $S = u_j^{\frac{1}{s_i + n_{i,j} + 1}}$.
 - 4) T is $u^{x_i} PRF_{u_j, t}(n_j)^R$, that is, $T = u^{x_i} u_j^{\frac{R}{t_i + n_{i,j} + 1}}$.
 - 5) $0 \leq n_{i,j} < k_j$
- The above can be abstracted as SPK_4 whose instantiation is shown in Section VI-D.

$$SPK_4 \left\{ \begin{array}{l} (s_i, e_i, s_i, t_i, x_i, n_{i,j}, w_{i,j}) : \\ \hat{e}(s_i, Zh^{e_i}) = \hat{e}(gg_0^{s_i} g_1^{t_i} g_2^{x_i}, h) \quad \wedge \\ S = u_j^{\frac{1}{s_i + n_{i,j} + 1}} \quad \wedge \\ T = u^{x_i} u_j^{\frac{R}{t_i + n_{i,j} + 1}} \quad \wedge \\ \hat{e}(V_j, h_j) = \hat{e}(w_{i,j}, Y_j h_j^{e_i}) \quad \wedge \\ 0 \leq n_{i,j} \quad \wedge \\ n_{i,j} < k_j \end{array} \right\} (m)$$

- AP_j verifies SPK_4 is correct. If yes, AP_j accepts and adds (SPK_4, S, T, R, m) to its authentication log.
- U_i increases its counter, $n_{i,j}$, by one.

PublicTracing. For two entries (SPK, S, T, R) and (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 $y_i := u^{x_i} = (\frac{T^{R'}}{T^R})^{((R' - R)^{-1})}$ 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 $q_i = h_0^{q_i}$ for $i = 1, \dots, t_{max}$, 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 needs to publish its identity and bound k . 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 $v = h_0^{\prod_{k=1}^{k=|\{e_j\}|} (e_k + q)}$. This quantity could be locally computed by both user and AP without knowledge of q . The user also computes the witness w by $h_0^{\prod_{k=1, k \neq i}^{k=|\{e_j\}|} (e_k + q)}$ such that $w^{(q+e_i)} = v$.

The rest of the protocol follows the original scheme, and the same SPK (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 $Sig(1), \dots, Sig(k)$. 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 $Sig(n^*)$ for some user where $n^* > k$. However, we can trust the AP would not issue $Sig(n)$ 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 $\mathfrak{A}_1 = \mathfrak{g}_1^{r_1} \mathfrak{g}_2^{r_2}$, $\mathfrak{A}_2 = \varsigma \mathfrak{g}_2^{r_1}$ for some randomly generated $r_1, r_2 \in_R \mathbb{Z}_p^*$. Then he conducts the following protocol with the verifier.

- (Commitment.) The prover randomly generates $\rho_{r_1}, \rho_{r_2}, \rho_{\beta_1}, \rho_{\beta_2}, \rho_r, \rho_{m_1}, \dots, \rho_{m_L}, \rho_e, \rho_s \in_R \mathbb{Z}_p^*$, computes $\mathfrak{T}_1 = \mathfrak{g}_1^{\rho_{r_1}} \mathfrak{g}_2^{\rho_{r_2}}$, $\mathfrak{T}_2 = \mathfrak{A}_1^{-\rho_e} \mathfrak{g}_1^{\rho_{\beta_1}} \mathfrak{g}_2^{\rho_{\beta_2}}$, $\mathfrak{T}_3 = \mathfrak{g}_0^{\rho_r} \mathfrak{g}_1^{\rho_{m_1}} \dots \mathfrak{g}_L^{\rho_{m_L}} \in \mathbb{G}_1$ and $\mathfrak{T}_4 = \hat{e}(\mathfrak{A}_2, h)^{-\rho_e} \hat{e}(\mathfrak{g}_2, Z)^{\rho_{r_1}} \hat{e}(\mathfrak{g}_2, h)^{\rho_{\beta_1}} \hat{e}(g_0, h)^{\rho_s} \hat{e}(g_1, h)^{\rho_{m_1}} \dots \hat{e}(g_L, h)^{\rho_{m_L}} \in \mathbb{G}_T$. The prover sends $(\mathfrak{T}_1, \mathfrak{T}_2, \mathfrak{T}_3, \mathfrak{T}_4)$ to the verifier.
- (Challenge.) The verifier chooses a random challenge $c \in_R \mathbb{Z}_p^*$ and sends c to prover.
- (Response.) The prover computes $z_{r_1} = \rho_{r_1} - cr_1$, $z_{r_2} = \rho_{r_2} - cr_2$, $z_{\beta_1} = \rho_{\beta_1} - c\epsilon r_1$, $z_{\beta_2} = \rho_{\beta_2} - c\epsilon r_2$, $z_r = \rho_r - cr$, $z_{m_1} = \rho_{m_1} - cm_1$, \dots , $z_{m_L} = \rho_{m_L} - cm_L$, $z_e = \rho_e - ce$, $z_s = \rho_s - cs$ and sends $(z_{r_1}, z_{r_2}, z_{\beta_1}, z_{\beta_2}, z_r, z_{m_1}, \dots, z_{m_L}, z_e, z_s)$ to verifier.
- (Verify.) The verifier outputs 1 if

$$\begin{aligned} \mathfrak{T}_1 &\stackrel{?}{=} \mathfrak{A}_1^c \mathfrak{g}_1^{z_{r_1}} \mathfrak{g}_2^{z_{r_2}} \\ \mathfrak{T}_2 &\stackrel{?}{=} \mathfrak{A}_1^{-z_e} \mathfrak{g}_1^{z_{\beta_1}} \mathfrak{g}_2^{z_{\beta_2}} \\ \mathfrak{T}_3 &\stackrel{?}{=} \mathfrak{C}_m^c \mathfrak{g}_0^{z_r} \mathfrak{g}_1^{z_{m_1}} \dots \mathfrak{g}_L^{z_{m_L}} \\ \mathfrak{T}_4 &\stackrel{?}{=} \frac{\hat{e}(\mathfrak{A}_2, Z)^c}{\hat{e}(g, h)} \hat{e}(\mathfrak{A}_2, h)^{-z_e} \hat{e}(\mathfrak{g}_2, Z)^{z_{r_1}} \hat{e}(\mathfrak{g}_2, h)^{z_{\beta_1}} \\ &\quad \hat{e}(g_0, h)^{z_s} \hat{e}(g_1, h)^{z_{m_1}} \dots \hat{e}(g_L, h)^{z_{m_L}} \end{aligned}$$

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.) U_i randomly generates $r_{s'_i}, r_{t_i}, r_{x_i} \in_R \mathbb{Z}_p^*$, computes $\mathfrak{T} = g_0^{r_{s'_i}} g_1^{r_{t_i}} g_2^{r_{x_i}} \in \mathbb{G}_1$ and sends \mathfrak{T} to GM.
- (Challenge.) GM chooses a random challenge $c \in_R \mathbb{Z}_p^*$ and sends c to U_i .
- (Response.) U_i computes $z_{s'_i} = r_{s'_i} - cs'_i$, $z_{t_i} = r_{t_i} - ct_i$ and $z_{x_i} = r_{x_i} - cx_i \in \mathbb{Z}_p$ and sends $(z_{s'_i}, z_{t_i}, z_{x_i})$ to GM.
- (Verify.) GM outputs 1 if $\mathfrak{T} \stackrel{?}{=} C^c g_0^{z_{s'_i}} g_1^{z_{t_i}} g_2^{z_{x_i}}$ and 0 otherwise.

