Protecting (Anonymous) Credentials with the Trusted Computing Group's TPM V1.2*

Jan Camenisch

IBM Research, Zurich Research Lab
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
jca@zurich.ibm.com

Abstract. Digital credentials and certificates can easily be shared and copied. For instance, if a user possesses a credential that allows her to access some service, she can easily share it with her friends and thereby let them use the service as well. While with non-anonymous credentials, this sharing can to some extend be detected by the fact that some credentials get used too often, such detection is not possible with anonymous credentials. Furthermore, the honest user is also at risk of identity theft: malicious software such as viruses and worms or phishing attacks can without too much difficulty steal her credentials.
One solution to the problem is to use tamper-resistant hardware tokens to which a credential is bound such that a credential can only be used in connection with the token. Although this approach is sometimes taken for isolated high security applications, it is not used widely because of the organizational overhead to distribute such tokens. Moreover, such tokens are usually very application specific and hence cannot be used with different applications (from different service providers).
Recently, however, manufacturers have started to embed into computers a tamper-resistant piece of hardware, called trusted platform modules (TPM), as specified by the Trusted Computing Group. In this paper we show that this module can in fact be used to secure anonymous as well as non-anonymous credentials. We provide a mechanism to insure that credentials can only be used with the TPM it got issued to. We then extend our solution to one that allows the use of credentials not only with the TPM they got issued to but also with other TPMs of the *same* user. Finally, we show how to secure a full-fledged anonymous credential system.

1 Introduction

Due to their nature, digital credentials can easily be copied and distributed. Thus, on the one hand, a computer virus or worm, for instance, can easily steal a user's credentials and, on the other hand, a user can even easier share her credentials (illegitimately) with her friends. In case credentials are used to protect access to valuable resources or services, such things should obviously be prevented and the credentials themselves be protected as well.

Although one can obtain some protection by software (such as storing credentials only in encrypted form and trying to secure the operating system), containment of credentials in a hardware token offers much better protection. While such protection is in

* A full version of this paper is available at www.zurich.ibm.com/~jca/publications.

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

principle possible also for credentials consisting of username and password (e.g., using so-called password-based key exchange protocols [3, 2, 17]), one would rather use public key cryptography because passwords have low entropy (so that they can be stored in a human brain). Credentials based on public key cryptography work as follows: A public/private key pair is generated (inside the hardware token) and the public key is sent to the issuer of the credential. The issuer either just stores the public key in a list of authenticated keys or actually issues a certificate on the public key, i.e., signs the public key together with some further (access) information or attributes. When the user then wants to access the service service or resource using her credentials, she sends the public key (and, if necessary, also the certificate) to the resource or service provider. The provider then either checks whether the public key is in the list of authorized keys or verifies the validity of certificate. Finally, using the secret key, the user (or the hardware token) identifies as the owner of the public key, and then will obtain access to the requested service or resource. If one uses so-called zero-knowledge protocols for this identification process (e.g., [15, 20]), this process does not divulge the secret key. Thus, if the only interface the hardware token offers is to generate a key pair, output the public key, and to run a zero-knowledge identification protocol w.r.t. this key pair, no information about the secret key is leaked from the hardware token and the credential cannot be used without the hardware token. Thus, the access credentials are protected from malicious software such as viruses and worms. Furthermore, assuming that the hardware token is tamper-resistant and it is ensured that the secret key is indeed held in the hardware token, users can no longer share credentials without sharing the token as well.

We note that this kind of protection is also applicable to so-called anonymous or private credentials [11, 6].

Today, credentials are only rarely protected by hardware tokens. One reason for this is that the cost of the deployment of such tokens is still too high, in particular as the tokens are not interoperable and can hence only be used for isolated applications. As a result, weak authentication is used, which in combination with the growing rate of frauds such as phishing attacks seriously hinder the growth of e-commerce on the Internet and even turn people away from using the Internet for financial transactions.

