4-8-2008

PPAA: Peer-to-Peer Anonymous Authentication (Extended Version)

Patrick P. Tsang
Dartmouth College

Sean W. Smith
Dartmouth College

PPAA: Peer-to-Peer Anonymous Authentication (Extended Version)*†

Patrick P. Tsang‡ and Sean W. Smith§

Department of Computer Science
Dartmouth College
Hanover, NH 03755
USA

Dartmouth Computer Science Technical Report
TR2008-615

April 8, 2008

Abstract

In the pursuit of authentication schemes that balance user privacy and accountability, numerous anonymous credential systems have been constructed. However, existing systems assume a client-server architecture in which only the clients, but not the servers, care about their privacy. In peer-to-peer (P2P) systems where both clients and servers are peer users with privacy concerns, no existing system correctly strikes that balance between privacy and accountability.

In this paper, we provide this missing piece: a credential system in which peers are *pseudonymous* to one another (that is, two who interact more than once can recognize each other via pseudonyms) but are otherwise anonymous and unlinkable across different peers. Such a credential system finds applications in, e.g., Vehicular Ad-hoc Networks (VANets) and P2P networks.

We formalize the security requirements of our proposed credential system, provide a construction for it, and prove the security of our construction. Our solution is efficient: its complexities are independent of the number of users in the system.

Keywords: privacy, anonymous authentication, credentials, secret handshakes, VANets, reputation systems

*This work was supported in part by the National Science Foundation under grant CNS-0524695. The views and conclusions do not necessarily represent those of the sponsors.

†This technical report is the extended version of the paper to appear in ACNS '08 under the same title [TS08].

‡patrick@cs.dartmouth.edu

§sws@cs.dartmouth.edu

Contents

1 Introduction

We live in an era where human activities happen electronically more than ever. People rely heavily on computer infrastructures, such as Web applications and peer-to-peer (P2P) networks, to share information, express opinions and trade goods. It is therefore paramount to protect the privacy of the users in these infrastructures by providing them with the option of acting anonymously, unlinkably and/or unobservably.

1.1 Balancing User Privacy and Accountability

It is impractical to pursue user privacy without taking accountability into consideration. Without the fear of being identified, held responsible and punished when they abuse the services, clients are likely to misbehave due to selfishness or malice, thereby disrupting system operations and harming everyone else. Accountability has traditionally been achieved through authentication mechanisms (often followed by access control and/or auditing), which verify the identity of a client who requests a service. In the classic examples of passwords, Kerberos and standard Public Key Infrastructures (PKIs), clients have to give up their privacy to be authenticated.

Anonymous Credential Systems In the pursuit of authentication schemes that balance privacy and accountability, numerous anonymous credential systems [CL01, CL02b], and closely related schemes such as k-times anonymous authentication (k-TAA) [TFS04, TS06], offline anonymous electronic cash (e-cash) systems [Cha82, CHL05] and group signatures [CvH91, ACJT00] have been constructed. An anonymous credential system allows a client to be authenticated by a server as a group member anonymously and unlinkably, and yet the anonymity can be revoked when certain conditions are met. Existing systems differ in their anonymity revocation mechanisms, and hence provide different balancing points between privacy and accountability for different application settings. For example, clients can be identified when they "double-spend" in an e-cash system; their authentications become linkable[1] when they are authenticated more than k times in k-TAA. In group signatures, an authority exists and is capable of arbitrarily revoking anonymity.

1.2 The Challenge: P2P Systems

All anonymous credential systems in existence today assume a client-server architecture in which only the clients, but not the servers, care about their privacy. However, in P2P systems where both clients and servers are peer users with privacy concerns, none of the existing credential systems correctly strike that balance between privacy and accountability.

More specifically, several existing anonymous credential systems provide client accountability by empowering servers to pseudonymize clients who are otherwise anonymous, and servers can thus decide whether and/or how to serve an anonymous client depending the past behavior of the client. In all such systems, however, a client must either (1) present to all servers the very same and hence linkable pseudonym, or (2) learn the identity or at least the pseudonym of a server and then present to that server a pseudonym specific to it. In the former case, client privacy is at risk because colluding servers can link connections from the same client; in the latter, server privacy is at risk because colluding clients can link connections to the same server.

[1] Two authentication runs are linkable (by some entity) *if and only if* it is possible (for that entity) to tell whether or not the two runs are executed by the same client.

We provide below two application scenarios to motivate the user's need for privacy not just as a client, but also as a server, in P2P systems. The opposing requirements of server privacy, client privacy, server accountability and client accountability in these scenarios illustrate the non-triviality of the challenge we overcome in this paper.

Vehicular Ad-Hoc Networks (VANets) To contribute to safer and more efficient roads, vehicles in VANets constantly exchange information such as road and weather conditions among each other and with roadside base stations. Research has shown that the provision of the necessary security and privacy in VANets is critical to the users who rely on these networks [RH07, CPHL07].

To protect the location privacy of the drivers when information is exchanged on the road between two vehicles, both vehicles should remain anonymous among all the vehicles in communication range. Furthermore, no one should be able to link reports by the same vehicle to different other vehicles or roadside base stations. This helps prevent a vehicle from being not only pseudonymized and thus tracked, but also deanonymized through drawing inferences from multiple reports made by the vehicle [Kru07].

From the accountability perspective, to distinguish legitimate data from rogue data, vehicles must be authenticated when reporting sensor readings. Moreover, so that repetitive reporting of the same information can be detected, vehicles should be pseudonymous to one another (that is, vehicle X can recognize some vehicle Y reporting again, without knowing anything else about Y). For instance, in VANets in which vehicles decide when to accelerate and break based on reports collected from the network, the failure to achieve these security goals can allow an attacker to paralyze traffic and/or induce accidents.

Reputation Systems for P2P Networks The existence of selfish users in P2P networks such as those for file sharing severely degrades system performance. Adversaries can reduce the availability of specific items in P2P networks by "poisoning" [CWC05] them, i.e., injecting lots of decoys into the network. Reputation systems provide a game-theoretic solution to these problems by introducing incentives for users to behave well. Unfortunately, reputation systems lacking privacy can also introduce disincentives to good behavior: if a reputation system reveals the pseudonym or even the identity of the serving peers, peers might refuse to serve others so as to stay anonymous.

A privacy-preserving reputation system for P2P networks where there is no (trusted) central server should have the following properties: users are pseudonymous to one another, so that a user Carol can decide whether to serve (or be served by) another user Dave based on her past experience with Dave, without knowing his actual identity. However, assuming the registration procedures make sure that users in the system can have at most one single membership, Dave shouldn't be able to start fresh after having established a bad reputation with respect to Carol, nor can he impersonate Carol for her high reputation, potentially even spoiling her reputation through misbehavior. Finally, connections between a peer Carol and different other peers should be unlinkable, as otherwise it might be possible for someone to trace Carol by studying those connections.

1.3 Our Contributions

In this paper, we overcome the challenge posed above by proposing the concept—and giving a construction and implementation—of *Peer-to-Peer Anonymous Authentication*, or PPAA for short,

a credential system in which peers are pseudonymous to individual peers but unlinkable across different peers. More specifically, we make the following contributions:

- We rigorously define the operations of PPAA and its security and privacy requirements, during which we introduce the notion of the *Linkability Context* of an authentication scheme as a tool for a more precise reasoning about the linkability property of an authentication scheme. We also formalize the threat model in which those security requirements must be satisfied.
- We provide the first construction for PPAA. Our construction is both secure and efficient. In particular, its complexities are independent of the number of users in the system. We also report empirical performance figures of a software implementation of our construction.

Paper Organization We review the related works in Section 2 and give an overview of our solution in Section 3. Section 4 covers the preliminary materials. In Section 5, we define the security model. We present our solution and analyze its security and efficiency in Section 6. We provide some discussions in Section 7 and conclude the paper in Section 8.

2 Related Works

We review the literature for related works, and argue why they fail to solve the problem posed in this paper. We make occasional but otherwise minimal use of mathematical notation without definition for the sake of conciseness.

***k*-Times Anonymous Authentication** *k*-TAA [TFS04, NSN05, TS06, CHK+06] and related schemes such as event-oriented linkable group/ring signatures [TWC+04, ASY06] are close candidates in overcoming the posed challenge. In essence, when a client Alice in these schemes is being authenticated by a server Bob, she provides Bob with a tag and convinces him that the tag is correctly formed. Bob can test if two authentications are linked to the same client by examining the associated tags.

These schemes do not solve the posed problem since authentication runs by the same user to different servers are linkable. This is because a user always uses the same tag when being authenticated by any server. More specifically, the tag of client i with secret x_i has the form of $t_i = g^{x_i}$ for some global parameter g.

We point out that, while they do not address server privacy, various *k*-TAA schemes and anonymous credential systems do provide several major ingredients for the solution we propose in this paper. For example, our proof system for group membership uses ideas from Camenisch and Lysyanskaya [CL02b] and Boneh et al. [BBS04]. Also, the concept of event identifiers in this paper stems from several other existing schemes [CL01, TFS04, TWC+04].

Secret Handshakes Secret handshake schemes (SHSs) [BDS+03, CJT04, XY04, ABK07] allow any two members of the same group to authenticate each other as a group member and share a session key without revealing their group affiliations to outsiders.

In the scheme due to Xu and Yung [XY04], secret handshakes are anonymous and unlinkable, but members are limited to shaking hands no more than some predefined number of times. The

state-of-the-art construction [ABK07] provides anonymity and unlinkability without such a limitation. Recently, Tsudik and Xu [TX06] extended secret handshakes into a multi-party and privacy-conserving setting: two or more group members can anonymously and unlinkably authenticate each other such that one's group affiliation is not revealed unless every other party's membership is ensured.

All anonymous secret handshakes proposed so far [XY04, ABK07, TX06] fail to solve the posed problem. As handshakes are unlinkable, a client Alice has no way to tell if the one she is shaking hands with is the same as the one behind some earlier handshakes. As a remedy, Alice may ask the person behind the handshake to reveal a secret, e.g., a random nonce, that she leaked in their last handshake. Unfortunately, this is problematic because the person does not know which secret to reveal as Alice is anonymous. Also, one could pretend to be new by "forgetting" the secrets.

3 Our Approach

In this section, we provide an overview of our approach to solve the posed challenge.

3.1 Putting Authentication Schemes into "Linkability Context"

We first introduce the notion of the *linkability context* in authentication.

Definition 1 (Linkability Context) The *Linkability Context*, or LC for short, of an authentication scheme is a collection of attributes that determines the linkability of authentication runs in the scheme. In particular, two authentication runs are *linkable if and only if* the two runs are executed when the attributes in the linkability context are all in the same condition. □

In k-TAA, for instance, authentication runs by the same client at the same "time" are linkable, while runs by the same client at different times, as well as those by different clients at the same time, are not linkable. The linkability context of k-TAA is thus $LC = \{\texttt{client-ID}, \texttt{time}\}$, i.e., the collection of client identity and time.

