Notarized Federated Identity Management for Web Services*

Michael T. Goodrich[1], Roberto Tamassia[2], and Danfeng Yao[2]

[1] Department of Computer Science, University of California
Irvine, CA 92697 USA
goodrich@acm.org

[2] Department of Computer Science, Brown University
Providence, RI 02912 USA
{rt, dyao}@cs.brown.edu

Abstract. We address the question of how to establish trust in federated identity management systems. We propose a *notarized* federated identity management model that supports efficient user authentication when providers are unknown to each other. Our model introduces a notary service, owned by a trusted third-party, to dynamically notarize assertions generated by identity providers. An additional feature of our model is the avoidance of direct communications between identity providers and service providers, which provides improved privacy protection for users. We present an efficient implementation of our notarized federated identity management model based on the Secure Transaction Management System (STMS).

We also give a practical solution for mitigating aspects of the identity theft problem and discuss its use in our notarized federated identity management model. The unique feature of our cryptographic solution is that it enables one to proactively prevent the leaking of secret identity information.

Keywords: Federated identity management, notary and notarization, SAML, identity theft, identity-based encryption

1 Introduction

Digital identity management is becoming an integral part of our lives, as consumers and businesses rely more and more on online transactions for daily tasks, such as banking, shopping, and bill payment. These transactions crucially depend on networked computer systems to communicate sensitive identity data across personal, company, and enterprise boundaries.

Unfortunately, the overuse of personal information in online transactions opens the door to identity theft, which poses a serious threat to personal finances

* This work was principally supported by IAM Technology, Inc. Additional partial support was provided by NSF grants CCF–0311510, CNS–0303577 and IIS–0324846.

and credit ratings of users and creates liabilities for corporations. Moreover, the increasing dangers of identity theft are negatively affecting people's collective confidence on the digital world for online financial transactions [12]. Thus, effective solutions for managing digital identity on both the individual and enterprise levels are urgently needed.

Additionally, end users are challenged with increasing numbers of websites that require access control and authentication. Studies show that users resort to using weak passwords or writing them down to alleviate the burden of memorizing multiple passwords. One well-known identity management solution that deals with this issue is the *single sign-on* (*SSO*) technique, which requires the user to authenticate only once to a website, and then automatically authenticates the user to other websites from then on, within a session. There are two primary approaches to single sign-on. One approach is based on browser cookies and redirects, and the other is based on cryptographic-enabled assertions.

The cookies-based approach is simple to implement, yet has several limitations. This approach is implemented by using browser cookies to maintain the state of the browser, so that re-authentication at a secure web site is not required. Because browser cookies are not transferred between different administrative domains, however, cookies obtained from one site are not sent (in any HTTP messages) to other sites where authentications are required. Thus, the cookies-based approach is only useful inside a single administrative domain. In fact, this problem also exists in single organizations that maintain separate divisional domains.

The approach based on cryptographic-enabled assertions is embodied by the *Security Assertion Markup Language* (*SAML*) [9]. Version 2.0 of SAML is generally believed to support general cross-domain authentication and SAML is quickly becoming the de-facto means for exchanging user credentials between trusted environments. The identity federation architecture of *Liberty Alliance* is compliant with the SAML 2.0 standard [16]. Indeed, SAML is specifically designed to support cross domain single sign-on, which is illustrated in the following example.

Suppose a user has a secure logon session to a website, (e.g., Airline.com), and is accessing resources on that site. Airline.com serves as the *identity provider* site in this example. At some point in the session, the user is directed to another web site in a different DNS domain for a related service, and this outside domain is called the *service provider* site (e.g., CarRental.com). The identity provider (Airline.com) asserts to the service provider (CarRental.com) that the user is known to the identity provider and gives to the service provide the user's name and session attributes (e.g., Gold member). Since the service provider trusts the assertions generated by the identity provider, it creates a session for the user based on the information received. Therefore, the user is not required to authenticate again when directed to the service provider site. Hence, single sign-on is achieved.

The *identity provider (IdP)* in SAML [9] is defined as the system, or administrative domain, that asserts information about a subject. An identity provider asserts that a user has been authenticated and has certain attributes. The *service provider (SP)* is defined as the system, or administrative domain, that relies on the information supplied to it by the identity provider.

1.1 Motivation for Notarized ID Federation

In existing federated identity management systems that support SAML, such as the *Liberty Identity Federation Framework* (*ID-FF*) [10] and *WS-Federation* [26], it is up to the service provider to decide whether it trusts the assertions provided to it. Service providers in SAML are also known as *relying parties* due to the fact that they rely on the information provided by an identity provider. This reliance implies that websites of different administrative domains need to trust each other's access control verdicts on end users. In fact, SAML single sign-on relies on the concept of *identity federation* in order for it to work at all. An identity federation is said to exist between an identity provider and a service provider, when the service provider accepts assertions regarding a user from the identity provider [9].

Methods for efficiently maintaining a federated identity infrastructure are vital, therefore, especially when member domains in the federation dynamically join or leave. Methods for effectively disseminating the member domains in the current federation serve an important role because service providers rely on this information for making access control decisions. These access control decisions directly protect the security of the resources of the service provider and have to be made with high assurance. Because the role of service providers and identity providers are sometimes inter-changeable in web services, all participating domains in federated identity management systems must face the trust decisions implied by all possible cross-domain interactions.

Nevertheless, these issues have not drawn much attention in the federated identity management literature [22]. In fact, most existing SSO solutions assume preexisting trust relationship among providers and do not provide concrete mechanisms for the trust establishment between providers. The WS-Federation specification [26] discusses several trust relationships between identity providers and service providers, including directed trust, indirected brokered trust, and chained trust. However, details on how the trust relationships and identity brokers can be instantiated are not given. This limitation hinders the wide deployment of SSO in web-service environments, because providers may be unknown to each other. Therefore, flexible, reliable, and secure trust establishment mechanisms need to be provided for federated identity management.

1.2 Our Contributions

In this paper, we present a notarized federated identity management protocol that supports flexible and efficient authentication of assertions, and enables a service provider to proactively obtain the trustworthiness information of unknown identity providers.

We also address important aspects of the problem of large-scale identity theft. In June 2005, CardSystems Solutions, a large credit card payment processor in Tucson, Arizona announced that forty million credit card numbers may have been stolen by computer hackers. The theft was a direct result of the company's illegal practice of retaining transaction records. It is insufficient to simply trust financial institutions' abilities and intentions for secure data management. Thus, rather than putting faith in the data management of financial institutions, we give a proactive solution for protecting the disclosure of user's sensitive personal data with a cryptographic approach.

