

Better Privacy for Trusted Computing Platforms

(Extended Abstract)

Jan Camenisch

IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland

Abstract. The trusted computing group (TCG) specified two protocols that allow a trusted hardware device to remotely convince a communication partner that it is indeed a trusted hardware device. In turn, This enables two communication partners to establish that the other end is a secure computing platform and hence it is safe exchange data. Both these remote identification protocols provide some degree of privacy to users of the platforms. That is, the communication partners can only establish that the other end uses some trusted hardware device but not which particular one. The first protocol achieves this property by involving trusted third party called Privacy CA in each transaction. This party must be fully trusted by all other parties. In practice, however, this is a strong requirement that is hard to fulfill. Therefore, TCG proposed a second protocol called direct anonymous attestation that overcomes this drawback using techniques known from group signature schemes. However, it offers less privacy than the one involving the Privacy CA. The reason for this is that the protocol needs to allow the verifier to detect rogue hardware devices while before this detection was done by the Privacy CA. In this paper we show how to extend the direct anonymous attestation protocols such that it offers the same degree of privacy as the first solution but still allows the verifier to detect rogue devices.

1 Introduction

Consider a trusted hardware module, called the trusted platform module (TPM) in the following, that is integrated into a platform such as a laptop or a mobile phone. Assume that the user of such a platform communicates with a verifier who wants to be assured that the user indeed uses a platform containing such a trusted hardware module, i.e., the verifier wants the TPM to authenticate itself. However, the user wants her privacy protected and therefore requires that the verifier only learns that she uses a TPM but not which particular one – otherwise all her transactions would become linkable to each other. This problem arose in the context of the Trusted Computing Group (TCG). TCG is an industry standardization body that aims to develop and promote an open industry standard for trusted computing hardware and software building blocks to enable more secure data storage, online business practices, and online commerce transactions while protecting privacy and individual rights (cf. [19]). TCG is the successor organization of the Trusted Computing Platform Alliance (TCPA).

In principle, the problem just described could be solved using any standard public key authentication scheme (or signature scheme): One would generate a secret/public key pair, and then embed the secret key into each TPM. The verifier and the TPM would then run the authentication protocol. Because all TPMs use the same key, they are indistinguishable. However, this approach would never work in practice: as soon as one hardware module (TPM) gets compromised and the secret key extracted and published, verifiers can no longer distinguish between real TPMs and fake ones. Therefore, detection of rogue TPMs needs to be a further requirement.

The solution first developed by TCG uses a trusted third party, the so-called privacy certification authority (Privacy CA), and works as follows [17]. Each TPM generates an RSA key pair called an Endorsement Key (EK). The Privacy CA is assumed to know the Endorsement Keys of all (valid) TPMs. Now, when a TPM needs to authenticate itself to a verifier, it generates a second RSA key pair called an Attestation Identity Key (AIK), sends the AIK public key to the Privacy CA, and authenticates this public key w.r.t. the EK. The Privacy CA will check whether it finds the EK in its list and, if so, issues a certificate on the TPM's AIK. The TPM can then forward this certificate to the verifier and authenticate itself w.r.t. this AIK. In this solution, there are two possibilities to detect a rogue TPM: 1) If the EK secret key was extracted from a TPM, distributed, and then detected and announced as a rogue secret key, the Privacy CA can compute the corresponding public key and remove it from its list of valid Endorsement Keys. 2) If the Privacy CA gets many requests that are authorized using the same Endorsement Key, it might want to reject these requests. The exact threshold on requests that are allowed before a TPM is tagged rogue depends of course on the actual environment and applications, and will in practise probably be determined by some risk-management policy.

This solutions has the obvious drawback that the Privacy CA needs to be involved in every transaction and thus highly available on the one hand but still as secure as an ordinary certification authority that normally operates off-line on the other hand. Moreover, if the Privacy CA and the verifier collude, or the Privacy CA's transaction records are revealed to the verifier by some other means, the verifier will still be able to uniquely identify a TPM. Similarly, the verifier needs to trust the Privacy CA not to issue certificate to just anybody, but only the genuine TPM. These trust relationships are in fact the more severe drawback as it is not clear who would provide such a Privacy CA service.