Understanding the precise linkability context of an authentication scheme helps reason about the privacy guarantees and hence implications of the scheme. At one end of the spectrum of client privacy, in conventional authentication schemes such as those using digital signatures, any two authentication runs are linkable. The linkability context of these schemes therefore consists of nothing, i.e., $LC = \emptyset$. At the other end of the spectrum, there are schemes such as ring authentication [RST01, DKNS04] in which no two authentication runs are linkable. In this case, the linkability context is the authentication run instance, i.e., $LC = \{\texttt{authen-run-ID}\}$.

Linkability Context in PPAA A correct choice of its linkability context is the first step towards a secure PPAA construction. In our design, the linkability context in PPAA is the collection of the *unordered* pair of client and server identity, and the event for which the PPAA authentication is executed, i.e.,

$$LC = \{\{\texttt{client-ID}, \texttt{server-ID}\}, \texttt{event-ID}\}.$$

In other words, we would like to design PPAA in such a way that authentication runs are linkable *if and only if* they are executed between the same pair of peers for the same event. In the example of VANets, if one sets the event to be *"speed on Highway I-89 on June 3rd, 2008,"* then only those PPAA-authenticated speed report made by the same vehicle to the same road-side base station on Highway I-89 on June 3rd, 2008 are linkable.

3.2 Key Ideas in Our PPAA Design

An Observation It should have become clear now that event-oriented linkable group/ring signatures and k-TAA fail as a secure PPAA construction because server identity is not in their linkability context. It would seem that one could bring server identity into the linkability context in an event-oriented linkable group/ring signature (resp. k-TAA) by mapping an event in (resp. one "time") into the identity of a server. Consequently, LC becomes $\{\texttt{client-ID}, \texttt{server-ID}\}$ and authentication runs by the same user to different servers become unlinkable. More specifically, the tag of client i with secret x_i with respect to server j has the form of $t_{i,j} = g_j^{x_i}$, where g_j is a server-specific parameter. Tags of the same client with respect to different servers are now unlinkable thanks to the underlying intractability assumption (the Decisional Diffie-Hellman assumption).

Unfortunately, to produce a tag and prove its correctness during an authentication run in the above modified scheme, a client must now ask the server for its g_j, which can be considered its pseudonym. Even if there existed a way in which a client could compute a tag for the server without knowing the pseudonym of the server, two colluding users can easily determine if they are being authenticated by the same server. In other words, using the tag design in event-oriented linkable group/ring signatures and k-TAA, it is impossible to devise a secure authentication scheme with $LC = \{\texttt{client-ID}, \texttt{server-ID}\}$.

The Need of a Novel Tag Construct As a result, constructing a secure PPAA requires a new tag design that possesses novel features:

- Tags must be dependent on the identity of the client, the server, and the event.
- Tags are linked *if and only if* they are produced by the same (unordered) pair of peers, and during the same event.
- Peers must be able to produce tags and prove their correctness in zero-knowledge through interacting with the other peers and without knowing the identity of the other peers.

In Section 6, we present such a tag design and how we use it to construct a secure PPAA.

4 Preliminaries

We provide the technical background necessary for understanding the rest of this paper.

Notations A function $f(\lambda)$ is negligible if for all polynomial $p(\lambda)$, $f(\lambda) < 1/p(\lambda)$ holds for all sufficiently large λ. A function is non-negligible if it is not negligible. The probability $\Pr[E]$ of an event E is overwhelming (in some parameter λ) if $1 - \Pr[E]$ is negligible (in λ).

Let λ be a sufficiently large security parameter. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups of prime order p with $|p| = \lambda$ such that group operation is efficiently computable. Let g_0 and h_0 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively such that there is an efficiently computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\psi(h_0) = g_0$.

We say that $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair if there exists an efficiently computable map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_T$ is also a cyclic group of prime order p, such that: $\hat{e}(A^x, B^y) = \hat{e}(A, B)^{xy}$ for all $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_p$, and $\hat{e}(g_0, h_0) \neq 1$.

Complexity Assumptions The security of our solution to be presented later in this paper relies on the validity of the DDH assumption in $\mathbb{G}_1$ and the q-SDH assumption on bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, which we define as the following.

- *The Decisional Diffie-Hellman (DDH) problem* in $\mathbb{G}_1$: On input of a quadruple $(g_0, g_0^a, g_0^b, g_0^c) \in \mathbb{G}_1^4$, where $a, b \in_R \mathbb{Z}_p$, and $c = ab$ or $c \in_R \mathbb{Z}_q$ equally likely, output 1 if $c = ab$ and 0 otherwise. We say that *the DDH assumption* in $\mathbb{G}_1$ holds if no probabilistic polynomial time (PPT) algorithm has non-negligible advantage over random guessing in solving the DDH problem in $\mathbb{G}_1$.

- *The q-Strong Diffie-Hellman (q-SDH) problem* in $(\mathbb{G}_1, \mathbb{G}_2)$: On input of a $(q+2)$-tuple $(g_0, h_0, h_0^x, h_0^{x^2}, \cdots, h_0^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, where $x \in_R \mathbb{Z}_p$, output a pair $(A, c) \in \mathbb{G}_1 \times \mathbb{Z}_p$ such that $A^{(x+c)} = g_0$. We say that *the q-SDH assumption* in $(\mathbb{G}_1, \mathbb{G}_2)$ holds 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 was introduced and proven to hold in generic groups [Sho97] by Boneh and Boyen [BB04]. The DDH assumption in $\mathbb{G}_1$ is the also known as the *eXternal Diffie-Hellman (XDH)* assumption in $(\mathbb{G}_1, \mathbb{G}_2)$ [CHL05, BBS04]. The validity of the XDH assumption implies that ψ is computationally one-way. The assumption is known is be false on supersingular curves [GR04], but is conjectured to hold for the Weil or Tate pairing on MNT curves with embedded degree greater than 1 and $\mathbb{G}_1$ defined over the ground field [BBS04].

Proofs of Knowledge In a *Zero-Knowledge Proof-of-Knowledge (ZKPoK)* protocol [GMR89], a prover convinces a verifier that some statement is true without the verifier learning anything except the validity of the statement. Σ-protocols are a special type of three-move ZKPoK protocols. They can be converted into non-interactive *Signature Proof of Knowledge (SPK)* schemes that are secure in the *Random Oracle (RO)* Model [BR93] (in the sense of Indistinguishability against chosen-message attacks, or IND-CMA [GMR88]).

In many anonymous credential systems, a client uses an SPK scheme to prove in zero-knowledge to a server her possession of a credential issued by the Group Manager when being authenticated by a server. The SPK schemes differ in these systems, which accounts for the differences in privacy and accountability guarantees and complexity assumptions. The SPK schemes we will use in our solution are based on the ZKPoK protocol due to Boneh and Boyen [BBS04].

We follow the notation introduced by Camenisch and Stadler [CS97] for the various ZKPoK protocols. For example, $PK\{(x) : y = g^x\}$ denotes a ZKPoK protocol that proves the knowledge of an integer x such that $y = g^x$ holds, where y and g are elements of some group $G = \langle g \rangle$. Using this notation, a ZKPoK protocol can be described by just pointing out its aim while hiding all the details. Moreover, we denote by $SPK\{(x) : y = g^x\}(M)$ the SPK scheme converted from the above ZKPoK protocol.

5 Model

This section formalizes PPAA. The entities involved in PPAA are the Group Manager (GM) and a set of peer users, or simply peers. The GM is responsible for registering peers. A peer can be a client, a server, or both. Clients are interested in accessing services provided by servers and servers are willing to serve the clients, as long as their privacy and accountability requirements are satisfied.

5.1 System Operations

Operations that take place in PPAA include the GM setting up the system (Setup) and registering peers into the system (Registration), and peers authenticating one another (Authentication) and testing if two authentication runs are linked (Linking). We highlight that only Setup and Registration involve a centralized authority, namely the GM; Authentication requires no centralized authority, which is a crucial property necessary for PPAA to be applicable to P2P systems with scalability.

The syntax for these operations are given as follows.

- Setup is a *Probabilistic Poly-Time (PPT)* algorithm invoked by the GM. On input a sufficiently large security parameter λ, the algorithm outputs GM's secret key `gsk` and the group public key `gpk`. The GM stores `gsk` privately and publishes `gpk` to the public. `gpk` is an implicit input to all the algorithms below.

- Registration is a two-party multi-round protocol between the $\mathsf{Register}^P$ PPT algorithm invoked by a peer and the $\mathsf{Register}^{GM}$ PPT algorithm invoked by the GM. The additional input to $\mathsf{Register}^{GM}$ is the GM's secret key `gsk`. Upon successful termination of a protocol run, $\mathsf{Register}^P$ outputs a credential, which the peer stores privately, and by doing so becomes a registered peer in the system.

- Authentication is a two-party multi-round protocol between the $\mathsf{Authenticate}^I$ PPT algorithm invoked by a registered peer Alice (as the *Initiator*, i.e. the one who initiates the protocol) and the $\mathsf{Authenticate}^R$ PPT algorithm invoked by another registered peer Bob (as the *Responder*). The common input to both parties is an event identifier `eid` upon which they have already agreed.[2] The additional inputs to $\mathsf{Authenticate}^I$ and $\mathsf{Authenticate}^R$ are Alice's credential and Bob's credential, respectively.

  A protocol run terminates successfully *if and only if* both algorithms output a tag, in which case we say that the authentication is *successful* and that Alice and Bob are mutually authenticated with one another, during an event with identifier `eid`. When we say that a peer Carol is involved in an authentication without specifying her role, then Carol can be either the initiator or the responder in that authentication.

- Linking is a (possibly probabilistic) poly-time algorithm any peer can invoke. On input two tags $\mathtt{tag}_1$ and $\mathtt{tag}_2$, the algorithm outputs a boolean value of either `linked` or `not-linked`.

  In the former (resp. the latter) case, the two tags, and also the two successful authentication runs from which the tags are resulted, are said to be *linked* (resp. *not linked*).

  Semantically, a peer Carol uses this algorithm to pseudonymize other peers with which she has mutually authenticated: for any two successful authentication runs during the same event, she thinks she is mutually authenticating with the same peer *if and only if* the two authentication runs are linked.

Any construction of PPAA must be correct:

Definition 2 (Correctness) An PPAA construction is correct if it has *authentication correctness* and *linking correctness*:

[2] In the VANet example given in Section 3.2, the `eid` can be `20080603||I-89||speed`.

- *Authentication Correctness.* If all entities in PPAA are honest (i.e. they all follow the system's specification), then, with overwhelming probability, any authentication between any two registered peers is successful.
- *Linking Correctness.* If all entities in PPAA are honest, then, with overwhelming probability, in any two successful authentication involving any registered peer Carol, the two tags output by Carol are linked *if and only if*, in those two authentications, both the event identifiers and the other peers involved are identical. □

5.2 Security Requirements

Roughly speaking, a PPAA construction is secure if it satisfies the following security requirements. A formal definition can be found in Appendix A.

