CERIAS Tech Report 2006-26

PRACTICAL IDENTITY THEFT PREVENTION USING AGGREGATED PROOF OF KNOWLEDGE

by A. Bhargav-Spantzel, A.C. Squicciarini, R. Xue, E. Bertino

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Practical Identity Theft Prevention using Aggregated Proof of Knowledge

Abhilasha Bhargav-Spantzel Anna C. Squicciarini Rui Xue Elisa Bertino

Abstract

The problem of identity theft, that is, the act of impersonating others' identities by presenting stolen identifiers or proofs of identities, has been receiving increasing attention because of its high financial and social costs. In this paper we address such problem by developing a solution for federated organizations. Our approach is based on the concept of privacy preserving multi-factor authentication achieved by the implementation of a new cryptographic primitive which uses aggregate signatures on commitments that are then used for aggregate zero-knowledge proof of knowledge (ZKPK) protocols. The resultant signatures are very short and the zero-knowledge proofs are succinct and efficient. We prove the security of our scheme under the co-GDH assumption for groups with bilinear maps. Our cryptographic scheme is superior in terms of the performance, flexibility and storage requirements than the existing efficient ZKPK techniques that may be used to prove, under zero-knowledge, the knowledge of multiple secrets.

1 Introduction

Digital identity can be defined as the digital representation of the information known about a specific individual or organization. As such, it encompasses not only login names (often referred to as *nyms*), but many additional information, referred to as *identity attributes* or *identifiers*. Managing identity attributes raises a number of challenges, due to conflicting requirements. On the one hand, identity attributes need to be shared to speed up and facilitate authentication of users and access control. On the other hand, they need to be protected as they may convey sensitive information about an individual and can be a target of attacks like identity theft. Here, by *identity theft* we mean the act of impersonating others' identities by presenting stolen identifiers or proofs of identities. Reports state that within the last twelve months about a million American adults became victims of digital identity fraud [18]. Most common identity theft attacks are perpetrated through password cracking, pharming, phishing, and database attacks where the attacker captures the personally identifying information of individuals and uses them to commit fraud.

When talking about identifiers, it is important to distinguish between *weak* and *strong identifiers*. A strong identifier uniquely identifies an individual in a population, whereas a weak identifier can be applied to many individuals in a population. Whether an identifier is strong or weak depends upon the size of the population and the uniqueness of the identity attribute. Examples of strong identifiers are a user's passport number or social security number. Weak identifiers are attributes like age and gender.

Digital identity management is usually coupled with the notion of *federation* [19, 16, 25]. The goal of federations is to provide users with protected environments to federate identities by the proper management of identity attributes. Federations provide a controlled method by which federation members can provide more integrated and complete services to a qualified group of individuals within certain sets of business transactions. By controlling the scope of access to participating sites, and by enabling secure, cross-domain transmission of user's personal information, federations can make the perpetration of identity frauds more difficult, as well as reduce their frequency, and their potential impact. Federations are usually composed of two main entities: identity providers (IP's), managing identities of individuals, and service providers (SP's), offering services to registered individuals. In a typical federated identity management system, the individual registers with his/her local IP and is assigned a username and password. Based on this information an individual can submit additional attributes and corresponding attribute release policies, which are stored at the local IP. The IP is from then on contacted whenever the user interacts with any other SP in the federation when additional user information is needed. The IP is in charge of sending the SP the submitted user attributes according to the attribute release policies.

The major drawback of current approaches [19, 16, 25] to federate digital identity management is that no specific techniques are provided to protect registered individuals from identity theft. Dishonest individuals can register fake attributes or impersonate other users of the federation. It is thus crucial that authentication protocols able to protect

against identity theft be provided as part of digital identity management solutions. Some relevant requirements that such protocols should verify include the following:

1. The protocols should preserve individuals' *privacy*, and enforce a "need to know" principle when requiring identifiers.
2. The protocols should be efficient, flexible, and should not require complex user interactions. This directly relates to *usability* which is one of the main aims of federations.
3. The protocols should be *robust*, in the sense that even if an adversary is able to obtain the value of the strong identifiers, it should not be able to impersonate the victim in the federation.
4. A federated identity system should ensure *consistency* of the data shared in the federation. Although validity of identifiers can only be checked with actual identifier issuers, which can be outside the federation, the system should be able to detect identity theft based on the information available within the federation.

The goal of the work presented in this paper is to address the problem of identity theft in federations by developing a solution that satisfies the above requirements. Our solution to *strong authentication* is based on multi-factor authentication, which consists of verifying the authenticity of an identity using more than one identity token at a time. Our multi-factor authentication protocols are supported by some new cryptographic primitives, also proposed in this paper, that result in high efficiency. Our protocols implement a mechanism to prove the knowledge of multiple strong identifiers stored as cryptographic commitments using aggregated zero-knowledge proofs. The commitments are signed by a special federation entity, referred to as *registrar*, and the corresponding signature can be verified in an aggregated fashion at the time of use. To achieve aggregate signature we develop techniques based on the approach originally proposed by Boneh *et al.* [2].

Zero-knowledge proof of knowledge (ZKPK) is extensively used for identity protection [7, 8]. Our scheme enhances such protocols by the use of multi-factor proof. Although a single ZKPK has been proven to be sufficiently efficient [6], multi-factor proofs cannot maintain the same performance if a large number of proofs is considered. To address this issue we develop aggregated ZKPK and reduce the proofs of several factors, that would require several ZKPK's, to one that uses only one ZKPK. In our protocols, users always need to compute a small constant number of exponentiations, while the verifier's (that is, either the SP or the registrar) computation of exponentials is dramatically reduced, which makes our protocols highly suited for lightweight devices.

A key advantage of our protocols is that they are flexible with respect to which commitments are aggregated. That is, any combination of commitments (among the ones available for a given user), can be aggregated for computing the signature at runtime. This approach allows different SP's in the federation to challenge the knowledge of different combinations of the committed identifiers. This is a substantial improvement with respect to existing approaches [8], which require the possible combinations of strong identifiers to be predefined or stored for computation of multi-factor proofs. Thus, under such protocols the space required is exponential with respect to the number of committed values. Our protocols instead require storing only the committed values and signatures.

Another main advantage of our solution is that, from an architectural point of view, it requires only minimal extensions. Besides the conventional set of IP's and SP's composing a federation, our approach only requires adding some registrars. The task of registrars is to enable users to register their strong identifiers without having to reveal their actual values. Registration in our approach means that users can establish cryptographic tokens which can be used subsequently for establishing *proof of knowledge* for the corresponding strong identifiers. Registrars are small and modular software components which can be easily added to the architectures of current IP systems. Our solution is also succinct and flexible, since we let the interacting entities exchanging only the information actually needed for the specific interaction; no extra information needs to be exchanged. Our protocols greatly reduce the amount of information revealed to the SP for authentication. We even provide a protocol that allows one to verify of the signature of a commitment without knowing the value of the commitment itself. Such property greatly enhances privacy while still assuring integrity and validity of committed data. Moreover the ZKPK commitments used are semantically secure requiring the enrollment of random secret along with the strong identifier. The use of such technique ensures that even if an adversary learns the values of the strong identifiers, that is, steals identity information, it cannot wrongly authenticate itself as the owner of this information.

The paper is organized as follows. In Section 2 we provide our notion of identity assurance and the mechanisms for its establishment and maintenance within a federated identity management system. In Section 3 we present the cryptographic scheme for the aggregate proof of knowledge protocols. This is followed by a detailed analysis of security, efficiency and system security in Section 4. In Section 5 we discuss the related work followed by the conclusion. To be more accessible for readers, our presentation also includes two extra sections. In Appendix A, we the explain

functions required for identity assurance. In Appendix B, we provide proofs of some of the theorems presented in Section 4.

2 Identity Assurance in Federations

The notion of identity assurance deals with the confidence about the truth of the claims related with the identity of an individual. Intuitively, weak identity assurance may increase the risk of identity theft, as provenance and authenticity of the identity data are not certain. Hence, strong cryptographic techniques built upon identifiers which have weak identity assurance are vulnerable to misuse. Strong identity assurance is thus a crucial requirements for any identity management system. Our approach to identity assurance relies on the use of multiple proofs of identity, that are stored in a data structure called *Identity Record* (IdR for short) stored and managed by the registrar. The IdR is a fundamental notion of our approach in that it actually provides a digital representation of user identities. In this section we first introduce a simple notion of federation that we will use throughout the discussion. We then define all components of IdR's and various notions of assurance related to such records. We describe the requirements and mechanisms to evaluate and maintain assurance about IdR's in subsection 2.3.

2.1 Federations

Our approach to federations involve four types of entities: principals, service providers, identity providers and registrars. Formally, a federation is modeled as a tuple $\mathcal{F} = \langle \mathcal{P}, \mathcal{SP}, \mathcal{IP}, \mathcal{R} \rangle$, where:

- $\mathcal{P}$ is the set of **principals** belonging to the federation. Principals are associated with a single sign-on (SSO) Id; each SSO Id represents a principal of a federation. An individual can be associated with several principals in a federation.
- $\mathcal{SP}$ is the set of **service providers** forming the federation. The services offered by service providers are protected by a set of policies defining the requirements principals have to satisfy for their use. Such policies are referred to as service disclosure policies [1, 29].
- $\mathcal{IP}$ corresponds to the set of **identity providers**. At least one IP has to be in place in a federation to keep track of the principals' attributes.
- $\mathcal{R}$ corresponds to the set of **registrars** which establish and maintain *identity commitments* used to establish proof of knowledge of strong identifiers. At least one registrar has to be in place to achieve strong authentication of principals.