Our contributions are summarized as follows.

- We address the important question of how to establish trust between providers in a decentralized environment. We propose a notarized federated identity management model that supports automatic user authentications when the providers are unknown to each other. Our model introduces a *notary server*, which is owned by a trusted third-party to dynamically notarize assertions generated by identity providers. Assertions are generated by identity providers and registered with a trusted notary server. When a service provider needs to verify an assertion, it queries the notary server to get a notarized assertion. The notary information shows that the identity provider is trusted by the notary server, and proves the trustworthiness of the identity provider that generates the assertion. As an extra feature provided by the notary server, our federated identity management model also reduces possible collusions between identity providers and service providers, and gives improved privacy protections for users.
- We describe an efficient implementation of the federated identity management protocol with the existing *Secure Transaction Management System* (*STMS*) [1, 15]. The notary server caches the assertions at a collection of responders deployed in the network. Even when the responders are located in insecure, untrusted locations, a service provider can easily identify a forged or tampered assertion so that the integrity of an assertions is maintained.

  Our protocol can be thought of as a concrete solution for a trust broker model proposed by existing federated identity management systems [26]. Besides brokering trust, our solution offers additional features. *Accountability* is supported by archiving signatures on requests and assertions. *User privacy* is achieved by encrypting assertions stored by the notary server. *Verification efficiency* is achieved by using the authenticated-dictionary technique (see, e.g., [1, 15, 20, 24] implemented in STMS.
- We also give a practical solution for mitigating aspects of the identity theft problem, and discuss how it is used in our federated identity management protocol. Our cryptographic solution is based on the *Identity-Based Encryption* (*IBE*) scheme [5]. The main feature of our cryptographic solution is that it enables one to proactively prevent the leaking of secret identity information.

1.3 Organization of the paper

Our model for notarized federated identity management is described in Section 2. The STMS implementation of the notarized federated identity management protocol is presented in Section 3. In Section 4, we present a cryptographic protocol that helps protect users against identity theft. The security of the federated identity management protocol and the IBE-based authentication protocol is analyzed in Section 5. Related work is given in Section 6.

2 Notarized Federated Identity Management

Our notarized federated identity management model introduces a *notary server*, a trusted third-party that dynamically maintains assertions generated by identity providers. Assertions are generated by identity providers and stored by the notary server. When a service provider needs to verify an assertion, it queries the notary server for a *notarized assertion* that shows the trustworthiness of the identity provider generating the assertion.

2.1 Notary server

In a notarized ID federation, a notary server is trusted by both identity providers and service providers. Identity providers that have good internet behavior and reputation are allowed to register with the notary server, and thus are trusted. The notary server stores the assertions generated by registered identity providers. A notary server supports two operations, SUBMIT and QUERY.

- SUBMIT(id, S_{id}, sig): a registered identity provider *IdP* authenticates itself to the notary server, and submits via a secure channel the tuple (id, assertion, signature), denoted by (id, S_{id}, sig), to the notary server. The assertion S_{id} states the attributes of an identity *id*, and the signature *sig* is signed by *IdP* on the assertion S_{id}. The notary server stores the tuple.
- QUERY(id): a service provider *SP* queries in a *public (insecure)* channel the notary server for assertions associated with identity *id*, and the notary server returns the *notarized assertion(s)*.

A notarized assertion has a proof showing that the assertion is indeed stored by the notary server, which implies that the identity provider that generates the assertion is trustworthy. The reason for not using a secure channel in QUERY is for higher efficiency and scalability in a distributed environment. The challenge, thus, becomes how to efficiently generate and verify the notarized assertion, even when it is transmitted in a insecure channel. Our solution is based on the authenticated dictionary technique (see, e.g., [1, 15, 20, 24]), which is more scalable than using a signature scheme.

The main purpose of the notary server is to provide the assurance of the trustworthiness of assertions when identity providers are unknown to the service providers. The notary server is a bridge of trust between providers in web-service transactions. Another advantage of storing assertions on the notary server is the prevention of direct contact between identity providers and service providers. A notarized assertion does not contain the name of the identity provider. This further increases the difficulty of collusions among providers to discover private user information.

In our model, we assume that the notary server is trustworthy, and is trusted by all entities (users, identity providers, service providers). The security properties of our notarized federated identity management protocol are summarized below and are analyzed in detail in Section 5.

– *Security* is defined as that no polynomial-time adversary can forge a notarized assertion that can be accepted by a service provider.
– *Secrecy* is defined intuitively as that the protocol does not leak any information about a notarized assertion to a (polynomial-time) adversary. This property provides privacy protection to the users.
– *Accountability* is defined as that identity providers should be held accountable for the assertions generated, and for any unauthorized information disclosure about the users.

Note that the notary server only certifies that the source of an assertion is trustworthy; it is not required to examine and certify the content of an assertion. In fact, our protocol, which is described next, deliberately avoids disclosing assertion contents to the notary server by encrypting the assertions. This feature is for the purpose of user privacy, and prevents the notary server from gaining knowledge of private user information.

2.2 Protocol

In this section, we present the protocol for our notarized federated identity management model. The following entities participate in the protocol: a user, an identity provider, a service provider, and a notary server. The protocol gives an instantiation of operations Submit and Query. Note that the roles of identity provider and service provider are interchangeable. For example, a bank can be the identity provider in one scenario and the service provider in another scenario.

We assumes that the notary server knows the public keys of registered identity providers. In addition, the public key of the notary server is known by all of the providers. A schematic drawing of the protocol is shown in Figure 1.