Mis-authentication Resistance Mis-authentication occurs when two peers successfully complete mutual authentication, but only one of them is an honest and registered peer. A secure PPAA construction must be resistant to mis-authentication.

For example, this property prevents vehicles in VANets from believing (malicious) data from rogue sensors.

Peer Accountability To subvert peer accountability, a coalition of $n \geq 1$ registered but malicious peer(s) attempts to run more than n successful mutual authentication involving the same honest peer Carol during the same event such that the tags Carol outputs in those authentication are all pairwise unlinked. A secure PPAA construction requires that no adversary can succeed in such an attempt.

In the example of P2P networks, this prevents a peer from starting fresh after having established a bad reputation with respect to another peer.[3]

Peer Privacy To subvert the privacy of an honest peer Carol involved in an authentication potentially executed with a malicious peer, the adversary, potentially with the GM's help, attempts to:

- deanonymize Carol in individual protocol runs, and/or
- pseudonymize Carol in protocol runs with different peers and/or during different events.

A secure PPAA construction requires that no adversary can succeed in any of the above attempts.

As an example, this ensures that communications of a vehicle in VANets with different other vehicles or roadside base stations cannot be linked.

Framing Resistance An honest peer Carol is *framed* when another honest peer Dave thinks that he is mutually authenticating with the same peer in two successful authentication runs, even though Carol is involved in exactly one of them. A secure PPAA construction requires that no adversary, even with the help of the GM, can frame an honest peer.

In the example of P2P reputation systems, this makes sure that peers can't impersonate other peers with high reputation.

[3]This assumes that a peer can't register more than once. We will discuss this issue further in Section 7.

6 Our Solution

We begin this section with a presentation of our first attempt to construct PPAA, which, although insecure by itself, illustrates our tag design as the core component of our full and secure PPAA construction. Then we proceed to present our actual PPAA construction. In Appendix B, we discuss our implementation of the construction and its empirical performance evaluation.

6.1 Our First Attempt

We call our first attempt Basic-PPAA.

Parameters Let $\mathbb{G}_1$ be a group as described in Section 4 in which the DDH assumption holds. Let $H : \{0,1\}^* \to \mathbb{G}_1$ be a secure cryptographic hash function. Event identifiers are strings of any length.

Credentials Each user is given by the GM one credential $\texttt{cred}$ in the form $\texttt{cred} = (A, x, y) \in \mathbb{G}_1 \times \mathbb{Z}_p^2$, where $x, y \in_R \mathbb{Z}_p$ and A is distinct in all credentials.

Tags In Basic-PPAA, a tag is the output of a function f that takes as inputs the credential of an initiating peer $\texttt{cred}_1 = (A_1, x_1, y_1)$, the credential of a responding peer $\texttt{cred}_2 = (A_2, x_2, y_2)$ and an event identifier $\texttt{eid}$. The function is defined as follows:

$$f : (\texttt{cred}_1, \texttt{cred}_2, \texttt{eid}) \mapsto \texttt{tag} \doteq \{\tau_1, \tau_2\}, \text{ where } \begin{cases} \tau_1 &= A_1^{x_2} H(\texttt{eid})^{y_1}, \\ \tau_2 &= A_2^{x_1} H(\texttt{eid})^{y_2}. \end{cases}$$

Thus, a tag is a set of two $\mathbb{G}_1$ elements.

The Skeleton Protocol The following steps describe a protocol run between an initiating peer Alice with credential $\texttt{cred}_1 = (A_1, x_1, y_1)$ and a responding peer Bob with credential $\texttt{cred}_2 = (A_2, x_2, y_2)$ during an event with identifier $\texttt{eid}$. When the protocol terminates, Alice and Bob output a tag.

1. Alice $\to$ Bob: $\langle U_1, V_1 \rangle = \langle A_1^{r_1}, H(\texttt{eid})^{r_1} \rangle$, where $r_1 \in_R \mathbb{Z}_p$.
2. Bob $\to$ Alice: $\langle U_2, V_2, W_2 \rangle = \langle A_2^{r_2}, H(\texttt{eid})^{r_2}, U_1^{x_2} V_1^{y_2} \rangle$, where $r_2 \in_R \mathbb{Z}_p$.
3. Alice $\to$ Bob: $\langle W_1, \tau_1 \rangle = \langle U_2^{x_1} V_2^{y_1}, W_2^{1/r_1} \rangle$.
4. Bob $\to$ Alice: $\langle \tau_2 \rangle = \langle W_1^{1/r_2} \rangle$.
5. Alice and Bob both output $\texttt{tag} = \{\tau_1, \tau_2\} = f(\texttt{cred}_1, \texttt{cred}_2, \texttt{eid})$ and terminate.

Properties The tags and the skeleton protocol given above have the following desirable properties:

1. Two tags $\texttt{tag} = f(\texttt{cred}_1, \texttt{cred}_2, \texttt{eid})$ and $\texttt{tag}' = f(\texttt{cred}_1', \texttt{cred}_2', \texttt{eid}')$ are the same *if and only if* $\{\texttt{cred}_1, \texttt{cred}_2\} = \{\texttt{cred}_1', \texttt{cred}_2'\}$ and $\texttt{eid} = \texttt{eid}'$, with overwhelming probability.

2. The protocol view of Alice $\langle \texttt{cred}_1, \texttt{eid}, r_1, U_2, V_2, W_2, \tau_2 \rangle$ can be simulated (computationally indistinguishably) by Alice if she is given `tag`. In other words, Alice learns no knowledge other than `tag` from running the skeleton protocol. Similarly, Bob learns no knowledge other than `tag` from running the skeleton protocol.

3. The tag produced by a peer Alice for another peer Bob during an event is indistinguishable from the tag produced by any peer for Bob during a different event; it is also indistinguishable from the tag produced by Alice for a different peer during the same event.

The validity of these properties are straightforward provided that the DDH assumption in $\mathbb{G}_1$ holds. We thus omit the proof.

Remark If not all entities are honest, Basic-PPAA results in an insecure PPAA construction. For instance, users can be authenticated without asking the GM for a credential, dishonest users may use an arbitrary credential instead of the one given by the GM to get away from being linked and a malicious GM can frame clients.

6.2 Our PPAA Construction

We now enumerate our PPAA construction. It can be thought of as the result of securing Basic-PPAA by adding to it all necessary mechanisms to force the entities to behave honestly, such as by accompanying each step in the skeleton protocol with a SPK scheme that proves the correctness of the step.

Parameters In addition to those in Basic-PPAA, our PPAA construction has the following parameters. Let $\mathbb{G}_2$ be a group as described in Section 4 such that $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair in which the q-SDH assumption holds. Let ℓ be a sufficiently large security parameter of size polynomial in λ. Let $g_1, \ldots, g_5 \in \mathbb{G}_1$ be generators of $\mathbb{G}_1$ such that the relative discrete logarithms among $g_1, \ldots, g_5$ and g_0 (from Section 4) are unknown. Let $\hat{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be a secure cryptographic hash function. $\hat{H}$ is utilized by the various SPKs in the construction.

Setup The GM randomly chooses $\gamma \in_R \mathbb{Z}_p$ and computes $w = h_0^{\gamma} \in \mathbb{G}_2$. The group secret key is $\texttt{gsk} = (\gamma)$ and the group public key is $\texttt{gpk} = (w)$.

Registration At the successful termination of a run of this protocol between a user Alice and the GM, Alice obtains a credential `cred` in the form of $\texttt{cred} = (A, e, x, y, z) \in \mathbb{G}_1 \times \mathbb{Z}_p^4$ such that $A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z$. The private input to the GM is his group secret key `gsk`. The protocol proceeds as follows.

1. The GM sends $\langle N_0 \rangle$ to Alice, where $N_0 \in_R \{0,1\}^{\ell}$ is a random challenge.

2. Alice sends $\langle C, \Pi_0 \rangle$ to the GM, where $C = g_1^x g_2^y g_3^{z'} \in \mathbb{G}_1$ is a commitment of $(x, y, z') \in_R \mathbb{Z}_p^3$ and Π_0 is a signature proof of knowledge of

$$SPK\left\{(x, y, z') : C = g_1^x g_2^y g_3^{z'}\right\}(M)$$

on message $M = N_0 || C$, which proves the correctness of C. We will refer to the above SPK as SPK_0.

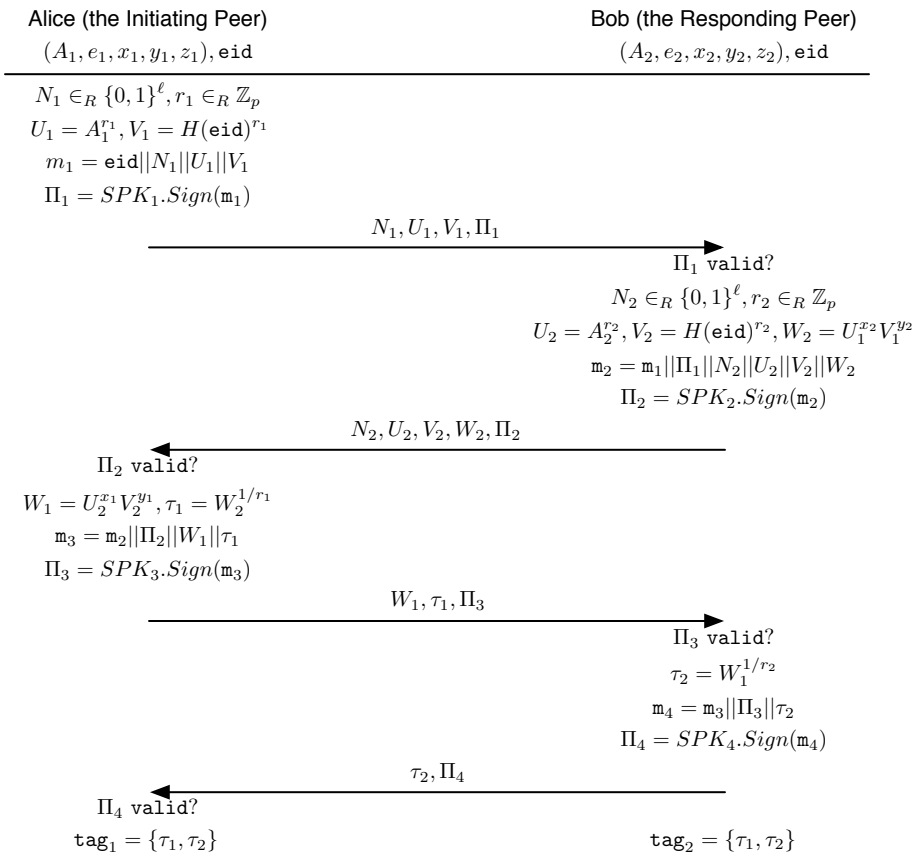

Figure 1: The Authentication Protocol

3. The GM terminates with `failure` if the verification of Π_0 returns `invalid`. Otherwise the GM sends $\langle A, e, z'' \rangle$ to Alice, where $e, z'' \in_R \mathbb{Z}_p$ and

$$A = (g_0 C g_3^{z''})^{\frac{1}{e+\gamma}} \in \mathbb{G}_1$$

4. Alice computes $z = z' + z''$. She terminates with `failure` if $\hat{e}(A, wh_0^e) \neq \hat{e}(g_0 g_1^x g_2^y g_3^z, h_0)$. Otherwise she outputs $\texttt{cred} = (A, e, x, y, z)$ as her credential.

