View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Institutional Knowledge at Singapore Management University

Singapore Management University
Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems | School of Information Systems

3-2007

Privacy-Preserving Credentials Upon Trusted Computing Augmented Servers

Yanjiang YANG
Singapore Management University, yjyang@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Feng BAO
Singapore Management University, fbao@smu.edu.sg

DOI: https://doi.org/10.1007/978-3-540-72163-5_15

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Information Security Commons

Citation

YANG, Yanjiang; DENG, Robert H.; and BAO, Feng. Privacy-Preserving Credentials Upon Trusted Computing Augmented Servers. (2007). *Information Security Practice and Experience: Third International Conference, ISPEC 2007, Hong Kong, China, May 7-9: Proceedings*. 4464, 177-192. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/388

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

Privacy-Preserving Credentials Upon Trusted Computing Augmented Servers

Yanjiang Yang, Robert H. Deng, and Feng Bao

School of Information Systems
Singapore Management University, Singapore 178902
Institute for Infocomm Research, Singapore 119613
{yjyang,robertdeng}@smu.edu.sg,baofeng@i2r.a-star.edu.sg

Abstract. Credentials are an indispensable means for service access control in electronic commerce. However, regular credentials such as X.509 certificates and SPKI/SDSI certificates do not address user privacy at all, while anonymous credentials that protect user privacy are complex and have compatibility problems with existing PKIs. In this paper we propose privacy-preserving credentials, a concept between regular credentials and anonymous credentials. The privacy-preserving credentials enjoy the advantageous features of both regular credentials and anonymous credentials, and strike a balance between user anonymity and system complexity. We achieve this by employing computer servers equipped with TPMs (Trusted Platform Modules). We present a detailed construction for ElGamal encryption credentials. We also present XML-based specification for the privacy-preserving credentials.

1 Introduction

It is well accepted that user privacy is an important issue in online services such as electronic commerce [1]. User privacy concerns actually result from the fact that online systems routinely enforce access control over the services they provide in order to distinguish qualified and illegitimate users, and current standard technologies implement access control/authorization through user identification. For example, a user provides her credential (e.g., a X.509 certificate) to a service provider in order to attest her qualification for the service in question. As a result, the service provider is enabled to log transactions and derive accurate dossiers of user activities.

Currently, there are mainly three kinds of credentials in PKI: X.509 certificates [17], SPKI/SDSI authorization certificates [12,23], and attribute certificates [18]. A X.509 certificate binds a public key to a globally unique user identity so as to enable the use of public keys at the discretion of user identities. X.509 certificates are thus known as *identity certificates*. In access control, however, it has been noted that a user's identity is almost never a factor in an authorization decision [11], and what really counts is whether the user has the required permissions. This gives rise to the concept of SPKI/SDSI *authorization certificate* and *attribute certificate*: an authorization certificate binds a set of user attributes that

E. Dawson and D.S. Wong (Eds.): ISPEC 2007, LNCS 4464, pp. 177–192, 2007.
© Springer-Verlag Berlin Heidelberg 2007

convey access permissions to a public key, while an attribute certificate binds user attributes to an identity. These standard credentials do not deal with user privacy[1].

Anonymous credentials (e.g., [7,6,9,8]) can be viewed as a special class of certificates for authorization. Instead of directly passing a credential to the verifier as with the regular credentials, use of anonymous credentials is through zero-knowledge proof protocols [14], where the verifier ends up learning whether or not the credential satisfy its access control policies but nothing beyond this fact. *Unlinkability* is a core feature of anonymous credentials, i.e., transactions using the same credential cannot be linked. While anonymous credentials offer strong user privacy protection, they have not been widely used in real world applications. A main reason was believed to be that they are not compatible with the existing PKIs [5]. Other reasons may attribute to the use of zero-knowledge proof techniques, which results in: (1) limited expressiveness. Zero-knowledge proof techniques are not effective in conveying complex relations between user attributes and access control policies; (2) low efficiency. Zero-knowledge proof techniques are in general expensive in terms of both computation and communication, and this makes it particularly difficult for resource-constraint users (e.g., wireless users) to use anonymous credentials.

Our Contributions. In this paper, we propose privacy-preserving credentials, which represent a concept between regular credentials and anonymous credentials. Specifically, the privacy-preserving credentials are built upon and thus compatible with regular credentials, but endeavor to achieve unlinkability as of anonymous credentials. For efficiency reasons, we avoid any zero-knowledge proof technique; rather, we manage to achieve *relaxed unlinkability* (see Section 3 for details). As a result, the privacy-preserving credentials enjoy the advantages of both regular credentials and anonymous credentials: compatibility with existing PKI and rich expressiveness, of regular credentials, and privacy-enhancing feature of anonymous credentials. Moreover, we implement partial disclosure of sensitive attributes based on "need to know".

A key challenge in constructing the privacy-preserving credentials lies in the public keys embedded in credentials, which are globally unique quantities. The public keys cannot be disclosed to the service providers, but they must still be usable for data encryption or data authentication. We solve this problem by running a specialized software program, PEM (Privacy Enhancing Module), at the sever side, which composes "chameleon public keys" by *blinding* the original public keys without compromising the usages of the keys. Trustworthiness of PEM is maintained through a TPM under the auspice of Trusted Computing Group (TCG) specifications [28] (more details on TCG/TPM are provided in Appendix). We design our protocol using TPM commands Version 1.2 [29].