Upon receiving a service request, a service provider opens a secure channel with the user. The service provider and the user jointly generate a random session ID for the user's request. The identity provider is given the session ID by the user in a secure channel, after successful authentication. The identity provider computes the hashed session ID using a collision-resistant one-way hash function. The assertion associated with the user is generated by the identity provider with the hashed session ID. To prevent information leaking, the assertion is blinded by the identity provider before it is signed and submitted to the notary server. A notary server only knows the blinded assertion and hashed

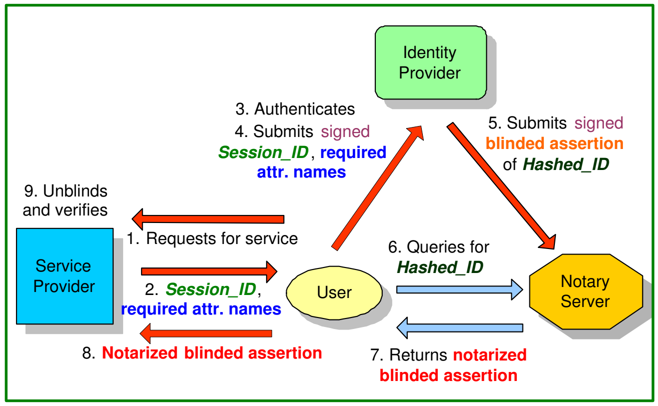

Fig. 1. Overview of the notarized federated identity management protocol. Red lines represent secure communication channels. Blue lines indicate that secure channels are *not* required.

session ID and cannot take advantage of the information for any service. This is important for the security of the protocol, when we distribute the notary service to untrusted notary responders in STMS in the next section. When submitting assertions, an identity provider has to first authenticate to the notary server, which ensures the provider is authorized.

In our protocol, the user only needs to authenticate once to an identity provider. Subsequent requests for service from multiple service providers do not require the user for authentication. Nevertheless, for protecting personal privacy, the user is given the ability to examine the contents of assertions to be given to the service providers in our protocol. If the assertions are generated by the identity provider according to the user's request, then they are passed on to the service providers. We argue that having the user involved in the identity management protocol for privacy purpose is a feasible solution. This concept was also proposed by other federated identity management solution [3]. The process can be automated to minimize the user's manual participation.

Public parameters include a collision-resistant one-way hash function, *Hash*, that takes a binary string of arbitrary length and hashes to a binary string of fixed length k: $Hash : \{0,1\}^* \rightarrow \{0,1\}^k$. For the blinding purpose, the public parameters also include two public strings P_1 and P_2. In addition, providers also agree on a symmetric-key encryption scheme for blinding and unblinding assertions. The encryption and decryption of a message x with a secret key K using this scheme are denoted as $E_K(x)$ and $D_K(x)$, respectively.

1. The user requests services from a service provider *SP*. *SP* requires attribute information of the user needed to complete the service.
2. *SP* opens a secure communication channel with the user. The user and *SP* each generate a random integer of the same length. They first exchange the cryptographic hashes of these integers as commitments using the secure

channel, and then they exchange the integers using the secure channel. The session ID N is finally computed as the XOR of the two integers. *SP* also informs the user of the attribute names that are needed for the service (e.g., billing address and age).

3. The user contacts and authenticates to her identity provider *IdP*. If the authentication is successful, the user opens a secure channel with *IdP*, and transmits a *signed* request that contains the session ID and the required attributes.
4. *IdP* verifies and stores the signed request by the user. The signature is for the accountability purpose in case of dispute (see Section 5).
5. *IdP* then computes the *index* of the assertion as the hash of session ID concatenated with the public parameter P_1: $h = Hash(N, P_1)$. It then generates an assertion S_h about the user using index h. For example, S_h states that h is a university student.
6. To prevent information leaking, *IdP* blinds the assertion as follows.
   (a) *IdP* computes the *blinding factor* K as the hash of the session ID concatenated with the public parameter P_2: $K = Hash(N, P_2)$.
   (b) *IdP* encrypts S_h with the symmetric encryption scheme, using K as the secret key. This gives the *blinded assertion* $S'_h = E_K(S_h)$.

   The blinded assertion S'_h is signed by *IdP* with its private key, which gives a signature sig_h.
7. *IdP* runs $\textsc{Submit}(h, S'_h, sig_h)$ with the notary server to submit tuple (h, S'_h, sig_h) through a secure channel as follows.
   (a) *IdP* first authenticates to the notary server to establish a secure communication channel.
   (b) *IdP* transmits tuple (h, S'_h, sig_h) to the notary server via the secure channel.
   (c) The notary server verifies signature sig_h, and stores (S'_h, sig_h) indexed by h. The signature is stored for accountability purposes.
8. The user computes the index $h = Hash(N, P_1)$ from N and P_1, and runs $\textsc{Query}(h)$ to obtain the assertion for h. The notary server processes the query as follows.
   (a) The blinded assertion S'_h associated with h is retrieved.
   (b) The notary server *notarizes* the assertion S'_h, and returns the notarized assertion. We describe two approaches for the realization of notarized assertion in the following sections. Note that the QUERY operation between the user and the notary server does not require a secure channel.
9. Once the user obtains the returned notarized blinded assertion, she unblinds it with the blinding factor $K = Hash(N, P_2)$. This is done by decrypting S'_h with K, which gives $S_h = D_K(S'_h)$. The user verifies that the assertion does not release any unauthorized personal information about her.
10. The notarized blinded assertion is then relayed from the user to the service provider, who verifies that it is notarized by the notary server. This implies that the identity provider *IdP* is trusted by the notary server. If the verification succeeds, the assertion S'_h is unblinded in the same way as in Step 9. The attribute information is obtained from the assertion, and h is compared

with the hash $Hash(N, P_1)$ of session ID N and P_1. The user is then granted the service if the verification passes. The service provider also stores the notarized assertion for accountability purposes.

The use of public parameters P_1 and P_2 decouples the blinding factor $Hash(N, P_2)$ and the index $Hash(N, P_1)$ of an assertion. This is done to prevent dictionary attacks that use the public index $Hash(N, P_1)$ of an assertion to evaluate the blinding factor $Hash(N, P_2)$ and thus infer the assertion. The security is analyzed in Section 5.

In our protocol, the randomness of the session ID N is important because it is used to blind the assertions of users. Therefore, the user participates in generating the random session ID in Step 2, where the user and the service provider each contribute a share of the session ID.

A straightforward realization of notarized assertions with signatures is described next in Section 2.3. A more sophisticated solution based on the Secure Management Transaction System(STMS) is presented in Section 3.

2.3 Realization of notarized assertions with signatures

Notarized assertions can be realized using simple time-stamped signatures. The notary server individually signs every assertion and the current time-stamp with its private key. The notarized assertion consists of this signature along with the assertion and time-stamp. To verify a notarized assertion, the service provider verifies the signature against the public key of the notary server, which can be obtained through usual means such as a public key certificate. Because the notary server is trusted, the correct verification of the signature establishes the authenticity of the assertion about the index identifier $Hash(N, P_1)$. Namely, the security of the signature-based realization of notarized assertions follows directly from the security of the underlying signature scheme adopted.