To overcome these problems, TCG included in the TPM v1.2 specification [18, 4] an alternative proposal called direct anonymous attestation. This new protocols draws on techniques that have been developed for group signatures [11, 8, 1], identity escrow [14], and credential systems [10, 5]. In fact, direct anonymous attestation 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). More precisely, it also employs a suitable signature scheme to issue certificates on a membership public key generated by a TPM. Then, to authenticate as a group member, or valid TPM, a

TPM proves that it possesses a certificate on a public key for which it also knows the secret key. To allow a verifier to detect rogue TPMs, the TPM is further required to reveal and prove correct of a value $N_V = \zeta^f$, where f is its secret key and ζ is a generator of an algebraic group where computing discrete logarithms is infeasible. As in the Privacy-CA solution, there are two possibilities for the verifier to detect a rogue TPM: 1) By comparing N_V with $\zeta^{\tilde{f}}$ for all \tilde{f} 's that are known to stem from rogue TPMs. 2) By detecting whether he has seen the same N_V too many times. Of course, the second method only works if the verifier always uses the same ζ , or at least changes it only rarely. However, ζ should not be a fixed system parameter as otherwise the user gains almost no privacy – in the TCG specification it is proposed that either ζ be somehow derived from the verifier's name, e.g., using an appropriate hash function. However, this has the drawback that it allows the verifier to link transactions by perfectly honest users and hence drastically reduces the privacy offered by the direct anonymous attestation. In fact, in this respect, direct anonymous attestation is inferior to TCG's initial Privacy CA solution. There, a user can get a new certificate from the Privacy CA for each transactions and hence a verifier can not link the different transactions (unless it colludes with the Privacy CA).

In this paper we show how direct anonymous attestation can be modified such that it offers the same degree of privacy as the Privacy CA solution but it is still possible to detect rogue TPM. That is, it allows the verifier to check the frequency a TPM accesses some service but prevents it from doing profiling. The idea is to separate the frequency check from the request of the service. This can be done as follows: the verifier checks that the TPM's authentication as a genuine TPM using the direct anonymous attestation protocol, performs the frequency check and, if the check succeeds, issues an anonymous one-time certificate to the TPM/host instead of providing the service. In the following we will often call this anonymous one-time certificate a *frequency certificate*. Later, the TPM/host can then access the service using the anonymous one-time certificate. Of course, this frequency certificate must be such that

- the issuing of the certificate and its use cannot be linked,
- it can only be used a single time and with a single verifier, and
- it is tied to the particular TPM/host, i.e., cannot be transferred.

The first property can be achieved either by using so-called blind signatures [9] or by using an the approach similar to the one used for direct anonymous attestation and group signatures [8, 4]. The second property can be obtained by including the verifier's name and something like a serial number in the certificate. Finally, the third property is obtained by ensuring that the DAA attestation and the frequency certificate share a common value that is unique for each TPM. This value must of course be hidden from the verifier and the Privacy CA, but the TPM/host needs to prove to the verifier that the attestation certificate and the frequency certificate share such a values. Thus, the TPM/host need to convince the verifier not only that they have obtained the frequency certificate from the Privacy CA but also that they got attestation and that these two credentials

share a common value. Of course, here the TPM/host must use a random base ζ when showing that they have obtained the attestation.

In this paper we will show how to implement all of this, using the same approach to issue and using frequency certificate that is used for anonymous attestation. For simplicity we assume that the frequency test are not performed by the verifier itself but by a third party which we will call Privacy CA. However, we stress that in the remainder of this document, the verifier can be the same entity as the Privacy CA without that any security and privacy property is compromised. We also note that the protocols we propose are compatible with the TCG TPM v1.2 specifications [18, 4]. That is, while we require some modification to the direct anonymous attestation protocol, the parts of it that are executed by the TPM (i.e., realized in hardware) need not to be modified!