Note that the above entities are logical and therefore the same federation host may provide the functions associated with several such entities.

2.2 Identity Records

As we mentioned, each principal P in a federation has associated one or more IdR's, each recorded at some registrar in the federation. Each IdR in turn consists of several *identity tuples*, denoted as τ's. Each identity tuple is associated with one strong identifier and records all information related to the verification of this identifier at the time of use. In particular, each strong identifier$m$ is associated with two types of commitments: a semantically secure commitment, and a deterministic commitment, denoted as M and $\widehat{M}$, respectively. M is a value signed by the registrar upon registration. The signature on M is denoted by σ and is part of the identity tuple associated with m. M is computed as $g^m h^r$, where g and h are generators in group G of prime order q. G and q are public parameters of the registrar and r is chosen randomly from $\mathbb{Z}_q$[1]. The public parameters are the same for all registrars in the federation. $\widehat{M}$ is the deterministic commitment corresponding to M, and it is calculated as $\widehat{M} = g^m$. Note that $\widehat{M}$ is conceptually tied to m, but does not need to be stored in the identity tuple. Its usage will be clarified in Section 3. P can execute a ZKPK to assure that both M (stored in τ) and $\widehat{M}$ (the deterministic value) refer to the same m. m is also tied to a set of weak identifiers, denoted by $\{w_1, \ldots w_k\}$. For example, assume 4040330043794877 to be a credit card number and 'Alice' and 'Smith' be the first and last name of an individual. Here, 4040330043794877 is the strong identifier value, while 'Alice' and 'Smith' are the associated weak identifiers. The IdR is also associated with some other public parameters required for the cryptographic protocols, as detailed in Section 3.

[1] More details of the cryptographic commitments and mechanisms are subject of Section 3

All strong identifier commitments and weak identifiers are tagged with an identifier descriptor tag and two types of assurance, namely *validity assurance* and *ownership assurance*. Validity assurance corresponds to the confidence about the validity of the identifier based on the verification performed at the identifier's original issuer. As such, it refers to the correctness of identifiers with respect to the real world information sources and the original issuers of the identifiers, which can possibly be external to the federation. For example an issuer (say $MasterCard$) can verify if a credit card number it issued is valid. Ownership assurance corresponds to the confidence about the claim that the principal presenting a given identifier is its true owner.
We introduce four levels of assurance: absolute assurance, tagged as 'A', corresponding to the absolute certainty about the claim; reasonable assurance, tagged as 'B', corresponding to case when one or more assertions from trusted parties exist regarding the certainty of the claim; unknown assurance, tagged as 'U', when there is no information to assert the certainty of the claim; and false assurance, tagged as 'F', denoting that the claim is incorrect.
We assume that absolute validity of a given strong identifier can only be determined by authorities which have issued the strong identifiers. This corresponds to value 'A' of the validity-assure of the associated strong identifier. Because such authorities may not always be part of a given federation, we assume that agreements are in place that allows the federation to verify validity of identifiers with such authorities. Note however that our approach also supports the case when such verifications are not possible. In fact, we mark as 'B' the validity assurance of a strong identifier the validity of which has been asserted by a principal, whose identity record has a validity assurance set to 'A'. If no entity other than the principal supports the validity of the strong identifier, the identifier is marked with unknown assurance 'U'. Identifiers might be immediately validated upon registration, or they might be validated later on when actually used by the principal. The latter corresponds to the concept of *lazy validation*.

The notation adopted to represent the various IdR elements is as follows:
IdR = $\{\{\tau_i\}$, cryptographic parameters $\}$
τ_i = $[(\sigma_i, M_i, tag, validity\text{-}assure, ownership\text{-}assure), \{W_{i_j}\}]$
W_{i_j} = $(w_{i_j}, tag, validity\text{-}assure, ownership\text{-}assure)$

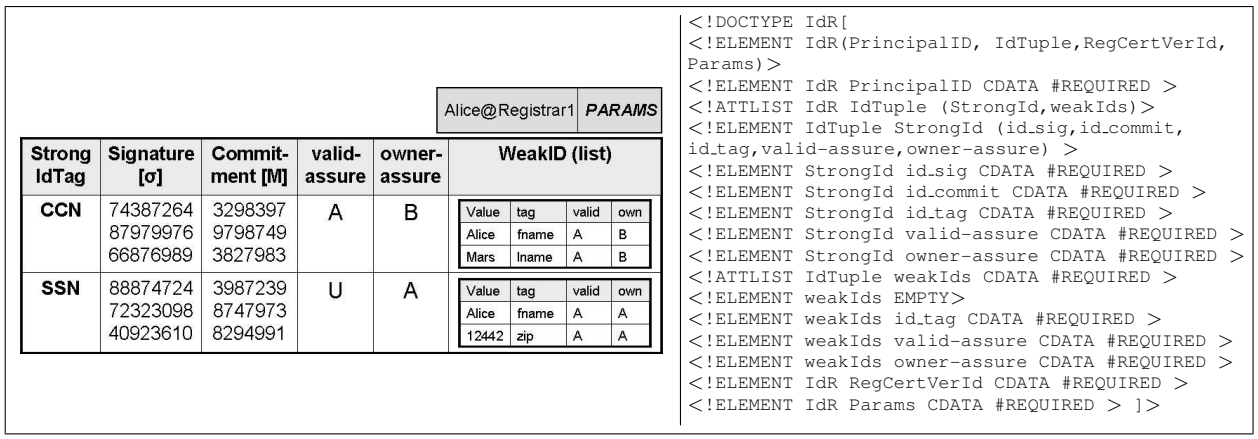

Figure 1: (a) Simplified graphical representation of an IdR (left). (b) DTD of an IdR (right).

For clarity of understanding we provide an illustrative example of an IdR as shown in Figure 1(a). Here the principle is known as Alice@Registrar1 with registrar Registrar1 and has enrolled 2 strong identifiers namely a CCN and SSN. We report the Document Type Definition [21] providing the structure of a IdR in Figure 1 (b).
We adopt the dot notation to denote an element in a given object. That is, a tuple τ' appearing in IdR is denoted by $IdR.\tau'$, and the tag descriptor of the strong identifier inτ' as $\tau'.tag$.
An important notion in our approach is represented by the *proof of knowledge.* Let m be the value of a strong identifier and let M be the corresponding semantically secure commitment. We say that principal P can provide a proof of knowledge by providing a verifiable cryptographic token used in a ZKP protocol (see Section 3.1) which asserts that P knows: (1) the actual value of m, and (2) cryptographic secret(s) associated with M. This is denoted as $M \rhd P$ (read "M belongs to P").
The conditions according to which a principal P can prove ownership of a given IdR are dictated by a policy π. Such policy expresses which of the committed strong identifiers in a given IdR need to be proven in order to ensure ownership of the whole record. Policy π can either be specified separately by the various registrars or it can be globally

Identity Assurance Function	Description
LocalConsistency(τ, IdR, P)	To check whether weak identifiers in τ are locally consistent.
FederationDuplicate Detection(τ, IdR,P)	To check whether duplicate values of the strong identifier commitment exists.
ExternalValidation(τ, IdR,P)	To validate the strong identifier m appearing in τ, by contacting the issuer authority.

Table 1: Identity Assurance Functions

Phase	Identity Assurance Functions	Cryptographic Protocols
Enrollment	LocalConsistency; FederationDuplicate Detection; ExternalValidation	Protocol 1 §3.2
Update	LocalConsistency; FederationDuplicate Detection	Protocol 1 §3.2
Usage	ExternalValidation	Protocols 2, 3(a,b) §3.3, Protocols 4(a,b) §3.4

Table 2: Roadmap of the Identity Protocols

defined as a part of the federation agreement policies. Ownership of IdR by P proven according to policy π is denoted by $IdR_P \blacktriangleright_\pi P$ and is formally defined as follows.

Definition 2.1 (Ownership of Identity Record) Let π be a policy and let ψ be the set of tags to be verified according to π. A principal P registered at registrar R can prove ownership of an identity record IdR_P if, for each $t \in \psi$ a $\tau \in IdR_P$ exists, such that $\tau.tag = t$ and the following conditions hold: (i) $M \rhd P$; (ii) P can provide proof that signature $\tau.\sigma$ is valid ; (iii) $\tau.validity\text{-}assure =$ 'A'; (iv) $\tau.ownerhip\text{-}assure =$ 'A' or 'B'. □

The definition states that ownership assurance of an identity record is the result of the ownership assurance of each tuple referred in the policy. That is, for each tuple indicated, it is required that the signature on the strong identifier be verified, the validity assurance of the strong identifier is set to 'A' and the ownership-assurance is set to level 'A' or 'B'.

Notice that identity assurance, that is, the main goal of our approach, is a broader concept than only ownership and validity assurance. Identity assurance is also related to the *consistency* of the IdR. Consistency of IdR is both a local and a global concept. Local consistency deals with the information recorded by a specific IdR. In order to be consistent, the collected strong and weak identifiers should qualify an individual with no evident errors. That is, no conflicting attribute values should be collected in a same IdR. For instance, if the weak identifier "age" appears in different weak identifiers, it should have the same value. Global consistency requires that no strong identifier be associated with multiple principals, as they are typically unique, unless some specific conditions, detailed below, hold. We formalize the concept of consistency in the following definition.