Organization. We review related work in Section 2, followed by discussions on the concept, general construction, and security features of our privacy-preserving

[1] While a SPKI/SDSI authorization certificate does not necessarily contain a user identity, the public key associated with the certificate uniquely indicates a user, which *links* all the transactions under the same certificate.

credentials in Section 3. An instantiation of the credentials for ElGamal public encryption keys is presented in Section 4. We implement credential specification using XML in Section 5, and Section 6 concludes the paper.

2 Related Work

Our work is clearly closely related to anonymous credentials (e.g., [7,6,9,8]). Anonymous credentials achieve *unlinkability*, which to our belief is an essential factor for any privacy-enhancing technique. As such, the privacy-preserving credentials we propose are endeavored to provide unlinkability. However, in order not to compromise efficiency, we shall avoid zero-knowledge proof techniques in our construction, so the privacy-preserving credentials attain *relaxed unlinkability*, striking a balance between user anonymity and system complexity.

Trust negotiation (e.g.,[24,31]) is a procedure whereby a user and a server establish trust through a gradual exchange of the user's credential attributes and the server's access control policies. The technique is on the one hand to protect sensitive attributes of users from unqualified server, while on the other to protect the server's access control policies against illegitimate users. After a successful negotiation, the server obtains the user credential. In other words, trust negotiation is not meant to protect user privacy from qualified server. In contrast, our privacy-preserving credentials are designed to protect user privacy from the server, be it qualified or unqualified.

The Oblivious Attribute Certificates proposed in [20] work in such a way that a user gets a service iff the attributes stored in her certificate satisfy the policies of the server, yet the server learns nothing about these attribute values. While the Oblivious Attribute Certificates do not rely on zero-knowledge proof techniques, they still have the limitations of anonymous credentials such as restricted expressiveness and low efficiency. The objective of our privacy-preserving credentials is not concealing attribute values from the server, but disclosing only those satisfying "need to know". Other certificate-based access control relates to ours include Secret Handshakes [3], Hidden Credentials [15], and Oblivious Signature Based Envelope [19]. They however work in different ways: the server sends an encrypted message to a user, and the user can decrypt iff she has a certificate having attribute values specified by the server's access control policies; but the server does not learn whether or not the user has such a certificate. [5] suggested a novel method to extend standard attribute certificates so as to achieve user privacy, but the resultant certificates are essentially linkable.

3 Privacy-Preserving Credentials

3.1 Concept

Conceptually, a credential contains a subject together with a set of subject attributes specified by name/value pairs (e.g., Issuer/ca, Age/28, Role/professor), attached with a digital signature over the credential content; the signature is

generated by a Credential Issuer using his private key, and the authenticity of the credential can be verified by using the Issuer's public key. In the context of authorization, the objective of a credential is to attest that the subject indicated (directly or indirectly) by the public key possesses the specified attributes, and authorization decisions must be made to the public key based upon the attributes. We point out that the subject of a SPKI/SDSI authorization certificate is directly the public key contained in the certificate; while the subject of a X.509 certificate refers to the user identity in the certificate, but what is really effective in authorization is the public key, and the access control decisions are eventually granted to the public key. It is important to observe that in a credential, the public key, i.e., the credential subject, must be a globally unique quantity, whereas other attributes are not necessarily unique. For example, a SPKI/SDSI certificate includes attributes such as Issuer, Delegation, Authorization and Validity, but none of them would have values that are unique. This is logical since in virtually any application in practice, there must be a group of users share an attribute value or a combination of attribute values, and thus have the same permission. As a result, even a user discloses the attribute values in her credential to a server while without revealing her public key (and possibly other unique quantities), the server is still not able to accurately link the current transaction to the user's previous transactions.

Based upon this observation, we propose the concept of *privacy-preserving credentials* outlined in Figure 1. In particular, the privacy-preserving credentials represent a kind of authorization tokens between regular credentials and

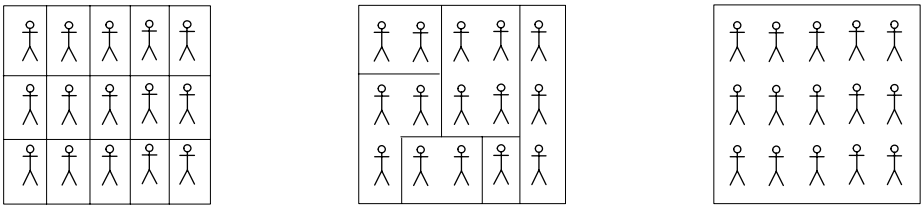

Fig. 1. Concept of Privacy-Preserving Credentials

anonymous credentials; regular credentials distinguish individual users, thereby not protecting user privacy at all (shown in Figure 1(a)), and anonymous credentials achieve unlinkability and recognize users as the whole user population, thereby fully protecting user privacy (shown in Figure 1(c)). The basic principle of the privacy-preserving credentials works as follows: the credential user does not reveal the credential subject (e.g., the public key and the user identity, and other unique quantities) to the verifier, and only discloses a minimal subset of attributes that satisfy "need to know" requirement of the verifier. As a result, the verifier distinguishes user cohorts among the whole user population (shown in

Figure 1(b)), where a cohort comprises users who have the same attribute values. In other words, the privacy-preserving credentials achieve *relaxed unlinkability* in the sense that the verifier can link an individual user to a cohort of users. The size of the cohorts relates only to the attribute values exposed to the verifier, and there exists the possibility that some particular users could be recognized as long as the size of the cohorts they belong to is one, but the majority of users cannot be differentiated (further discussion is given in subsection 4.3).