TCG's Trusted Platform Module. Recently, manufacturers have started to embed so-called trusted platform modules (TPM) [21] into computers. These modules are one of the building blocks for trusted computing that are specified by the Trusted Computing Group (TCG). They are essentially smartcard chips build into a computing device. These modules can, among other things, generate public and secret key pairs. Moreover, they have a mechanism to remotely convince another party that a key pair was indeed generated by them and that the secret is protected by the module (i.e., cannot not be extracted from the module except by physically breaking it open). The mechanism is called direct anonymous attestation (DAA) [21, 5] and can be seen as a group signature scheme without the capability to open signatures (or anonymity revocation) but with a mechanism to detect rogue members (TPMs in this case). In a group signature scheme, there is a group manager (the manufacturer of TPMs in our case) who controls who are the members of the group (all TPMs produced by the manufacturer). Group signatures allow member of the group to sign anonymously on behalf of the group. Thus, the capa-

bility to sign proves group membership and hence, as TPMs are instructed to sign only public keys to which they holds the secret key, the desired mechanism results.

Our Contribution. In principle one could use the TPM to secure credentials by having the TPM generate a key pair, proving that the key pair was indeed generated by the TPM (using the DAA protocol), and then issue a certificate on the signed public key. However, this approach does not protect the user's privacy as the issuing and each use of the certificate can be linked.

In this paper we show how the TPM's functionalities can be used to secure credentials and how to do it in a privacy friendly way. More precisely, we analyze the DAA protocol and show how the TPM's part of that protocol can be used to obtain a new, DAA-like protocol that allows one to issue (anonymous) credentials to users in such a way that they can only be used in conjunction with the TPM they got issued to. This protocol provides privacy to users, i.e., the issuer of the credential and the provider of the service or resource cannot link the two transactions. Essentially, we obtain an anonymous credential system where a user can only use her credentials in cooperation with her TPM. We note that our new protocol does not require any modification of the TPM and hence our solution can be realized with the TPM following the V1.2 specification which are already found in some computing devices today.

While our solution offers users protection against viruses and service providers protection against fraudulent users, it has the drawback that users can use a credential only with the device (resp., the TPM embedded into this device) the credential was issued to. Thus, credentials are not portable as they would for instance be if they were tied to a smartcard or a USB key. We therefore extend our solution to allow each user to use all *her* credentials with all *her* devices. We also show how to protect conventional (i.e., non-anonymous) credentials as well how to extend our scheme to a full-fledged anonymous credential system.

We believe that once TPMs according to the V1.2 specification [21] become widely available, our solution can enable widespread strong authentication and privacy protection for electronic transactions and help ignite the staggering growth of e-commerce on the Internet.

2 Model and Security Requirements of Our Scheme

A system for protecting anonymous credentials for use with all of a user's devices and *only* with those is a follows. It consists of five kinds of parties, device-credential issuers, application-credential issuers, service (or resource) providers, devices, and users. For simplicity we will consider only a single device-credential issuer, a single application-credential issuer and a single provider. We stress, however, that our scheme will also achieve all security properties if multiple such parties participate. The system features the following procedure.

KeyGenDevCredI and KeyGenAppCredA. These are the key generators for the device-credential and the application-credential issuer. On input a security parameter λ, they output a public/secret key pair (PK_D, SK_D), resp., (PK_A, SK_A), for the device-credential and the application credential issuer, respectively.

GetDevCred. This is a protocol between a user's device and a device-credential issuer. The input to both parties are the latter's public key, PK_D, and the user's identity ID_U. The issuer's additional input is his secret signing key SK_D. In case the user is eligible to register the device (e.g., because she has not yet registered too many devices), the user's device's output of the protocol is a device credential and some related secret keys (which will be held inside the device's TPM).

GetAppCred. This is a protocol between a user's device and an application-credential issuer. The input to both parties are the latter's public key PK_A. The user's device gets as additional input the identity ID_U. The application-credential issuer's additional input is his secret signing key SK_A. In case the user is eligible to obtain an application credential, the user's device's output of the protocol is an application credential and some related secret keys, all of which the device outputs.

UseAppCred. This is a protocol between a user's device and a service provider. The input to both parties are the public keys PK_A and PK_D of the device- and application-credential issuers, respectively. The device's input consists of its device credential and the related secret keys (the latter are kept in the device's TPM), the user's identity, as well as an application credential the user has obtained with one of her devices and the related secret keys. The output of the service or resource provider is either accept or reject.