Our protocols as well as direct anonymous attestation [18, 4] are based on prior work on group signatures and credential systems [1, 5, 6] that relies on the strong RSA assumption. While one could also base our protocol on the recently proposed groups signature scheme by Boneh, Boyen, Shacham [3] and Camenisch and Lysyanskaya [7] that use bi-linear maps, this seems not preferable as this would introduce additional computational assumptions to the ones already employed by direct anonymous attestation.

2 Informal Model

This section provides an informal model of what our protocols is supposed to achieve. A formal model is beyond the scope of this extended abstract. However, it is not too hard to modify and adapt the model provided in [4] to our purposes.

The parties in our model are several trusted platform modules (TPMs), several hosts, an attester, several Privacy CAs, and several verifiers. Each host has exclusive access to a TPM. As a TPM is embedded into a host, all communication to and from a TPM is routed via its host. We call the pair host and TPM also platform.

The systems consists of the following procedures.

Key Generation and Setup. There are key generation algorithms for the attester, the Privacy CA and the TPM. We assume that each of these parties has made its public key authentically available.

Attestation Protocol. This is a protocol involving the attester, a TPM and the corresponding host. The attester's input to the protocol is its secret and public keys as well as a list of public keys of valid TPMs, the inputs to the TPM consists of its secret and public keys and the public key of the attester, and the input to the host are the public keys of the attester and the TPM.

If the protocol terminates successfully, the TPM and host will have obtained a attestation certificate from the attester.

Frequency-Certification protocol. This is a protocol involving the a Privacy CA, a TPM, and the corresponding host. The Privacy CA's input to the protocol is its secret and public keys as well as the public key of the attester, the

inputs to the TPM consists of its secret and public keys, the public key of the attester and an attestation certificate, and the input to the host are the public keys of the attester, the Privacy CA, and the TPM and an attestation certificate. Furthermore, the Privacy CA's input also contains a policy stating how frequently a TPM is allowed to obtain a *frequency certificate* from it. The inputs to the TPM consists of its secret and public keys, the public key of the attester, and an attestation certificate. and the input to the host are the public keys of the attester, the Privacy CA, and the TPM and an attestation certificate.

If the protocol terminates successfully, the TPM and host will have obtained a frequency certificate from the Privacy CA while the Privacy CA will have obtained a pseudonym of the TPM.

Verify. This is a protocol involving a verifier, a TPM, and the corresponding host. The input of the verifier consists of the public keys of the attester and a Privacy CA, the inputs to the TPM consists of its secret and public keys, the public key of the attester, an attestation certificate, and a frequency certificate. and the input to the host are the public keys of the attester, the Privacy CA, and the TPM, an attestation certificate, and a frequency certificate.

The security requirements are as follows.

Unforgeability. A TPM-host pair should only obtain a frequency certificate from a Privacy CA if it has obtained an attestation certificate from the attester and if it has not asked the same Privacy CA for too many frequency certificates within a given time. This requires that a TPM-host pair can have only a single pseudonym with a given Privacy CA. Furthermore, a frequency certificate should be usable only once and with a single verifier.

Anonymity. All transactions by the TPM-host pair should be unlinkable, except transactions with the same Privacy CA.

3 Preliminaries

Let $\{0, 1\}^\ell$ denote the set of all binary strings of length ℓ . We often switch between integers and their representation as binary strings, e.g., we write $\{0, 1\}^\ell$ for the set $[0, 2^\ell - 1]$ of integers. Moreover, we often use $\pm\{0, 1\}^\ell$ to denote the set $[-2^\ell + 1, 2^\ell - 1]$.

We need some notation to select the high and low order bits of an integer. Let $\text{LSB}_u(x) := x - 2^u \lfloor \frac{x}{2^u} \rfloor$ and $\text{CAR}_u(x) := \lfloor \frac{x}{2^u} \rfloor$. Let $(x_k \dots x_0)_b$ denote the binary representation of $x = \sum_{i=0}^k 2^i x_i$, e.g., $(1001)_b$ is the binary representation of the integer 9. Also note that $x = \text{LSB}_u(x) + 2^u \text{CAR}_u(x)$.