Definition 2.2 (Consistency of Identity Records) Let $\mathcal{F} = \langle \mathcal{P}, \mathcal{SP}, \mathcal{IP}, \mathcal{R} \rangle$ be a federation. Let P be a principal in $\mathcal{P}$, enrolled at registrar $R \in \mathcal{R}$. Let IdR_P be an identity record of P. IdR_P is *consistent* if it is *locally consistent* with respect to R and *globally consistent* with respect to $\mathcal{F}$.
Idr_P is locally consistent with respect to R if $\forall \tau_i \in Idr_P, \nexists \tau_i.W_{ih}, \tau_j.W_{jk}, j \neq i | tag_{W_{ih}} = tag_{W_{jk}}$ and $\tau_i.W_{ih} \neq \tau_j.W_{jk}$.
Idr_P is globally consistent if one of the following conditions holds:

1. $\nexists P_j \in \mathcal{P}, P \neq P_j | \widehat{M_i} = \widehat{M_j}$ and $\tau_j \in Idr_{P_j}$ and $Idr_{P_j} \blacktriangleright_\pi P_j$
2. if $\exists P_j \in \mathcal{P}, P \neq P_j | \widehat{M_i} = \widehat{M_j}$ and $\tau_j \in Idr_{P_j}$, then $Idr_{P_j} \blacktriangleright_\pi P$. □

Local consistency checks are executed in order to verify that there are no weak identifiers in the same IdR with the same descriptor tags and different values. For example the value of the weak identifier tagged by *firstname* should the same in all strong identifiers in which it appears. With respect to global consistency, the first condition requires that no duplicates of strong identifiers exist in a federation; in most cases, a strong identifier is unique to an individual and a duplicate may represent an inconsistency. However, as stated by the second condition, if a duplicate is detected in another identity record, then the principal should be able to prove ownership of this identity record. Therefore, our approach also allows multiple principals to commit the same value for strong identifiers, under the condition that ownership of the duplicate strong identifier can be proven. For instance, we let two principals share a same credit card, if both can prove the ownership of the corresponding IdR.
Identity assurance can be achieved enforcing specific checks at registrars. We have designed a set of functions implementing the controls needed to verify identity assurance. These functions are summarized in Table 1 and details are provided in Appendix A.

2.3 Management of Identity Records

The management of identity in our approach is characterized by three main phases: enrollment, during which individuals register with the federation; usage of identity, requiring the verification of identity information; update of identity, allowing individuals to modify their IdR. In what follows we discuss such phases in more details.

Enrollment process. Individuals are required to submit strong identifiers to enroll in the federation, according to the policy of the registrar. Since our approach is based on multi-factor verification of identity, we assume that a minimum number of identifiers is needed to actively participate in the federation. The exact type and numbers of identifiers to register is part of the registrar policy and is assumed to be publicly available from the registrar. For example, a registrar may require that a principal submits at least three strong identifiers for enrolling in the federation. A policy language for expressing enrollment conditions is required, which we are currently developing. According to its policy the registrar executes the ExternalValidation function (see Table 1) according to a *pull* strategy[2] in order to obtain an assurance level equal to 'A' for the committed values required by this policy. Further, the registrar will execute the LocalConsistency and FederationDuplicateDetection functions to check if the strong identifiers are locally consistent and if one or more duplicates are present in any other IdR. In case duplicates are found, the principal is asked to prove ownership of the IdR conveying the duplicate. If the principal is unable to provide proof of the ownership, then the enrollment is aborted. Once this check is completed the newly created IdR is consistent according to Definition 2.2. At the time of enrollment, ownership of the claimed strong identifiers also needs to be ensured. Individuals can enroll either through a face-to-face registration process or online. Face-to-face registration is executed when an individual physically enrolls at a specific registrar office by showing credentials proving its identity and hence proving the ownership of the claimed identifiers. For example the individual can go to the physical location acting as registrar where it shows its SSN card. A trusted official in the registrar confirms the validity of the physical card and supervises the enrollment procedure ensuring that the correct SSN number is entered into the system and thus stored as a commitment. Here the individual trusts the registrar's system not to store extraneous information other than the commitments needed for the enrollment. Face-to-face registration guarantees ownership and therefore has $ownership\text{-}assure$ level equal to 'A'. An alternative approach is online registration. Online registration is based on the concept of *digitally introduction* by strongly identified principals. The principal acting as a*grantor* for the enrolling principal needs to have a valid IdR (say $IdR_{grantor}$) and $IdR_{grantor} \blacktriangleright_{\rho} grantor$ where ρ is the authentication policy of the registrar. The *grantor* essentially asserts that the enrolling principal actually possesses the strong identifiers the commitments of which are presented to the registrar. Since such an assurance is based on the level of trust of *grantor*, the $ownership\text{-}assure$ of this type of registration has a level equal to 'B'. Online registration will thus require the individuals to present in addition to the minimum number of strong identifier commitments, one assertion from at least one grantor.

IdR Update process. The IdR may have to be updated for 1) adding strong identifier commitments, 2) revoking strong identifier commitments and 3) changing the$validity - assure$ status of the strong identifier commitments in the IdR. When a principal P requires adding a strong identifier commitment M_{new} to its IdR (say IdR_P) at registrar R, it presents an identity tuple τ_{new}, collecting M_{new} and a set of weak identifiers W_{new}; the following steps are then executed:

1. P proves ownership of IdR_P based on the policy of R, denoted asπ_R. Hence $IdR_P \blacktriangleright_{\pi_R} P$.
2. The LocalConsistency(τ_{new}, IdR,P) function is executed to confirm local consistency of the new identity tuple with respect to the weak identifiers.
3. The FederationDuplicateDetection(τ_{new}, IdR,P) function is executed in order to check for duplicates of $\widehat{M}_{new}$. If a duplicate is found at another IdR, say IdR_{dup}, then P has to prove $IdR_{dup} \blacktriangleright P$.
4. The $ownership\text{-}assure$ level is determined based on the following three cases. If P is performing a face-to-face update, then $ownership\text{-}assure$ = 'A', else if P uses digital introduction $ownership\text{-}assure$ = 'B', and finally if no assurance is given,$ownership\text{-}assure$ = 'U'.
5. M_{new} is then signed by R in order to generate the signature σ_{new}.
6. Finally τ_{new} is added to the IdR with the $validity - assure$ of each identifier set as unknown.

[2]When the registrar needs to determine the validity status before the value of commitment M is signed and used in the federation, it checks the validity immediately. This refers to the pull strategy. Details are provided in Appendix A

Revocation of strong identifier commitment is executed by changing the $validity - assure$ to 'F'. More generally, changes in the level of validity assurance are the result of the execution of ExternalValidation function in a push mode[3].

Usage of IdR. Any combination of the signed values of the commitments can be required for authentication purposes by a SP. The content of the IdR must thus be available when the principal requests service from a SP. Availability can be ensured according to two strategies. One strategy is to let principal P indicate its registrar, so that SP can directly retrieve the required content of P's IdR. This requires the registrar to be online. Another option is to let P store the content of the IdR in a - per tuple - signed structure called $RegCert$. $RegCert$ contains all fields of the IdR and can be encoded in any portable language, like XML [20]. To ensure security against stale $RegCert$'s we limit the lifetime of the $RegCert$ so that fresh values are reloaded from the registrar after a period of time specified by the federation security parameters.

Table 2 provides a roadmap indicating which identity assurance functions and cryptographic protocols are used in the different stages. The details of the protocols are presented next in Section 3.

3 Protocols

In this section we propose our protocols to enable principals to enroll with registrars, and authenticate using privacy preserving multi-factor authentication mechanism. More specifically, we provide detailed protocols based on ZKPK which are employed in the enrollment of the strong identifiers, and the signing of the commitments. We also show how such commitments can be used in the verification phase. Our approach is based on aggregation techniques of committed values to provide flexible and efficient zero-knowledge proofs. We also extend the aggregation protocols to provide signature verification with hidden commitments in Section 3.4.

3.1 Preliminary Concepts

Following are the preliminary concepts regarding commitments, aggregate signatures and zero-knowledge proofs, and the corresponding protocol notation.

Pedersen commitments: Let g and h be generators of group G of prime order q. A value m is committed by choosing r randomly from $\mathbb{Z}_q$ and giving commitment $C = g^m h^r$. Commitment C is opened (or revealed) by disclosing m and r, and the opening is verified by checking that$C$ is indeed equal to $g^m h^r$. A prover can prove by using zero-knowledge proof that it knows how to open such commitment without revealing either m or r.

Bilinear maps: For a security parameter k, let q be a prime of length k, and G_1, G_2, G_T be groups of order q. Suppose $g_1 \in G_1$, $g_2 \in G_2$ to be generators. Function $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear mapping satisfying the following properties:

1. Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g_1, g_2) \neq 1 \in G_T$.
3. There exists a computable isomorphism ψ from G_2 to G_1, with $\psi(g_2) = g_1$.

Bilinear aggregate signatures: The aggregate signature concept has been proposed by Boneh *et. al.* [2] who provide an efficient aggregate signature scheme referred to as BGLS from bilinear maps. Informally, aggregate signatures are signatures that allow multiple signatures to be aggregated into one signature that is verifiable with respect the public keys of the signers and the signed messages. Formally, the BGLS scheme consists of five algorithms: $KeyGen, Sign, Verify, Aggregate$ and $AggVer$. Any principal P uses $KeyGen$ to generate the private and public key pair (χ, v) such that $v = g_2^{\chi}$, where $g_2 \in G_2$, χ is the private key, and v is the public key.

The $Sign$ algorithm computes the signature on input message m_i. Its main step is the mapping of m_i into G_1 by a mapping $h : \{0,1\}^* \rightarrow G_1$. The output message $\sigma_i = h(m_i)^{\chi} \in G_1$ is the signature for m_i.

The $Aggregate$ algorithm aggregates the signatures $\sigma_1, \sigma_2, \cdots, \sigma_t$ for t different messages $m_1, m_2, \cdots, m_t$ into one signature $\sigma = \prod_{i=1}^{t} \sigma_i$.