The registration of a user's device does not necessarily require the user's identity – it can in principle also be done anonymously. We discuss this issue in §4.2, but for now require the user to identify herself to register a device.

The security requirements are as follows:

Unforgeability. We require that no adversary, who can register as many devices and retrieve as many application credentials as he wishes, can successfully run the protocol UseAppCred with a service-provider with an unregistered device or can successfully run the protocol UseAppCred with more (hidden) identities that he has obtained application credentials.

Anonymity. We require that even if the device-credential issuer, the application-credential, and the service provider collude, they cannot link different transactions by the same user except those transactions to register devices that were conducted under the same identity.

We note that the unforgeability property ensures that an adversary can only use an application credential with devices that were registered w.r.t. some particular user.

3 Preliminaries

3.1 Known Discrete-Logarithm-Based, Zero-Knowledge Proofs

In the common parameters model, we use several previously known results for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [20] or a composite [16, 14], (2) proof of knowledge of equality of representation modulo two (possibly different) prime [12] or composite [8] moduli, (3) proof that a commitment opens to the product of two other committed values [8,

4], (4) proof that a committed value lies in a given integer interval [10, 8], and also (5) proof of the disjunction or conjunction of any two of the previous [13]. These protocols modulo a composite are secure under the strong RSA assumption and modulo a prime under the discrete logarithm assumption. When referring to the proofs above, we will follow the notation introduced by Camenisch and Stadler [9] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

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

denotes a "*zero-knowledge Proof of Knowledge of integers* α, β, *and* δ *such that* $y = g^{\alpha}h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\delta}$ *holds, where* $u \leq \alpha \leq v$," where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. We apply the Fiat-Shamir heuristic [15] to turn such proofs of knowledge into signatures of knowledge on some message m; denoted as, e.g., $SPK\{(\alpha) : y = g^{\alpha}\}(m)$.

3.2 Direct Anonymous Attestation

The direct anonymous attestation (DAA) protocol [5, 21] applies the Camenisch-Lysyanskaya signature scheme [7] and the related protocols to issue a certificate (attestation) to a computing platform that it is genuine and to allow a platform to remotely prove to a service provider that it is indeed genuine while protecting the platform user's privacy. More precisely, the attestation is issued to the trusted platform module (TPM) embedded into the platform. Let us describe the DAA protocol in more detail.

The involved parties are an attester, a device, a platform, a TPM embedded into the device, and a verifier who wants to get convinced by the platform that it got attestation. To communicate with the outside world, the TPM is dependent on the other components of the device. We call all these other components the platform, i.e., a device consist of two (separate) parties, a TPM and a platform. the TPM communicates only with the platform.

The idea underlying DAA is to use Camenisch-Lysyanskaya (CL) signature [7] and the related protocols to sign as attestation secret messages, i.e., "messages" m_0 and m_1 chosen by the TPM. The reason that the TPM chooses two instead of just a single secret message is technical – it allows one to keep the message space small and thus to keep small the primes e that need to be generated for each signature by the attester which in turn makes generating them during signature generation much more efficient) while guaranteeing sufficient entropy in the TPM's secret.

Now, when the platform wants to prove to the verifier that it has embedded an attested TPM, the TPM generates a commitment (pseudonym) N_V on (m_0, m_1) and then proves that it has gotten a signature from the attester on the committed messages. This proof is done using the protocol provided together with CL signature [7] with the difference that N_V is a special commitment, i.e., $N_V = \zeta^{m_0+2^{\ell}m_1}$ for some given ζ and ℓ. We refer to [5] for more details on N_V.