In our scheme we will use various protocols to prove knowledge of and relations among discrete logarithms. To describe these protocols, we use notation introduced by Camenisch and Stadler [8] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance, $PK\{(\alpha, \beta, \gamma) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma \wedge (u \leq \alpha \leq v)\}$

denotes a “zero-knowledge Proof of Knowledge of integers α , β , and γ such that $y = g^\alpha h^\beta$ and $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma$ 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 $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote the quantities the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, a proof protocol can be described by just pointing out its aim while hiding all details.

In the random oracle model, such protocols can be turned into signature schemes using the Fiat-Shamir heuristic [12, 16]. We use the notation $SPK\{(\alpha) : y = g^\alpha\}(m)$ to denote a signature obtained in this way and call it proof signature.

The Camenisch-Lysyanskaya Signature Scheme. The direct anonymous attestation scheme as well as our new scheme are both based on the Camenisch-Lysyanskaya (CL) signature scheme [6]. Unlike most signature schemes, this one is particularly suited for our purposes as it allows for efficient protocols to prove knowledge of a signature and to retrieve signatures on secret messages efficiently using discrete logarithm based proofs of knowledge [6]. As we will use somewhat different (and also optimized) protocols for these tasks than those provided in [6], we recall the signature scheme here and give an overview of discrete logarithm based proofs of knowledge in the following subsection.

Key generation. On input 1^k , choose a special RSA modulus $n = pq$, $p = 2p' + 1$, $q = 2q' + 1$. Choose, uniformly at random, $R_0, \dots, R_{L-1}, S, Z \in \text{QR}_n$. Output the public key $(n, R_0, \dots, R_{L-1}, S, Z)$ and the secret key p . Let ℓ_n be the length of n .

Signing algorithm. Let ℓ_m be a parameter. On input (m_0, \dots, m_{L-1}) with $m_i \in \pm\{0, 1\}^{\ell_m}$, choose a random prime number e of length $\ell_e > \ell_m + 2$, and a random number v of length $\ell_v = \ell_n + \ell_m + \ell_r$, where ℓ_r is a security parameter. Compute the value A such that $Z \equiv R_0^{m_0} \dots R_{L-1}^{m_{L-1}} S^v A^e \pmod{n}$. The signature on the message (m_0, \dots, m_{L-1}) consists of (e, A, v) .

Verification algorithm. To verify that the tuple (e, A, v) is a signature on message (m_0, \dots, m_{L-1}) , check that $Z \equiv A^e R_0^{m_0} \dots R_{L-1}^{m_{L-1}} S^v \pmod{n}$, and check that $2^{\ell_e} > e > 2^{\ell_e - 1}$.

Theorem 1 ([6]). *The signature scheme is secure against adaptive chosen message attacks [13] under the strong RSA assumption.*

The original scheme considered messages in the interval $[0, 2^{\ell_m} - 1]$. Here, however, we allow messages from $[-2^{\ell_m} + 1, 2^{\ell_m} - 1]$. The only consequence of this is that we need to require that $\ell_e > \ell_m + 2$ holds instead of $\ell_e > \ell_m + 1$.

The Direct Anonymous Attestation Protocol. In this section we review the idea underlying the direct anonymous attestation scheme. The idea is in fact similar to the one of the Camenisch-Lysyanskaya anonymous credential system [5, 6]: A trusted hardware module (TPM) chooses a secret “message” f , obtains a Camenisch-Lysyanskaya (CL) signature (aka attestation) on it from

the attester via a secure two-party protocol, and then can convince a verifier that it got attestation anonymously by a proof of knowledge of a signature on a secret message. To allow the verifier to recognize rogue TPMs, a TPM must also provide a pseudonym N_V and a proof that the pseudonym is formed correctly, i.e., that it is derived from the TPM’s secret f contained in the attestation and a base determined by the verifier. We discuss different ways to handle rogue TPMs later.