3.2 General Construction

The way we achieve privacy-preserving credentials is to use the credentials upon servers that are equipped with TCG-conformant TPMs. TPM at the server together with the Privacy Enhancing Module (PEM) constitutes a trusted computing platform that cannot be tampered with regardless of software or physical attacks (see Figure 2). PEM is a specialized software taking charge of enhancing user privacy, and users trust it to execute certain functions and not reveal information to the server. PEM is a protected application under the auspices of TPM. According to the TCG specifications, TPM takes integrity measurement of PEM and reports the integrity metrics to remote users through attestation. As a result, unless TPM is tampered with, the server cannot compromise PEM. It should be noted that while PEM colocates with the server, it essentially act as an extension of the user side.

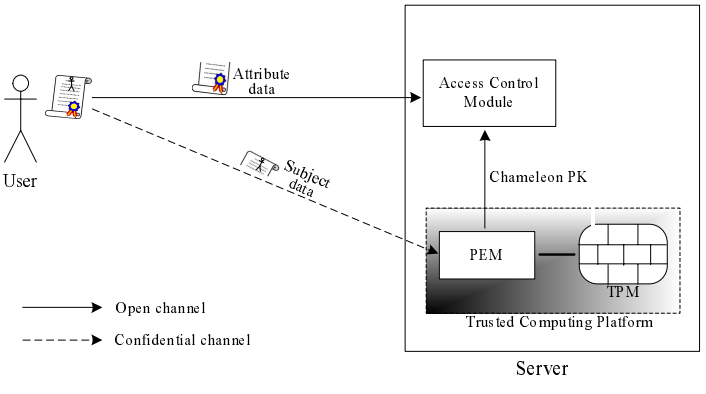

Fig. 2. General Construction of Privacy-Preserving Credentials

Our general construction works as follows. We partition the content of a credential into two parts: one is *subject data*, and the other is *attribute data*. The subject data, denoted as SubjDTA, include the data that uniquely indicate a subject, e.g., the public key, the unique user name if any, the digital signature over the credential (denoted as credSIG), and possibly some other data (e.g., the credential serial number if any, and some auxiliary data that are necessary for the verification of credSIG). The attribute data, denoted as AttrbDTA, include

all the subject attributes that affect access control decisions. As we made it clear the attribute data almost never uniquely identify a user, since it is unlikely that the attribute values in a credential are unique to a single user. As such, our way to achieve privacy-preserving credentials is that the attribute data are submitted to the server (precisely to the access control module that manages access control), but the subject data are given to PEM, shown in Figure 2.

The main task of PEM is to examine the validity of the credential, and derive a *chameleon public key* from the public key contained in SubjDTA and pass it to the access control module if the credential is valid. The access control module is an integral part of the server, responsible for evaluating the attribute values against its access control policies and making the final authorization decision. In our system, the server actually entrusts *validation of credentials* to PEM, but still takes the full responsibility in enforcing access control; and PEM does not in any way involve into the enforcement of access control. It is important to note that while PEM takes root in TPM, it still uses the resources (computation and storage) of the server platform for execution, so there is no efficiency penalty upon PEM. TPM does not perform any application-specific function, only taking charge of keeping the trusted state of PEM. Therefore, although TPM is strongly limited by its computation and storage capability, the overall system does not subject to the hardware constraint of the coprocessor.

3.3 Security Features

We desire the following security features upon the privacy-preserving credentials in the above construction.

- *Unforgeability*: Unforgeability is a fundamental feature of any credential system, which requires that nobody other than the Credential Issuer can issue valid credentials.
- *Partial disclosure of attributes*: A credential normally includes some sensitive attributes, and the credential user may be reluctant to reveal them to the verifier beyond "need to know". A user thus should be enabled to choose to disclose only the attributes that are absolutely necessary for the fulfilment of the server's access control requirements. The server should not be able to learn the hidden attribute values.
- *Relaxed Unlinkability*: The server should not be able to link transactions by inspecting the subject data that could obviously lead to linkability, and the extent of linkability is only dependent on the attribute values disclosed by the user. This suggests that the channel from the user to PEM (dote line in Figure 2) must be confidential against the server.
- *Usability of public keys*: In certificate-based access control, an authorization decision is often made to the public key contained in a credential, which is either for the purpose of data encryption or data authentication. We know that for any online service, access control is the first step whereby the server determines whether the user who uses a credential has the permission to the service in question; and what after access control is the *service provision*

procedure, where the public key of the credential must be used for either data encryption or verification of data signed by the user. The privacy-preserving credentials thus must not disable the usage of the public keys.

Goals of Adversary: Adversary behaviors towards the privacy-preserving credentials include: users may wish to break the unforgeability feature to forge credentials, and to defeat non-repudiation of digital signatures; the server may attempt to compromise partial disclosure of attributes by inferring the attribute values of hidden attributes, as well as to compromise relaxed unlinkability by linking individual users.

4 Concrete Instantiation

In this section, we give a concrete instantiation of the privacy-preserving credentials upon TPM-augmented servers, according to the above general construction. We know that the public key contained in a credential may correspond to either public key encryption or digital signature, but for limit of space we only instantiate ElGamal type digital signature credentials. Our instantiation can also be extended to Elgamal public key encryption credentials and even RSA credentials.