C. Details of PoK₃

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

- (Commitment.) U_i randomly generates $r_{s_i}, r_{t_i}, r_{x_i} \in_R \mathbb{Z}_p^*$, computes $\mathfrak{T}_1 = u^{r_{x_i}} \in \mathbb{G}_p$ and $\mathfrak{T}_2 = g_0^{r_{s_i}} g_1^{r_{t_i}} g_2^{r_{x_i}} \in \mathbb{G}_1$ and sends $(\mathfrak{T}_1, \mathfrak{T}_2)$ to GM.
- (Challenge.) GM chooses a random challenge $c \in_R \mathbb{Z}_p^*$ and sends c to U_i .
- (Response.) U_i computes $z_{s_i} = r_{s_i} - cs_i$, $z_{t_i} = r_{t_i} - ct_i$ and $z_{x_i} = r_{x_i} - cx_i \in \mathbb{Z}_p$ and sends $(z_{s_i}, z_{t_i}, z_{x_i})$ to GM.
- (Verify.) GM outputs 1 if $\mathfrak{T}_1 \stackrel{?}{=} y_i^c u^{z_{x_i}}$ and $\mathfrak{T}_2 \stackrel{?}{=} (C' g_0^{s'_i})^c g_0^{z_{s_i}} g_1^{z_{t_i}} g_2^{z_{x_i}}$; and 0 otherwise.

D. Details of SPK₄

To conduct SPK₄, U_i computes $\mathfrak{A}_1 = \mathfrak{g}_1^{e_i} \mathfrak{g}_2^{\rho_1}$, $\mathfrak{A}_2 = \mathfrak{g}_1^{x_i} \mathfrak{g}_2^{\rho_2}$, $\mathfrak{A}_3 = \varsigma_i \mathfrak{g}_1^{\rho_3}$, $\mathfrak{A}_4 = w_{i,j} \mathfrak{g}_1^{\rho_4}$, $\mathfrak{A}_5 = \mathfrak{g}_1^{n_{i,j}} \mathfrak{g}_2^{\rho_5}$ for some randomly generated $\rho_1, \rho_2, \rho_3, \rho_4, \rho_5 \in_R \mathbb{Z}_p^*$. Next, U_i sends $\mathfrak{A}_1, \mathfrak{A}_2, \mathfrak{A}_3, \mathfrak{A}_4, \mathfrak{A}_5$ to AP_j and computes the following two SPK's.

$$\text{SPK}_{4A} \left\{ \begin{array}{l} \left(\begin{array}{l} e_i, s_i, t_i, x_i, n_{i,j}, \rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \\ \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8 \end{array} \right) : \\ \mathfrak{A}_1 = \mathfrak{g}_1^{e_i} \mathfrak{g}_2^{\rho_1} \quad \wedge \\ \mathfrak{A}_2 = \mathfrak{g}_1^{x_i} \mathfrak{g}_2^{\rho_2} \quad \wedge \\ \mathfrak{A}_5 = \mathfrak{g}_1^{n_{i,j}} \mathfrak{g}_2^{\rho_5} \quad \wedge \\ 1 = \mathfrak{A}_1^{-\rho_3} \mathfrak{g}_1^{\beta_1} \mathfrak{g}_2^{\beta_2} \quad \wedge \\ 1 = \mathfrak{A}_1^{-\rho_4} \mathfrak{g}_1^{\beta_3} \mathfrak{g}_2^{\beta_4} \quad \wedge \\ 1 = \mathfrak{A}_2^{-t_i} \mathfrak{g}_1^{\beta_5} \mathfrak{g}_2^{\beta_6} \quad \wedge \\ 1 = \mathfrak{A}_2^{-n_{i,j}} \mathfrak{g}_1^{\beta_7} \mathfrak{g}_2^{\beta_8} \quad \wedge \\ \frac{\hat{e}(\mathfrak{A}_3, Z)}{\hat{e}(g, h)} = \hat{e}(g_0, h)^{s_i} \hat{e}(g_1, h)^{t_i} \\ \quad \hat{e}(g_2, h)^{x_i} \hat{e}(\mathfrak{g}_1, Z)^{\rho_3} \\ \quad \hat{e}(\mathfrak{g}_1, h)^{\beta_1} \hat{e}(\mathfrak{A}_3, h)^{-e_i} \quad \wedge \\ \frac{\hat{e}(\mathfrak{A}_4, Y_j)}{\hat{e}(V_j, h_j)} = \hat{e}(\mathfrak{A}_4, h_j)^{-e_i} \hat{e}(\mathfrak{g}_1, h_j)^{\beta_3} \\ \quad \hat{e}(\mathfrak{g}_1, Y_j)^{\rho_4} \quad \wedge \\ \frac{u_j}{S} = S^{s_i} S^{n_{i,j}} \quad \wedge \\ \frac{u^R}{T} = u^{-\beta_5} u^{-\beta_7} u^{-x_i} T^{t_i} T^{n_{i,j}} \end{array} \right\} (m)$$

$$\text{SPK}_{4B} \left\{ (n_{i,j}, \rho_5) : \mathfrak{A}_5 = \mathfrak{g}_1^{n_{i,j}} \mathfrak{g}_2^{\rho_5} \wedge 0 \leq n_{i,j} < k_j \right\} (m)$$

Instantiation of SPK_{4B} is a simple range proof and is discussed in Section VI-E. Below we show how to instantiate SPK_{4A}.