Even though the service provider cannot tell which identity provider generated the assertion, trusting the notary server is sufficient for authenticating the user in most applications such as on-line shopping. In addition, not knowing the source of assertions prevents, to some degree, the service providers from colluding with identity providers to discover information about the user. Because of the use of session ID, the notary server and the service providers do not know the actual identities associated with assertions.

In this simple signature-based approach, notarizing assertions can be a performance bottleneck because the notary server needs to sign every individual assertion. To improve the efficiency of the notary server, we give an improved realization of notarized assertions using authenticated dictionary techniques in the next section.

3 STMS-Based Implementation

In this section, we describe an approach for realizing notarized assertions using the secure transaction management system (STMS). The main advantage of STMS in comparison to the simple time-stamped signature approach is its high efficiency of computation. The notary server only needs to generate one signature as opposed to a signature for each assertion. In addition, STMS also provides a distributed architecture for fast real-time dissemination of assertion updates. Next, we first introduce the components and algorithms of STMS, then we describe how to use STMS to scale up the notary service.

3.1 Secure Transaction Management System (STMS)

The computational abstraction underlying STMS is a data structure called an *authenticated dictionary* (see, e.g., [1, 15, 20, 24]), which is a system for publishing data and supporting authenticated responses to queries about the data. In an authenticated dictionary, the data originates at a secure central site, called *STMS source* and is distributed to servers scattered across the network, called *STMS responders*. The responders answer queries on behalf of the source about the data made by clients. It is desirable to delegate query answering to the responders for two reasons: (1) The source is subject to risks such as denial-of-service attacks if it provides services directly on the network, and (2) The large volume and diverse geographic origination of the queries require distributed servers to provide responses efficiently.

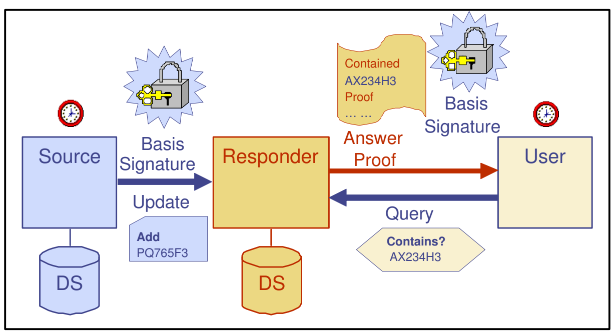

Fig. 2. Overview of STMS. The source pushes updates containing a signed basis to the responder. The responder then answers user queries with a proof of the answer, and a copy of the signed basis from the source.

The main feature of STMS is that it maintains trust even when responders are located in insecure, untrusted locations. That is, when a client submits a query to an STMS responder, it gets back not only an answer but also a proof of the answer. The client can easily validate the answer and determine that the responder has not been tampered with, while relying solely on trusted statements signed by the source. The design of STMS allows untrusted responders, which do not store private keys, to provide verifiable authentication services on behalf of a trusted source. This nonintuitive yet mathematically provable fact is the key to achieve cost effectiveness.

Figure 2 shows a high-level description of the STMS parties and protocol. The source sends real-time updates to the responders together with a special signed time-stamped fingerprint of the database called the *basis*. A user's query to the responder asks whether an element is contained in the authenticated dictionary maintained by STMS source. A responder replies to the query with an authenticated response. This consists of the answer to the query, the proof of the answer, the basis and its signature signed by the STMS source. Informally speaking, the proof is a *partial fingerprint* of the database that, combined with the subject of the query, should yield the fingerprint of the entire database. A proof consists of a very small amount of data (less than 300 bytes for most applications) and can be validated quickly. The client finally evaluates the risk associated with trusting the answer using the freshness of the time-stamp.

The signature of the basis is verified using the source public-key and the current time quantum. If the signature is not valid, then the basis is not valid. This may indicate that the basis or the source public-key is tampered by the STMS responder from which the values are obtained. The user verifies the answer for element x by simply hashing the values of the returned sequence $Q(x)$ of hash values in the given order, and comparing the result with the signed value $f(s)$, where s is the basis value. If the two values agree, then the user is assured of the validity of the answer at the time given by the time-stamp. The authenticated dictionary data structure can be implemented using Merkle hash tree [19]. The data structure [15] used in STMS system is based on skip list, which is more efficient than a Merkle hash tree. We refer readers to the authenticated dictionary literature [1, 15] for more information.

3.2 Implementing notarized assertions with STMS

Using STMS, a notary server consists of a *notary source* and several *notary responders*. The notary source needs to be a trusted server that stores assertion inputs from identity providers. Notary responders can be strategically placed in geographically dispersed locations to accommodate fast queries. They obtain real-time updates from the notary source, and answer queries from users. Notary responders do not need to be trusted servers. The notarized assertions returned by them can be authenticated by verifying against the public key of the notary source by anyone.

With STMS, a notarized assertion returned by Query operation consists of two parts: assertion itself and a STMS proof. As described in the previous section, the proof is a sequence of hash values of elements in the notary server for proving the existence of the assertion. The size of the proof is quite compact, even for large number of items in the notary server. Therefore, transmitting the proof can be quite fast. The service provider then obtains the signed STMS basis of the current time quantum from the notary responder, if it does not yet have it. The proof of the assertion is verified against the basis, and the signature of the basis is verified against the public key of the notary source. If the verification is successful, the request is granted. The signed basis remains the same for the duration of a time quantum, therefore it only needs to be obtained once for each time quantum. The rest of the notarized federated identity management protocol with STMS follows the protocol in Section 2.2, and is not repeated here.

Because notary responders are not required to be trusted servers, storing session ID in the clear is not secure – a notary responder may attempt to impersonate a user with the session ID for service. Note that opening a secure communication between the service provider and the notary responder does not solve this problem. Our notarized federated identity management protocol in Section 2 is resilient to this problem, because assertions use hashed session ID rather than the plain session ID. In addition, the service provider transmits the plain session ID to the user in a secure channel. A schematic drawing of the STMS implemented notarized federated identity management protocol is shown in Figure 3. The time quantum can be set to as short as orders of milliseconds to allow fast dissemination of assertions. Due to space limit, the protocol and security of STMS implemented notarized federated identity management are not presented. The security is based on the security of STMS, which has been previously proved [1].