4.1 Preliminaries

We shall use the following notations in the sequel. p, q, g are parameters of ElGamal public key encryption scheme, where p, q are two large primes such that $q|p-1$, and $g \in Z_p^*$ is of order q. $h(.)$ is a collision resistant hash function such as SHA-1. $\{.\}_k$ denote secret key encryption by a secret key k. $\mathcal{E}_{PK}(.)$ denotes public key encryption by a public key PK, and $\mathcal{S}_{SK}(.)$ denotes signature signing by a private key SK.

ElGamal Public Key Encryption. A user has a key pair $(PK = y, SK = x)$, where $y = g^x \pmod p$ is the public key and x is the private key. The encryption of a message m generates a ciphertext (c_1, c_2), where $c_1 = g^t \pmod p$, $c_2 = my^t = mg^{xt} \pmod p$, with $t \in_R Z_q$. The decryption by the user using x works as $c_2/c_1^x = mg^{xt}/g^{tx} = m \pmod p$.

Merkle Hash Tree. The Merkle hash tree [22] is an efficient method to authenticate a set of data in such a way that given a signature over the whole data set along with some *auxiliary authenticating data*, a subset of data can be verified while in the absence of the remaining data. We illustrate the Merkle hash tree by a simple example shown in Figure 3, which is to authenticate a set of data $\{d_1, d_2, d_3, d_4\}$.

The construction of the Merkle hash tree is as follows: each leaf node of the tree is assigned a hash value of a datum, so the values represented by the leaf nodes are $h_1 = h(d_1), h_2 = h(d_2), h_3 = h(d_3), h_4 = h(d_4)$, respectively. The value of each internal node including the root node is derived from its child nodes. For example,

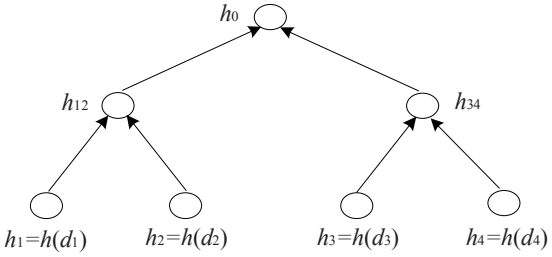

Fig. 3. Construction of Merkle Hash Tree

the value of node h_{12} is $h_{12} = h(h_1||h_2)$, where $h(.)$ is a collision-resistant one-way hash function and $||$ denotes concatenation. Similarly, $h_{34} = h(h_3||h_4)$ and $h_0 = h(h_{12}||h_{34})$. With a signature issued upon the root value h_0, any subset of the data set can be authenticated with the help of some auxiliary authenticating data while without disclosing the remaining data. For example, d_1 can be authenticated by given the authenticating data h_2 derived from d_2 and h_{34} derived from d_3 and d_4, while in the absence of d_2, d_3 and d_4; d_1 and d_2 can be authenticated if the authenticating data h_{34} is given, while without knowing d_3 and d_4. The efficiency of the Merkle hash tree rests with the fact that the number of the authenticating data is linear to $\log_2 N$, where N is the size of the whole data set. It is clear that given a root value of a data set, it is computationally infeasible to find a different data set that has the same root value, which amounts to the security of the Merkle hash tree method.

4.2 Protocol

Security Assumptions

- The hardware layer of TPM is tamper resistant regardless of hardware attacks and software attacks by any party.
- PEM is running in a protected execution environment, within which different applications run in isolation, free from observed or compromised by other processes running in the same protected partition, or by processes in any insecure partition that may exit in parallel. TPM defined by the TCG specifications itself does not suffice to afford this kind of protected execution environments, but a TPM with slightly extended mechanisms, such as the Intel's LaGrande Technology (LT) [16], can achieve this objective.

Overview. Let us first give some insights on our instantiation. First, the main challenge in constructing the privacy-preserving credentials is to simultaneously achieve relaxed unlinkability and usability of public keys. The feature of relaxed unlinkability requires the public key in a credential to be hidden from the server, while the feature of usability of public keys suggests the server must use the public key in the subsequent service provision procedure for either data encryption or

data authentication. Our solution is that PEM composes and gives a "chameleon public key" by "blinding" the actual public key to the server such that the usability of the public key is enabled, yet the server is not able to compute the actual public key.

Second, to enable a user to selectively disclose credential attributes (i.e., partial disclosure of attributes), we organize the content of a credential into a Merkle hash tree, and the credential signature credSIG by the Credential Issuer is issued upon the root value of the Merkle hash tree. Note that credSIG is a unique quantity, so it must be hidden from the server. In fact, it is included in SubjDTA, and never revealed to the server.

Third, TPM is responsible for integrity measurement and reporting of the platform including the protected software, PEM. In particular, PCR values of TPM record the integrity metrics of the platform from booting, to loading of operation system, to loading of PEM. Before sending a credential to the server, a user must first make sure that the protected computing platform is running in the expected status. TPM reports the platform configuration and status through *platform attestation* (it will be clear shortly how our protocols implement platform attestation).

Finally, recall the general construction that to achieve relaxed unlinkability, the subject data sent to PEM must be through a confidential channel against the server. We thus suppose PEM has a certified key pair (PK_{PEM}, SK_{PEM}) that corresponds to a standard public key encryption scheme, so that one can encrypt and send messages to it using the public key, and the server cannot decrypt and learn the messages. To achieve better security, we protect the secret key SK_{PEM} by sealed storage of TPM (invoking TPM_Seal to seal SK_{PEM}), and the integrity metrics of the platform is bound with the seal.