To keep the TPM as small a chip as possible, the direct anonymous attestation protocol was designed such that all operations for retrieving a signature and proving possession of a signature that are not security critical are be outsourced to the platform

and computed there. In particular, as a platform could always destroy the privacy of its TPM, all operations that are necessary for privacy but not for security are performed by the platform. For instance, in the protocol to prove possession of a signature (attestation), the TPM computes the "commitment" N_V and the prover's part of the proof protocol

$$PK\{(m_0, m_1, v): \quad U \equiv R_0^{m_0} R_1^{m_1} S^v \pmod{n} \wedge N_V \equiv \zeta^{m_0 + m_1 2^\ell} \wedge m_0, m_1 \in \{0,1\}^{\ell_m + \ell_\varnothing + \ell_\mathcal{H} + 2}\} \ ,$$

where ℓ, ℓ_m, $\ell_\varnothing + \ell_\mathcal{H}$ are security parameters (cf. [5]). Note that this proof reveals U, i.e., does not hide the TPM's (and thus the platform's) identity. The platform then transform the TPM's messages of that proof, using its knowledge of A and e, into the prover's messages of the following proof protocol

$$SPK\{(m_0, m_1, \tilde{v}, e): \quad Z \equiv \pm A'^e S^{\tilde{v}} R_0^{m_0} R_1^{m_1} \pmod{n} \wedge N_V \equiv \zeta^{m_0 + m_1 2^\ell} \wedge m_0, m_1 \in \{0,1\}^{\ell_m + \ell_\varnothing + \ell_\mathcal{H} + 2} \wedge (e - 2^{\ell_e}) \in \{0,1\}^{\ell'_e + \ell_\varnothing + \ell_\mathcal{H} + 1}\} \ ,$$

where $A' := AS^r$ for a randomly chosen r.

4 Securing Credentials with a TPM

The fact that the operation of the "signature receiver" in the protocols to obtain a signature and to prove knowledge of a signature are split between the TPM and the platform and the fact that the platform's operations in the protocol are implemented in software, allow us to "abuse" the DAA protocol and to extend it for our purposes by modifying the software parts of the protocol.

In this section, we first describe how to extend the DAA-protocol such that a DAA-certificate can include attributes while keeping the TPM-part of the protocol unaltered. That is, the modified protocol will allow one to obtain a DAA-certificate containing, besides the secret messages m_0 and m_1 (that will only be known by the TPM), messages $m_2, \ldots m_{L-1}$ that are potentially only known to the platform. The issuer will only learn a commitment to these messages unless the (user of the) platform decides to reveal some of them. Similarly, the modified protocol will allow the platform to prove that the DAA-certificate contains further messages $m_2, \ldots m_{L-1}$ where the individual messages can either be revealed to the verifier, be given in a commitment to him, or be hidden from him. Both protocols will require the cooperation of the TPM to be successfully executed. In particular, the protocols maintain the fact that one cannot prove possession of a DAA-certificate without involving the TPM to which the certificate has been issued. Thus, the user is protected from malicious code such as viruses aiming to steal her certificates and a credential issuer is protected from malicious users because as the certificate can not longer be shared by different users (or at least only by the users of the same platform).

However, as a TPM itself is tied to a particular device, this is not very convenient for users who use a number of devices (e.g., laptop, PDA, mobile phone, home-computer,

office-computer, etc) and want to use their credentials with all their devices as this would mean retrieving a particular credential for each of these platforms. Indeed, users would like to have their credentials portable as it would for instance be the case if the credentials were tied to token such as a smartcard or a USB-token. To overcome this drawback, we present a scheme that allows a user to use a credential issued to her with all *her* devices.

Finally, we will show how our basic schemes can be extended to full-fledged credential system and to traditional (non-privacy protecting) certificates and credentials.

Let the public key of the signer consist of an ℓ_n-bit RSA modulus n and elements $R_0, \ldots, R_{L-1}, S, Z \in \mathrm{QR}_n$ for some L and the secret key consists of Also let g_i be random group elements of QR_n such that $g_i \in \langle g_0 \rangle$ and such that $\log_{g_0} g_i$ is unknown.

4.1 Extending DAA-Credentials to Include Attributes

A DAA-certificate entails values (A, v, e) such that $Z \equiv A^e S^v R_0^{m_0} R_1^{m_1} \pmod{n}$, where v, m_0, and m_0 are only known to the TPM [21,5]. We now want to extend the DAA protocols to retrieved a signature and to prove knowledge of a signature such that (A, v, e) will be a signature on additional messages such that the TPM's part of the DAA protocols remain unchanged.