- (Commitment.) U_i randomly generates $r_{e_i}, r_{s_i}, r_{t_i}, r_{x_i}, r_{n_{i,j}}, r_{\rho_1}, r_{\rho_2}, r_{\rho_3}, r_{\rho_4}, r_{\rho_5}, r_{\beta_1}, r_{\beta_2}, r_{\beta_3}, r_{\beta_4}, r_{\beta_5}, r_{\beta_6}, r_{\beta_7}, r_{\beta_8} \in_R \mathbb{Z}_p^*$ and computes $\mathfrak{T}_1 = \mathfrak{g}_1^{r_{e_i}} \mathfrak{g}_2^{r_{\rho_1}}$, $\mathfrak{T}_2 = \mathfrak{g}_1^{r_{x_i}} \mathfrak{g}_2^{r_{\rho_2}}$, $\mathfrak{T}_3 = \mathfrak{g}_1^{r_{n_{i,j}}} \mathfrak{g}_2^{r_{\rho_5}}$, $\mathfrak{T}_4 = \mathfrak{A}_1^{-r_{\rho_3}} \mathfrak{g}_1^{r_{\beta_1}} \mathfrak{g}_2^{r_{\beta_2}}$, $\mathfrak{T}_5 = \mathfrak{A}_1^{-r_{\rho_4}} \mathfrak{g}_1^{r_{\beta_3}} \mathfrak{g}_2^{r_{\beta_4}}$, $\mathfrak{T}_6 = \mathfrak{A}_2^{-r_{t_i}} \mathfrak{g}_1^{r_{\beta_5}} \mathfrak{g}_2^{r_{\beta_6}}$, $\mathfrak{T}_7 = \mathfrak{A}_2^{-r_{n_{i,j}}} \mathfrak{g}_1^{r_{\beta_7}} \mathfrak{g}_2^{r_{\beta_8}}$ in \mathbb{G}_1 , $\mathfrak{T}_8 = \hat{e}(g_0, h)^{r_{s_i}} \hat{e}(g_1, h)^{r_{t_i}} \hat{e}(g_2, h)^{r_{x_i}} \hat{e}(\mathfrak{g}_1, Z)^{r_{\rho_3}} \hat{e}(\mathfrak{g}_1, h)^{r_{\beta_1}} \hat{e}(\mathfrak{A}_3, h)^{-r_{e_i}}$, $\mathfrak{T}_9 = \hat{e}(\mathfrak{A}_4, h_j)^{-r_{e_i}} \hat{e}(\mathfrak{g}_1, h_j)^{r_{\beta_3}} \hat{e}(\mathfrak{g}_1, Y_j)^{r_{\rho_4}}$ in \mathbb{G}_T and $\mathfrak{T}_{10} = S^{r_{s_i}} S^{r_{n_{i,j}}}$, $\mathfrak{T}_{11} = u^{-r_{\beta_5}} u^{-r_{\beta_7}} u^{-r_{x_i}} T^{r_{t_i}} T^{r_{n_{i,j}}}$ in \mathbb{G}_p . U_i sets $\mathfrak{T} = (\mathfrak{T}_1, \dots, \mathfrak{T}_{11})$.

- (Challenge.) U_i computes challenge $c = H(m||AP_j||\mathfrak{T})$ using a cryptographic hash function H .
- (Response.) U_i computes, in \mathbb{Z}_p , $z_{e_i} = r_{e_i} - ce_i$, $z_{s_i} = r_{s_i} - cs_i$, $z_{t_i} = r_{t_i} - ct_i$, $z_{x_i} = r_{x_i} - cx_i$, $z_{n_{i,j}} = r_{n_{i,j}} - cn_{i,j}$, $z_{\rho_1} = r_{\rho_1} - c\rho_1$, $z_{\rho_2} = r_{\rho_2} - c\rho_2$, $z_{\rho_3} = r_{\rho_3} - c\rho_3$, $z_{\rho_4} = r_{\rho_4} - c\rho_4$, $z_{\rho_5} = r_{\rho_5} - c\rho_5$, $z_{\beta_1} = r_{\beta_1} - c\rho_3 e_i$, $z_{\beta_2} = r_{\beta_2} - c\rho_3 \rho_1$, $z_{\beta_3} = r_{\beta_3} - c\rho_4 e_i$, $z_{\beta_4} = r_{\beta_4} - c\rho_4 \rho_1$, $z_{\beta_5} = r_{\beta_5} - cx_i t_i$, $z_{\beta_6} = r_{\beta_6} - c\rho_2 t_i$, $z_{\beta_7} = r_{\beta_7} - cx_i n_{i,j}$, $z_{\beta_8} = r_{\beta_8} - c\rho_2 n_{i,j}$. U_i sets $\mathfrak{z} = (z_{e_i}, z_{s_i}, z_{t_i}, z_{x_i}, z_{n_{i,j}}, z_{\rho_1}, z_{\rho_2}, z_{\rho_3}, z_{\rho_4}, z_{\rho_5}, z_{\beta_1}, z_{\beta_2}, z_{\beta_3}, z_{\beta_4}, z_{\beta_5}, z_{\beta_6}, z_{\beta_7}, z_{\beta_8})$.
- (Output.) U_i outputs (c, \mathfrak{z}) as SPK_{4A} .
- (Verify.) Upon receiving $\text{SPK}_{4A} := (c, \mathfrak{z})$, AP_j computes the following.

$$\begin{aligned}\mathfrak{T}'_1 &= \mathfrak{A}_1^c \mathfrak{g}_1^{z_{e_i}} \mathfrak{g}_2^{z_{\rho_1}} \\ \mathfrak{T}'_2 &= \mathfrak{A}_2^c \mathfrak{g}_1^{z_{x_i}} \mathfrak{g}_2^{z_{\rho_2}} \\ \mathfrak{T}'_3 &= \mathfrak{A}_3^c \mathfrak{g}_1^{z_{n_{i,j}}} \mathfrak{g}_2^{z_{\rho_5}} \\ \mathfrak{T}'_4 &= \mathfrak{A}_1^{-z_{\rho_3}} \mathfrak{g}_1^{z_{\beta_1}} \mathfrak{g}_2^{z_{\beta_2}} \\ \mathfrak{T}'_5 &= \mathfrak{A}_1^{-z_{\rho_4}} \mathfrak{g}_1^{z_{\beta_3}} \mathfrak{g}_2^{z_{\beta_4}} \\ \mathfrak{T}'_6 &= \mathfrak{A}_2^{-z_{t_i}} \mathfrak{g}_1^{z_{\beta_5}} \mathfrak{g}_2^{z_{\beta_6}} \\ \mathfrak{T}'_7 &= \mathfrak{A}_2^{-z_{n_{i,j}}} \mathfrak{g}_1^{z_{\beta_7}} \mathfrak{g}_2^{z_{\beta_8}} \\ \mathfrak{T}'_8 &= \left[\frac{\hat{e}(\mathfrak{A}_3, Z)}{\hat{e}(g, h)} \right]^c \hat{e}(g_0, h)^{z_{s_i}} \hat{e}(g_1, h)^{z_{t_i}} \hat{e}(g_2, h)^{z_{x_i}} \\ &\quad \hat{e}(\mathfrak{g}_1, Z)^{z_{\rho_3}} \hat{e}(\mathfrak{g}_1, h)^{z_{\beta_1}} \hat{e}(\mathfrak{A}_3, h)^{-z_{e_i}} \\ \mathfrak{T}'_9 &= \left[\frac{\hat{e}(\mathfrak{A}_4, Y_j)}{\hat{e}(V_j, h_j)} \right]^c \hat{e}(\mathfrak{A}_4, h_j)^{-z_{e_i}} \hat{e}(\mathfrak{g}_1, h_j)^{z_{\beta_3}} \hat{e}(\mathfrak{g}_1, Y_j)^{z_{\rho_4}} \\ \mathfrak{T}'_{10} &= \left[\frac{u_j}{S} \right]^c S^{z_{s_i}} S^{z_{n_{i,j}}} \\ \mathfrak{T}'_{11} &= \left[\frac{u_j^R}{T} \right]^c u^{-z_{\beta_5}} u^{-z_{\beta_7}} u^{-z_{x_i}} T^{z_{t_i}} T^{z_{n_{i,j}}}\end{aligned}$$