Let us now describe this more detailed. Let (n, S, Z, R_0, R_1, R_2) be the attester’s public key for the CL signature scheme. First, for efficiency reasons, the TPM’s secret f is split into two ℓ_f -bit messages to be signed. These (secret) messages are denoted by f_0 and f_1 (instead of m_0 and m_1). This split allows for smaller primes e as their size depends on the size of the messages that get signed, which will make issuing of signature more efficient. The two-party protocol to sign secret messages is as follows (cf. [6]). First, the TPM sends the attester a commitment to the message-pair (f_0, f_1) , i.e., $U := R_0^{f_0} R_1^{f_1} S^{v'}$, where v' is a value chosen randomly by the TPM to “blind” the f_i ’s. Also, the TPM computes $N_I := \zeta_I^{f_0+f_1 2^{\ell_f}} \bmod \Gamma$, where ζ_I is a quantity derived from the attester’s name, and sends U and N_I to the attester. Next, the TPM convinces the attester that U and N_I correctly formed (using a proof of knowledge a representation of U w.r.t. the bases R_0, R_1, S and N_I w.r.t. ζ_I) and that the f_i ’s lie in $\pm\{0, 1\}^{\ell_f+\ell_{\mathcal{H}}+\ell_{\emptyset}+2}$, where $\ell_f, \ell_{\mathcal{H}}$, and ℓ_{\emptyset} are security parameters. This interval is larger than the one from which the f_i ’s actually stem because of the proof of knowledge we apply here. If the attester accepts the proof, it compares N_I with previous such values obtained from this TPM to decide whether it wants to issue a certificate to TPM w.r.t. N_I or not. (The attester might not want to grant too many credentials to the TPM w.r.t. different N_I , but should re-grant a credential to a N_I it has already accepted.) To issue a credential, the attester chooses a random ℓ_v -bit integer v'' and a random ℓ_e -bit prime e , signs the hidden messages by computing $A := \left(\frac{Z}{US^{v''}}\right)^{1/e} \bmod n$, and sends the TPM (A, e, v'') . The attester also proves to the TPM that she computed A correctly. The signature on (f_0, f_1) is then $(A, e, v := v' + v'')$, where v should be kept secret by the TPM (for f_0 and f_1 to remain hidden from the attester), while A and e can be public. This concludes the description of the *DAA-Join* protocol.

A TPM can now prove that it has obtained attestation by proving that it got a CL-signature on some values f_0 and f_1 . This can be done by a zero-knowledge proof of knowledge of values f_0, f_1, A, e , and v such that $A^e R_0^{f_0} R_1^{f_1} S^v \equiv Z \pmod{n}$. Also, the TPM computes $N_V := \zeta^{f_0+f_1 2^{\ell_f}} \bmod \Gamma$ and proves that the exponent here is related to those in the attestation, where $\zeta \in \langle \gamma \rangle$, i.e., the subgroup of \mathbb{Z}_Γ^* of order ρ . This proof of knowledge is turned into a signature scheme using the Fiat-Shamir heuristic. The specification provides to modes for this signature scheme: in one mode, an arbitrary message can be signed, in the other mode an attestation identity key (AIK) generated by the TPM is signed. This ensures that the verifier can tell whether a signed AIK was indeed produced by and is held inside a TPM. The resulting procedure is called *DAA-Sign*.

The base ζ is either chosen randomly by the TPM or is generated from a basename value \mathbf{bsn}_V provided by the verifier. As mentioned in the introductions, the value N_V serves two purposes. The first one is rogue-tagging: If a rogue TPM is found, the values f_0 and f_1 are extracted and put on a blacklist. The verifier can then check N_V against this blacklist by comparing it with $\zeta^{\hat{f}_0 + \hat{f}_1 2^{\ell_f}}$ for all pairs (\hat{f}_0, \hat{f}_1) on the black list. Note that i) the black list can be expected to be short, ii) the exponents $\hat{f}_0 + \hat{f}_1 2^{\ell_f}$ are small (e.g., 200-bits), and iii) batch-verification techniques can be applied [2]; so this check can be done efficiently. Also note that the blacklist need not be certified, e.g., by a certificate revocation agency: whenever $f_0, f_1, A, e,$ and v are discovered, they can be published and everyone can verify whether (A, e, v) is a valid signature on f_0 and f_1 . The second purpose of N_V is its use as a pseudonym, i.e., if ζ is not chosen randomly by the TPM but generated from a basename then, whenever the same basename \mathbf{bsn}_V is used, the TPM will provide the same value for N_V . This allows the verifier to link different transactions made by the same TPM while not identifying it, and to possibly reject a N_V if it appeared too many times. By defining how often a different basename is used (e.g., a different one per verifier and day), one obtains the full spectrum from full identification to pseudonymity to full anonymity. The way the basename is chosen, the frequency it is changed, and the threshold on how many times a particular N_V can appear before a verifier should reject it, is a question that depends on particular scenarios/applications and is outside of the scope of this paper.