Obtaining a Credential That is Tied to a TPM The DAA-join protocol can be extended as follows to have the issuer in addition sign the messages $m_2, \ldots, m_{L-1} \in \pm\{0,1\}^{\ell_m}$, where the issuer is given the m_i's only as a commitment (should an application require the issuer to learn some of the m_i, they can just be reveal and then it can be proven that one has revealed in indeed to correct ones).

It is often useful that the issuer or the verifier do not learn all of the messages $m_2, \ldots, m_{L-1}$. In the following we assume that the issuer learns the messages $m_{L'}, \ldots, m_{L-1}$ and is given a commitment C to the other messages so that, if needed, the platform could prove properties about them using C independently of our protocol.

In the DAA protocol, the TPM sends the issuer (attester) a values $U = S^{v'} R_0^{m_0} R_1^{m_1}$ and proves to the issuer knowledge of v', m_0, m_1, and then the issuer will use U to produce the signature. Now, the idea to extend the DAA-certificate is to send the issuer a second values U' that contains the messages $m_2 \ldots, m_{L'-1}$ and then have the issuer to use that value as well in the signature generation. Let us describe this in more details. Let $C = \mathsf{Com}(m_2, \ldots, m_{L'-1}; r)$ for some suitably chosen r.

1. While the TPM computes U (and some other value N_I serving as pseudonym with the issuer), the platform chooses a random $\hat{v} \in_R \{0,1\}^{\ell_n + \ell_\varnothing}$, computes $U' := S^{\hat{v}} \prod_{i=2}^{L'-1} R_i^{m_i} \bmod n$ and sends to the signer the value U' together with the U produced by the TPM. Furthermore, the value U is authenticated by the TPM using its endorsement key EK. We refer to [5,21] for the details on how this is achieved.
2. Next, the issuer needs to get convinced that U and U' are constructed correctly and that U' corresponds to C. Using the TPM's prover-messages of the original

protocol, the platform computes the proof

$$SPK\{(m_0,\ldots,m_{L'-1},v',\tilde{v},r):$$
$$U \equiv \pm S^{v'} R_0^{m_0} R_1^{m_1} \pmod{n} \ \wedge\ N_I := \zeta_I^{m_0+m_1 2^{\ell_m}} \pmod{\Gamma} \ \wedge$$
$$U' \equiv \pm S^{\tilde{v}} \prod_{i=2}^{L'-1} R_i^{m_i} \pmod{n} \ \wedge\ C \equiv g_0^r \prod_{i=2}^{L'-1} g_{i-1}^{m_i} \ \wedge$$
$$m_0,\ldots,m_{L'-1} \in \{0,1\}^{\ell_m+\ell_\varnothing+\ell_\mathcal{H}+2} \ \wedge\ v',\tilde{v} \in \{0,1\}^{\ell_n+\ell_\varnothing+\ell_\mathcal{H}+2}\} \ .$$

This protocol convinces the issuer that the TPM has computed U and N_I correctly, the platform did build U' correctly and that U' contains the messages committed to in C.

3. Now the signature is generated by the signer as in the DAA protocol with the exception that A is computed as

$$A := \left(\frac{Z}{UU'S^{v''}\prod_{i=L'}^{L-1} R_i^{m_i}}\right)^{1/e} \bmod n \ ,$$

where e is a randomly chosen ℓ_e-bit prime e and v a randomly chosen $(\ell_n+\ell_\varnothing)$-bit integer. The signer sends the values (A,e,v'') to the platform.

4. The signer's proof that A is correctly formed is done in the same way as in the DAA protocol with the adaption to the two values U and U', i.e., the proof protocol becomes

$$SPK\{(d): \quad A \equiv \pm\left(\frac{Z}{UU'S^{v''}\prod_{i=L'}^{L-1} R_i^{m_i}}\right)^d \pmod{n}\} \ .$$

5. The platform forwards v'' to the TPM which sets $v := v'' + v'$, and stores (m_0,m_1,v). The platform stores A, e, $\hat{v}$ and $m_2,\ldots,m_L$.