Based on these ideas, we next give a protocol to construct the privacy-preserving credentials that contain ElGamal public keys for encryption.

ElGamal Public Key Encryption Credentials. We suppose a user Alice has an ElGamal key pair $(PK_A = y = g^x \pmod p, SK_A = x)$, and the public key contained in her credential is thus PK_A. Moreover, without loss of generality, we suppose Alice needs to submit a subset of her credential attributes to a server in order to access a service. In such a case, the attribute data AttrbDTA comprises this subset of attribute values, while the auxiliary authenticating data that are derived from the hidden attribute values should be included in the subject data SubjDTA (see the general construction). We have to hide the auxiliary authenticating data of the hidden attributes from the server, since otherwise the server could infer the hidden attribute values in case the domains of the hidden attributes are small. To see this, suppose the server is given the auxiliary authenticating data d_{34} in order to authenticate d_1 and d_2 in Figure 3. While the server is not able to directly get d_3 and d_4 from d_{34}, it can enumerate every value in the domains of d_3 and d_4 to find out the actual values that amount to d_{34}, as long as the domains are small.

For a public key encryption credential, the public key PK_A will be used by the server to send sensitive data to Alice in the subsequent service provision

procedure. In our context, the server should not directly get PK_A, and PEM composes a chameleon public key in order to conceal PK_A. The protocol for platform attestation and credential processing is described as follows ($A \rightarrow B : m$ denotes that entity A sends m to B).

1. Alice → PEM: Attestation Request, R_A. R_A is a nonce generated by Alice.
2. PEM → TPM: TPM_Quote($h(R_A||ID_S)||indx(I)$). The TPM_Quote command instructs TPM to attest the platform status. The parameters given to this command include the indices of the PCRs that record the platform integrity metrics, I. TPM_Quote may also be given 160 bits of externally supplied data which, in our case, is the hash value of R_A and the server ID ID_S.
3. TPM → PEM: $\mathcal{S}_{AIK}(h(R_A||ID_S)||I)$. TPM returns a signature upon $h(R_A||ID_S)||I$ issued using its AIK.
4. PEM → Alice: I, $\mathcal{S}_{AIK}(h(R_A||ID_S)||I)$, AIK certificate. PEM sends platform integrity metrics I, the signature, and AIK certificate to Alice.
5. Alice: first checks the validity of $\mathcal{S}_{AIK}(h(R_A||ID_S)||I)$; then checks $h(R_A||ID_S)$ to ensure the message is fresh; finally decides whether the integrity metrics I represents a trustworthy state of the platform. From I, Alice can know whether PEM has been compromised or not, and whether it is running as expected.
6. Alice → PEM: AttrbDTA, c. If all checks pass, Alice encrypts SubjDTA using PK_{PEM}, the public key of PEM, to generate $c = (\mathcal{E}_{PK_{PEM}}(k), \{\text{SubjDTA}\}_k)$, where k is a random secret key for a standard secret key encryption scheme. Then Alice sends AttrbDTA and c to PEM.
7. PEM → TPM: TPM_Unseal(SK_{PEM}). PEM instructs TPM to unseal SK_{PEM}. SK_{PEM} is unsealed only if the platform is in the agreed state, i.e., the integrity metrics I matches the PCR values stored together with the protected SK_{PEM} at the time TPM_Seal was invoked.
8. TPM → PEM: SK_{PEM}. TPM returns SK_{PEM} to PEM.
9. PEM: decrypts c using SK_{PEM} to get k, and decrypts $\{\text{SubjDTA}\}_k$ using k to get SubjDTA; then verifies the authenticity of the credential by checking the validity of credSIG included in SubjDTA:
   (a) credSIG is valid: PEM picks a random blinding element $\alpha \in_R Z_q$ and composes a chameleon public key $PK'_A = g^{\alpha} PK_A = g^{x+\alpha} \pmod{p}$; it then encrypts α using PK_A to generate $c' = \mathcal{E}_{PK_A}(\alpha)$, and sends AttrbDTA, PK'_A, c', and VALID (a symbol indicating the credential is valid), to the server (precisely to the access control module of the server).
   (b) credSIG is invalid: PEM sends INVALID (a symbol indicating the credential is invalid) to the server.
10. Server: If receiving INVALID, it simply rejects Alice and aborts the transaction; otherwise, it evaluates AttrbDTA against its access control policies to determine whether the permission is granted to the user. If the permission is granted, the server sends c' to the user, and will use PK'_A for data encryption in the subsequent service provision.

11. Alice: if granted access permission and receiving c', she decrypts c' to get α, and computes and uses $SK'_A = SK_A + \alpha = x + \alpha \pmod q$ as her private key in the service provision procedure. (PK'_A, SK'_A) clearly constitutes a valid ElGamal key pair.

4.3 Security Analysis

We next discuss how the above instantiation achieve the security features in Section 3.

Unforgeability. Unforgeability of credentials is trivial due to the security of the Merkle hash tree and the digital signature of the Credential Issuer.

Partial disclosure of attributes. The feature of partial disclosure of attributes is also clear, since a credential is signed by the Credential Issuer upon the root value of the Merkle hash tree organized by the content of the credential, so the user is enabled to only reveal a subset of attributes that satisfy the server's access control policies. Furthermore, the server cannot see the auxiliary authenticating data derived from the hidden attributes, thereby learning nothing on the hidden attributes even their domains are small.