The $AggVer$ algorithm verifies signature and works like the*Aggregate* signature algorithm. For a set $m_1, m_2, \cdots, m_t$

[3]Push mode of validation is when any SP in the federation receiving the actual value of the strong identifier$m$ consults the issuer and subsequently sends the validation result to the registrar storing the corresponding IdR

of different messages, and public keys $v_1, v_2, \cdots, v_t$ and a signature σ, the verifier checks if $e(\sigma, g_2) = \prod_i e(h_i, v)$, where $h_i = h(m_i)$ and e is the bilinear mapping.

Zero-knowledge proof of knowledge: In our approach we use the techniques by Camenisch and Stadler in [9] for the various ZKPK of discrete logarithms and proofs of the validity of statements about discrete logarithms. We also conform to the same notation as [9]. For instance to denote the ZKPK of values α and β such that $y = g^{\alpha}h^{\beta}$ holds, and $u \leq \alpha \leq v$, we use the following notation:

$$PK\{(\alpha, \beta) \, : \, y = g^{\alpha}h^{\beta} \wedge (u \leq \alpha \leq v)\}$$

The convention is that Greek letters denote quantities the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, the proof protocol is described by just pointing out its aim while hiding all details.

3.2 Commitments and Signatures at Enrollment

As stated earlier, there are two types of commitment created for each strong identifier at the time of enrollment: the deterministic commitment (denoted by $\widehat{M}$) and the semantically secure one (denoted by M). Only M is signed and stored in the IdR, while $\widehat{M}$ is used to detect duplicates. In the following protocol we show how these commitments can be created by the principal and how the principal can prove that the two commitments refer to the same secret m. Finally, we illustrate how M is signed by the registrar. Formally, the protocol is composed of the following steps.

Protocol 1: Computing a signature on a information-theoretic hiding committed value.

1. *Registrar's parameters.* The registrar runs generation algorithm GenKey on input 1^k to generate the public parameters: a prime q of length k, three groups G_1, G_2, G_T of order q. Two generators g_1, h_1 in G_1 are specified such that $\log_{g_1} h_1$ is unknown. An additional generator $g_2 \in G_2$ is needed, as well as a secret key $\chi \in \mathbb{Z}_q$ and the public key $v = g_2^{\chi}$. The resulting set of public parameters is $(G_1, G_2, G_T, g_1, h_1, g_2, v)$.
2. *Commitment of a value* $m \in \mathbb{Z}_q$. The principal chooses a $r \in \mathbb{Z}_q$, and computes $M = g_1^m h_1^r$. The principal also computes $\widehat{M} = g_1^m$.
3. *Zero-knowledge proof of committed value*. The principal gives a ZKPK of opening of the commitments M and $\widehat{M}$ and that they commit the same secret m.

$$PK\{(\alpha, \beta) \colon M = g_1^{\alpha}h_1^{\beta} \ \wedge \ \widehat{M} = g_1^{\alpha}, \ \alpha, \beta \in \mathbb{Z}_q\}$$

4. *Signing of a committed value*. After performing the security checks on the committed value, the registrar executes the $Sign$ algorithm on the commitment M to output M^{χ} as the signature where χ is the secret key of the registrar.

3.3 Multi-factor Authentication Verification

Assume that principal P requests a service from a SP which requires P to first authenticate by proving that it knows how to open a specified set of commitments. To indicate this set of commitments a set of tags is given which is denoted by ψ_{proof}. Moreover, to be authorized for the service the SP usually requires the principal to open or reveal in clear values of some of the strong identifiers in its IdR. We denote this set of tags as ψ_{open}. The signatures and public parameters are retrieved from the principal's IdR or $RegCert$, introduced in Section 2.3. Since multiple commitments have to be verified and proven, and can change according to the specific SP's policy, ψ_{proof} and ψ_{open} are not predetermined. As such, we need an aggregation technique to combine any given combination of commitments and signatures for verification. In what follows we illustrate how this can be achieved. Precisely, Protocols 2 and 3 provide aggregate proof of knowledge of the commitments corresponding to ψ_{proof} and ψ_{open} respectively. The protocols are two party computations, in which the principal is the prover and the SP is the verifier (we use the two terms interchangeably).

Protocol 2: Proving aggregated signature on committed values. The principal performs the ZKP of the aggregated commitments corresponding to the tags given in ψ_{proof} and aggregated signature for verification.

1. *Principal's aggregation*. Let $\sigma_1, \sigma_2, \cdots, \sigma_t$, be the signatures corresponding to the tags in ψ_{proof}. The principal aggregates the signatures into $\sigma = \prod_{i=1}^{t} \sigma_i$, where σ_i is the signature of committed value $M_i = g_1^{m_i} h_1^{r_i}$. It also computes $M = \prod_{i=1}^{t} M_i = g_1^{m_1+\cdots+m_t} h_1^{r_1+\cdots+r_t}$. Finally the principal sends σ, M, M_i for $1 \leq i \leq t$ to verifi er.

2. *Zero-knowledge proof of aggregate commitment*. The principal and the verifi er SP carry out the following ZKP protocol:
$$PK\left\{(\alpha, \beta) \colon M = g_1^{\alpha} h_1^{\beta},\ \alpha, \beta \in \mathbb{Z}_q\right\}$$

3. *Verifi cation of aggregate signature* After the verifi er accepts the zero-knowledge proof of the commitments, it checks if the following verifi cations succeed:
$$M = \prod_{i=1}^{t} M_i \qquad \text{and} \qquad e(\sigma, g_2) = e(M, v)$$

Only if steps 2 and 3 are successful, the SP will consider the signatures as valid. Step 2 provides an effi cient way of performing the ZKPK for each M_i in an aggregated manner which avoids carrying out a proof for each of the M_i's. Similarly, the aggregate signature in step 3 provides an effi cient way of checking the signature for each of the commitment indicated in ψ_{proof} .

Protocol 3a: Opening the committed value. In order to satisfy SP's request to open in clear the principal's strong identifi ers and verify the corresponding signatures, the principal has to show the corresponding values along with the commitments as given in ψ_{open}, as well as the aggregated signature.
The protocol relies on a random oracle hash function H which is known to all entities. Formally,

1. *Principal's aggregation and preparation*. Upon SP's requirement to show $m_1, m_2, \cdots, m_t$
   (a) The principal aggregates the signatures $\sigma_1, \sigma_2, \cdots, \sigma_t$ into $\sigma = \prod_{i=1}^{t} \sigma_i$ where σ_i is the signature of committed value $M_i = g_1^{m_i} h_1^{r_i}$.
   (b) Using H, principal computes the random values $(x_1, x_2, \ldots, x_t) = H(m_1 \parallel \cdots \parallel m_t \parallel M_1 \parallel \cdots \parallel M_t)$[4]. It also computes $r = \sum_{i=1}^{t} r_i x_i$ which is used in the zero-knowledge proof in the next step.

   The principal sends $m_1, \ldots, m_t, M_1, \ldots, M_t, (x_1, x_2, \ldots, x_t)$ to the verifi er SP.

2. *Zero-knowledge proof of aggregate commitment*. The principal and SP compute $M = \prod_{i=1}^{t} M_i^{x_i}$ and $\widehat{M} = \prod_{i=1}^{t} g_1^{m_i x_i}$ and carry out the following ZKPK:
$$PK\{(\beta) \ : \ M/\widehat{M} = h_1^{\beta}\}$$

3. *Verifi cation of aggregate signature* After the verifi er receives$M_1, M_2, \cdots, M_t$, and accepts the zero-knowledge proof of the commitments, it checks if
$$M = \prod_{i=1}^{t} M_i \qquad \text{and} \qquad e(\sigma, g_2) = e(M, v)$$

Only if all above checks are successful, SP validates the signatures and the values $m_1, m_2, \cdots, m_t$.
Steps 1-3 are executed to ensure that the opened values $m_1, m_2, \cdots, m_t$ are the same as the the ones originally committed $\{M_1, \ldots, M_t\}$. Moreover the knowledge of the m_i's is not suffi cient to perform a successful proof of knowledge since also the committed random value r_i is needed to complete the proof. This requirement prevents possible misuse of the m_i's by the verifi er SP. Note also that step 1b) corresponds to the challenge creation in a random oracle model, to enable a non-interactive ZKP according to the Fiat-Shamir [15, 28] paradigm.