We remark that the security of the system requires that no two instances of the Registration protocol may run concurrently. To enforce this rule, the GM registers users one after the other.

Authentication Alice (as the initiator) and Bob (as the responder) would like to mutually authenticate with each other during an event with identifier $\texttt{eid} \in \{0,1\}^*$. The common input to both Alice and Bob is `eid`. The private input to Alice and Bob is their own credentials $(A_1, e_1, x_1, y_1, z_1)$ and $(A_2, e_2, x_2, y_2, z_2)$ respectively. The following describes the steps in the 4-round protocol for authentication.

1. Alice sends $\langle N_1, U_1, V_1, \Pi_1 \rangle$ to Bob, where:

- $N_1 \in_R \{0,1\}^\ell$, $r_1 \in_R \mathbb{Z}_p$,
- $U_1 = A_1^{r_1} \in \mathbb{G}_1$, $V_1 = H(\texttt{eid})^{r_1} \in \mathbb{G}_1$, and
- Π_1 is a signature proof of knowledge of

$$SPK\left\{(A,e,x,y,z,r): \begin{array}{c} A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z \quad \wedge \\ U_1 = A^r \ \wedge \ V_1 = H(\texttt{eid})^r \end{array}\right\}(M)$$

on message $M = \mathtt{m}_1 = \texttt{eid}||N_1||U_1||V_1 \in \{0,1\}^*$, which Alice can produce using her knowledge of $(A_1, e_1, x_1, y_1, z_1, r_1)$. We will refer to the above SPK as SPK_1.

2. Bob terminates with `failure` if verification of Π_1 returns `invalid`. Otherwise he sends $\langle N_2, U_2, V_2, W_2, \Pi_2\rangle$ to Alice, where:

   - $N_2 \in_R \{0,1\}^\ell$, $r_2 \in_R \mathbb{Z}_p$,
   - $U_2 = A_2^{r_2} \in \mathbb{G}_1$, $V_2 = H(\texttt{eid})^{r_2} \in \mathbb{G}_1$, $W_2 = U_1^{x_2} V_1^{y_2}$, and
   - Π_2 is a signature proof of knowledge of

$$SPK\left\{(A,e,x,y,z,r): \begin{array}{c} A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z \quad \wedge \\ V_2 = H(\texttt{eid})^r \ \wedge \ U_2 = A^r \quad \wedge \\ W_2 = U_1^x V_1^y \end{array}\right\}(M)$$

   on message $M = \mathtt{m}_2 = \mathtt{m}_1||\Pi_1||N_2||U_2||V_2||W_2 \in \{0,1\}^*$, which Bob can produce using his knowledge of $(A_2, e_2, x_2, y_2, z_2, r_2)$. We will refer to the above SPK as SPK_2.

3. Alice terminates with `failure` if verification of Π_2 returns `invalid`. Otherwise she sends $\langle W_1, \tau_1, \Pi_3\rangle$ to Bob, where:

   - $W_1 = U_2^{x_1} V_2^{y_1} \in \mathbb{G}_1$, $\tau_1 = W_2^{1/r_1} \in \mathbb{G}_1$, and
   - Π_3 is a signature proof of knowledge of

$$SPK\left\{(A,e,x,y,z,r): \begin{array}{c} A^{e+\gamma} = g_0 g_1^x g_2^y g_3^z \quad \wedge \\ U_1 = A^r \ \wedge \ V_1 = H(\texttt{eid})^r \quad \wedge \\ W_1 = U_2^x V_2^y \ \wedge \ W_2 = \tau_1^r \end{array}\right\}(M)$$

   on message $M = \mathtt{m}_3 = \mathtt{m}_2||\Pi_2||W_1||\tau_1 \in \{0,1\}^*$, which Alice can produce using her knowledge of $(A_1, e_1, x_1, y_1, z_1, r_1)$. We will refer to the above SPK as SPK_3.

4. Bob terminates with `failure` if verification of Π_A returns `invalid`. Otherwise he sends $\langle \tau_2, \Pi_4\rangle$ to Alice, where:

   - $\tau_2 = W_1^{1/r_2}$, and
   - Π_4 is a signature proof of knowledge of

$$SPK\left\{(r): W_1 = \tau_2^r \ \wedge \ V_2 = H(\texttt{eid})^r\right\}(M)$$

   on message $M = \mathtt{m}_4 = \mathtt{m}_3||\Pi_3||\tau_2 \in \{0,1\}^*$, which Bob can produce using his knowledge of (r_2). We will refer to the above SPK as SPK_4.

   Bob outputs $\mathtt{tag}_2 = \{\tau_1, \tau_2\}$ and terminates.

5. Alice terminates with `failure` if verification of Π_4 returns `invalid`. Otherwise she outputs $\mathtt{tag}_1 = \{\tau_1, \tau_2\}$ and terminates.

Figure 1 is a diagrammatic representation of the protocol.

	Number of Operations (without precomputation)		
	$\mathbb{G}_1$ multi-EXPs	$\mathbb{G}_T$ multi-EXPs	Pairings
Alice (the Initiator)	12 (28)	4 (10)	2 (4)
Bob (the Responder)	16 (26)	8 (11)	4 (5)

Table 1: Timing complexity of the Authentication protocol

Linking On input two tags $\texttt{tag}_1, \texttt{tag}_2$, this algorithm returns `linked` if they are equal and `not-linked` otherwise.

6.3 SPK Instantiation

The instantiation of SPK_0 to SPK_4 and their computational costs in terms of the number of pairing computation and multi-exponentiations (multi-EXPs)[4] can be found in Appendix C.

6.4 Analysis

Our PPAA construction has correctness, which is a straightforward consequence of the correctness of the skeleton protocol and the correctness of the various SPK schemes. We omit the proof for conciseness.

Security The security of our construction hinges on the correctness of the skeleton protocol and the security properties of the various SPK schemes surrounding it. We now state the following theorem. Its proof is sketched in Appendix D.

Theorem 1 (Security) Our proposed PPAA construction is secure in the random oracle model if the XDH assumption and the q-SDH assumption hold in $(\mathbb{G}_1, \mathbb{G}_2)$. □

Complexities Our solution scales extremely well: all operations have constant computational and communication complexities, regardless on the number of peers, events and authentication runs. Registration is a one-time process per user in the system. Linking involves only an equality testing of two sets of two $\mathbb{G}_1$ elements.

Authentication is the dominating operation, thus we provide a more detailed analysis on its costs. Alice, the initiating peer, needs to do an SPK_1 and an SPK_3 signing, and an SPK_2 and an SPK_4 verification. The number of $\mathbb{G}_1$ multi-EXPs, $\mathbb{G}_T$ multi-EXPs and pairings are 24, 10 and 4 respectively. Some of these operations can be precomputed before the the start of an authentication; with precomputation, those numbers become 10, 4 and 2 respectively. Bob, the responding peer, needs to an SPK_2 and an SPK_4 signing, and an SPK_1 and an SPK_3 verification. The number of $\mathbb{G}_1$ multi-EXPs, $\mathbb{G}_T$ multi-EXPs and pairings are 22, 11 and 5 respectively. With precomputation, they become 14, 8 and 4. In addition to these calculation, Alice and Bob also need to compute several $\mathbb{G}_1$ multi-EXPs during the protocol.

Table 1 summarizes the computational costs for Alice and Bob.

[4] A multi-EXP computes the product of exponentiations faster than performing the exponentiations separately. We assume that one multi-EXP operation multiplies up to 3 exponentiations.

7 Discussion

Resilience to Sybil Attacks Sybil attacks [Dou02] are attacks during which an individual entity masquerades as multiple simultaneous identities. Any authentication mechanisms including PPAA must defend Sybil attacks launched against user registration. Approaches exist to ensure that only legitimate users can register and that no legitimate user can register more than once. They include *trusted certification* such as X.509 [AF99], *resource testing*, where resources could be IP addresses or "friendship" in social networks or PGP-like web of trust, *recurring costs* imposed by cryptographic puzzles or CAPTCHAs [vABHL03], and *trusted devices* with certain degree of tamper-resistance, such as Trusted Platform Modules (TPMs) [TPM07].

The practicality of the above approaches depends on the application scenarios. In the example of VANets, the Department of Motor Vehicles (DMV) can play the role of the GM with little overhead. Additionally, the makers of the vehicles can install a trusted device preloaded with a credential in each of vehicle they manufacture. In the example of P2P systems over a public network such the Internet, demonstrating the possession of IP addresses is a pragmatic and thus more popular approach, even though it does not have the highest resilience to Sybil attacks.

Revocation Any practical authentication mechanism must allow for credential revocation. In the settings of PPAA, one might want to revoke a credential because the peer user in possession of that credential is compromised or misbehaving. For example, in VANets, the credential issued to a vehicle should be revoked when the vehicle is reported to have been stolen. Revocation allows for easier identification and thus tracking of stolen vehicles while maintaining the privacy of other vehicles as stolen cars with revoked credentials can no longer be anonymously authenticated by, e.g. a highway toll booth.

Our construction of PPAA can be modified in a straightforward manner to allow for credential revocation by adopting existing standard techniques [CL02a, BS04]: Alice and Bob verifiably encrypt part of their credentials during SPK_1 and SPK_2 respectively during the authentication under the public key of an entity usually referred to as the Revocation Manager. Now in addition to the original authentication, Alice and Bob have to convince one another that they have not been revoked. In the approach of verifier-local revocation [BS04], each user keeps a list of revoked users; in the approach of dynamic accumulators [CL02a], each non-revoked user updates their credential when someone else's has been revoked.

Authenticated Key-exchange The authentication protocol in PPAA can be easily turned into an authenticated Diffie-Hellman key-exchange. Specifically, Alice additionally includes in $\mathtt{m}_1$ an element g_0^a with $a \in_R \mathbb{Z}_p$ in Step 1 of the authentication protocol, while Bob additionally includes in $\mathtt{m}_2$ an element g_0^b with $b \in_R \mathbb{Z}_p$ in Step 3. When the protocol terminates, both of them can derive a shared session key as $g_0^{ab} = (g_0^a)^b = (g_0^b)^a$. Since $\mathtt{m}_1$ and $\mathtt{m}_2$ are signed with SPK_1 and SPK_2 respectively, Alice and Bob can use the session key to establish a confidential channel with the same privacy and accountability guarantees as in PPAA.