Relaxed Unlinkability. Clearly, relaxed unlinkability is achieved by the approach that the server is not allowed to inspect the subject data SubjDTA that obviously leads to linkability (all unique quantities including credSIG are included in SubjDTA). From the instantiation, the extent of linkability depends totally on the attribute values contained in AttrbDTA, since the chameleon public key and the ciphertext of the blinding element are random quantities. "Relaxability" (relaxed unlinkability) comes from the fact that users would be linked to cohorts, each consists of users having the same attribute values. It is important to note that we do not rule out the possibility that some particular users can be linked, in which case the size of the cohort a user belongs to is 1. For example, a particular user may have a set of attribute values distinct from anyone else. We next give an analysis on the conditions under which no individual user is linked.

Suppose a type of privacy-preserving credentials has κ attributes (attribute 1 to attribute κ), and attribute i takes v_i values, $i = 1..\kappa$. Note that these v_i values do not necessarily constitute the domain of attribute i, and they are the values that are actually assigned to users. There are thus $\prod_{i=1}^{\kappa} v_i$ combinations of attribute values in total, and clearly each combination determines a cohort. In order that no individual user is linked, there must be at least two users to take every combination of the attribute values, i.e., the size of every cohort must be at least 2. As such, there are at least $2\prod_{i=1}^{\kappa} v_i$ users. Further, consider each particular attribute: for each of the v_i values of attribute i, it must occur at least $2v_1...v_{i-1}v_{i+1}...v_\kappa = 2\prod_{j=1,j\neq i}^{\kappa} v_j$ times when in combination with the remaining attributes, which suggests that there must be at least $2\prod_{j=1,j\neq i}^{\kappa} v_j$ users to take the value. As a result, we have the following claim.

CLAIM. *There would be no individual user be linked as long as the following conditions are satisfied:*

1. *There are at least* $2\prod_{i=1}^{\kappa} v_i$ *users registered to the Credential Issuer; and*
2. *For each value of the* v_i *values of attribute* i, $i = 1..\kappa$, *there are at least* $2\prod_{j=1,j\neq i}^{\kappa} v_j$ *users taking the value.*

Of course, the second condition already implies the first one. These conditions can be used as a criteria to evaluate the relaxed unlinkability of the privacy-preserving credentials.

Usability of public keys. The usability of public keys is determined by the chameleon public keys and the corresponding private keys. It can be easily seen that what we construct in the above instantiation is a valid ElGamal encryption key pair.

4.4 Discussions

We discuss the advantages and limits of the privacy-preserving credentials. Our constructions do not use any zero-knowledge proof technique, hence the privacy-preserving credentials have efficiency advantage over anonymous credentials. We do not give the exact comparison result, as this depends on the specific anonymous credential schemes to be compared, but a casual estimate on the overhead of the privacy-preserving credentials is simply several operations of signature generation/verification and encryption/decryption (note that platform attestation involves essentially no more than one signature signing operation and one signature verification operation.). Moreover, there is no major architectural change at the user side, so we believe resource-constraint users such as mobile devices will not be affected in our system. A more appealing advantage is that since the privacy-preserving credentials are totally compatible with regular credentials, they have no expressiveness problem; more importantly, this makes them implementable directly upon existing standard PKIs. In contrast, a major limit of anonymous credentials is their incompatibility with PKIs.

The main limit of the privacy-preserving credentials we can imagine is that they have to use TPM at the server side. It should be noted that TPM can only be trusted up to the level of its hardware tamper resistance, and should be assumed to deter only the least resourceful attackers [13]. On the bright side however, numerous techniques to take hardware tamper resistance and the threat from the local host users into account in the design of trusted systems have been studied extensively, and much progress has been made in recent years (e.g., [21,25,26,27]). It is thus reasonable to expect that as tamper resistant hardware becomes more widely adopted, high quality tamper resistant hardware will become affordable due to economy of scale.

5 Credential Specification

We implement XML-based credential specification for the privacy-preserving credentials, compatible with the structure of regular credentials such as X.509 certificates and SPKI/SDSI authorization certificates. Observe that while the

```
<DOCTYPE CashBank_Customer[
   <!ELEMENT CashBank_Customer(issuer, validity, profession,
         city, dateBirth, credLevel, cusID, publicKey, credSIG, extension)>
   <!ELEMENT issuer ANY>
   <!ELEMENT validity (#PCDATA|NULL)>
   <!ELEMENT profession (#PCDATA|NULL)>
   <!ELEMENT city (#PCDATA|NULL)>
   <!ELEMENT dateBirth (#PCDATA|NULL)>
   <!ELEMENT credLevel (HIGH|MEDIUM|LOW|NULL)>
   <!ELEMENT extension (extAttr*)>
   <!ELEMENT extAttr ANY>
   <!ATTLIST extAttr attrSeq CDATA #REQUIRED
      attrV CDATA #REQUIRED>
   <!ATTLIST issuer XML:LINK CDATA #FIXED "SIMPLE"
      HREF CDATA #REQUIRED
      signKey CDATA #REQUIRED>
      signAlg CDATA #REQUIRED>
   <!ATTLIST CashBank_Customer cusID ID NULL>
   <!ATTLIST CashBank_Customer publicKey CDATA NULL)>
   <!ATTLIST CashBank_Customer credSIG CDATA NULL)>
]>
```

```
<CashBank_Customer cusID='NULL' publicKey='NULL' credSIG='NLL'>
<issuer HREF='http://www.cashbank.com'
   signKey='2ABG64897HJ' signAlg='RSA'>
<validity> 01-10-1006 </validity>
<profession> software_engineer </profesion>
<city> NULL </city>
<dateBirth> NULL </dateBirth>
<credLevel> MEDIUM </credLevel>
<extension>
   <extAttr attrSeq= '4' attrV='♯◇†‖⌜‡⌟↯♭ℓ'/>
   <extAttr attrSeq= '5' attrV='↯®‖ℑØ§∃℘§⌜£ℑ'/>
   <extAttr attrSeq= '7' attrV='⌜Ø©ℜ√⊤§‡¶✠'/>
   <extAttr attrSeq= '8' attrV='¶ℓ℘ℜℑ⊤♣♯♮†'/>
   <extAttr attrSeq= '9' attrV='∞□△ℵℏ∂♡℧'/>
</extension>
</CashBank_Customer>
```