Set $\mathfrak{T}' = (\mathfrak{T}'_1, \dots, \mathfrak{T}'_{11})$. Output 1 if $c \stackrel{?}{=} H(m||AP_j||\mathfrak{T}')$ 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 $k = 2^\kappa$, a simple range proof of order $O(\kappa)$ can be constructed easily.

For the ease of presentation, we let $\mathfrak{C}_n = \mathfrak{g}_0^n \mathfrak{g}_1^r$ be a commitment of n and the goal is the following zero-knowledge proof-of-knowledge.

$$\text{PoK}_{\text{RANGE}} \{ (n, r) : \mathfrak{C}_n = \mathfrak{g}_0^n \mathfrak{g}_1^r \wedge 0 \leq n < 2^\kappa \}$$

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 ℓ -th bit of n such that ℓ starts from 0. For $\ell = 0$ to $\kappa - 1$, compute $\mathfrak{C}_\ell = \mathfrak{g}_1^{n[\ell]} \mathfrak{g}_2^{r_\ell}$ such that $r_\ell \in_R \mathbb{Z}_p^*$.

Conduct the following two SPK 's.

$$\text{SPK}_{5A} \left\{ \begin{array}{l} (n, r, \alpha) : \\ \mathfrak{C}_n = \mathfrak{g}_0^n \mathfrak{g}_1^r \wedge \\ \prod_{\ell=0}^{\kappa-1} \mathfrak{C}_\ell^{2^\ell} = \mathfrak{g}_1^n \mathfrak{g}_2^\alpha \end{array} \right\} (m)$$

$$\text{SPK}_{5B} \left\{ \begin{array}{l} (r_0, \dots, r_{\kappa-1}) : \\ \bigwedge_{\ell=0}^{\kappa-1} \left(\mathfrak{C}_\ell = \mathfrak{g}_2^{r_\ell} \vee \right. \\ \left. \mathfrak{C}_\ell / \mathfrak{g}_1 = \mathfrak{g}_2^{r_\ell} \right) \end{array} \right\} (m)$$

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 randomly generates $\rho_n, \rho_r, \rho_\alpha \in_R \mathbb{G}_p$ and computes $\mathfrak{T}_1 = \mathfrak{g}_0^{\rho_n} \mathfrak{g}_1^{\rho_r}$, $\mathfrak{T}_2 = \mathfrak{g}_1^{\rho_n} \mathfrak{g}_2^{\rho_\alpha}$. Set $\mathfrak{T} = (\mathfrak{T}_1, \mathfrak{T}_2)$.
- (Challenge.) The prover computes challenge $c = H(m||\mathfrak{T})$ using a cryptographic hash function H .
- (Response.) The prover computes $z_n = \rho_n - cn$, $z_r = \rho_r - cr$ and $z_\alpha = \rho_\alpha - c \sum_{\ell=0}^{\kappa-1} 2^\ell r_\ell \in \mathbb{Z}_p$ and sets \mathfrak{z} as (z_n, z_r, z_α) .
- (Output.) The prover outputs (c, \mathfrak{z}) as SPK_{5A} .
- (Verify.) The verifier computes $\mathfrak{T}'_1 = \mathfrak{C}_n^{z_n} \mathfrak{g}_1^{z_r}$ and $\mathfrak{T}'_2 = (\prod_{\ell=0}^{\kappa-1} \mathfrak{C}_\ell^{2^\ell})^c \mathfrak{g}_1^{z_n} \mathfrak{g}_2^{z_\alpha}$. The verifier sets $\mathfrak{T}' = (\mathfrak{T}'_1, \mathfrak{T}'_2)$ and outputs 1 if $c \stackrel{?}{=} H(m||\mathfrak{T}')$ and 0 otherwise.

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

- (Commitment.) For $\ell = 0$ to $\kappa - 1$, randomly picks $c_{\ell,1-n[\ell]}, z_{r_{\ell,1-n[\ell]}}, \rho_{r_{\ell,n[\ell]}} \in_R \mathbb{Z}_p$ and computes $\mathfrak{T}_{\ell,1-n[\ell]} = \left(\frac{\mathfrak{C}_\ell}{\mathfrak{g}_1^{1-n[\ell]}} \right)^{c_{\ell,1-n[\ell]}} \mathfrak{g}_2^{z_{r_{\ell,1-n[\ell]}}}$ and $\mathfrak{T}_{\ell,n[\ell]} = \mathfrak{g}_2^{\rho_{r_{\ell,n[\ell]}}}$. For $\ell = 0$ to $\kappa - 1$, set $\mathfrak{T}_\ell = (\mathfrak{T}_{\ell,0}, \mathfrak{T}_{\ell,1})$.
- (Challenge.) For $\ell = 0$ to $\kappa - 1$, the prover computes challenge $c_\ell = H(m||\mathfrak{T}_\ell)$ using a cryptographic hash function H .
- (Response.) For $\ell = 0$ to $\kappa - 1$, the prover computes $c_{\ell,n[\ell]} = c_\ell - c_{\ell,1-n[\ell]}$. The prover computes $z_{r_{\ell,n[\ell]}} = \rho_{r_{\ell,n[\ell]}} - c_{\ell,n[\ell]} r_\ell$. Sets $\mathfrak{c} = (c_{\ell,0}, c_{\ell,1})_{\ell=0}^{\kappa-1}$ and $\mathfrak{z} = (z_{r_{\ell,0}}, z_{r_{\ell,1}})_{\ell=0}^{\kappa-1}$.
- (Output.) The prover outputs $(\mathfrak{z}, \mathfrak{c})$ as SPK_{5B} .
- (Verify.) For $\ell = 0$ to $\kappa - 1$, verifier computes $\mathfrak{T}'_{\ell,0} = \mathfrak{C}_\ell^{c_{\ell,0}} \mathfrak{g}_2^{z_{r_{\ell,0}}}$ and $\mathfrak{T}'_{\ell,1} = \mathfrak{C}_\ell^{c_{\ell,1}} \mathfrak{g}_2^{z_{r_{\ell,1}}}$. Set $\mathfrak{T}'_\ell = (\mathfrak{T}'_{\ell,0}, \mathfrak{T}'_{\ell,1})$. Output 1 if $c_{\ell,0} + c_{\ell,1} = H(m||\mathfrak{T}'_\ell)$ for all $\ell = 0$ to $\kappa - 1$ and 0 otherwise.