Proving Possession of a Credential that is tied to a TPM To show possession of an extended DAA-certificate, i.e., a signature by the DAA-issuer that now is not only a signature on the TPM's secret keys m_0 and m_1 but also on the messages $m_2,\ldots,m_{L-1} \in \pm\{0,1\}^{\ell_m}$, we have to modify the DAA-sign protocol as follows. Let I_r, I_c, and I_h be three sets. We assume that the verifier is given the messages m_i with $i \in I_r$ and a commitment C to messages m_i with $i \in I_c$ but that the messages m_i with $i \in I_h$ are hidden from the verifier. As before, we need to build the protocol such that the TPM's part of it are as in the original DAA-sign protocol. Thus, the platform now needs to transform the TPM's proof-protocol messages into ones of a proof that convinces the verifier that the messages $m_2,\ldots,m_{L-1}$ are contained as attributes in the DAA-certificate. That is, the platform receives from the TPM prover-messages for the the original protocol and needs to transform them into the prover-messages of the

protocol

$$PK\{(e, \tilde{v}, \{m_i | i \in I_c \cup I_h\}, r') :$$
$$Z \prod_{i \in I_r} R_i^{-m_i} \equiv \pm A'^e S^{\tilde{v}} \prod_{i \in I_c \cup I_h} R_i^{m_i} \pmod{n} \wedge C \equiv {g_0}^{r'} \prod_{i \in I_c} {g_i}^{m_i} \wedge$$
$$m_i \in \{0,1\}^{\ell_m + \ell_\varnothing + \ell_\mathcal{H} + 2} \, (i \in I_c \cup I_h) \wedge (e - 2^{\ell_e}) \in \{0,1\}^{\ell'_e + \ell_\varnothing + \ell_\mathcal{H} + 1}\} \ ,$$

where the value A' is derived from A as $A' = AS^r$ for a suitable chosen r (such that A' will be a random element) and I_c and I_h are set of indices of the messages to which the verifier receives the commitments C and which remain hidden from the verifier, respectively.

4.2 Using Credentials with Several Devices

We now describe how to realize our limited transferable anonymous certificate scheme using TPM V1.2 chips. The idea is that a user registers (the TPM's of) all her devices with some authority, that is, the registration authority issues all the user's TPMs an extended DAA-credential that contains as one of the extra messages signed (e.g., m_2) an identifier that is unique to that user, e.g., her identity. We call such a credential an *device credential*. We will call the "real" credentials, i.e., those that will allows the user to access some resource or service, *application credential*. These are then issued to the user as described in [7] except that we always set one of the message includes in the credential, e.g., m_0, to the user's identity. Note that these credentials are not directly tied to the TPM. In case the user's identity should not get known to the issuer of application credentials, the user can provide him with a commitment C to m_0 (her identifier), use the protocol presented in §4.1 to show that she has obtained a device-credential that contains as message m_2 the same value that is committed to by C, and then the two can run the protocol in [7] to obtain a signature on a committed message. Now, whenever a user wants to use an application credential, we require the user to provide a commitment C to her identifier and then to prove that she has obtained a device credential that contains as second message the value committed to by C and that she has further obtained an application credential that contains as 0-th message the value committed to by C. This she can do using the protocols provided in §4.1, respectively.

It is not hard to see that this solution provides what we want: Application credentials cannot be used without having access to a device that got registered w.r.t. the identity that is contained in the application credential. Thus, if the device credential issuer makes sure that only a limited number of devices get registered under the same identity, application credentials can only be used with a limited number of devices. In fact, it is not important that the string encoded into the device credentials and the application credentials is the user's real identifier. The device-credential issuer only needs to make sure that only a limited amount of devices get registered per user/identifier. Depending on the scenarios, one might have different limits here. Thus, the registration of a device could even be pseudonymous. That is, the first time a user registers a device, she would establish a pseudonym with the device credential issuer who would then choose an identifier for the user to be encoded into the device credential.