Blending Secret handshakes into PPAA As discussed, anonymous SHSs such as Ateneise et al.'s [ABK07] do not provide the linkability desired by the servers. On the other hand, PPAA leaks the initiating peer's group affiliation to any responding peer who might not be a group member. Hence, each of them has its advantage over the other. Fortunately, one can enjoy the advantages of

both by composing the two schemes. Specifically, two group members first execute an anonymous secret handshake to authenticate the group membership of one another and establish a secure channel, then they execute an PPAA authentication within that channel.

Furthermore, carrying out PPAA authentication within a secure channel has the additional benefit of preventing eavesdroppers from linking authentication traffic.

Fairness In our PPAA construction, a malicious responding peer Bob might decide to stop after receiving Alice's protocol message at step 3 of the authentication protocol so that he could learn Alice's tag without Alice being able to learn his. The revealing of tags between Alice and Bob is thus not guaranteed to be fair in our construction.

Borrowing ideas from optimistic fair exchange [ASW97, ASW98], one could augment fairness to PPAA by modifying it as follows. Alice requires Bob to additionally send a verifiable encryption of r_2 under the public key of some Trusted Third Party (TTP) in step 2 also that in case Bob stops before step 4, Alice can still reconstruct the tag with the help of the TTP. However, such a modification puts Bob's privacy at risk, as the collusion between Alice the TTP can identity Bob. We leave the exploration of how to provide fairness without sacrificing privacy as future work.

8 Conclusion

In this paper, we have introduced *Peer-to-Peer Anonymous Authentication* (PPAA), a credential system that correctly balances user privacy and accountability in P2P systems where not just clients but also servers are concerned with their privacy. We have shown that such a credential system finds applications in many P2P systems such as VANets. We have presented the first PPAA construction, which is both secure and very efficient.

Acknowledgments

The authors would like to thank Man Ho Au for his suggestion on an initial tag design, the anonymous reviewers for their insightful reviews, and the members in the Security Reading Group at Dartmouth College (SRG@Dartmouth)[5] who discussed this paper for their helpful feedback.

[5] `https://wiki.cs.dartmouth.edu/srg/`

References

[ABK07] Giuseppe Ateniese, Marina Blanton, and Jonathan Kirsch. Secret Handshakes with Dynamic and Fuzzy Matching. In *NDSS*. The Internet Society, 2007.

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.

[AF99] C. Adams and S. Farrell. Internet X.509 Public Key Infrastructure Certificate Management Protocols. Internet Engineering Task Force: RFC 2510, 1999.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic -taa. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *LNCS*, pages 111–125. Springer, 2006.

[ASW97] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *ACM Conference on Computer and Communications Security*, pages 7–17, 1997.

[ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT*, pages 591–606, 1998.

[ASY06] Man Ho Au, Willy Susilo, and Siu-Ming Yiu. Event-oriented k-times revocable-iff-linked group signatures. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP*, volume 4058 of *LNCS*, pages 223–234. Springer, 2006.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BDS+03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret Handshakes from Pairing-Based Key Agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Computer Society, 2003.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM Press, 1993.

[BS04] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security*, pages 168–177. ACM, 2004.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security*, pages 201–210. ACM, 2006.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.

[CJT04] Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *LNCS*, pages 293–307. Springer, 2004.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[CL02a] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.

[CL02b] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN*, volume 2576 of *LNCS*, pages 268–289. Springer, 2002.

[CPHL07] Giorgio Calandriello, Panos Papadimitratos, Jean-Pierre Hubaux, and Antonio Lioy. Efficient and robust pseudonymous authentication in vanet. In *VANET '07: Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks*, pages 19–28, New York, NY, USA, 2007. ACM Press.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *LNCS*, pages 410–424. Springer, 1997.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[CWC05] Nicolas Christin, Andreas S. Weigend, and John Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *ACM Conference on Electronic Commerce*, pages 68–77. ACM, 2005.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 609–626. Springer, 2004.

[Dou02] John R. Douceur. The sybil attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS*, volume 2429 of *LNCS*, pages 251–260. Springer, 2002.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GR04] Steven D. Galbraith and Victor Rotger. Easy Decision-Diffie-Hellman Groups. *LMS Journal of Compuation and Mathematics*, 7:201–218, 2004.

[Kru07] John Krumm. Inference attacks on location tracks. In Anthony LaMarca, Marc Langheinrich, and Khai N. Truong, editors, *Pervasive*, volume 4480 of *LNCS*, pages 127–143. Springer, 2007.

[NIS01] NIST. FIPS 180-2: Secure hash standard (SHS). Technical report, National Institute of Standards and Technology (NIST), 2001. `http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf`.

[NSN05] Lan Nguyen and Reihaneh Safavi-Naini. Dynamic k-times anonymous authentication. In John Ioannidis, Angelos D. Keromytis, and Moti Yung, editors, *ACNS*, volume 3531 of *LNCS*, pages 318–333, 2005.

[RH07] Maxim Raya and Jean-Pierre Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.

[TAKS07] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: blocking misbehaving users without TTPs. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 72–81, New York, NY, USA, 2007. ACM.

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-times anonymous authentication (extended abstract). In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *LNCS*, pages 308–322. Springer, 2004.

[TPM07] TPM Work Group. TCG TPM Specification Version 1.2 Revision 103. Technical report, Trusted Computing Group, 2007.

[TS06] Isamu Teranishi and Kazue Sako. k-times anonymous authentication with a constant proving cost. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *LNCS*, pages 525–542. Springer, 2006.

[TS08] Patrick P. Tsang and Sean W. Smith. PPAA: Peer-to-peer anonymous authentication. In *ACNS*, LNCS. Springer, 2008. To appear.

[TWC+04] Patrick P. Tsang, Victor K. Wei, Tony K. Chan, Man Ho Au, Joseph K. Liu, and Duncan S. Wong. Separable linkable threshold ring signatures. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *LNCS*, pages 384–398. Springer, 2004.

[TX06] Gene Tsudik and Shouhuai Xu. A flexible framework for secret handshakes. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4258 of *LNCS*, pages 295–315. Springer, 2006.

[vABHL03] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Captcha: Using hard ai problems for security. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *LNCS*, pages 294–311. Springer, 2003.

[XY04] Shouhuai Xu and Moti Yung. k-anonymous secret handshakes with reusable credentials. In *ACM Conference on Computer and Communications Security*, pages 158–167. ACM, 2004.

A Formal Security Definitions

We formally define the security of PPAA. For each security requirement given in Section 5.2, we define a corresponding game played between the Challenger $\mathcal{C}$ and the Adversary $\mathcal{A}$. A security requirement is satisfied in a PPAA construction if no Probabilistic Poly-Time (PPT) adversary can win in the corresponding game with certain probabilities. To break the security of a construction, $\mathcal{A}$ thus tries to win in at least one of the games with the required probabilities. $\mathcal{A}$'s capabilities when trying to win in a game is modeled as oracles maintained by $\mathcal{C}$.

Let set S and integer n be states kept by these oracles, initially set to ϵ and 0 respectively. The oracles are defined as follows:

- Reg simulates a run of the registration protocol between an honest user and an honest GM and returns the resulting protocol transcript to the adversary. The newly registered honest peer is given the index n. n is added to S and then incremented.
- Reg-U acts on behalf of honest user and interacts with the corrupt GM in a run of the registration protocol. Upon successful termination, the newly registered honest peer is given the index n. n is added to S and then incremented.
- Reg-G acts on behalf of the honest GM and interacts with a corrupt peer in a run of the registration protocol. Upon successful termination, the newly registered corrupt peer is given the index n. n is then incremented.
- Corr(i) the corruption of honest user $i \in S$ by the adversary. It returns the user's credential to the adversary. n is then removed from S.
- Auth($\texttt{eid}, i, j$) simulates a run of the authentication protocol between honest client $i \in S$ and honest server $j \in S \backslash \{i\}$ during event `eid` and returns the resulting transcript to the adversary.
- Auth-C($\texttt{eid}, i$) acts on behalf of honest client $i \in S$ and interacts with a corrupt server in the authentication protocol during event `eid`.
- Auth-S($\texttt{eid}, j$) acts on behalf of honest server $j \in S$ and interacts with a corrupt client in the authentication protocol during event `eid`.

$\mathcal{A}$ can query arbitrarily and adaptively the oracles throughout the game, subject to appropriate restrictions specified in the games. Accesses to Corr are atomic. Accesses to Reg or Reg-G must not overlap. Without loss of generality, Corr(i) is disabled during queries to Auth($\cdot, i, \cdot$), Auth($\cdot, \cdot, i$), Auth-C($\cdot, i$) and Auth-S($\cdot, i$).

A.1 Mis-authentication Resistance and Peer Accountability

Mis-authentication Resistance is implied by Peer Accountability: if an unregistered client (resp. server) Alice can be successfully authenticated, then she can be successfully authenticated by a server (resp. client) during an event not linked to any authentication by the same server (resp. client) during the same event. The following game between the Challenger $\mathcal{C}$ and the Adversary $\mathcal{A}$ is defined for Peer Accountability.

- *(Setup Phase)* $\mathcal{C}$ executes Setup on a sufficiently large security parameter and gives gpk to $\mathcal{A}$.
- *(Probing Phase)* $\mathcal{A}$ is given access to all oracles except Reg-U.
- *(End Game Phase)* $\mathcal{A}$ outputs $(\mathtt{eid}^*, j^*)$.

Let n^* be the number of successful queries to Auth-S$(\mathtt{eid}^*, j^*)$ or Auth-C$(\mathtt{eid}^*, j^*)$ such that all the associated tags are pairwisely not linked. Let n be the sum of the number of queries to Reg-G and that to Corr. $\mathcal{A}$ wins in the game if $n^* > n$. Peer Accountability is satisfied if no PPT adversary can win in the game with non-negligible probability.

A.2 Peer Privacy

The following game between the Challenger $\mathcal{C}$ and the Adversary $\mathcal{A}$ are defined for Peer Privacy.

- *(Setup Phase)* $\mathcal{C}$ runs Setup on a sufficiently large security paramter and gives both gpk and gsk to $\mathcal{A}$.
- *(Probing Phase I)* $\mathcal{A}$ is given access to all oracles except Reg and Reg-G.
- *(Challenge Phase)* $\mathcal{A}$ gives $\mathcal{C}$ $\mathtt{eid}^*$ and distinct i_0^*, i_1^* and j^* such that the following condition holds:

  Condition 1 For $b \in \{0, 1\}$, all statements below are true:

  1. $\nexists$Corr(i_b^*).
  2. $\nexists$Auth$(\mathtt{eid}^*, i_b^*, j^*)$ and $\nexists$Auth$(\mathtt{eid}^*, j^*, i_b^*)$.
  3. If Corr(j^*) exists, then there exists no Auth-C$(\mathtt{eid}^*, i_b^*)$ or Auth-S$(\mathtt{eid}^*, i_b^*)$ after Corr(j^*).