The two SPK 's consists of $(4\kappa + 4)$ elements in \mathbb{Z}_p , and κ \mathfrak{C}_ℓ 's in \mathbb{G}_1 . In our protocol, total size of the range proof is $(4 + 4\kappa) * 170 + \kappa * 171$ 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 $\iota \in \mathbb{Z}_p$ be a secret value of an AP, and $I = h^\iota$ be a public value. Specifically, a weakly-secure Boneh-Boyen short signature[27] $\text{Sig}(n)$ on n is $\text{Sig}(n) = g^{\frac{n}{\iota+n}}$. Note that each $(\text{Sig}(n), n)$ pair satisfies $\hat{e}(\text{Sig}(n), I h^n) = \hat{e}(g, h)$. The AP also publishes $\text{Sig}(1), \dots, \text{Sig}(k)$ as public parameter. Then, SPK_{4B} can be instantiated as SPK_{5C}

$$\text{SPK}_{5C} \left\{ \begin{array}{l} (n, r, \text{Sig}(n)) : \\ \mathfrak{A}_5 = \mathfrak{g}_0^n \mathfrak{g}_1^r \quad \wedge \\ \hat{e}(\text{Sig}(n), I h^n) = \hat{e}(g, h) \end{array} \right\} (m)$$

To conduct SPK_{5C} , the prover first computes $\mathfrak{A}_6 = \mathfrak{g}_1^{r_1} \mathfrak{g}_2^{r_2}$, $\mathfrak{A}_7 = \text{Sig}(n) \mathfrak{g}_2^{r_1}$ for some randomly generated $r_1, r_2 \in_R \mathbb{Z}_p^*$.

- (Commitment.) The prover randomly generates $\rho_{r_1}, \rho_{r_2}, \rho_n, \rho_{\beta_1}, \rho_{\beta_2}, \rho_r \in_R \mathbb{Z}_p^*$, computes $\mathfrak{T}_1 = \mathfrak{g}_1^{\rho_{r_1}} \mathfrak{g}_2^{\rho_{r_2}}$, $\mathfrak{T}_2 = \mathfrak{A}_6^{-\rho_n} \mathfrak{g}_1^{\rho_{\beta_1}} \mathfrak{g}_2^{\rho_{\beta_2}}$, $\mathfrak{T}_3 = \hat{e}(\mathfrak{A}_7, h)^{-\rho_n} \hat{e}(\mathfrak{g}_2, I)^{\rho_{r_1}}$, $\hat{e}(\mathfrak{g}_2, h)^{\rho_{\beta_1}} \in \mathbb{G}_T$ and $\mathfrak{T}_4 = \mathfrak{g}_1^{\rho_n} \mathfrak{g}_2^{\rho_r}$. The prover sets $\mathfrak{T} = (\mathfrak{T}_1, \mathfrak{T}_2, \mathfrak{T}_3, \mathfrak{T}_4)$.
- (Challenge.) The prover computes challenge $c = H(m || \mathfrak{T})$.
- (Response.) The prover computes $z_{r_1} = \rho_{r_1} - c r_1$, $z_{r_2} = \rho_{r_2} - c r_2$, $z_{\beta_1} = \rho_{\beta_1} - c n r_1$, $z_{\beta_2} = \rho_{\beta_2} - c n r_2$, $z_r = \rho_r - c r$, $z_n = \rho_n - c n$ and sets $\mathfrak{z} = (z_{r_1}, z_{r_2}, z_{\beta_1}, z_{\beta_2}, z_r, z_n)$.
- (Output.) The prover outputs (c, \mathfrak{z}) as SPK_{5C} .
- (Verify.) Upon receiving $\text{SPK}_{5C} := (c, \mathfrak{z})$, verifier computes the following.

$$\begin{aligned} \mathfrak{T}'_1 &= \mathfrak{A}_6^c \mathfrak{g}_1^{z_{r_1}} \mathfrak{g}_2^{z_{r_2}} \\ \mathfrak{T}'_2 &= \mathfrak{A}_6^{-z_n} \mathfrak{g}_1^{z_{\beta_1}} \mathfrak{g}_2^{z_{\beta_2}} \\ \mathfrak{T}'_3 &= \left[\frac{\hat{e}(\mathfrak{A}_7, I)}{\hat{e}(g, h)} \right]^c \hat{e}(\mathfrak{A}_7, h)^{-z_n} \hat{e}(\mathfrak{g}_2, I)^{z_{r_1}} \hat{e}(\mathfrak{g}_2, h)^{z_{\beta_1}} \\ \mathfrak{T}'_4 &= \mathfrak{A}_5^c \mathfrak{g}_1^{z_n} \mathfrak{g}_2^{z_r} \end{aligned}$$

Set $\mathfrak{T}' = (\mathfrak{T}'_1, \dots, \mathfrak{T}'_4)$. Output 1 if $c \stackrel{?}{=} H(m || \mathfrak{T}')$ and 0 otherwise.

VII. SECURITY ANALYSIS

A. Analysis of BBS+

Capabilities of the Forger. \mathcal{F} is allowed to issue two types of signing query, namely, normal signing and signing through the signature generation protocol. \mathcal{F} is a successful forger if, after issuing q signature queries in total, it can output $q + 1$ valid and distinct message-signature pairs.

Simulation and Reduction. 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 in a bilinear group pair.