Depending on the scenario and the limits on how many devices a user can register, one might nevertheless want to ensure that users with few devices register their devices w.r.t. the same identifier. In the non-pseudonymous case, one could achieve this by requiring that the user fully identifies herself. In the pseudonymous case, one would require that the user prove ownership of her pseudonym (of course only after she has registered the first device) and apply one of the known methods to prevent the user from sharing her pseudonym (cf. [6]). One such method is for instance to bind the secret key required to prove ownership of the pseudonym to, e.g., the user's bank account. Thus, if the user would share her pseudonym, she would also share her bank account. The details of all of this, however, are out of the scope of this paper and we assume for simplicity in the following description of our solution that each user has a unique identifier.

Setup. For simplicity we consider only a single issuer of device credentials and a single issuer of application credentials. However, the scheme is easy extendable to several registration authorities. Let $(n, R_0, \ldots, R_{L-1}, S, Z)$ be the former's public key for the CL-signature scheme and $(\hat{n}, \hat{R}_0, \ldots, \hat{R}_{L-1}, \hat{S}, \hat{Z})$ be the one of the latter. That is, the key generation algorithms KeyGenDevCredI and KeyGenAppCredA are the key generation algorithms of the CL-signature scheme. We assume each user is somehow assigned an identity ID_U.

Registering a Device. The procedure GetAppCred to register a device is as follows. The user's device runs the protocol to obtain a signature that is tied to a TPM as described in §4.1 with the parameters $L' = 2$ and $L = 3$ and $m_2 = ID_U$. Thus, the user's device will obtain a device credential, i.e., values (A, e, v) such that

$$Z \equiv A^e R_0^{m_0} R_1^{m_1} R_2^{ID_U} S^v \pmod{n}$$

holds, where the values m_0, m_1, and v are known only to the TPM of the device the user just has registered. Thus, the user can only prove ownership of this device credential when involving the TPM in this proof.

Obtaining a Credential That is Tied to All of the User's TPMs. We now describe the procedure GetAppCred. To obtain an application credential, the user computes a commitment C to her identity, sends C to the issuer, and then runs with the issuer the protocol to obtain a signature on a committed message as described in [7]. Depending on the applications, the application credential might contain other messages (attributes). Thus, we assume that as a result of this protocol the user obtained a CL-signature $(\hat{A}, \hat{e}, \hat{v})$ (application credential) on the messages $ID_U, \hat{m}_1, \ldots, \hat{m}_{\hat{L}-1}$.

We note that there is per se no need for the application issuer to verify that the user has obtained a device credential nor that she has provided the right identity in C, because later on the user won't be able to use the obtained application credential if she has not obtained a device credential containing the identifier contained in C.

Also note that in practise, a user usually will not obtain an application credential without fulfilling some prerequisite. This might include the need to show the issuer some other (application) credential. In this case it might be important that the credential

issued and the one that the user proves possession of all encode the same identifier ID_U and thus that the issuer verifies this.

Anonymously Using a Secured Credential. We now describe the procedure UseApp–Cred. Let $(\hat{A}, \hat{e}, \hat{v})$ be an application credential, i.e., a signature on the messages ID_U, $\hat{m}_1, \ldots, \hat{m}_{\hat{L}-1}$, that the user wants to prove possession of to a service or resource provider (called verifier in the following) using a device she registered, i.e., for which she got the device credential (A, e) (for which that device's TPM keeps secret the corresponding m_0, m_1, and v values).

For simplicity, we assume that all the attributes (messages) $\hat{m}_1, \ldots \hat{m}_{\hat{L}-1}$ contained the application credential are hidden from the verifier. However, it is easy to modify the below protocol as to reveal (some of) the message or to provide the verifier only commitments to (some of) them.

1. The platform computes a commitment C to ID_U: it selects a random $r \in_R \{0,1\}^{\ell_n+\ell_\varnothing}$, computes $C := g_1^{ID_U} g_0^r$, and sends C to the verifier.
2. The platform proves possession of an application credential whose 0-th message is committed to in C. The other attributes (i.e., messages) $\hat{m}_1, \ldots \hat{m}_{\hat{L}-1}$ contained in the application credential can be handled as revealed, committed-to, or hidden messages, depending on the application, we here assume that they are all hidden. That is, the platform
   (a) chooses $\hat{r} \in_R \{0,1\}^{\ell_n+\ell_\varnothing}$, computes $\hat{A}' := \hat{A}\hat{S}^{\hat{r}}$, and sends $\hat{A}'$ to the application-credential issuer.
   (b) and then runs the following