  $\mathcal{C}$ then picks $b^* \in_R \{0, 1\}$.
- *(Probing Phase II)* $\mathcal{A}$ is given access to all oracles, and four new ones: Auth$^*_C(j)$, Auth$^*_S(j)$, Auth-C$^*()$ and Auth-S$^*()$, which respectively behave as Auth$(\mathtt{eid}^*, i_{b^*}^*, j)$, Auth$(\mathtt{eid}^*, j, i_{b^*}^*)$, Auth-C$(\mathtt{eid}^*, i_{b^*}^*)$ and Auth-S$(\mathtt{eid}^*, i_{b^*}^*)$. Condition 1 must still hold. In addition, if Corr(j^*) exists, then there exists no Auth-C$^*()$ or Auth-S$^*()$ after Corr(j^*).
- *(End Game Phase)* $\mathcal{A}$ outputs b'.

$\mathcal{A}$ wins in the game if $b' = b^*$. Peer Privacy is satisfied if no PPT adversary can win in any of the two games with probability non-negligibly greater than 1/2.

Remarks Item 1 in Condition 1 requires that peers i_0^* and i_1^* under attack are *honest*. It makes no sense to protect the privacy of a peer corrupted by the adversary from the adversary. Item 2 requires that neither of honest peers i_0^* and i_1^* mutually authenticates with peer j^* during the event with identifier $\mathtt{eid}^*$ when peer j^* is still honest, because otherwise the adversary would have known the corresponding tag and could use it to identify peer i_0^* or i_1^* when the peer mutually authenticates with peer j^* during the same event. Restricting peers i_0^* and i_1^* as such does not

weaken the adversary's capabilities because PPAA is designed to provide linking correctness as defined in Definition 2. Item 3 further forbids peer i_0^* and i_1^* to mutually authenticate during the event with identifier $\texttt{eid}^*$ with any peer not known to be still honest after the adversary has corrupted peer j^*, because otherwise the adversary could act as peer j^* and achieve the same effect as in Item 2.

A.3 Framing Resistance

The following game between the Challenger $\mathcal{C}$ and the Adversary $\mathcal{A}$ is defined for Framing Resistance.

- *(Setup Phase)* $\mathcal{C}$ executes Setup on a sufficiently large security parameter and gives both *gpk* and *gsk* to $\mathcal{A}$.
- *(Probing Phase)* $\mathcal{A}$ is given access to all oracles except Reg and Reg-G.
- *(End Game Phase)* $\mathcal{A}$ outputs $(\texttt{eid}^*, i_0^*, i_1^*, b)$, where $b \in \{0, 1\}$.

$\mathcal{A}$ wins in the game if there exists no query to $\textsc{Corr}(i_b^*)$, there exist a successful query Q_1 to $\textsc{Auth}(\texttt{eid}^*, i_0^*, i_1^*)$ and a successful query Q_2 to $\textsc{Auth-S}(\texttt{eid}^*, i_b^*)$ or $\textsc{Auth-C}(\texttt{eid}^*, i_b^*)$ such that the tags associated with Q_1 and Q_2 are linked. Framing Resistance is satisfied if no PPT adversary can win in the game with non-negligible probability.

Remarks Peer i_b^* is the honest user being framed. Q_1 is an authentication he was actually involved. Q_2 is an authentication he wasn't involved, but is linked to Q_1.

B Implementation

B.1 Software Prototyping

We prototyped our proposed PPAA construction as a software library written in C. We used the Pairing-Based Cryptography (PBC) Library[6] (version 0.4.12) for the underlying elliptic-curve and pairing operations, which in turn uses the GNU MP Bignum (GMP) Library (version 4.2.1) for field arithmetics. We also used OpenSSL for some of its cryptographic routines such as SHA-1 [NIS01] for instantiating the secure cryptographic hash functions.

We chose MNT curves with embedding degree 6 that contain a subgroup of prime order as the DDH assumption is conjectured to hold in $\mathbb{G}_1$ instantiated as such. In particular, we used the curve known as 9563-201-181 in the PBC Library, where 181 denotes $|p|$, i.e. the security parameter λ.[7]

B.2 Empirical Performance Results

The machine on which we ran our experiments was an Lenovo/IBM Thinkpad T60 laptop with an Intel dual-core 2GHz CPU and 1.5GB of RAM, running Ubuntu GNU/Linux 7.04. Notice that this machine is similar to a typical end-user machine in P2P systems in terms of computational

[6] The Pairing-Based Cryptography Library. `http://crypto.stanford.edu/pbc/`

[7] The hardness of solving the DDH problem in $\mathbb{G}_1$ instantiated this way is believed to be comparable solving the same problem in a group of integers of size 1024 bits, which is in turn believed to be intractable.

	Alice (the Initator)	Bob (the Responder)
Precomputation	2.21s	2.03s
Online computation	0.35s	0.34s
Protocol Latency	0.35s	

Table 2: Experiments show that our authentication protocol is efficient: Alice and Bob experience only a latency of only 0.35s, irrespective of the number of peers in the system.

power. The performance figures gathered from the experimentation thus closely reflect those in a real deployment. All the timings reported below are averaged over 10 randomized runs of the same experiments. We only report timing figures on the authentication protocol.

Before the actual authentication run, a user Alice who is going to play the role of the initiating peer initializes a initiator context structure in our prototype, during which all the precomputable operations are performed. This took Alice 2.21s. Similarly, a user Bob initializes a responder context structure, foreseeing that he will play the role of the responding peer in a future authentication run. This took Bob 2.03s. With their respective context structures initialized, Alice and Bob experienced a total latency of 0.35s executing the PPAA authentication protocol.

These figures are also listed in Table 2.

C SPK Instantiation

We detail the instantiation of the SPKs in Section 6.2. Let $\hat{e}_i$ denote $\hat{e}(g_i, h_0)$ for all $i = 0$ to 4. It is straightforward to show that the instantiation of each SPK is an non-interactive honest-verifier zero-knowledge proof-of-knowledge protocol with special soundness under the appropriate assumptions in the random oracle model. We thus omit the proof for conciseness.

C.1 SPK_0

Signing To sign a signature for SPK_0 on message $M \in \{0,1\}^*$, do the following using the knowledge of (x, y, z'):

- *(Commitment.)* Compute $T = g_1^{r_x} g_2^{r_y} g_3^{r_{z'}}$, where $r_x, r_y, r_{z'} \in_R \mathbb{Z}_p$.
- *(Challenge.)* Compute $c = \hat{H}(T||M)$.
- *(Response.)* Compute $s_x = r_x - cx$, $s_y = r_y - cy$ and $s_{z'} = r_{z'} - cz'$.

Output the signature Π_0 as $(c, s_x, s_y, s_{z'})$.

Verification To verify a signature Π_0 for SPK_0 on message $M \in \{0,1\}^*$, compute $T' = g_1^{s_x} g_2^{s_y} g_3^{s_{z'}} C^c$ and then $c' = \hat{H}(T'||M)$. Output `valid` if $c = c'$. Output `invalid` otherwise.

C.2 SPK_1

Signing To sign a signature Π_1 for SPK_1 on message $M \in \{0,1\}^*$, do the following using the knowledge of (A, e, x, y, z, r):

- *(Auxiliary commitment.)* Pick $\rho_1, \rho_2 \in_R \mathbb{Z}_p$. Compute $C_1 = g_4^{\rho_1} g_5^{\rho_2}$ and $C_2 = A g_4^{\rho_2}$.
- Sign a signature for SPK_1' on message $M' = C_1||C_2||M$, where SPK_1' denotes:

$$SPK\left\{\left(\begin{array}{c} e, x, y, z, r, \\ \rho_1, \rho_2, \\ \alpha_1, \alpha_2, \beta_1, \beta_2 \end{array}\right) : \begin{array}{c} \frac{\hat{e}(C_2,w)}{\hat{e}_0} = \hat{e}(C_2,h_0)^{-e}\hat{e}_1^{x}\hat{e}_2^{y}\hat{e}_3^{z}\hat{e}_4^{\alpha_2}\hat{e}(g_4,w)^{\rho_2} \quad \wedge \\ C_1 = g_4^{\rho_1} g_5^{\rho_2} \quad \wedge \\ 1 = C_1^{-e} g_4^{\alpha_1} g_5^{\alpha_2} \quad \wedge \quad 1 = C_1^{-r} g_4^{\beta_1} g_5^{\beta_2} \quad \wedge \\ U_1 = C_2^{r} g_4^{-\beta_2} \quad \wedge \quad V_1 = H(\texttt{eid})^{r} \end{array}\right\},$$

  using the knowledge of $(e, x, y, z, r, \rho_1, \rho_2, \alpha_1, \alpha_2, \beta_1, \beta_2)$, where $\alpha_i = e\rho_i$ and $\beta_i = r\rho_i$ for $i = 1, 2$, which can be done in the following three steps.
- *(Commitment.)* Pick $r_e, r_x, r_y, r_z, r_r, r_{\rho_1}, r_{\rho_2}, r_{\alpha_1}, r_{\alpha_2}, r_{\beta_1}, r_{\beta_2} \in_R \mathbb{Z}_p$. Compute

$$\begin{array}{lllll} T_1 = \hat{e}(C_2,h_0)^{-r_e}\hat{e}_1^{r_x}\hat{e}_2^{r_y}\hat{e}_3^{r_z}\hat{e}_4^{r_{\alpha_2}}\hat{e}(g_4,w)^{r_{\rho_2}}, & & & & \\ T_2 = g_4^{r_{\rho_1}} g_5^{r_{\rho_2}}, & & & & \\ T_3 = C_1^{-r_e} g_4^{r_{\alpha_1}} g_5^{r_{\alpha_2}}, & T_4 = C_1^{-r_r} g_4^{r_{\beta_1}} g_5^{r_{\beta_2}}, & & & \\ T_5 = C_2^{r_r} g_4^{-r_{\beta_2}}, & T_6 = H(\texttt{eid})^{r_r}. & & & \end{array}$$

- *(Challenge.)* Compute $c = \hat{H}(T_1||\ldots||T_6||M')$.
- *(Response.)* Compute

$$\begin{array}{llllllllllll} s_e = r_e - ce, & s_x = r_x - cx, & s_y = r_y - cy, \\ s_z = r_z - cz, & s_r = r_r - cr, & & & & & & \\ s_{\rho_1} = r_{\rho_1} - c\rho_1, & s_{\rho_2} = r_{\rho_2} - c\rho_2, & & & & & & \\ s_{\alpha_1} = r_{\alpha_1} - c\alpha_1, & s_{\alpha_2} = r_{\alpha_2} - c\alpha_2, & & & & & & \\ s_{\beta_1} = r_{\beta_1} - c\beta_1, & s_{\beta_2} = r_{\beta_2} - c\beta_2. & & & & & & \end{array}$$

Output the signature Π_1 as $(C_1, C_2, c, s_e, s_x, s_y, s_z, s_r, s_{\rho_1}, s_{\rho_2}, s_{\alpha_1}, s_{\alpha_2}, s_{\beta_1}, s_{\beta_2})$.