\mathcal{S} is given an instance of the q -SDH problem $(g', h', h'^\mu, \dots, h'^{\mu^q})$ and its goal is to output a pair (A', e') such that $A'^{e'+\mu} = g'$. This pair satisfies $\hat{e}(A', h'^{e'}) = \hat{e}(g', h')$. \mathcal{S} first randomly chooses, for $i = 1$ to $q - 1$, $e_i \in_R \mathbb{Z}_p^*$ and denotes the $q - 1$ degree polynomial $f(x) = \prod_{i=1}^{q-1} (x + e_i)$. \mathcal{S} also randomly chooses $e^*, k^*, a^* \in_R \mathbb{Z}_p^*$ and computes $h = h'^{f(\mu)}$, $Z = h^\mu = h'^{\mu f(\mu)}$ and $h_0 = h^{\frac{(e^*+\mu)k^*-1}{a^*}}$. Next \mathcal{S} randomly chooses $\mu_j \in_R \mathbb{Z}_p^*$ and sets $h_j = h_0^{\mu_j}$ for $j = 1$ to L . Finally, \mathcal{S} computes $g = \psi(h)$ and $g_j = \psi(h_j)$ for $j = 0$ to L and gives (h, Z, g_0, \dots, g_L) to \mathcal{F} as the public key of the BBS+ signature.

\mathcal{F} is allowed to issue up to q signature queries. If it is a signing query through the signature generation protocol,

\mathcal{S} needs to rewind \mathcal{F} during execution of PoK_0 to obtain the multi-block message committed. Then each signing query through the signature generation protocol can be handled in the same way as the normal signing query. Due to the need of rewinding, BBS+ signature generation protocol cannot be run in parallel. Below we only describe how normal signing query is handled.

For the i -th query, denote the multi-block message to be signed as $(m_{1,i}, \dots, m_{\ell_i,i})$ such that $\ell_i \leq L$. For each query, \mathcal{S} computes $M_i = \sum_{j=1}^{\ell_i} m_{j,i} \mu_j$.

Out of these q queries, \mathcal{S} randomly chooses one, called query $*$ which shall be handled differently. For the other $q - 1$ queries, \mathcal{S} randomly picks $s_i \in_R \mathbb{Z}_p^*$, computes $S_i = s_i + M_i$ and $\varsigma_i = (g g_0^{S_i})^{\frac{1}{e_i + \mu}}$. Note that

$$\begin{aligned} \varsigma_i &= g g_0^{S_i} \\ &= \left(g^{\frac{(e^* + \mu)k^* + a^* - 1}{a^*}} \right)^{\frac{1}{e_i + \mu}} \\ &= \left(g'^{\frac{f(\mu)}{(e_i + \mu)}} \right)^{\frac{a^* - 1}{a^*}} \left(g'^{\frac{f(\mu)(e^* + \mu)}{e_i + \mu}} \right)^{k^*} \end{aligned}$$

and is computable by \mathcal{S} even though μ is unknown since $(e_i + \mu)$ divides $f(\mu)$ and $(e^* + \mu) \left(\frac{f(\mu)}{e_i + \mu} \right)$ is a degree q polynomial. \mathcal{S} returns (ς_i, e_i, s_i) as the answer of the i -th signature query.

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

$$\begin{aligned} (g^{k^*})^{e^* + \mu} &= g g_0^{a^*} \\ &= g g_0^{a^*} \\ &= g g_0^{s^*} g_0^{M_*} \\ &= g g_0^{s^*} g_1^{m_{1,*}} \dots g_{\ell_*}^{m_{\ell_*,*}} \end{aligned}$$

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 (ς', e', s') , $(m'_1, \dots, m'_{\ell'})$. Denote $S' = s' + \sum_{j=1}^{\ell'} m_{j,\ell'}$.

There are three possibilities.

- Case I [$e' \notin \{e_i, e^*\}$]:

$$\begin{aligned} \varsigma'^{e' + \mu} &= g g_0^{S'} \\ \varsigma'^{e' + \mu} &= g^{\frac{a^* - S'}{a^*}} g^{\frac{(e^* + \mu)k^* S'}{a^*}} \end{aligned}$$

Since $e' \notin \{e_i, e^*\}$, $(e' + \mu)$ does not divide $f(\mu)$ (respectively $f(\mu)(\mu + e^*)$) and \mathcal{S} computes a $q - 2$ (resp. $q - 1$) degree polynomial $Q(\mu)$ (resp. $Q^*(\mu)$) and constant Q (resp. Q^*) such that $f(\mu) = Q(\mu)(\mu + e') + Q$ (resp. $f(\mu)(\mu + e^*) = Q^*(\mu)(\mu + e') + Q^*$). Thus,

$$\begin{aligned} \varsigma' &= \left(g'^{\frac{a^* - S'}{a^*}} \right)^{Q(\mu) + \frac{Q}{e' + \mu}} \left(g'^{\frac{(e^* + \mu)k^* S'}{a^*}} \right)^{Q^*(\mu) + \frac{Q^*}{\mu + e'}} \\ g'^{\frac{1}{\mu + e'}} &= (\varsigma' (g'^{Q(\mu)})^{\frac{S' - a^*}{a^*}} (g'^{Q^*(\mu)})^{\frac{-k^* S'}{a^*}})^{\frac{1}{Q + Q^*}} \end{aligned}$$

Thus, $(g'^{\frac{1}{\mu + e'}}, e')$ is the solution to the q -SDH problem.

- Case II: [$e' = e_i$ and $\varsigma' = \varsigma_i$]: 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 $\varsigma' \neq \varsigma_i$: With probability, $1/q$, $e' = e^*$.

$$\begin{aligned}\varsigma'^{e^*+\mu} &= gg_0^{S'} \\ \varsigma'^{e^*+\mu} &= g^{\frac{(e^*+\mu)k^*S'+a^*-S'}{a^*}} \\ \varsigma' &= g^{\frac{k^*S'}{a^*}} g^{\frac{a^*-S'}{(e^*+\mu)a^*}}\end{aligned}$$

Since $(e^* + \mu)$ does not divide $f(\mu)$, \mathcal{S} computes $q - 2$ degree polynomial $Q(\mu)$ and constant Q such that $f(\mu) = (e^* + \mu)Q(\mu) + Q$. Thus,

$$\begin{aligned}\varsigma' &= g^{\frac{k^*S'}{a^*}} (g^{Q(\mu) + \frac{Q}{e^*+\mu}})^{\frac{a^*-S'}{a^*}} \\ g'^{\frac{1}{\mu+e'}} &= (\varsigma' g^{-\frac{k^*S'}{a^*}} g^{\frac{S'-a^*}{a^*}})^{\frac{1}{Q}}\end{aligned}$$

Thus, $(g'^{\frac{1}{\mu+e'}}, e')$ is the solution to the q -SDH problem.