The actual anonymous attestation protocols outsources some operations of the TPM to the host in which the TPM is embedded. In particular, these are operation that are related to hiding the TPM's identity but not to the capability of producing a proof-signature of knowledge of an attestation. The rationale behind this is that the host can always reveal a TPM's identity.

4 Better Privacy using a Two Stage Authorization

In this section we describe how we can achieve better privacy if we combine the direct anonymous attestation protocols with the Privacy CA approach. We will first describe the solution on a high level, similarly to the description of the direct anonymous attestation in the previous section. We will then describe the different protocols of our new scheme in more detail.

High-Level Idea. Recall that using the DAA-join protocol, a TPM can get a certificate (A, e, v) from an attester such that $A^e R_0^{f_0} R_1^{f_1} S^v \equiv Z \pmod{n}$ holds, where $f_0, f_1,$ and v are only known to the TPM and (R_0, R_1, S, Z, n) are values of the attester's public key. Using the DAA-sign protocol, a TPM can convince a verifier that it has obtained such a certificate without revealing any other information. That is, it can convince the verifier that it knows values $A, e, f_0, f_1,$ and v such that the above equation holds.

Now for our scheme, we need to modify the DAA-Join protocol such that the attester not only signs the secret keys of the TPM but also some value k_t

the attester chooses such that it is unique for each TPM. That is, the certificate (A, e, v) the TPM/host obtains from an attester will satisfy

$$A^e R_0^{f_0} R_1^{f_1} R_2^{k_t} S^v \equiv Z \pmod{n},$$

and R_2 is an additional value of the attester's public key. From this it also follows that k_t should be an ℓ_f -bit value. Of course, the attester has to send k_t to the TPM. This value will then also be part of all other the certificates issued to the platform. The value of k_t could for instance be derived from the TPM's endorsement key EK or in some other way; there is no cryptographic requirement on how k_t is chosen such as requiring that k_t is a cryptographic hash of EK .

Let us now look at how a frequency certificate is issued. Let (n, S, Z, R_0, R_1, R_2) be the Privacy CA's public key for the CL signature scheme. Now, in order to get an anonymous frequency certificate from the Privacy CA, the TPM/host show the attestation certificate to the Privacy CA using a long-term named base ζ for the computation of N_V and, if N_V passes the Privacy CA's frequency test, they will obtain a frequency certificate (A, e, v) from the Privacy CA such that

$$A^e R_0^{k_0} R_1^{k_1} R_2^{k_t} R_3^{\text{exp-date}} S^v \equiv Z \pmod{n}$$

holds, where k_t is the same values as in the attestation, k_0 and k_1 are values chosen by the host, and exp-date is an encoding of the expiration date of the certificate (the different possible values of exp-date should be small to guarantee privacy). In getting this certificate from the Privacy CA, the TPM will need to be only involved in showing the attestation but not in anything related to the frequency certificate. The k_0 and k_1 should identify the verifier and some random number in a cryptographically secure way, e.g., $k_0 := \text{LSB}_{\ell_f}(H(\text{verifier-name}||\text{ran}))$ and $k_1 := \text{CAR}_{\ell_f}(H(\text{verifier-name}||\text{ran}))$.