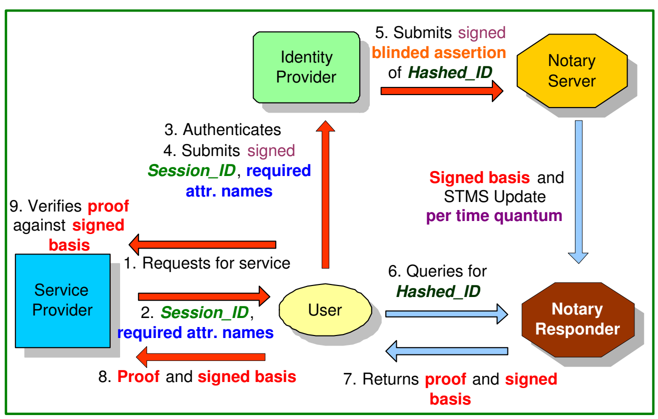

Fig. 3. A schematic drawing of the STMS implemented notarized federated identity management protocol. At each time quantum, the notary source sends the signed basis and updates of the authenticated dictionary to the notary responder. The notary responder answers a query for assertion by returning the signed basis and the proof corresponding to the queried element.

Next, we present an authentication protocol that effectively reduces the identity theft problem. We also describe how to integrate the authentication protocol with our notarized federated identity management protocol.

4 Reducing the Risks of Identity Theft

Identity theft is a major problem for identity management systems. Recently, several practical solutions against on-line identity theft have been proposed [3, 18]. Single sign-on systems have been criticized to provide weak protection against identity theft, because once an attacker successfully logs onto one site, she will have no problem requesting services from other sites. Although intuitively this is true, single sign-on does not necessarily make identity thieves' life any easier. Madsen, Koga, and Takahashi argued in [18] that in single sign-on systems, users only need to use one password and therefore are more likely to choose and memorize strong passwords. Strong passwords are shown to be an effective way to prevent break-ins.

In this section, we first analyze the causes of a successful identity theft. Then, we give a practical solution that, if used by identify mangagers, could mitigating major aspects of the identiy-theft problem. We also describe how to use our scheme in our notarized federated identity management protocol.

4.1 Identity theft and its causes

Identity theft is a type of crime in which an imposter obtains key pieces of personal information, such as Social Security or driver's license numbers, in order to impersonate someone else. Although an identity thief might crack into a database to obtain personal information, it is believed that a thief is more likely to obtain information using Trojans or even old-fashion methods such as dumpster diving and shoulder surfing.

We observe that the current authentication protocols, both physical and digital ones, are fundamentally susceptible to identity theft, even if an individual is careful in protecting her sensitive information. Physical authentication protocols include the procedures for obtaining a driver's license at a government office, opening a bank account, and applying for mortgage. Digital authentication protocols include the corresponding on-line transactions. In current solutions, key pieces of personal information are usually communicated in the clear or stored in the clear. This makes stealing of information easier for identity thieves. Although the SSL protocol encrypts communications between a user and a server, this does not prevent Trojan keyloggers, or shoulder surfing, because the user still needs to disclose and type over and over sensitive information such as her social security number.

We argue that this fundamental characteristic of the existing authentication protocols is one of the main causes of identity theft, namely using sensitive information in clear form for authentication. We propose a simple and practical cryptographic protocol for authentication. Our solution ties personal information to random secrets, which are used to prove *interactively* the ownership of the personal information but are *never disclosed*.

4.2 Motivation for using IBE

In public key encryption schemes, the private key information is never disclosed. Yet, a challenge-response process can be used by a user to prove the possession of the private key to an identity provider. The private key is usually protected by encrypting it with a passphrase, and storing it in a portable device, such as a smart card or a USB flash drive. Observe that the private key is never disclosed in clear during transactions, hence it never appears in any printed form or display. Therefore, it is difficult for attackers to retrieve someone's private key using standard identity theft techniques. To steal the private key, an attacker would need to obtain the physical device and know the passphrase.

In order to associate public keys with identity information, we use the Identity-Based Encryption (IBE) scheme [5, 23]. A public key in IBE will be the personal information (e.g., the social security number of an individual). For authentication, an individual not only needs to know her personal information (e.g., social security number), but also needs to prove the possession of the corresponding private key for authentication. In the rest of this section, we introduce identity-based encryption and related schemes. In the next section, we describe our authentication protocol for identity management.

The idea of an identity-based encryption (IBE) scheme is that an arbitrary string can serve as a public key. The motivation for using IBE as opposed to conventional public-key encryption schemes is as follows. IBE reduces the need for public key certificates and certificate authorities, because a public key can be associated with identity information such as a user's social security number. On the contrary, conventional encryption schemes such as RSA does not allow an arbitrary string to be used as a public key, and hence requires key certification.

Using IBE, a user can disclose her social security number to an identity provider without worrying about identity theft attacks such as shoulder surfing and dumpster diving. This is because the number is only used as a public key, and the user also needs to prove the possession of the corresponding private key. The identity provider encrypts a challenge nonce using the user's social security number as the public key. The user receives the encrypted challenge nonce and decrypts it with his private key corresponding to the social security number. The private key is obtained by a third party, called Private Key Generator (PKG) in IBE literature. To do that, the user authenticates himself to the PKG in the same way as he would authenticate himself to a passport office and obtains his private key from PKG. The user returns the decryption result to the identity provider. Without knowing the private key associated with the social security number, an attack cannot make use of the number.

4.3 Identity-based encryption scheme

The first scheme for identity-based encryption was based on the bilinear Diffie-Hellman assumption in the random oracle model by Boneh and Franklin [5]. In IBE schemes, the private key generator (PKG) is responsible for generating private keys for all users, and therefore is a performance bottleneck for organizations with large number of users. Hierarchical identity-based encryption (HIBE) schemes [13] were proposed to alleviate the workload of a root PKG by delegating private key generation and identity authentication to lower-level PKGs.

Boneh-Franklin IBE contains four operations: SETUP, EXTRACT, ENCRYPT, and DECRYPT [5]. In SETUP, the PKG takes a security parameter k, and returns *params* (system parameters) and the root secret key SK. The root private key is used to derive private keys for all other users and is only known to the PKG. In EXTRACT, the PKG uses SK to generate the private key SK_{id} for a user with an ID. In ENCRYPT, a sender inputs *params*, a message M and the ID of the intended message recipient, and computes a ciphertext C. In DECRYPT, a user with an ID inputs *params*, C, and its private key $SKid$, and returns the message M. These operations are used in our authentication protocol.

4.4 A cryptographic authentication protocol

We propose to use ID-based encryption scheme for implementing an authentication protocol for sensitive personal information. Our protocol minimizes the exposure of secret personal information and thus is more robust against identity theft than existing authentication methods.