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

B. Analysis of Our k -TAA

D-Detectability. Let \mathcal{A} be an adversary who executes f Join protocol with simulator \mathcal{S} acting as the GM. Let $\mathcal{E}_{\text{PoK}_2}, \mathcal{E}_{\text{PoK}_3}$ be extractors of the PoK_2 and PoK_3 respectively. For each join request, \mathcal{S} acts exactly as an honest GM would, except during step 3, where \mathcal{S} runs the extractor $\mathcal{E}_{\text{PoK}_3}$ to extract the values (s, t, x) . From the value s , \mathcal{S} compute the tuple $w_l = (S_1, \dots, S_k, s, t, x)$ such that $S_i = u_0^{1/(J_{AP}+s+1)}$ for $i = 1, \dots, k$. 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} . Moreover, such extraction requires rewind simulation and thus the Join protocol cannot be executed concurrently.

For each AP, let $u_j = H(AP_j) = u_0^{r_j}$ for some randomly chosen r_j . Due to the soundness of the underlying SPK, A_f , together with the set $\{r_j\}$ contains all *valid* one-time pass 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} 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) \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 one-time pass 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) $\varsigma^{e+\mu} = gg_0^s g_1^t g_2^x$
- 2) $w_j^{e+\gamma_j} = V_j$
- 3) $S = u_j^{1/(n_j+s+1)}$
- 4) $T = u_0^x u_j^{R/(n_j+t+1)}$
- 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 $k = 2^*$, 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 \mathcal{A} in case 1 is negligible.

Consider case (2), it has 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 one-time pass S , to authenticated more than k times for the same AP, it must use 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 $S^{1/r_j} \in A_f$. Since the AP is honest, $R_1 \neq R_2$ with high probability. We are to show that $T_1 = u_0^x u_j^{R_1/(n_j+t+1)}$ and $T_2 = u_0^x u_j^{R_2/(n_j+t+1)}$ 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 tracing tags to accompany the duplicated one-time pass 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 overwhelming probability.

During the course of the running of \mathcal{A} , it is allowed to query several oracles. We outline how \mathcal{S} simulates these oracles. The join oracle is simulated by invoking the signing oracle of the BBS+ signature (\mathcal{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 \mathcal{A} , colluding with the GM and all AP's, creates 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} simulates as follow.

- \mathcal{S} is given *seed* and compute $R = H(\text{seed})$.
- \mathcal{S} random chooses s, t, x and a random $n_j \in_R \{1, \dots, k\}$ and compute $S = u_j^{1/(n_j+s+1)}$, $T = u_0^x u_j^{R/(n_j+t+1)}$.
- \mathcal{S} simulates a proof of $\varsigma, w, e, s, t, x, n_j$ such that (1) $\varsigma^{e+\mu} = gg_0^s g_1^t g_2^x$, (2) $w^{e+\gamma_j} = V_j$, (3) $S = u_j^{1/(n_j+s+1)}$, (4) $T = u_0^x u_j^{R/(n_j+t+1)}$, (5) $0 \leq n_j \leq k$.
- 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 \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 [31], S and T are indistinguishable from randomly elements under the k -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. 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_4 . Assume m is also an element in \mathbb{Z}_p^* , the total communication cost consists of 2 elements in \mathbb{G}_p , 7 elements in \mathbb{G}_1 and 24 elements in \mathbb{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 \mathbb{G}_1 , 2 multi-EXP's in \mathbb{G}_p , 4 multi-EXP's in \mathbb{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 \mathbb{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 \mathbb{G}_1 , 2 multi-EXP's in \mathbb{G}_2 , 2 multi-EXP's in \mathbb{G}_p , 7 multi-EXP's in \mathbb{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 \mathbb{Z}_p , \mathbb{G}_1 can be represented by 24 and 25 bytes respectively. As for \mathbb{G}_p , we have two choices. We could take \mathbb{G}_p as \mathbb{G}_1 if we assume DDH problem is difficult in \mathbb{G}_1 . Such a assumption is formally called the external Diffie-Hellman (XDH) assumption, which implies there is no efficient mapping from \mathbb{G}_1 to \mathbb{G}_2 . On the other hand, we could take \mathbb{G}_p as \mathbb{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 \mathbb{G}_1 takes 2.32ms (resp. 2.36ms).

When we take \mathbb{G}_p to be \mathbb{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 \mathbb{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 to identify practical application for authentication scheme with a “nice” level of privacy.