Protocol 3b: Hidden Strong Identifi er Validation In the following protocol we tackle the specifi c case of transactions where the actual values of strong identifi ers are not required to be released to the SP. As an example, a widely

[4]Here the random function H is from $\{0,1\}^*$ onto $\{0,1\}^{ck}$, where c is a constant, k is the security parameter. For any $x \in \{0,1\}^*$, let $y = H(x)$. For any given $ck > t > 0$, let $m = \lfloor |y|/t \rfloor$, to denote x_i is substring in y of length m for $1 \leq i \leq t-1$, and x_t is the suffi x of y with length of $|y| - (t-1)m$, such that $y = x_1 x_2 \ldots x_t$. We denote it as $(x_1, \ldots, x_t) = H(x)$.

used strong identifier is the credit card number (CCN), the actual value of which may not be needed by a SP if the issuer of the CCN, possibly a bank, can credit the SP with the required amount of money. In the protocol the strong identifier is in fact revealed only to the issuer of that identifier and the deterministic commitments are sent to the SP instead of the clear values. Moreover, an additional cryptographic token is passed to the SP which is forwarded to the issuer. Here, we assume that the principal knows the public key of the issuer. Formally,

1. *Principal's aggregation.* Upon SP's requirement to provide deterministic commitments for strong identifiers $m_1, m_2, \cdots, m_t$, the following steps are executed:
   (a) The principal aggregates the signatures $\sigma_1, \sigma_2, \cdots, \sigma_t$ into $\sigma = \prod_{i=1}^{t} \sigma_i$ where σ_i is the signature of committed value $M_i = g_1^{m_i} h_1^{r_i}$.
   (b) The principal computes $\widehat{M_i} = g_1^{m_i}$ for $1 \leq i \leq t$ and $r = \sum_{i=1}^{t} r_i x_i$ which is used in the ZKP in the step 2. Using H, it also computes the random values $(x_1, x_2, \ldots, x_t) = H(\widehat{M_1} \parallel \cdots \parallel \widehat{M_t} \parallel M_1 \parallel \cdots \parallel M_t)$.
   (c) The principal constructs the following message for each issuer.
   $$\mathsf{Enc}_{Issuer}(\{m_1, \ldots, m_t, timestamp\})$$
   The principal sends $\{\widehat{M_1}, \ldots, \widehat{M_t}\}$, $\{M_1, \ldots, M_t\}$, σ and $\mathsf{Enc}_{Issuer}(\{m_1, \ldots, m_t, timestamp\})$ to SP.
2. *Zero-knowledge proof of aggregate commitment.* The principal and SP compute $M = \prod_{i=1}^{t} M_i^{x_i}$ and $\widehat{M} = \prod_{i=1}^{t} (\widehat{M_i})^{x_i}$ and carry out the following ZKPK:
   $$PK\{(\beta) : M/\widehat{M} = h_1^{\beta}\}$$
3. *Verification of aggregate signature* The principal sends σ to the SP which can verify the signature as follows:
   $$M = \prod_{i=1}^{t} M_i \qquad \text{and} \qquad e(\sigma, g_2) = e(M, v)$$

Only if steps 2 and 3 are valid, validator SP will accept the truth of the signatures and send the message.[5] SP will append all the deterministic commitments to the message for verification by the issuer as follows:

$$\mathsf{Enc}_{Issuer}(\{m_1, \ldots, m_t, timestamp\}), (\widehat{M_1}, \cdots, \widehat{M_t})$$

The issuer can then verify each of the m_i's for its validity as well as the freshness of the message using the $timestamp$. It can also check that m_i, $1 \leq i \leq t$, corresponds to the deterministic commitments as checked by the SP.

3.4 Signature Verification with hidden commitments

The tags associated with committed strong identifiers may potentially leak information about the individual. For example, if a SSN number is enrolled it would imply that the individual has some source of income within the U.S. This may be not be acceptable in some scenarios, as highlighted by next example.

Example 1 *Consider a registrar R_{govt} which enrolls only the government officials and requires high identity assurance for each of the commitments it signs. The commitments of the individuals in R_{govt} may correspond to the role of the individual in the organization. Suppose principal P_A has enrolled its* "secret service officer ID number" *with R_{govt} and has received a signed commitment corresponding to it. Consider now a hotel Ht which provides discounts to government officials. P_A while booking a room at Ht wants to apply for the discount. For this purpose P_A needs to prove the commitment signed by R_{govt}. If the commitment and the corresponding tags are given in clear, they will leak information regarding P_A being a secret service officer. Therefore, it is desirable that P_A be able to prove that it has enrolled* some *identifier with R_{govt} without revealing the exact commitment or the tag associated with it.*

[5] If more than one issuers for are required to be contacted, more bilinear mapping computations will be involved in the step 3.

The above example can be generalized to the case where multiple commitments should be proven issued by a known registrar without actually knowing the values of the commitment itself or the corresponding tags. To achieve this feature we introduce a new cryptographic primitive.

Protocol 4a: Integrating the zero-knowledge proof into the verification

1. *Principal's aggregation*. Upon SP's requirement to prove $\sigma_1, \sigma_2, \cdots, \sigma_t$, the principal aggregates the signatures into $\sigma = \prod_{i=1}^{t} \sigma_i$ where σ_i is the signature of committed value $M_i = g_1^{m_i} h_1^{r_i}$. The principal also computes $m = m_1 + \cdots + m_t \pmod q$, $r = r_1 + \cdots + r_t \pmod q$.

2. *Zero-knowledge proof of aggregate commitment*. The principal sends σ to SP, and carries out the following ZKP protocol with SP:
$$PK\left\{(\alpha, \beta) \colon e(\sigma, g_2) = e(g_1, v)^{\alpha} e(h_1, v)^{\beta},\ 0 < \alpha, \beta < q\right\}$$

Note that the only information sent by the principal is σ, while in protocol 3 also the tags and the commitments were known. If the above checks are valid, verifier SP will validate the signatures.

Protocol 4b: Zero-knowledge proof the aggregated signature.
Protocol 4a is a secure protocol which hides the tags and commitments. However, it is not a ZKP protocol because different instances of the signature verification performed by the same principal can be linked by the SP since the signatures themselves are deterministic. Moreover, if the principal had revealed in an earlier transaction, its M_i (and possibly the tag associated with it) then the SP can link the σ_i with it. To address this we provide a final protocol variant where even the actual signature is not revealed. More specifically, a randomized signature is used to verify that the original signature has been issued by a given registrar. Note that since the signatures are randomized and the proof of validity is zero-knowledge, one signature cannot be distinguished from the other. The succinct ZKPK is to convince the verifier of possession of knowledge of one signature on a committed value, rather than which one it is. The final submitted value is independent of any of the actual signatures. Therefore it is necessary that only one signature be verified. Any further verification of additional randomized signatures does not provide any additional information. This protocol has the advantage of assuring that a principal remains unlinkable and anonymous even if it had initially revealed its strong identifiers and commitments to the verifying SP.

1. *Principal's aggregation*. Upon SP's requirement to prove a signature σ, principal chooses $r \in \mathbb{Z}_q$ at random, and sends the messages $\delta := \sigma^r$ to SP.

2. *Zero-knowledge proof of aggregate commitment*. The principal carries out the following zero-knowledge proof protocol with the verifier SP:
$$PK\left\{(\alpha, \beta) \colon e(\delta, g_2) = e(g_1, v)^{\alpha} e(h_1, v)^{\beta},\ 0 < \alpha, \beta < q\right\}$$

4 Analysis of the Protocols

In this section we analyze our solution. We first provide a formal analysis of the security of the cryptographic protocols introduced in Section 3. We then evaluate the computational complexity of the main protocols characterizing our approach. Finally, based on the properties of our protocols and on the identity assurance methodologies presented in Section 2 we briefly analyze how identity theft prevention is achieved in the resulting identity system.

4.1 Security Analysis of the Protocols

Before proving the security properties of our protocols, we identify the properties that characterize the cryptographic techniques used. The security of such cryptographic techniques relies on the assumption of co-gap Diffie-Hellman (co-GDH) problem [2], which is summarized as follows.
For multiplicative cyclic groups G_1, G_2, G_T of order q, let g_1 be a generator of G_1 and g_2 be a generator of G_2. Let ψ be a computable isomorphism from G_1 to G_2, with $\psi(g_1) = g_2$ and e a computable bilinear map e: $G_1 \times G_2 \rightarrow G_T$. Note that ψ and e can be computed efficiently. Co-GDH gap problem is relating two problems used in cryptography which are as follows:
Decisional Co-Diffie-Hellman problem: Given $\langle g_1, g_2, g_1^a, g_2^b, g_2^c \rangle$ for some $a, b, c \in \mathbb{Z}_q^*$, to decide if $c = ab \mod q$.
Computational Co-Diffie-Hellman problem: Given $\langle g_1, g_2, g_1^a, g_2^b \rangle$ for some $a, b \in \mathbb{Z}_q^*$, to compute $g_2^{ab} \in G_2$.

Groups G_1, G_2 are said to be **Co-GDH groups** if there exists an effi cient algorithm to solve the Decisional Co-DH problem and there is no polynomial-time (in $|q|$) algorithm to solve the Computational Co-DH problem. The existence of a cryptographic bilinear map ensures the existence of Co-GDH groups.
Since the co-GDH assumption implies discrete logarithm assumption, the results stated in the next lemma concerning the ZKPs appearing in Protocols 1, 2, 3a, 3b and 4a are derived from [11, 12, 9].

Lemma 4.1 *Let G_1, G_2 be Co-GDH groups of prime order q with respect to generators $g_1 \in G_1$ and $g_2 \in G_2$. Let $h_1 \in G_1$ be a generator with $\log_{g_1} h_1$ unknown. The ZKPK appearing in protocols 1, 2, 3a, 3b and 4 hold true for the specifi ed parameters. More precisely:*

1. *Step 3, protocol 1, and step 2 in protocol 3b are ZKPs of knowledge of the values $m_i, r_i \in \mathbb{Z}_q$ such that the same m_i is committed in both $\widehat{M}_i$ and M_i.*
2. *Step 2 in protocol 2 is a ZKPK of the values $\sum m_i \mod q$, $\sum r_i \mod q$.*
3. *Step 2 in protocols 3a and 3b is a ZKPK of the values $\sum(m_i \times x_i) \mod q$, $\sum(r_i \times x_i) \mod q$ where $x_i \in \mathbb{Z}_q$ are random challenges.*
4. *Step 2 in protocol 4a is a ZKPK of the values $\sum m_i \mod q$, $\sum r_i \mod q$ satisfying the signature verifi cation relation.*

We now show that all protocols are two-party secure computations. Security is ensured by proving *correctness* and *unforgeability* of each protocol.
Correctness of protocols means that honest users can, with correct data, carry out the protocols successfully, while unforgeability guarantees that an adversary, with forged data, cannot execute the protocols successfully. Our results on unforgeability for protocol 2 are derived from Lemma 4.3.
Proving the security of the fi rst protocol is straightforward. The following lemma is given.