Signing requires 7 multi-EXPs in $\mathbb{G}_1$, 3 multi-EXPs in $\mathbb{G}_T$ and 1 pairing. All operations can be precomputed by Alice during an authentication.

Verification To verify a signature Π_1 for SPK_1 on message $M \in \{0,1\}^*$, do the following:

- Compute

$$\begin{array}{lllll} T_1' = \hat{e}(C_2,h_0)^{-s_{e_2}}\hat{e}_1^{s_x}\hat{e}_2^{s_y}\hat{e}_3^{s_z}\hat{e}_4^{s_{\alpha_2}}\hat{e}(g_4,w)^{s_{\rho_2}}\left(\frac{\hat{e}(C_2,w)}{\hat{e}_0}\right)^{c}, & & & & \\ T_2' = g_4^{s_{\rho_1}} g_5^{s_{\rho_2}} C_1^{c}, & & & & \\ T_3' = C_1^{-s_e} g_4^{s_{\alpha_1}} g_5^{s_{\alpha_2}}, & T_4' = C_1^{-s_r} g_4^{s_{\beta_1}} g_5^{s_{\beta_2}}, & & & \\ T_5' = C_2^{s_r} g_4^{-s_{\beta_2}} U_1^{c}, & T_6' = H(\texttt{eid})^{s_r} V_1^{c}. & & & \end{array}$$

- Compute $c' = \hat{H}(T_1'||\ldots||T_6'||M')$, where $M' = C_1||C_2||M$.

Output `valid` if $c = c'$. Output `invalid` otherwise.

Verification requires 5 multi-EXPs in $\mathbb{G}_1$, 4 multi-EXPs in $\mathbb{G}_T$ and 2 pairings.

C.3 SPK_2

Signing To sign a signature Π_2 for SPK_2 on message $M \in \{0,1\}^*$, do the following using the knowledge of (A, e, x, y, z, r):

- *(Auxiliary commitment.)* Pick $\rho_1, \rho_2 \in_R \mathbb{Z}_p$. Compute $C_1 = g_4^{\rho_1} g_5^{\rho_2}$ and $C_2 = A g_4^{\rho_2}$.
- Sign a signature for SPK_2' on message $M' = C_1||C_2||M$, where SPK_2' denotes:

$$SPK\left\{\left(\begin{array}{c} e, x, y, z, r, \\ \rho_1, \rho_2, \\ \alpha_1, \alpha_2, \beta_1, \beta_2 \end{array}\right) : \begin{array}{rcl} \frac{\hat{e}(C_2,w)}{\hat{e}_0} = \hat{e}(C_2,h_0)^{-e}\hat{e}_1^x\hat{e}_2^y\hat{e}_3^z\hat{e}_4^{\alpha_2}\hat{e}(g_4,w)^{\rho_2} & \wedge \\ C_1 = g_4^{\rho_1} g_5^{\rho_2} & \wedge \\ 1 = C_1^{-e} g_4^{\alpha_1} g_5^{\alpha_2} \wedge 1 = C_1^{-r} g_4^{\beta_1} g_5^{\beta_2} & \wedge \\ U_2 = C_2^r g_4^{-\beta_2} \wedge V_2 = H(\texttt{eid})^r & \wedge \\ W_2 = U_1^x V_1^y & \end{array}\right\},$$

  using the knowledge of $(e, x, y, z, r, \rho_1, \rho_2, \alpha_1, \alpha_2, \beta_1, \beta_2)$, where $\alpha_i = e\rho_i$ and $\beta_i = r\rho_i$ for $i = 1, 2$, which can be done in the following three steps.

- *(Commitment.)* Pick $r_e, r_x, r_y, r_z, r_r, r_{\rho_1}, r_{\rho_2}, r_{\alpha_1}, r_{\alpha_2}, r_{\beta_1}, r_{\beta_2} \in_R \mathbb{Z}_p$. Compute $T_1, T_2, \ldots, T_6$ as in SPK_1. Further compute
$$T_7 = U_1^{r_x} V_1^{r_y}.$$
- *(Challenge.)* Compute $c = \hat{H}(T_1||\ldots||T_7||M')$.
- *(Response.)* Same as that in SPK_1.

Output the signature Π_1 as $(C_1, C_2, c, s_e, s_x, s_y, s_z, s_r, s_{\rho_1}, s_{\rho_2}, s_{\alpha_1}, s_{\alpha_2}, s_{\beta_1}, s_{\beta_2})$.

Signing requires 8 multi-EXPs in $\mathbb{G}_1$, 3 multi-EXPs in $\mathbb{G}_T$ and 1 pairing. Among them only 1 multi-EXP in $\mathbb{G}_1$, namely T_7, can't be precomputed by Bob during an authentication.

Verification To verify a signature Π_1 for SPK_1 on message $M \in \{0,1\}^*$, do the following:

- Compute $T_1', T_2', \ldots, T_4'$ as in SPK_1. Further compute
$$T_5' = C_2^{s_r} g_4^{-s_{\beta_2}} U_2^c,\ T_6' = H(\texttt{eid})^{s_r} V_2^c,\ T_7' = U_1^{s_x} V_1^{s_y} W_2^c.$$
- Compute $c' = \hat{H}(T_1'||\ldots||T_7'||M')$, where $M' = C_1||C_2||M$.

Output `valid` if $c = c'$. Output `invalid` otherwise.

Verification requires 6 multi-EXPs in $\mathbb{G}_1$, 4 multi-EXPs in $\mathbb{G}_T$ and 2 pairings.

C.4 SPK_3

Signing To sign a signature Π_3 for SPK_3 on message $M \in \{0,1\}^*$, do the following using the knowledge of (A, e, x, y, z, r):

- *(Auxiliary commitment.)* Pick $\rho_1, \rho_2 \in_R \mathbb{Z}_p$. Compute $C_1 = g_4^{\rho_1} g_5^{\rho_2}$ and $C_2 = A g_4^{\rho_2}$.

- Sign a signature for SPK_3' on message $M' = C_1||C_2||M$, where SPK_2' denotes:

$$SPK\left\{\begin{pmatrix} e, x, y, z, r, \\ \rho_1, \rho_2, \\ \alpha_1, \alpha_2, \beta_1, \beta_2 \end{pmatrix} : \begin{array}{rclcrcll} \frac{\hat{e}(C_2,w)}{\hat{e}_0} & = & \hat{e}(C_2,h_0)^{-e}\hat{e}_1^x\hat{e}_2^y\hat{e}_3^z\hat{e}_4^{\alpha_2}\hat{e}(g_4,w)^{\rho_2} & \wedge & & & & \wedge \\ & & C_1 = g_4^{\rho_1}g_5^{\rho_2} & & & & & \wedge \\ 1 & = & C_1^{-e}g_4^{\alpha_1}g_5^{\alpha_2} & \wedge & 1 & = & C_1^{-r}g_4^{\beta_1}g_5^{\beta_2} & \wedge \\ U_1 & = & C_2^r g_4^{-\beta_2} & \wedge & V_1 & = & H(\texttt{eid})^r & \wedge \\ W_1 & = & U_2^x V_2^y & \wedge & W_2 & = & \tau_1^r & \wedge \end{array}\right\},$$

  using the knowledge of $(e, x, y, z, r, \rho_1, \rho_2, \alpha_1, \alpha_2, \beta_1, \beta_2)$, where $\alpha_i = e\rho_i$ and $\beta_i = r\rho_i$ for $i = 1, 2$, which can be done in the following three steps.

- *(Commitment.)* Pick $r_e, r_x, r_y, r_z, r_r, r_{\rho_1}, r_{\rho_2}, r_{\alpha_1}, r_{\alpha_2}, r_{\beta_1}, r_{\beta_2} \in_R \mathbb{Z}_p$. Compute $T_1, T_2, \ldots, T_6$ as in SPK_1. Further compute

$$T_7 = U_2^{r_x}V_2^{r_y},\ T_8 = \tau_1^{r_r}.$$

- *(Challenge.)* Compute $c = \hat{H}(T_1||\ldots||T_8||M')$.

- *(Response.)* Same as that in SPK_1.

Output the signature Π_1 as $(C_1, C_2, c, s_e, s_x, s_y, s_z, s_r, s_{\rho_1}, s_{\rho_2}, s_{\alpha_1}, s_{\alpha_2}, s_{\beta_1}, s_{\beta_2})$.

Signing requires 9 multi-EXPs in $\mathbb{G}_1$, 3 multi-EXPs in $\mathbb{G}_T$ and 1 pairing. Among them only 2 multi-EXPs in $\mathbb{G}_1$, namely T_7 and T_8, can't be precomputed by Alice during an authentication.

Verification To verify a signature Π_1 for SPK_1 on message $M \in \{0,1\}^*$, do the following:

- Compute $T_1', T_2', \ldots, T_6'$ as in SPK_1. Further compute

$$T_7' = U_2^{s_x}V_2^{s_y}W_1^c,\ T_8' = \tau_1^{s_r}W_2^c.$$

- Compute $c' = \hat{H}(T_1'||\ldots||T_8'||M')$, where $M' = C_1||C_2||M$.

Output `valid` if $c = c'$. Output `invalid` otherwise.

Verification requires 7 multi-EXPs in $\mathbb{G}_1$, 4 multi-EXPs in $\mathbb{G}_T$ and 2 pairings.

C.5 SPK_4

Signing To sign a signature for SPK_4 on message $M \in \{0,1\}^*$, do the following using the knowledge of (r):

- *(Commitment.)* Compute $T_1 = H(\texttt{eid})^{r_r}$ and $T_2 = \tau_2^{r_r}$, where $r_r \in_R \mathbb{Z}_p$.

- *(Challenge.)* Compute $c = \hat{H}(T_1||T_2||M)$.

- *(Response.)* Compute $s_r = r_r - cr$.

Output the signature Π_4 as (c, s_r).

Signing requires 2 multi-EXPs in $\mathbb{G}_1$, one of which, namely T_2, can be precomputed.

SPK	Algorithm	Number of Operations		
		$\mathbb{G}_1$ multi-EXPs	$\mathbb{G}_T$ multi-EXPs	Pairings
SPK_0	Signing	1	0	0
	Verification	2	0	0
SPK_1	Signing	7	3	1
	Verification	5	4	2
SPK_2	Signing	8	3	1
	Verification	6	4	2
SPK_3	Signing	9	3	1
	Verification	7	4	2
SPK_4	Signing	2	0	0
	Verification	2	0	0

Table 3: Timing complexity of the SPKs

Verification To verify a signature Π_4 for SPK_0 on message $M \in \{0,1\}^*$, do the following:

- Compute $T_2' = H(\texttt{eid})^{s_r} V_2^c$ and $T_2' = \tau_2^{s_r} W_1^c$.
- Compute $c' = \hat{H}(T_1'||T_2'||M)$.