REFERENCES

- [1] M. H. Au, W. Susilo, and Y. Mu, “Constant-Size Dynamic k -TAA,” in *Security and Cryptography for Networks (SCN)*, ser. Lecture Notes in Computer Science, vol. 4116. Springer, 2006, pp. 111–125.
- [2] I. Teranishi, J. Furukawa, and K. Sako, “ k -Times Anonymous Authentication (Extended Abstract),” in *Advances in Cryptology - ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 3329. Springer, 2004, pp. 308–322.
- [3] L. Nguyen and R. Safavi-Naini, “Dynamic k -Times Anonymous Authentication,” in *Applied Cryptography and Network Security (ACNS)*, ser. Lecture Notes in Computer Science, vol. 3531, 2005, pp. 318–333.
- [4] J. Camenisch and A. Lysyanskaya, “A Signature Scheme with Efficient Protocols,” in *Security in Communication Networks (SCN)*, ser. Lecture Notes in Computer Science, vol. 2576. Springer, 2002, pp. 268–289.
- [5] —, “Signature Schemes and Anonymous Credentials from Bilinear Maps,” in *Advances in Cryptology - CRYPTO*, ser. Lecture Notes in Computer Science, vol. 3152. Springer, 2004, pp. 56–72.
- [6] D. Boneh, X. Boyen, and H. Shacham, “Short Group Signatures,” in *Advances in Cryptology - CRYPTO*, ser. Lecture Notes in Computer Science, vol. 3152. Springer, 2004, pp. 41–55.
- [7] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, “Blacklistable Anonymous Credentials: Blocking Misbehaving Users without TTPs,” in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, Oct./Nov. 2007, pp. 72–81.
- [8] J. Camenisch, M. Dubovitskaya, and G. Neven, “Oblivious Transfer with Access Control,” in *ACM Conference on Computer and Communications Security (CCS)*, 2009, pp. 131–140.
- [9] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki, “Revocable group signature schemes with constant costs for signing and verifying,” in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, S. Jarecki and G. Tsudik, Eds., vol. 5443. Springer, 2009, pp. 463–480.
- [10] S. S. M. Chow, “Real Traceable Signatures,” in *Selected Areas in Cryptography (SAC)*, ser. Lecture Notes in Computer Science, vol. 5867. Springer, 2009, pp. 92–107.
- [11] J. Camenisch, M. Dubovitskaya, and G. Neven, “Unlinkable priced oblivious transfer with rechargeable wallets,” in *Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Sion, Ed., vol. 6052. Springer, 2010, pp. 66–81.
- [12] Y. Yang, J. Zhou, J. W. Wong, and F. Bao, “Towards Practical Anonymous Password Authentication,” in *Annual Computer Security Applications Conference (ACSAC)*, 2010, pp. 59–68.
- [13] J. Camenisch, M. Dubovitskaya, G. Neven, and G. M. Zaverucha, “Oblivious transfer with hidden access control policies,” in *Public Key Cryptography*, ser. Lecture Notes in Computer Science, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., vol. 6571. Springer, 2011, pp. 192–209.
- [14] A. Sudarsono, T. Nakanishi, and N. Funabiki, “Efficient Proofs of Attributes in Pairing-Based Anonymous Credential System,” in *Privacy Enhancing Technologies (PETs)*, ser. Lecture Notes in Computer Science, vol. 6794. Springer, 2011, pp. 246–263.
- [15] M. H. Au, A. Kapadia, and W. Susilo, “BLACR: TTP-free blacklistable anonymous credentials with reputation,” in *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2012.
- [16] M. H. Au and A. Kapadia, “PERM: Practical reputation-based blacklisting without TTPs,” in *The 19th ACM Conference on Computer and Communications Security (CCS)*, To appear. ACM, Oct. 2012.
- [17] J. Camenisch, R. Chaabouni, and A. Shelat, “Efficient Protocols for Set Membership and Range Proofs,” in *Advances in Cryptology - ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 5350. Springer, 2008, pp. 234–252.
- [18] J. Camenisch, M. Kohlweiss, and C. Soriente, “An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials,” in *Public Key Cryptography (PKC)*, ser. Lecture Notes in Computer Science, vol. 5443. Springer, 2009, pp. 481–500.
- [19] J. Groth and A. Sahai, “Efficient Non-interactive Proof Systems for Bilinear Groups,” in *Advances in Cryptology - EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 4965. Springer, 2008, pp. 415–432.
- [20] A. Kiayias, Y. Tsiounis, and M. Yung, “Traceable Signatures,” in *Advances in Cryptology - EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 3027. Springer, 2004, pp. 571–589.
- [21] I. Teranishi and K. Sako, “ k -Times Anonymous Authentication with a Constant Proving Cost,” in *Public Key Cryptography (PKC)*, ser. Lecture Notes in Computer Science, vol. 3958. Springer, 2006, pp. 525–542.
- [22] J. Camenisch and A. Lysyanskaya, “Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials,” in *Advances in Cryptology - CRYPTO*, ser. Lecture Notes in Computer Science, vol. 2442. Springer, 2002, pp. 61–76.
- [23] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, “Compact E-Cash,” in *Advances in Cryptology - EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 3494. Springer, 2005, pp. 302–321.
- [24] S. S. M. Chow, S.-M. Yiu, and L. C. K. Hui, “Efficient Identity Based Ring Signature,” in *Applied Cryptography and Network Security (ACNS)*, ser. Lecture Notes in Computer Science, vol. 3531, 2005, pp. 499–512.
- [25] S. S. M. Chow, W. Susilo, and T. H. Yuen, “Escrowed Linkability of Ring Signatures and Its Applications,” in *International Conference on Cryptology in Vietnam (VIETCRYPT)*, ser. Lecture Notes in Computer Science, vol. 4341. Springer, 2006, pp. 175–192.
- [26] M. H. Au, W. Susilo, and S.-M. Yiu, “Event-Oriented k -Times Revocable-iff-Linked Group Signatures,” in *Australasian Conference on Information Security and Privacy (ACISP)*, ser. Lecture Notes in Computer Science, vol. 4058. Springer, 2006, pp. 223–234.
- [27] D. Boneh and X. Boyen, “Short Signatures Without Random Oracles,” in *Advances in Cryptology - EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 3027. Springer, 2004, pp. 56–73.
- [28] S. Goldwasser, S. Micali, and C. Rackoff, “The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract),” in *ACM Symposium on Theory of Computing (STOC)*, 1985, pp. 291–304.
- [29] A. Fiat and A. Shamir, “How to Prove Yourself: Practical Solutions to Identification and Signature Problems,” in *Advances in Cryptology - CRYPTO*, ser. Lecture Notes in Computer Science, vol. 263. Springer, 1986, pp. 186–194.
- [30] J. Camenisch and M. Stadler, “Efficient Group Signature Schemes for Large Groups (Extended Abstract),” in *Advances in Cryptology - CRYPTO*, ser. Lecture Notes in Computer Science, vol. 1294. Springer, 1997, pp. 410–424.
- [31] Y. Dodis and A. Yampolskiy, “A Verifiable Random Function with Short Proofs and Keys,” in *Public Key Cryptography (PKC)*, ser. Lecture Notes in Computer Science, vol. 3386. Springer, 2005, pp. 416 – 431.
- [32] T. P. Pedersen, “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing,” in *Advances in Cryptology - CRYPTO*, ser. Lecture Notes in Computer Science, vol. 576. Springer, 1991, pp. 129–140.

- [33] L. Nguyen, “Accumulators from Bilinear Pairings and Applications,” in *The Cryptographers’ Track at the RSA Conference (CT-RSA)*, ser. Lecture Notes in Computer Science, vol. 3376. Springer, 2005, pp. 275–292.
- [34] 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.
- [35] F. Boudot, “Efficient Proofs that a Committed Number Lies in an Interval,” in *Advances in Cryptology - EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 1807. Springer, 2000, pp. 431–444.
- [36] S. S. M. Chow, J. K. Liu, and D. S. Wong, “Robust Receipt-Free Election System with Ballot Secrecy and Verifiability,” in *Network and Distributed System Security (NDSS)*, 2008, pp. 81–94.
- [37] L. D. Nguyen, “Cryptographic Privacy Enhancing Technologies,” Ph.D. dissertation, University of Wollongong, 2005.
- [38] M. Abe, S. S. M. Chow, K. Haralambiev, and M. Ohkubo, “Double-Trapdoor Anonymous Tags for Traceable Signatures,” in *Applied Cryptography and Network Security (ACNS)*, ser. Lecture Notes in Computer Science, vol. 6715, 2011, pp. 183–200.