$$SPK\{(\hat{e}, \hat{v}, ID_U, \hat{m}_1, \ldots \hat{m}_{\hat{L}-1}, r) :$$
$$\hat{Z} \equiv \pm \hat{A}'^{\hat{e}} \hat{S}^{\hat{v}} \hat{R}_0^{ID_U} \prod_{i=1}^{\hat{L}-1} \hat{R}_i^{\hat{m}_i} \ \wedge \ C \equiv \pm g_1^{ID_U} g_0^r \ \wedge$$
$$\hat{m}_1, \ldots, \hat{m}_{\hat{L}-1}, ID_U \in \{0,1\}^{\ell_m+\ell_\varnothing+\ell_\mathcal{H}+2} \wedge (e - 2^{\ell_e}) \in \{0,1\}^{\ell'_e+\ell_\varnothing+\ell_\mathcal{H}+1}\}$$

3. To prove that it has also obtained a device-credential, the platform proceeds as follows:
   (a) The platform chooses $r \in_R \{0,1\}^{\ell_n+\ell_\varnothing}$, computes $A' := AS^r$, and sends A' to the application-credential issuer.
   (b) The platform as prover (with the help of its TPM) executes the protocol

$$SPK\{(e, \tilde{v}, m_0, m_1, ID_U, r) :$$
$$Z \equiv \pm A'^e S^{\tilde{v}} R_0^{m_0} R_1^{m_1} R_2^{ID_U} \pmod{n} \ \wedge \ C \equiv \pm g_1^{ID_U} g_0^r \bmod n \ \wedge$$
$$m_0, m_1, ID_U \in \{0,1\}^{\ell_m+\ell_\varnothing+\ell_\mathcal{H}+2} \ \wedge \ (e - 2^{\ell_e}) \in \{0,1\}^{\ell'_e+\ell_\varnothing+\ell_\mathcal{H}+1}\}$$

   with the application-credential issuer as verifier. To be able to prove the first term, the platform needs to transform the corresponding prover's messages obtained from the TPM via the DAA-protocol in the way described in §4.1.

5 Conclusion

We have provided means to bind traditional as well as privacy-protecting certificates and credentials to TPM chips such that they cannot be used without access to the particular TPM chip they were issued. We then extended our scheme so that a user can use her certificates and credential with all of *her* devices but not with any other device. Thus, we provide the users protection of their certificates and credentials from theft by, e.g., malware or phishing attacks. We also provide protection for the service providers from malicious users sharing their credentials. We believe that our solutions overcome two of main barriers of electronic commerce: the insecurity of the currently used authentication schemes and their lack of privacy.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, vol. 1880 of *LNCS*, pp. 255–270. Springer Verlag, 2000.
2. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, vol. 1087 of *LNCS*, pp. 139–155.
3. S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM CCS*, pp. 244–250, 1993.
4. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 318–333. Springer Verlag, 1997.
5. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM CSS*, pp. 225–234. acm press, 2004.
6. J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *EUROCRYPT 2001*, vol. 2045 of *LNCS*, pp. 93–118. Springer Verlag, 2001.
7. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, vol. 2576 of *LNCS*, pp. 268–289. Springer Verlag, 2003.
8. J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In *EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 107–122.
9. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424. Springer Verlag, 1997.
10. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 561–575. Springer Verlag, 1998.
11. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.
12. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, vol. 740 of *LNCS*, pp. 89–105. Springer-Verlag, 1993.
13. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, vol. 839 of *LNCS*, pp. 174–187. Springer Verlag, 1994.
14. I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, vol. 2501 of *LNCS*. Springer, 2002.
15. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194.

16. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO '97*, vol. 1294 of *LNCS*, pp. 16–30.
17. R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT 2003*, vol. 2656 of *LNCS*, pp. 524–543. Springer Verlag,
18. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, vol. 576 of *LNCS*, pp. 129–140. Springer Verlag, 1992.
19. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.
20. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
21. Trusted Computing Group. TCG TPM specification 1.2. Available at www.trustedcomputinggroup.org, 2003.