Lemma 4.2 *In protocol 1, let G_1, G_2, be Co-GDH groups of prime order q with respect to generators $g_1 \in G_1$ and $g_2 \in G_2$. Let $h_1 \in G_1$ be a generator, with $\log_{g_1} h_1$ unknown. The protocol is secure.*

The truth of Lemma 4.2 is based on the statistical hiding and computational binding properties of Pedersen commitments. Therefore, signatures and aggregation computed on such commitments will continue to hold those properties. The independent techniques employed in this protocol are conventional, and have been investigated separately in several related works [4, 22, 26, 27]. The correctness proofs are similar to the ones elaborated in Theorem 4.4 and are therefore omitted.

Lemma 4.3 (Unforgeability of Aggregation of Pedersen Commitment) *Let G be a group of prime order q, in which the discrete logarithm is hard to compute. Elements $g, h \in G$ are generators with $\log_g h$ unknown. $M_i = g^{m_i} h^{r_i}$ are Pedersen commitments to messages $m_i \in \mathbb{Z}_q$, and random numbers $r_i \in \mathbb{Z}_q$, with $1 \leq i \leq t$. Let $M = \prod_{i=1}^{t} M_i$. Then, it is infeasible, given only $M_1, M_2, \ldots, M_t$, to compute m, r such that $M = g^m h^r$ if at least one of m_i or r_i is unknown.*

Proof. Suppose that $m_1, \ldots, m_{t-1}$ and $r_1, \ldots, r_t$ are known, and m_t is unknown. If adversary can compute $m, r \in \mathbb{Z}_q$, where $m = \sum_{i=1}^{t} m_i, r = \sum_{i=1}^{t} r_i$ such that $M = g^m h^r$, then it can get $g^{m_t} g^{m_1+m_2+\cdots+m_{t-1}} h^{r_1+\cdots+r_t} = g^m h^r$, which means $g^{m_t} = g^{m-m_1-\cdots-m_{t-1}} h^{r-r_1-\cdots-r_t}$, that implies $m_t \equiv m - m_1 - \cdots - m_{t-1} \mod q$ and $r \equiv r_1 + \cdots + r_t \mod q$. This in turn implies that the adversary can solve the discrete logarithm $g^{m_t} = M/(g^{m_1+\cdots+m_{t-1}} h^r)$ with respect to g. Since m_t is an arbitrary element in $\mathbb{Z}_q$, that is contradictory with respect to the discrete log problem (DLP) assumption. □

Theorem 4.4 *For co-GDH groups G_1, G_2, the protocol 2 is a secure two-party computation.*

Proof. We show that prover needs to know all the committed values and that the associated signatures need to be valid in order to successfully execute the protocol.
Correctness: Let $M_i = g_1^{m_i} h_1^{r_i}$, $\sigma_i = M_i^{\chi}$, then $M = \prod_{i=1}^{t} M_i = \prod_{i=1}^{t} g_1^{m_i} h^{r_i} = g_1^m h_1^r$, where $m = \sum_{i=1}^{t} m_i$, $r = \sum_{i=1}^{t} r_i$. The prover is able to execute

$$PK\left\{(\alpha, \beta)\colon M = g_1^{\alpha} h_1^{\beta},\ \alpha, \beta \in \mathbb{Z}_q\right\}$$

		Protocol 2	Protocol 3a	Protocol 3b	Protocol 4a	Protocol 4b
Our	provers	$2+2$	$3+2$	$3+2$	2	2
Protocols	verifiers	3	$2t+3$	$2t+3$	3	3
Without	provers	$2t$	$4t$	$4t$	$2t$	$\times$
Aggregation	verifiers	$3t$	$5t$	$5t$	$3t$	$\times$

Table 3: Comparison on the number of exponentiations for proving t factors.

with the knowledge of $\alpha = m$ and $\beta = r$ according to Lemma 4.1.
To prove correctness for step 3 of the protocol, which verifies the validity of the aggregated signature, we note that $\sigma = \prod_{i=1}^{t} \sigma_i = \prod_{i=1}^{t} M_i^{\chi}$, and

$$e(\sigma, g_2) = e\left(\prod_{i=1}^{t} M_i^{\chi}, g_2\right) = \prod_{i=1}^{t} e(M_i, g_2)^{\chi} = \prod_{i=1}^{t} e(M_i, v) = e(M, v).$$

Unforgeability: We prove this property by showing that if the prover does not know even one of the messages $\{m_i\}_{1\leq i\leq t}$ and $\{r_i\}_{1\leq i\leq t}$, OR one of σ_i, $1 \leq i \leq t$, is not valid, then protocol fails.
If the prover does not know all the secrets and the proof is executed successfully, this would mean that there exists a knowledge extractor that can extract two values m' and r' such that $M = g_1^{m'} h_1^{r'}$. However, according to Lemma 4.3 this is infeasible.
For the case in which any one of the signatures is not valid, the step 3 of the protocol will not succeed because of the security of the aggregated signature as given in [2]. □

Theorem 4.5 *For co-GDH groups G_1, G_2, the following conclusions hold:*

1. *Protocol 3a is a secure two-party computation. It guarantees that 1) principal has knowledge of values r_i, 2) the values m_i are correctly committed in M_i, and 3) signatures $\sigma_1, \sigma_2, \ldots, \sigma_t$ are valid.*
2. *Protocol 3b is a secure two-party computation. It guarantees that 1) principal has knowledge of values r_i, 2) the values committed in M_i and $\widehat{M_i}$ are the same, and 3) signatures $\sigma_1, \sigma_2, \ldots, \sigma_t$ are valid.*

Proof is provided in Appendix B.1.

Theorem 4.6 *For co-GDH groups G_1, G_2, Protocol 4a is a secure two-party computation in random oracle model.*

Proof is provided in Appendix B.2.

Theorem 4.7 *The protocol 4b is a ZKP of a signature on a message under signature scheme of protocol 1.*

Proof is provided in Appendix B.3.

4.2 Efficiency Evaluation of the Protocols

Our ZKPK is based on the hardness of discrete logarithm which is implied by the assumption of co-GDH groups. To compute the proof of $PK\{(\alpha, \beta) : y = g^{\alpha} h^{\beta}, \alpha, \beta \in \mathbb{Z}_q\}$, 5 exponentiations are used [13]. If separate proof of the knowledge for t commitments were used, then $5t$ exponentials would need to be computed. In some of our protocols, we reduce the number of exponentiations to a constant that does not depend on the number of commitments to be proved. In our protocols, principals always need compute a constant number of exponentiations, while the verifier's computation of exponentials is mostly dramatically reduced (see Table 3). These simple considerations prove the efficiency and practical features of our approach. Table 3 reports a comparison of the exponentiations computed by the principals or provers and verifiers in our aggregate protocols and in the case when they are not aggregated.
Since we adopted the Pedersen commitment and the short signature from [3], our signatures on commitments are short and the storage complexity is smaller than the ones computed with existing techniques [8]. As an example, even the simplest version of signature is three times length than ours.
Camenisch *et al.* also considered signatures on the commitments on a set of messages (see [8], page 10 and Theorem 3.) Compared to their methods, our approach is more flexible in that whenever n messages are committed for a user, the user is able to to prove $2^n - 1$ many combinations of them which does not appear possible in the scheme by Camenisch *et al.* Since we make use of the aggregation signatures developed in [2] to sign the Pedersen's commitments, the verification of the signature is more efficient than if the verification were executed separately.

Moreover, in our case, since the signatures stored in a particular IdR are assigned only by the registrar which enrolls it, the verification becomes even more cost effective. We compare to the case where aggregated proofs are not used. Non aggregated proofs need $2t$ many bilinear mapping computations for the verification of t signatures, while each of our protocols needs only 2 bilinear mappings, which is again a constant and is independent of the number of signatures proved.
To summarize: No matter how many factors needing to be proved, with users having to perform only constant exponentials (at most 5) in the proof, it is practical to execute our protocol in lightweight devices such as Palm, smart card, mobile devices and so on. No matter how many factors needing to be proved, there are only two bilinear mapping computations needed in each protocol, and the number of exponentials for verifier are dramatically reduced.These are the most efficient paradigms at present.

4.3 Security analysis of the Federation System

We discuss how our identity assurance techniques and cryptographic protocols together guarantee the security of the federated identity management system with respect to robustness and confidentiality. In our context, robustness means that no identity theft can be perpetrated within the federation. Confidentiality means that no unauthorized third party can gain access to the data exchanged during the registration and the usage protocol.
Robustness against identity theft. An important property that our protocol must ensure is that no matter how the federation entities might collude, it must not be possible for any entity to succeed in stealing identities of other principals. Thus, it must not be possible that a principal P uses a strong identifier m belonging to another principal $P' \in \mathcal{P}$ unless P can prove ownership of m as well. Further, the SP can ensure that an adversary will not succeed in using strong identifiers belonging to any other individual, even if such individual has never registered the identifier with the federation. To show robustness we will focus on the most interesting misbehavior by the different entities.

(i) Dishonest principal P. At the time of registration, two possibilities arise: the first is the case in which P impersonates an already registered individual, $P' \in \mathcal{P}$, by trying to register m which is owned by P'. P fails registering m because the deterministic commitment $\widehat{M}$ is in fact already recorded by the FederationDuplicateDetection function (see Table 1), when P' enrolled it. The other possible case is that P is impersonating an individual not known to the federation by registering a strong identifier m. Here, theft by P is detected since ExternalValidation is executed for a minimum number of strong identifiers as defined by the federation. Protocols 2, 3 and 4 provide efficient and flexible approaches to perform the multi-factor authentication at the time of usage. Each of them are secure as proved by Theorems $4.4 - 4.7$. Therefore impersonation can only be achieved with the compromise of all the required identifiers.