Entities in our protocol include a user, an ID authority, an identity provider, and a revocation server controlled by the ID authority. Our authentication protocol has the following operations: SETUP, REGISTER, AUTHENTICATE, and REFRESH. It requires an on-line revocation server maintained by the ID authority. The operations are defined as follows.

SETUP is run by an ID authority who takes as input a security parameter and outputs public parameters and master secret.

REGISTER is run by the ID authority to generate secret keys for personal information of a user. The ID authority takes as input the identity information of a user and outputs the corresponding private key. The private key is transmitted to the user via a secure channel.

AUTHENTICATE is for a user to authenticate his identity information to an identity provider. The identity provider first queries the revocation server to ensure that the user's public key is valid. If yes, the user then proves to the identity provider that he possesses the secret key associated with the identity information. If the user successfully proves the possession of secret key, the output is true, otherwise, the output is false.

REFRESH is for the ID authority to re-generate the private key associated with a user's identity information. The input is the identity information, and the output is a new private key. Refresh is run when the previous secret key expires, or the key is compromised and a new key needs to be generated. The revoked public key is put on a revocation server.

Refreshing the secret key of identity information can be tricky, because the identity information typically does not change, e.g. social security number. We show later how to use multiple pieces of identity information and on-line revo-

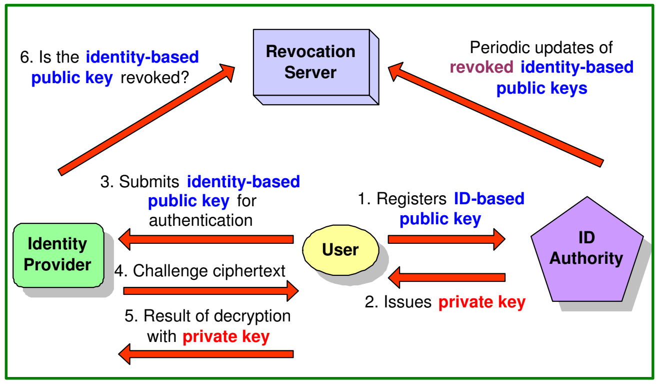

Fig. 4. A diagram of the ID-based authentication protocol. Identity information is used as the public key in this protocol. This identity information *cannot* be used by others for impersonation without knowing the corresponding private key.

cation checking to leverage this. A diagram of the protocol is shown in Figure 4. Here, we describe the realization of the above operations with IBE scheme.

1. Setup: The ID authority runs the PKG Setup operation of IBE.
2. Register: A user requests an identity private key from an ID authority. The user needs to be physically present in the ID office, for example the passport office, with paper identifications such as passport, birth certificate. The ID authority authenticates the user's identity.
   If the user's identity is verified, the ID authority runs the Extract operation of IBE with the user's identity information concatenated with a unique serial number l. For example, l can be the driver's license number. l is used for revocation purpose. Because the identity information such as social security number cannot be revoked, we need an addition replaceable field l. Note that l cannot be any random number, because using a random value as public key requires public-key certification, which defies the purpose of identity-based encryption. In what follows, we use the driver's license number as l. The user's driver's license can be equipped with a smart card chip and store the private key.
3. Authenticate: The user and the identity provider engage in a challenge-response protocol as follows.
   (a) The user gives his public key to the identity provider, which is the identity information concatenated with the driver's license number l to the identity provider.
   (b) The identity provider picks a random nonce m. It runs Encrypt of IBE to encrypt m using the user's identity information concatenated with l as the public key.
   (c) The ciphertext is given to the user, who runs Decrypt of IBE with his private key. If the user is unable to correctly decrypt the ciphertext, the authentication fails and returns false.

(d) The identity provider queries the revocation server maintained by the ID authority for the number l in the public key of the user. If l has been revoked, then the authentication fails. Otherwise, the authentication is successful and returns true.

4. Refresh: The ID authority refreshes the private key of the user as follows.

   (a) The user authenticates his identity information and current driver's license number l to the ID authority.

   (b) The ID authority puts the the driver's license number l on the revocation server to indicate that l has been revoked.

   (c) The ID authority generates a new driver's license number l' for the user. The new public key of the user associated with his identity information is that identity information concatenated with l'. For example, the public key is 999-99-9999 $\circ$ 1234567890, where 999-99-9999 is the social security number and 1234567890 is the new driver's license number l'.

   (d) The ID authority runs Extract of IBE to compute a new private key, which is transmitted to the user via a secure channel. The user stores the private key in his smart card.

The main advantage of our authentication protocol is that the secret personal information is not released during the transaction, which minimizes identity theft attacks such as dumpster diving and shoulder surfing. Our protocol can be used in any user authentication applications. In particular, it can be used in any federated identity management system when a user authenticates his personal information with an identity provider. For example, a user is required to run Authenticate with the identity provider when an assertion of his social security number is to be generated. Without the corresponding private key, it is impossible for an identity thief is to accomplish this.

The above protocol is suitable for authenticating extremely sensitive, or unique and permanent identity information such as social security number or credit card number. It is also suitable for less sensitive information such as age, phone number, address. Multiple attributes can be aggregated to form one key, in order to reduce the number of private keys required. Revocation in our protocol uses on-line revocation servers. Some sensitive information such as social security number cannot be changed for a user. Therefore, we make the public key contain not only the social security number, but also a replaceable number such as driver's license number. Efficient and scalable revocation service has been implemented [14, 20] and is not repeated here.

Remark: Our solution is not aimed to solve identity thefts that involve stealing paper credentials (such as birth certificate, passport) to impersonate others. In our scheme, an attacker may be able to gather enough paper credentials of a victim and register in person with the ID authority a new identity-based public key as the victim. (This may be possible because the attacker is a close friend or relative of the victim.)

5 Security Analysis

In this section, we first analyze the security of the notarized federated identity management protocol, and then analyze the ID-based authentication protocol.

5.1 Notarized federated identity management

The security of our notarized federated identity management protocol is analyzed from the perspectives of the user, the identity provider, the service provider, and the notary server, as each of them has different requirements on the security provided by the system. In what follows, we assume the existence of a signature scheme that is secure against existential forgery by polynomial-time adversaries in the security parameter of the signature scheme. Existential forgery means that an adversary forges a signature that the notary server has not signed in the past. An adversary in our protocol can monitor traffics in unsecured channels, request for services, request the identity provider to blind assertions of her choice, and request the notary server to notarize assertions of her choice.