Fig. 4. Example of Privacy-Preserving Template and Credential

exact fields contained in regular credentials may be different, they have similar structure, e.g., they usually include fields such as Serial Number, Issuer, Validity Period, Subject Name, Public key, etc., and an Extension field. Our basic idea for constructing the privacy-preserving credentials is utilizing the Extension field to encode the data to be submitted to PEM. While we can directly extend the X.509 certificates or the SPKI/SDSI certificates by placing all the application-specific attributes and the data intended for PEM in the Extension field, the examples we give below do not follow this method, simply for illustration purposes. XML [30] has extensive support in practice, so the implementation of XML-based specification entitles the privacy-preserving credentials wider applicability. For instance, we can use the privacy-preserving credentials in a trust negotiation system that enforces P3P privacy policies.

To simplify the management of credentials, we define *credential templates*. A credential template specifies a type of credentials specific to a particular application. We model a credential template as a XML DTD [30]. The upper part of Figure 4 shows an example of a privacy-preserving credential template CashBank_Customer. To facilitate partial disclosure of attributes, template fields are

assigned a default value NULL, which is a special symbol indicating the field may be concealed from the server.

A credential is an instance of the credential template, specifying the attribute values that characterize a user. A privacy-preserving credential is thus a valid XML document conforming to the corresponding DTD credential template. The lower part of Figure 4 gives an instance of the CashBank_Constomer template.

The attributes to be concealed from the server are assigned the special symbol NULL, and the auxiliary authenticating data derived from them according to the Merkle hash tree are encoded in the "extension" field. To simplify credential parsing, each piece of auxiliary authenticating data is encrypted as a separate extended attribute "extAttr". In particular, the example credential in Figure 4 (lower part) is as follows: the cleartexts for cusID, publicKey, and credSIG are removed, and the ciphertexts of them are encoded in the extension filed as "attrSeq= '7' attrV='⌜∅©ℜ√⊤§‡¶✠"', "attrSeq= '8' attrV='¶℘ℜℑ⊤♣♯♮†"', and "attrSeq= '9' attrV='∞□△ℵℏ∂♡℧"', respectively; the ciphertext of the hidden attribute "city" is represented by "attrSeq= '4' attrV='♯◇†‖⌜‡⊥♮♭ℓ"' and the hidden attribute "dataBirth" is encoded as "attrSeq= '5' attrV='♮®‖ℑ∅§∃℘§⌜£ℑ"'.

6 Conclusions

The new initiatives of trusted computing by placing TPM at the server machine are a promising paradigm in addressing user privacy in online services. Upon such servers, we proposed privacy-preserving credentials that represent a concept between regular credentials and anonymous credentials, in the sense that the privacy-preserving credentials are compatible with regular credentials while incorporating the privacy-enhancing features of anonymous credentials. We gave concrete construction of the privacy-preserving credentials containing ElGamal encryption keys for data encryption. We also implemented XML-based credential specification.

References

1. A. Acquisti. Privacy in Electronic Commerce and the Economics of Immediate Gratification. *Proc. ACM. Electronic Commerce (EC 04)*, pp. 21-29, 2004.
2. E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation, *Proc. ACM Conference on Computer and Communications Security, CCS'04*, pp. 132-145, 2004.
3. D. Balfanz, et al. Secret HandShakes from Pairing-based Key Agreement. *Proc. IEEE Symposium on Security and Privacy*, pp. 180-196, 2003.
4. R. Bradshaw, J. Hlot, K. Seamons. Concealing Complex Policies with Hidden Credentials. *Proc. ACM Conference on Computer and Communication Security*, pp. 146-157, 2004.
5. V. Benjumea, J. Lopez, J. A. Montenegro, and J. M. Troya, A First Approach to Provide Anonymity in Attribute Certificate. *Proc. Public Key Cryptography, PKC'04*, LNCS 2947, pp. 402-415, 2004.