(ii) Honest principal P with dishonest registrar. Within the federation even a registrar cannot misuse the data, since it cannot prove the ownership of a valid IdR. This is ensured by the ZKPK protocol presented in Section 3. Another possible misbehavior of the registrar not strictly related with identity theft is related with corrupting the IdR. Precisely, the value of one or more stored commitments in IdR may be changed to an incorrect value. However, because the principal generates these values independently from the registrar's input, such errors can be detected.

(iii) Honest principal P with dishonest SP. Even in case a dishonest SP attempts an identity theft, it cannot reuse the proofs or the signatures to prove ownership of the corresponding strong identifier. This condition holds even if the SP knows the actual value of the strong identifiers, due to the semantically secure Pedersen's commitment. Moreover as illustrated in Protocol 3b the $timestamp$ prevents replay attack of final token sent to the issuer.

Confidentiality. Confidentiality of strong identifiers is achieved through combination of PKI techniques and the security of the protocols. Precisely, confidentiality is achieved as follows. Concerning identifiers registration, as illustrated in Protocol 2, only the commitments of the strong identifiers are revealed. From Lemmas 4.1 and 4.2, it follows that the values of the strong identifiers in the commitments remain confidential. With respect to usage of identifiers our protocols preserve minimality, in that, if the values of the strong identifiers are not required to be revealed at the time of usage, then as illustrated in Protocols 3b, 4a and 4b, we derive that the confidentiality of the strong identifier is assured. Concerning the confidentiality of weak identifiers and strong identifiers' tags, Protocol 4(a,b) provides an elegant way to hide the entire IdR. Moreover, subsequent usage of the the signatures cannot be linked in Protocol 4b which is proved in Theorem 5.6 part 2. Protocol 4(a,b) directly implies that SP's do not have access to the tags of the committed values and they cannot infer which strong identifiers have been committed.

5 Related Work

In the corporate world there are several emerging standards for identity federation like Liberty Alliance [19] (LA) and WS-Federation [25]. They aim towards standards for SSO[6] with decentralized authentication. SSO allows a user to sign-on once in order to be seamlessly signed-on when navigating to another site without the need to authenticate again. Shibboleth [16] is an initiative by universities that are members of Internet2 [17]. The goal of such initiative is to develop and deploy middleware technologies that can facilitate inter-institutional collaboration and access to digital contents by secure attribute based authentication and sharing. Concerning the problem of identity theft, the above initiatives and other organizations like Better Business Bureau and Federal Trade Commission have initiated some efforts aiming at educating consumers and preventing identity theft. However, to the best of our knowledge, no comprehensive techniques have been proposed to deal with the problem of identity theft. Our solution extends the concept of federation with the concept of proofs of identity and multi-factor authentication.
Several privacy-enabled identity management systems have been proposed based on the notion of anonymous credentials [5, 6, 10]. In anonymous credential systems, organizations know the users only by pseudonyms. Different pseudonyms of the same user cannot be linked. Yet, an organization can issue a credential to a pseudonym. The corresponding user can prove possession of this credential to another organization (which knows this user by a different pseudonym), without revealing anything more than the fact that the user owns such a credential [14]. Idemix [5] is the first system implementing anonymous credentials in a federated identity management system. Idemix provides mechanisms for efficient multi-show[7] credentials and a flexible scheme for issuing and revoking anonymous credentials. It also provides a mechanism for *all or nothing* sharing and a PKI-based non-transferability. Anonymous credentials may not be adequate for several real e-commerce applications and web services that require strong identifiers. In our approach we do not require the user identity to be hidden, even if we protect its identity attributes. More specifically, we do not only protect user privacy but also protect the use of its strong identifiers without requiring anonymity. Strong identifiers can indeed be securely used, and can be either certified or uncertified. By using privacy preserving multi-factor authentication and ZKPK we can provide strong authentication and thus prevent identity theft. Moreover, unlike existing approaches, we can also timely detect possible identity theft by checking for duplicates of strong identifiers.
The work most closely related to ours concerning the cryptographic schemes proposed, is the signature schemes and anonymous credentials [8] work from Camenisch *et al.* They propose efficient protocols that allow one to prove in zero-knowledge the knowledge of a signature on a committed (or encrypted) message and to obtain a signature on a committed message. Such approach also provides a signature scheme that is based on an assumption introduced by [23] and uses bilinear maps. In Section 4 we show how our protocols is substantially better for the purposes of multi-factor authentication. We in fact combine our ZKPKs with the aggregate signature scheme presented in [2] and establish a new cryptographic primitive for aggregate proof of knowledge. Our scheme is more flexible and efficient and requires less storage than the protocols in [8]. The paper by Boneh *et al.* [2] presents several applications for aggregate signatures and proposes an efficient aggregate signature mechanism based on bilinear maps. They however, do not investigate signatures on commitments which can be used later for ZKPK protocols. Also, in our case since the signatures are aggregated by the same registrar, the aggregation and verification are more efficient.

6 Conclusion

We have presented a solution to identity theft prevention based on multi-factor authentication, implemented by our new cryptographic notion of aggregated proof of knowledge. We have shown that our aggregated multi-factors ZKPs are more efficient than separate cases ZKPs. Moreover users only need to execute constant many exponentials, no matter how many signatures and commitments are to be proved. Our proof of knowledge of signature on commitment is more computationally and storage efficient than existing approaches [8]. Our aggregate proof is more flexible and storage saving and the verification of the aggregated signature is also efficient. The small security parameters like q used by the Weil Pairing can be efficiently implemented better than the RSA version on small devices like smart cards. Based on additional composite protocols for maintaining identity assurance we provide a comprehensive solution to prevent identity theft in a federated identity management system.

[6] Single Sign-On

[7] Credentials can be used multiple times. Possession of a multi-show credential can be demonstrated an arbitrary number of times; these demonstrations cannot be linked to each other [5].

References

[1] Elisa Bertino, Elena Ferrari, and Anna C. Squicciarini. Trust-χ: A Peer-to-Peer Framework for Trust Establishment. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):827– 842, July 2004.

[2] Dan Boneh, Craig Gentry, Hovav Shacham, and Ben Lynn. Aggregate and verifi ably encrypted signatures from bilinear maps. In *Proceedings of Advances is Cryptology – Eurocrypt'03, LNCS. Springer-Verlag, 2003.*, 2003.

[3] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Lecture Notes in Computer Science*, 2248:514, 2001.

[4] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifi able secret sharing and proactive cryptosystems. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 88–97, New York, NY, USA, 2002. ACM Press.

[5] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, New York, NY, USA, 2002. ACM Press.

[6] Jan Camenisch and Anna Lysyanskaya. An effi cient system for non-transferable anonymous credentials with optional anonymity revocation. *Lecture Notes in Computer Science*, 2045:93+, 2001.

[7] Jan Camenisch and Anna Lysyanskaya. A signature scheme with effi cient protocols. In*International Conference on Security in Communication Networks – SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2002.

[8] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO '04*, 2004.

[9] Jan Camenisch and Markus Stadler. Effi cient group signature schemes for large groups. In*Advances in Cryptology – CRYPTO '97*, pages 410–424, 1997.

[10] David Chaum. Security without identifi cation: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[11] David Chaum. Zero-knowledge undeniable signatures (extended abstract). In *EUROCRYPT '90: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 458–464, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[12] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 89–105, London, UK, 1993. Springer-Verlag.

[13] Ivan Damgard and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 125–142, London, UK, 2002. Springer-Verlag.

[14] IBM Zurich Research Laboratory: Privacy enhancing Cryptography and Pseudonym Management. http://www.zurich.ibm.com/security/privacy/.

[15] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identifi cation and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[16] http://shibboleth.internet2.edu. Shibboleth, Internet2.

[17] http://www.internet2.edu/. Internet2.

[18] http://www.javelinstrategy.com/reports. 2005 Identity Fraud Survey Report.

[19] http://www.projectliberty.org. Liberty alliance project.

[20] http://www.w3.org/XML/. Extensible markup language (xml).

[21] http://www.w3schools.com/dtd/default.asp. DTD tutorial.

[22] Jiangtao Li, Ninghui Li, and William H. Winsborough. Automated trust negotiation using cryptographic credentials. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 46–57, New York, NY, USA, 2005. ACM Press.

[23] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography 1999*, pages 184–199, 1999.

[24] Gurmeet Singh Manku. *Dipsea: A Modular Distributed Hash Table*. PhD thesis, Stanford University, 2004.

[25] Sven Overhage and Peter Thomas. WS-Specification: Specifying Web Services Using UDDI Improvements. In *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, pages 100–119, London, UK, 2003. Springer-Verlag.

[26] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK, 1992. Springer-Verlag.

[27] Birgit Pfitzmann and Michael Waidner. Strong loss tolerance of electronic coin systems. *ACM Trans. Comput. Syst.*, 15(2):194–213, 1997.

[28] Claus P. Schnorr. Efficient identification and signatures for smart cards. In *EUROCRYPT '89: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 688–689, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[29] Ting Yu, Marianne Winslett, and Kent E. Seamons. Interoperable strategies in automated trust negotiation. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 146–155, New York, NY, USA, 2001. ACM Press.

A Enforcing Identity Assurance

Here we elaborate the functions described in table 1.