Finally, when the host wants to show a verifier that it has obtained a certificate from the Privacy CA, the TPM/host convince the verifier that 1) they have a valid attestation (using a random base ζ in this proof), 2) they have a certificate from the Privacy CA, 3) the attestation and the certificate are linked by a common values (i.e., k_t), and 4) the certificates contains verifier-name , ran , and exp-date . That is, the TPM/host jointly prove knowledge of values $f_0, f_1, A, e,$ and v, k_t, A, e, v such that $A^e R_0^{f_0} R_1^{f_1} R_2^{k_t} S^v \equiv Z \pmod{n}$ and $A^e R_0^{k_0} R_1^{k_1} R_2^{k_t} R_3^{\text{exp-date}} S^v \equiv Z \pmod{n}$ hold (note that the verifier can compute k_0 and k_1 himself).

Security Parameters. Our protocol employ the same security parameters as the direct anonymous attestation protocol [4], i.e., $\ell_n, \ell_f, \ell_e, \ell'_e, \ell_v, \ell_\emptyset, \ell_{\mathcal{H}}, \ell_r, \ell_\Gamma,$ and ℓ_ρ , where ℓ_n (2048) is the size of the RSA modulus, ℓ_f (104) is the size of the f_i 's (information encoded into the certificate), ℓ_e (368) is the size of the e 's (exponents, part of certificate), ℓ'_e (120) is the size of the interval the e 's are chosen from, ℓ_v (2536) is the size of the v 's (random value, part of certificate), ℓ_\emptyset (80) is the security parameter controlling the statistical zero-knowledge property,

$\ell_{\mathcal{H}}$ (160) is the output length of the hash function used for the Fiat-Shamir heuristic, ℓ_r (80) is the security parameter needed for the reduction in the proof of security, ℓ_{Γ} (1632) is the size of the modulus Γ , and ℓ_{ρ} (208) is the size of the order ρ of the sub group of \mathbb{Z}_{Γ}^* that is used for rogue-tagging (the numbers in parentheses are our proposal for these parameters). We require that: $\ell_e > \ell_{\emptyset} + \ell_{\mathcal{H}} + \max\{\ell_f + 4, \ell'_e + 2\}$, $\ell_v > \ell_n + \ell_{\emptyset} + \ell_{\mathcal{H}} + \max\{\ell_f + \ell_r + 3, \ell_{\emptyset} + 2\}$, and $\ell_{\rho} = 2\ell_f$.

The parameters ℓ_{Γ} and ℓ_{ρ} should be chosen such that the discrete logarithm problem in the subgroup of \mathbb{Z}_{Γ}^* of order ρ with Γ and ρ being primes such that $2^{\ell_{\rho}} > \rho > 2^{\ell_{\rho}-1}$ and $2^{\ell_r} > \Gamma > 2^{\ell_r-1}$, has about the same difficulty as factoring ℓ_n -bit RSA moduli (see [15]).

Finally, let \mathcal{H} be a collision resistant hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell_{\mathcal{H}}}$.

Key Generation and Setup. The attester runs the following key-generation algorithm.

1. It chooses a RSA modulus $n = pq$ with $p = 2p' + 1$, $q = 2q' + 1$ such that p, p', q, q' are all primes and n has ℓ_n bits.
2. It chooses a random element $g' \in_R \mathbb{Z}_n$ of order $p'q'$.
3. Next, it chooses random integers $x_0, x_1, x_2, x_z, x_s, x_h, x_g \in [1, p'q']$ and computes $g := g'^{x_g} \bmod n$, $h := g'^{x_h} \bmod n$, $S := h^{x_s} \bmod n$, $Z := h^{x_z} \bmod n$, $R_0 := S^{x_0} \bmod n$, $R_1 := S^{x_1} \bmod n$, $R_2 := S^{x_2} \bmod n$.
4. It produces a non-interactive proof *proof* that R_0, R_1, S, Z, g , and h are computed correctly, i.e., that $g, h \in \langle g' \rangle$, $S, Z \in \langle h \rangle$, and $R_0, R_1, R_2 \in \langle S \rangle$ (see [4] for how this is done).
5. Output the public key $(n, g', g, h, S, Z, R_0, R_1, R_2)$ and the secret key (p, q) .