We assume that the notary server is trustworthy, and is trusted by all entities (users, identity providers, service providers). All entities are assumed to follow the federated identity management protocol presented in Section 2. The most important security requirement in our notarized federated identity management model is the security of notarized assertions, which is defined in Section 2 as the nonforgeability of a notarized assertion. Our implementations satisfy this property, which is stated in the following theorem.

Theorem 1. *In the notarized federated identity management protocol, no polynomial-time adversary can successfully forge a valid notarized assertion that is not generated by the notary server.*

Proof sketch: We give two implementations of the notarized assertion. One is based on a simple signature scheme, the other is based on STMS. In both implementations, forging a notarized assertion is equivalent to forging the signature of the notary server at a time quantum. This is infeasible, assuming the existence of a signature scheme that is secure against existential-forgery attacks. Therefore, the theorem holds. □

For the privacy protection of a user, an important privacy requirement is the secrecy of assertions. This is defined as that the protocol does not leak any information of the assertion to the adversary. Our notarized federated identity management protocol satisfies the secrecy requirement, which is summarized in the following theorem.

Theorem 2. *Assume the existence of a collision-resistant one-way hash function, and a secure symmetric key encryption scheme. In the notarized federated identity management protocol, a polynomial-time adversary and untrusted notary responders cannot obtain any information from a blinded assertion.*

Proof sketch: We will prove that (1) the key is difficult to guess and (2) the blinded assertion is pseudorandom. An assertion is encrypted by the identity provider using a symmetric key encryption scheme that is secure in the sense of an adversary's inability to distinguish the output from a random string [2]. The secret key for the encryption/decryption is computed as $H(N, P_2)$, where H is a collision-resistant one-way hash function, N is the session ID, P_2 is a public parameter, and "," denotes concatenation. Given the public index $H(N, P_1)$ of an assertion, where P_1 is another public parameter, the secret key is still difficult to guess. This is because of the collision-resistant and one-way properties of the hash function H. In addition, the blinded assertion is indistinguishable from a random string, because of the security of the encryption scheme. Therefore, adversaries and untrusted notary responders cannot obtain any information from the blinded assertions. □

For decentralized authorization systems such as the federated identity management, an important security requirement is accountability. To prevent possible disputes, identity providers should be held accountable for the assertions that they have generated. In addition, to prevent unauthorized information exchange among providers, users should be able to dispute any fraudulent assertion requests. These properties are achieved in our protocol.

Theorem 3. *In the notarized federated identity management protocol, the identity provider is held accountable for the assertions that it generates.*

Proof: The notary server stores the signed (blinded) assertion submitted by an identity provider in Step 7c of our notarized federated identity management protocol. In case of a dispute between a service provider and an identity provider on the validity of an assertion, the notary server reveals the signature, which is used to hold the identity provider accountable for generating the assertion. Therefore, Theorem 3 holds. □

Theorem 4. *In the notarized federated identity management protocol, providers are held accountable for any unauthorized information exchange among them.*

Proof: In our protocol, an identity provider should only generates assertions based on a *signed* request from a user. The identity provider is required to keep the signed requests for its own record in Step 4 of our notarized federated identity management protocol. Once unauthorized information sharing among providers is detected, the identity provider is not able to show any signed request by the user. Hence, it is responsible for the information leak. □

Theorem 5. *The notarized federated identity management protocol is secure against replay attacks.*

It is easy to see that Theorem 5 holds, because the session ID is randomly generated for each service request and the notarized assertions are generated by the notary server with the time-stamp information.

5.2 ID-based authentication

The security of the ID-based authentication protocol is defined as the adversary's inability of impersonating a user. Formally, an ID-based authentication protocol is secure if no polynomial-time adversary can distinguish with non-negligible probability the challenge ciphertexts of two nonces of her choice.

This security property is equivalent to the semantic security in the identity-based encryption scheme [5], which intuitively means that the adversary does not learn anything about the messages from observing the ciphertexts. The adversary in this protocol is allowed to monitor traffic in unsecured channels, request for the private keys of identities of her choice, request the identity authority to decrypt the challenge ciphertexts of her choice, and choose two challenge nonces whose ciphertexts are to be distinguished.

Theorem 6. *Given an identity-based encryption scheme that is semantic-secure against an adaptive polynomial-time adversary, the ID-based authentication protocol is secure.*

It is easy to see that theorem holds because of the assumption of a secure IBE scheme. This theorem shows that it is infeasible for an adversary to impersonate a user if the user's private key is not compromised. This leads to the following conclusion that states the property of our authentication protocol in preventing identity theft. In the ID-based authentication protocol, an adversary cannot successfully impersonate a user without stealing the user's tamper-resistant device that stores the identity private key.

6 Related Work

Our approach of using privacy protection as a means to avoid identity theft is related to anonymous credential systems [11, 17]. Anonymous credential systems (also called pseudonym systems) allow anonymous yet authenticated and accountable transactions between users and service providers. One of the main design goal of these systems is to achieve unlinkability of multiple showing of credentials. The Identity mix (idemix) project [7] is an anonymous credential system using the cryptographic protocols presented in [8]. Such a system consists of users and organizations. Organizations know the users only by pseudonyms. Different pseudonyms of the same user cannot be linked. An organization can issue a credential to a pseudonym, and the corresponding user can prove the possession of this credential to another organization (who knows her by a different pseudonym), without revealing anything more than the fact that she owns such a credential.

Existing anonymous credential systems are different from our single sign-on system, in that they do not consider a federated identity infrastructure behind the providers. In comparison, our system focuses on how to manage user authentication in the more realistic setting of a federation of providers. Our system achieves simple pseudonym solutions and efficient single sign-on by taking advantages of the federated structure. In particular, we do not need a credential system, because the assertions can be short-lived and generated on-line by identity providers.

In the past decade, the European Union and its member states have implemented a legal framework to provide guidance on processing of personal data with the specific aim to restore citizens' control over their data. To complement the legal framework, Camenisch *et al.* presented the architecture of PRIME

(Privacy and Identity Management for Europe), which implements a technical framework for processing personal data [6]. PRIME focuses on enabling users to actively manage and control the release of their private information. Thus, the PRIME system places a significant burden on users.