6. S. Brands. Rethinking Public Key Infrastructures and Digital Certificates: Building in Privay. MIT Press, 2000.
7. D. Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete, *Communications of the ACM*, Vol 28, No. 10, pp. 1030-1044, 1985.
8. J. Camenisch and E. V. Herreweghen, Design and Implementation of the Idemix Anonymous Credential System. *Proc. ACM Conference on Computer and Communication Security, CCS'02*, 2002.
9. J. Camenisch and A. Lysyanskaya, An Efficient Non-Transferable Anonymous Multi-Show Credential System with Optional Anonymity Revocation. *Proc. Advances in Cryptology, Eurocrypt'01*, LNCS 2656, pp. 93-118, 2001.
10. W. Diffie, M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644-654, 1976.
11. C. M. Cllison. The Nature of A Usable PKI. *Computer Networks*, Vol. 31, No. 8, Elsevier, pp. 823-830, 1999.
12. C. M. Ellison, B. Frantz, B. Lampson, R.Rivest, B. Thomas, T. Ylnen. SPKI Certificate Theory, RFC 2693.
13. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. A Virtual Machine-Based PLatform for Trusted Computing. *Proc. 9th ACM Symposium on Operating Systems Principles*, pp. 193-206, 2003.
14. S. Goldwasser, S. Micali, C. Rackoff. The Knowledge Complexity of Interactive Proof-systems. 17^{th} *ACM Symposium on Theory of Computing*, pp. 291-304, 1985.
15. J. Holt, R. Bradshaw, K. Seamons, H. Orman. Hidden Credentials. *Proc. ACM Workshop in the Electronic Society*, pp. 1-8, 2003.
16. LaGrande technology architecrure. Intel Developer Forum, 2003.
17. ISO/IEC 9594-8, Information Technology - Open Systems Interconnections - The Directory: Authentication Framework.
18. ITU-T X.509 Recommandation, 2000, available at http://www.itu.int/rec/T-REC-X.509-200003-I/
19. N. Li, W. Du, D. Boneh. Oblivious Signature-based Envelope. *Proc. ACM Symposium on Principles of Distributed Computing*, pp. 182-189, 2003.
20. J. Li, N. L. Oacerts: Oblivious Attribute Certificates. *Proc. Applied Cryptography and Network Security*, LNCS 3531, pp. 301-307, 2005.
21. C. Mitchell. Trusted Computing, pp. 115-141, The Instituition of Electronic Engineers, London, 2005.
22. R. Merkle. A Certified Digital Signature. *Proc. Advances in Cryptology, Crypto'89*, LNCS 0435, pp. 218-238, 1989.
23. R. Rivest, B. Lampson. SDSI - A Simple Distributed Security Infrastructure. http://theory.lcs.mit.edu/ rivest/sdsi10.html, 1996.
24. K. E. Seamons, M. Winslett, T. Yu. Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation. *Proc. Network and Distributed System Security Symposium*, 2001.
25. S. Smith. *Trusted Computing Platforms - Design and Applications*. Springer 2005.
26. R. Sandhu, X. Zhang. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. *Proc. ACM symposium on Access control models and technologies*, pp. 147-158, 2005.
27. R. Sailer, X. Zhang, T. Jaeger, L, van Doorn. *Design and Implementation of a TCG-based Integrity Measurement Architecture*, USENIX Security Symposium, pp. 223-238, USENIX, 2004.
28. Trusted Computing Group. www.trustedcomputinggroup.org

29. TCG. TPM Main, Part 3 Commands, TCG Specification Ver. 1.2, Revision 62, www.trustedcomputinggroup.org, 2003.
30. World Wide Consortium. http://www.w3.org
31. T. Yu, M. Winslett. A Unified Scheme for Resource Protection in Automated Trust Negotiation. *IEEE Symposium on Security and Privacy*, pp. 110-122, 2003.

Appendix: TCG/TPM

The Trusted Computing Group (TCG) [28] defines a set of specifications aiming to provide hardware-based root of trust and a set of mechanisms to propagate trust to applications as well as across platforms. The root of trust in TCG is TPM (Trusted Platform Module), a tamper resistant secure coprocessor. TPM provides cryptographic functions, such as random number generation and RSA algorithms. Security mechanisms offered by TPM include integrity measurement and reporting, and sealed storage for secret data such as cryptographic keys. We next give a brief introduction on them. A TPM contains a set of Platform Configuration Registers (PCRs). PCR values record the integrity and state of a running platform from booting to loading of operation system to of loading applications [26]. With the integrity measurement and storage, TPM (attestator) can attest to a remote challenging platform the integrity of the platform under its protection through *platform attestation*. In particular, the challenging platform sends a challenge message to the attestator platform, who in turn returns the related PCR values signed by its Attestation Identity Key (AIK); the challenging platform verifies this attestation by comparing the signed values with expected values. The TPM command that instructs TPM to report the signed PCR values is TPM_Quote, whose input parameters specify the indices of the PCRs to be reported. Attestation can also be anonymous through Direct Anonymous Attestation [2].

TPM provides *sealed storage* that protect sensitive data with integrity values. Besides applying an encryption key (public key encryption) to encrypt the data, one or more PCR values are stored together with protected data during encryption. Consequently, TPM releases a protected data only if the current PCR values match those stored during encryption. The encryption key is protected either by a storage root key (SRK) that resides within TPM or by a key protected by the SRK. This actually forms a key hierarchy with the root being the SRK. The TPM commands that relate to sealed storage include TPM_Seal and TPM_Unseal. The TPM_Seal operation allows the invoking entity to explicitly state the future "trusted" configuration that the platform must be in for the secret to be revealed. The TPM_Seal operation also implicitly includes the relevant platform configuration (PCR values) when the TPM_Seal operation was performed. The TPM_Unseal operation will reveal TPM_Seal'ed data only if it was encrypted on this platform and the current configuration is the one named as qualified to decrypt it.