Output `valid` if $c = c'$. Output `invalid` otherwise.

Verification requires 2 multi-EXPs in $\mathbb{G}_1$.

C.6 Computational Costs

Table 3 summarizes the computational costs in terms of the number of pairing computation and multi-exponentiations (multi-EXPs).

D Proof Sketches

Define $[n] = \{1, 2, \ldots, n\}$ for all non-negative integer n. If S is a set, $a \in_R S$ means that a is assigned to an element drawn from S uniformly at random.

We now sketch the proof of Theorem 1.

D.1 Peer Accountability

We assume the contrary that there exists a PPT the adversary $\mathcal{A}$ who can win in the game defined for Peer Accountability with non-negligible probability, and show how to construct a PPT simulator $\mathcal{S}$ that solves a given q-SDH problem instance with non-negligible probability, contradicting to the q-SDH assumption.

On input an q-SDH problem instance, $\mathcal{S}$ generates the system's public parameters and the GM's key pair using the same technique in [BB04], which is also used by schemes such as [ASM06, TAKS07]. In this way, $\mathcal{S}$ obtains $(q-1)$ SDH pairs from the problem instance and the knowledge of any new distinct SDH pair enables $\mathcal{S}$ to solve the given problem instance. Let q be the maximum number of peer users registered throughout the game. $\mathcal{S}$ can correctly answer the oracle queries that involve all but one peer user. $\mathcal{S}$ picks that user uniformly at random. That is, that user is the q^*-th registered user, where $q^* \in_R [q]$.

More concretely, $\mathcal{S}$ uses assigned a distinct SDH pair to each of the $(q-1)$ users and creates a credential for that user using the assigned SDH pair, while the credential for user q^* is calculated differently. We refer the readers to [BB04] for the details. Simulating REG is easy because of the HVZK-ness of SPK_0 in the registration protocol. Registering users through REG-G requires $\mathcal{S}$ to first rewind the adversary during SPK_0 to extract (x, y, z'). Finally, the HVZK-ness of SPK_1 to SPK_4 allows $\mathcal{S}$ to correctly simulate AUTH, by first selecting V_1, V_2 and then simulating protocol transcript for such V_1 and V_2. AUTH-C and AUTH-S can be correctly simulated in a similar way.

Eventually, $\mathcal{A}$ has won in the game. Denote by $\mathtt{tag}^{(1)}, \mathtt{tag}^{(2)}, \ldots, \mathtt{tag}^{(n^*)}$ the n^* pairwisely not linked tags, i.e. $\mathtt{tag}^{(i)} \neq \mathtt{tag}^{(j)}$ for all distinct $i, j \in [n^*]$, output by AUTH-C$(\mathtt{eid}^*, j^*)$ or AUTH-S$(\mathtt{eid}^*, j^*)$. Let the credential of peer j^* be $(A^*, e^*, x^*, y^*, z^*)$. Then the special soundness property of SPK_0 to SPK_4 implies that $\mathcal{A}$ knows credential $(A_i, e_i, x_i, y_i, z_i)$ such that $\mathtt{tag}^{(i)} = \{A_i^{x^*} H(\mathtt{eid})^{y_i}, A^{*x_i} H(\mathtt{eid})^{y^*}\}$ for all $i \in [n^*]$. More preciously, such knowledge can be extracted from $\mathcal{A}$ by rewinding two times, one at each of, in the case of AUTH-S, SPK_1 and SPK_3 or, in the case of AUTH-C, SPK_2 and SPK_4.

Therefore, for all distinct $i, j \in [n^*]$, $A_i^{x^*} H(\mathtt{eid})^{y_i} \neq A_j^{x^*} H(\mathtt{eid})^{y_j}$ or $A^{*x_i} H(\mathtt{eid})^{y^*} \neq A^{*x_j} H(\mathtt{eid})^{y^*}$, which implies $A_i \neq A_j$ or $x_i \neq x_j$ or $y_i \neq y_j$, which in turn implies $(A_i, x_i, y_i) \neq (A_j, x_j, y_j)$. Now if $A_i = A_j$, then $x_i \neq x_j$ or $y_i \neq y_j$, which implies $e_i \neq e_j$ unless DL problem is easy. As a result, we have $(A_i, e_i) \neq (A_j, e_j)$ for all distinct $i, j \in [n^*]$. Now, since $n^* > n = q$, there exists an $i^* \in [n^*]$ such that (A_{i^*}, e_{i^*}) can be used to compute, with probability at least $1/q$, a distinct SDH pair as the solution to the problem instance. □

D.2 Peer Privacy

We assume the contrary that there exists a PPT the adversary $\mathcal{A}$ who can win in the game defined for Peer Privacy with probability non-negligibly greater than 1/2, and show how to construct a PPT simulator $\mathcal{S}$ that solves a given DDH problem instance with probability non-negligibly greater than 1/2, contradicting to the XDH assumption.

Let q be the upper bound on the number of queries to REG-U made by $\mathcal{A}$ throughout the game. Let $q^* \in_R [q]$. Given a DDH problem instance $(\mathfrak{g}, \mathfrak{h}, \mathfrak{u}, \mathfrak{v})$, $\mathcal{S}$ sets g_0 in the system's public parameters to $\mathfrak{g}$ and generates the rest of the system's public parameters and the GM's key pair honestly, i.e. by executing the respective algorithms according to specification. $\mathcal{S}$ simulates all oracles, with the exception of the hash oracle, honestly when the peer involved is not the q^*-th registered peer. To simulate the random oracle for H, $\mathcal{S}$ assigns, for each $\mathtt{eid}$ appeared in the game, $H(\mathtt{eid})$ to $\mathfrak{g}^{\rho_{\mathtt{eid}}}$, where $\rho_{\mathtt{eid}} \in_R \mathbb{Z}_p$.

To simulate REG-U for the q^*-th registered user, $\mathcal{S}$ simulates the registration protocol as if y in the user's credential is $\log_{\mathfrak{g}} \mathfrak{h}$ without actually knowing the value of $\log_{\mathfrak{g}} \mathfrak{h}$. $\mathcal{S}$ can do so correctly because, for any $x \in \mathbb{Z}_p$ and $y = \log_{\mathfrak{g}} \mathfrak{h}$, there exists an $z' \in \mathbb{Z}_p$ such that $C = g_1^x g_2^y g_3^{z'}$. To simulate AUTH-C$(\mathtt{eid}, q^*, \cdot)$, AUTH-C$(\mathtt{eid}, \cdot, q^*)$, AUTH-C$(\mathtt{eid}, q^*)$ and AUTH-S$(\mathtt{eid}, q^*)$, $\mathcal{S}$ first computes V for peer q^* using $\mathfrak{h}$ and $\rho_{\mathtt{eid}}$ and then uses such V to simulate the protocol execution. Note that S can't answer CORR(q^*) correctly.

Now with probability at least $1/q$, $\mathcal{A}$ have not never queried CORR on q^* and one of i_0^* and i_1^* equals q^*. $\mathcal{S}$ then computes V for peer q^* using $\mathfrak{u}$ and $\mathfrak{v}$ instead and then uses such V to simulate the protocol execution. Note that restricting $\mathcal{A}$'s queries to satisfy Condition 1 in the game is a necessary condition for the correctness of the above simulation.

When $\mathcal{A}$ eventually returns his guess, S decides that the given problem instance is a DDH-tuple if the guess is correct; otherwise, S randomly guesses an answer. If the given problem instance is

a DDH-tuple, then $\mathcal{S}$'s answer is correct if $\mathcal{A}$'s answer is correct. Otherwise, $\mathcal{S}$'s answer is correct with probability 1/2. Combining with the probability arguments above, $\mathcal{S}$ can use $\mathcal{A}$ to solve the given problem instance with probability non-negligibly greater than 1/2. □

D.3 Framing Resistance

We assume the contrary that there exists a PPT the adversary $\mathcal{A}$ who can win in the game defined for Framing Resistance with non-negligible probability, and show how to construct a PPT simulator $\mathcal{S}$ that solves a given Discrete Logarithm (DL) problem instance with non-negligible probability, contradicting to the XDH assumption. (The XDH assumption implies the DDH assumption, which in turn implies the DL assumption.)

Let q be the maximum number of queries to Reg-G. Let $q^* \in_R [q]$. Given a DL problem instance $(\mathfrak{g}, \mathfrak{u})$, $\mathcal{S}$ sets g_0 in the system's public parameters to $\mathfrak{g}$ and generates the rest of the system's public parameters and the GM's key pair honestly. Let peer q^* be the peer registered into the system as a result of querying Reg-G for the q^* time. S answers all queries to all the oracles, except the hash oracle, by executing the algorithms honestly if the queries do not involve peer q^*. To simulate the random oracle for H, $\mathcal{S}$ assigns, for each $\mathtt{eid}$ appeared in the game, $H(\mathtt{eid})$ to $\mathfrak{g}^{\rho_{\mathtt{eid}}}$, where $\rho_{\mathtt{eid}} \in_R \mathbb{Z}_p$.

S responds to oracle queries involving peer q^* as follows. For Reg-U, $\mathcal{S}$ simulates the registration protocol as if y in the user's credential is $\log_{\mathfrak{g}} \mathfrak{u}$ without actually knowing the value of $\log_{\mathfrak{g}} \mathfrak{u}$. $\mathcal{S}$ can do so correctly because for any $x \in \mathbb{Z}_p$ and $y = \log_{\mathfrak{g}} \mathfrak{u}$, there exists an $z' \in \mathbb{Z}_p$ such that $C = g_1^x g_2^y g_3^{z'}$. Upon receiving (A, e, z'') from $\mathcal{A}$, $\mathcal{S}$ records (A, e). To simulate Auth$(\mathtt{eid}, q^*, \cdot)$, Auth$(\mathtt{eid}, \cdot, q^*$, Auth-C$(\mathtt{eid}, q^*)$ and Auth-S$(\mathtt{eid}, q^*)$, $\mathcal{S}$ first computes V for peer q^* using $\mathfrak{u}$ and $\rho_{\mathtt{eid}}$ and then simulates the protocols using such V. As long as $\mathcal{A}$ does not corrupt peer q^*, all oracles can be simulated correctly throughout the game.

And with non-negligible probability, $\mathcal{A}$ eventually has won in the game. Let $\mathtt{tag}_1$ and $\mathtt{tag}_2$ be the two linked tags associated with queries Q_1 and Q_2 respectively. With probability at least $1/q$, Q_1 involves peer q^*. S can rewind $\mathcal{A}$ to extract from it the knowledge of (A^*, x^*, y^*) that constitutes to $\mathtt{tag}_2$. Since $\mathtt{tag}_2$ is linked to $\mathtt{tag}_1$, $(A^*, x^*, y^*, \cdot)$ must be a valid credential of peer q^*. With this information, S can compute $\log_{\mathfrak{g}} \mathfrak{u}$ and solve the DL problem instance. □