The federated identity management solution proposed by Bhargav-Spantzel, Squicciarini, and Bertino [3] emphasizes the need for proving the knowledge of personal information without actually revealing it, in order to help prevent identity theft. In their solution, personal data such as a social security number is never transmitted in the clear. Commitment schemes and zero-knowledge proofs are used by a user to commit data and prove the knowledge of the data. Our identity-based solution has a similar goal to this approach, but there is one important difference. We allow personal data such as social security numbers and credit card numbers to be transmitted in the clear. Yet, every time this information is used, the user needs to prove the possession of corresponding private keys. This requires minimal changes to the existing financial and administrative infrastructure, as personal information in our scheme is stored the same way as it is currently. Identity-based encryption [5] conveniently makes this possible, and, interestingly, this approach is also more efficient than zero-knowledge proof-of-knowledge protocols.

BBAE is the federated identity-management protocol proposed by Pfitzmann and Waidner [21]. They give a concrete browser-based single sign-on protocol that aims at the security of communications and the privacy of user's attributes. Their protocol is based on a standard browser, and therefore does not require the user to install any program. The security is based on extensive uses of SSL/TLS. The main difference with this and our approach is that we provide a notary mechanism for authenticating assertions when *IdP* and *SP* are not previous known to each other.

In the access control area, the closest work to ours is the framework for regulating service access and release of private information in web-services by Bonatti and Samarati [4]. They study the information disclosure using a language and policy approach. We designed cryptographic solutions to control and manage information exchange. Their framework mainly focuses on the client-server model, whereas our architecture include two different types of providers.

A counter measure for identity theft through location cross-checking and information filtering was recently proposed [25]. This paper addresses the identity cloning problem, and proposes to use personal location devices such as GPS and central monitoring systems to ensure the uniqueness of identities. However, the central monitoring system in their solution is likely to be a performance bottleneck. Moreover, because identity thieves are geographically dispersed, distributing the monitoring task into several locations is not feasible. In comparison, our solution is simple and efficient to adopt. Because we tie the secret identification information to a tamper-resistant smart card (e.g., driver's license), card theft can be easily noticed and reported by the card owner.

We compare our solutions with existing federated identity management proposals in Table 1.

Systems	Notarized FIM	BBAE [21]	ZK-based FIM [3]	idemix [7]	SAML [9]
Unknown providers	Yes	No	No	No	No
Brokering trust	Yes	No	No	No	No
***IdP/SP* separation**	Yes	No	Yes	Yes	No
ID-theft mitigation	Yes	No	Yes	Yes	No
Browser-based	No	Yes	No	No	Yes

Table 1. Comparisons of federated identity management systems. The term **unknown providers** indicates support for identity providers and service providers who do not have pre-established trust. **Brokering trust** refers to whether the protocol supports unknown providers to establish trust via trusted third-party, which is the notary server in our protocol. The term ***IdP/SP* separation** refers to the lack of direct communication between an identity provider and a service provider about the user's information. This separation benefits the user in terms of privacy protection.

7 Acknowledgements

We are grateful to David Croston, of IAM Technology, Inc., for suggesting the research topic of this paper and for useful comments.

References

1. A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia. Persistent authenticated dictionaries and their applications. In *Proc. Information Security Conference (ISC 2001)*, volume 2200 of *LNCS*, pages 379–393. Springer-Verlag, 2001.
2. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, December 2000.
3. A. Bhargav-Spantzel, A. C. Squicciarini, and E. Bertino. Establishing and protecting digital identity in federation systems. In *Proceedings of the 2005 ACM Workshop on Digital Identity Management*, pages 11–19, November 2005.
4. P. A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.
5. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto '01*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.
6. J. Camenisch, abhi shelat, D. Sommer, S. Fischer-Hübner, M. Hansen, H. Krasemann, G. Lacoste, R. Leenes, and J. Tseng. Privacy and identity management for everyone. In *Proceedings of the 2005 ACM Workshop on Digital Identity Management*, pages 20–27, November 2005.
7. J. Camenisch and E. V. Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, 2002.
8. J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Verlag, 2001.

9. S. Cantor, F. Hirsch, J. Kemp, R. Philpott, E. Maler, J. Hughes, J. Hodges, P. Mishra, and J. Moreh. Security Assertion Markup Language (SAML) V2.0. Version 2.0. OASIS Standards.
10. S. Cantor and J. Kemp. Liberty ID-FF Protocols amd Schema Specification. Version 1.2. Liberty Alliance Project. `http://www.projectliberty.org/specs/`.
11. D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
12. Cyber Security Industry Alliance. Internet security national survey, No. 2, December 2005. https://www.csialliance.org/StateofCyberSecurity2006/National_Survey_121305.PDF.
13. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology — Asiacrypt '02*, volume 2501 of *LNCS*, pages 548–566. Springer-Verlag, 2002.
14. M. T. Goodrich, M. Shin, R. Tamassia, and W. H. Winsborough. Authenticated dictionaries for fresh attribute credentials. In *Proc. Trust Management Conference*, volume 2692 of *LNCS*, pages 332–347. Springer, 2003.
15. M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proc. 2001 DARPA Information Survivability Conference and Exposition*, volume 2, pages 68–82, 2001.
16. Liberty Alliance Project. `http://www.projectliberty.org`.
17. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*. Springer Verlag, 1999.
18. P. Madsen, Y. Koga, and K. Takahashi. Federated identity management for protecting users from ID theft. In *Proceedings of the 2005 ACM Workshop on Digital Identity Management*, pages 77–83, November 2005.
19. R. C. Merkle. Protocols for public key cryptosystems. In *Proc. Symp. on Security and Privacy*, pages 122–134. IEEE Computer Society Press, 1980.
20. M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*, pages 217–228, 1998.
21. B. Pfitzmann and M. Waidner. Federated identity-management protocols. In *Security Protocols Workshop*, pages 153–174, 2003.
22. B. Shafiq, E. Bertino, and A. Ghafoor. Access control management in a distributed environment supporting dynamic collaboration. In *Proceedings of the 2005 ACM Workshop on Digital Identity Management*, pages 104–112, November 2005.
23. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology — Crypto '84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.
24. R. Tamassia. Authenticated data structures. In *Proc. European Symp. on Algorithms*, volume 2832 of *Lecture Notes in Computer Science*, pages 2–5. Springer-Verlag, 2003.
25. P. van Oorschot and S. Stubblebine. Countering identity theft through digital uniqueness, location cross-checking, and funneling. In *Proceedings of Financial Cryptography and Data Security (FC '05)*, pages 31–43, 2005.
26. Web Services Federation Language (WS-Federation), 2003. `ftp://www6.software.ibm.com/software/developer/library/ws-fed.pdf`.