LocalConsistency(τ, IdR, P) . This function checks whether the weak identifiers $W = (w_j, tag, validity - assure, ownership - assure)$ appearing on the input identity tuple τ are locally consistent. That is, if the same tag descriptors are present in any other identity tuple τ' of the same IdR, the value of the corresponding tags should be equal to w'_j. For example, if a credit card number (CCN) was committed along with weak identifiers $firstname$ and $lastname$, when a new identifier, say SSN, is committed the values of the weak identifiers $firstname$ and $lastname$ associated with the SSN should be identical to that for CCN[8].

FederationDuplicateDetection(τ, IdR,P) This function checks whether duplicate values of m exist, where m is a strong identifier appearing in tuple τ. Duplicate detection is based on use of Distributed Hash Table (DHT) [24]. Such table keeps track of the deterministic strong identifier commitments which have validity status equal to 'A'. The DHT is maintained by the registrars of the federation. The entries in the DHT are tuples of the form $\langle \widehat{M}, P, R \rangle$, where P denotes the principal, and R the identifier of the registrar storing the principal's IdR. $\widehat{M}$ is a deterministic value for the committed strong identifier m. Duplicate detection is actually executed by running algorithm $lookup$ on the DHT. Because the tables are distributed, the duplicate lookup is efficient in that it does not require an exhaustive search. If a duplicate is found the algorithm returns false and further actions are taken to detect whether a misuse is occurred.

ExternalValidation(τ, IdR,P) This function validates the strong identifier m appearing in τ, by contacting the issuer authority, which provides validity assurance. If the issuer successfully validates m, the associated $validity - assure$ value is set to 'A'. It is important that the weak identifiers used for the external validation correspond to the ones enrolled in τ.

External validation can be initiated according to two strategy. Under the first strategy, referred to as *push* mode, any SP in the federation receiving the actual value of the strong identifier m consults the issuer. The SP sends

[8] We assume the same tag names corresponding to the same semantic.

the validation result to the registrar storing the corresponding IdR accordingly so that the IdR can be updated accordingly. Under the second strategy, referred to as *pull* mode, the registrar needs to determine the validity status before the value of commitment M is signed and used in the federation. In this case the principal encrypts its strong identifiers with the issuer's public key which we assume to be available and sends it to the registrar. The registrar then appends the deterministic commitment $\widehat{M}$ to it and sends it to the issuer for verification. Details of this procedure are given in Protocol 4b). The registrar, once the verification from the issuer is completed, computes a signature σ on the commitment M and added to τ.

B Proofs

B.1 Proof of Theorem 4.5

We show that it is correct and unforgeable.
Correctness: We prove that for the honest prover, with values $m_1, \cdots, m_t, M_i = g^{m_i} h^{r_i}$, the protocols execute correctly. After computing the values $(x_1, \cdots, x_t)$ the prover calculates $M = \prod_{i=1}^{t} M_i^{x_i} = g_1^m h_1^r$ and $\widehat{M} = g_1^m$. Here, $m = \sum_{i=1}^{t} m_i x_i$ and $r = \sum_{i=1}^{t} r_i x_i$. This directly results in $M/\widehat{M} = h_1^r$ and the prover is able to carry out the ZKPK $PK\{(\beta) : M/\widehat{M} = h_1^\beta\}$. As such, only by knowing all $r_i, 1 \leq i \leq t$ the correct value r can computed and substituted for β. Items 1) and 2) of the thesis are thus proved for both protocols 3a and 3b.
The correctness of the signatures $\sigma_1, \sigma_2, \ldots, \sigma_t$ is derived from the validity of aggregated signature σ. We omit the correctness proof for σ since it is similar to the proof given for Theorem 4.4.
Unforgeability:

1. *Protocol 3a* We show that the successful execution of the protocol guarantees that a prover cannot forge even one of r_i or m_i by contradiction.
   Assume that an adversary executes successfully the protocol based on its knowledge of $r_2, \cdots, r_t$ and $m_1, \cdots, m_t$ and that it does not know r_1. In order to execute step 2 of Protocol 3a (3b), a zero-knowledge extractor that extracts r has to exist. Thus the adversary can feasibly compute r and, from $r = \sum_{i=1}^{t} x_i r_i \mod q$, it can deduce r_1. Thus it can feasibly compute r_1 which is a contradiction with respect to the assumption that r_1 is unknown.

   Unforgeability of m_i means that it is infeasible for a prover to reveal a set of $\{m'_1, \cdots, m'_t\}$ which is not exactly the same as $\{m_1, \cdots, m_t\}$ corresponding to the original identifiers committed in $M_1, \cdots, M_t$ and successfully execute the protocol. In this case at least one m'_i with $m'_i \neq m_i \mod q$ exists which would result in the random challenges to be calculated as $(x_1, \cdots, x_n) = H(m'_1 \parallel \cdots \parallel m'_t \parallel M_i \parallel \cdots \parallel M_t)$. Step 2 of protocol 3a performs the ZKPK, showing that $M/\widehat{M} = h_1^r, r = \sum_{i=1}^{t} x_i r_i \mod q$, $M = \prod_{i=1}^{t} M_i^{x_i} = g_1^m h_1^r$ and $\widehat{M} = \prod_{i=1}^{t} g_1^{m'_i x_i} = g_1^{m'}$. Here, $m = \sum_{i=1}^{t} x_i m_i$, $m' = \sum_{i=1}^{t} x_i m'_i$. Since $\log_{g_1} h_1$ is unknown, $g^m = g^{m'}$ implies $m - m' = 0 \mod q$. That is, $\sum_{i=1}^{t} x_i (m'_i - m_i) = 0 \mod q$. Since there exists at least one i such that $m'_i \neq m_i \mod q$, and since $(x_1, \cdots, x_n)$ is random, it is infeasible that $\sum_{i=1}^{t} x_i (m'_i - m_i) = 0 \mod q$.

2. *Protocol 3b* The same reasoning as before applies to protocol 3b, with the only difference that SP does not explicitly know the values $m_1, m_2, \ldots, m_t$.
   By using messages $\mathsf{Enc}_{Issuer}(\{m_1, \ldots, m_t, timestamp\}), (\widehat{M_1}, \cdots, \widehat{M_t})$, the issuer will check if $g_1^{m_i} = \widehat{M_i}$. If they are all valid, and from the correctness of the proof, we know that m_i is the value committed in M_i for $1 \leq i \leq t$, which were signed by registrar. □

B.2 Proof of Theorem 4.6

Correctness: From the signatures $\sigma_1, \sigma_2, \ldots, \sigma_t$ assigned by the registrar for messages $M_1 = g_1^{m_1} h_1^{r_1}, \ldots, M_t = g_1^{m_t} h_1^{r_t}$, the principal computes

$$\sigma = \prod_{i=1}^{t} \sigma_i = \prod_{i=1}^{t} M_i^{\chi} = \prod_{i=1}^{t} g_1^{m_i \chi} h_1^{r_i \chi} = g_1^{m\chi} h_1^{r\chi}$$

Where $m = \sum_{i=1}^{t} m_i,\ r = \sum_{i=1}^{t} r_i$. Then

$$e(\sigma, g_2) = e(g_1^{m\chi} h_1^{r\chi}, g_2) = e(g_1^m h_1^r, g_2)^\chi = e(g_1, v)^m e(h_1, v)^r$$

where χ and $v = g_2^\chi$ are respectively the private and public keys of the registrar. The principal is able to successfully carry out the ZKPK m, r as given in step 2 of the protocol.

Unforgeability: The successful protocol execution should guarantee that the prover has valid signatures σ_i and knowledge of all m_i committed in M_i. If a knowledge extractor exists for the ZKPK at step 2 that extracts two message m' and r', such that $e(\sigma, g_2) = e(g_1, v)^{m'} e(h_1, v)^{r'}$ then it would mean $e(\sigma, g_2) = e(g_1^{m'\chi} h_1^{r'\chi}, g_2)$. That is, $e(g_1^{m\chi} h_1^{r\chi}, g_2) = e(g_1^{m'\chi} h_1^{r'\chi}, g_2)$. Since G_1, G_2 are Co-GDH groups, it implies principal knows the values m and r. By Lemma 4.3, we know that the prover has the knowledge of all the values m_i committed in the messages M_i for $1 \leq i \leq t$. Thus, the validity of signatures $\sigma_1, \ldots, \sigma_t$ is obtained from the security of aggregation signature [2]. □

B.3 Proof of Theorem 4.7

To show the zero-knowledge property, we construct a simulator S as follows. Since the message that the principal sent in the first step is independent of any actual signature, S randomly chooses $s_1, s_2 \in \mathbb{Z}_q$, and forms $g_1^{s_1} h_1^{s_2}$ which has the following property:

$$e(g_1^{s_1} h_1^{s_2}, g_2) = e(g_1, g_2)^{s_1} e(h_1, g_2)^{s_2} = e(g_1, v)^{s_1/\chi} e(h_1, v)^{s_2/v}$$

The above results in the correct form of the required signature. Since in step 2, the principal and SP execute a ZKP, it follows that there exists a simulator S' for that step. When S' is run, it is easy to deduce that the simulator S constructed is the zero-knowledge simulator for the protocol.

Next, we show that protocol 4b is a proof of knowledge. Suppose a prover can give an acceptance proof following the protocol, the knowledge extractor for it will obtain values $m_0, r_0 \in \mathbb{Z}_q$, such that

$$e(\sigma, g_2) = e(g_1, v)^{m_0} e(h_1, v)^{r_0} = e(g_1, g_2)^{m_0\chi} e(h_1, g_2)^{r_0\chi} = e((g_1^{m_0} h_1^{r_0})^\chi, g_2)$$

This forms a signature pair $(g_1^{m_0} h_1^{r_0}, \sigma)$. □