We denote the resulting public key for the attester by $PK_I := (n, g', g, h, S, Z, R_0, R_1, R_2)$. The Privacy CA runs the same key generation algorithm, except that it also generates a third base R_3 in the same way that R_0, \dots, R_2 are generated. We denote the thus generated public key of the Privacy CA by $PK_P := (n, g', g, h, S, Z, R_0, R_1, R_2, R_3)$.

The attester furthermore makes available a group $\langle \gamma \rangle$ of prime order ρ , i.e., it chooses primes ρ and Γ such that $\Gamma = r\rho + 1$ for some r with $\rho \nmid r$, $2^{\ell_r-1} < \Gamma < 2^{\ell_r}$, and $2^{\ell_{\rho}-1} < \rho < 2^{\ell_{\rho}}$. Choose a random $\gamma' \in_R \mathbb{Z}_{\Gamma}^*$ such that $\gamma'^{(\Gamma-1)/\rho} \not\equiv 1 \pmod{\Gamma}$ and set $\gamma := \gamma'^{(\Gamma-1)/\rho} \bmod \Gamma$.

Finally, each TPM has generated an endorsement key, i.e., an RSA encryption public key, and has made it available to the attester. We refer to the TCG TPM specification [18, 4] for how this is done.

The Attestation Protocol. Let $PK_I := (n, g', g, h, S, Z, R_0, R_1, R_2, \gamma, \Gamma, \rho)$ be the public key of the attester. Let $\zeta_I \equiv (H_{\Gamma}(1 \parallel \mathbf{bsn}_I))^{(\Gamma-1)/\rho} \bmod \Gamma$, where \mathbf{bsn}_I is the attester's long-term base name.

We assume that the TPM somehow authenticates itself towards the attester using its endorsement key before the protocol below is run. We refer to the TCG TPM specification [18, 4] for how this is done.

The protocol's input to the TPM is $(n, R_0, R_1, S, \rho, \Gamma)$ and its input to the host is $(n, g', g, h, S, Z, R_0, R_1, R_2, \gamma, \Gamma, \rho)$.

1. The host computes $\zeta_I := (H_\Gamma(1 \parallel \text{bsn}_I))^{(\Gamma-1)/\rho} \bmod \Gamma$ and sends ζ_I to the TPM.
2. The TPM checks whether $\zeta_I^\rho \equiv 1 \pmod{\Gamma}$.
3. Let $i := \lfloor \frac{\ell_\rho + \ell_\varnothing}{\ell_\kappa} \rfloor$ (i will be 1 for values of the parameters selected in Section 4).

The TPM computes

$$f := H(H(\text{seed} \parallel H(PK'_I)) \parallel \text{cnt} \parallel 0) \parallel \dots \parallel H(H(\text{seed} \parallel H(PK'_I)) \parallel \text{cnt} \parallel i) \pmod{\rho},$$

$f_0 := \text{LSB}_{\ell_f}(f)$, $f_1 := \text{CAR}_{\ell_f}(f)$, $v' \in_R \{0, 1\}^{\ell_n + \ell_\varnothing}$, $U := R_0^{f_0} R_1^{f_1} S^{v'} \bmod n$, and $N_I := \zeta_I^{f_0 + f_1 2^{\ell_f}} \bmod \Gamma$, and sends U and N_I to the attester.

4. The TPM proves to the attester knowledge of f_0 , f_1 , and v' : it executes as prover the proof signature

$$\text{SPK}\{(f_0, f_1, v') : f_0, f_1 \in \{0, 1\}^{\ell_f + \ell_\varnothing + \ell_\kappa + 2} \wedge v' \in \{0, 1\}^{\ell_n + \ell_\varnothing + \ell_\kappa + 2} \wedge U \equiv \pm R_0^{f_0} R_1^{f_1} S^{v'} \pmod{n} \wedge N_I \equiv \zeta_I^{f_0 + f_1 2^{\ell_f}} \pmod{\Gamma}\}$$

with the attester as the verifier.

5. The attester chooses $\hat{v} \in_R \{0, 1\}^{\ell_v - 1}$, a prime $e \in_R [2^{\ell_e - 1}, 2^{\ell_e - 1} + 2^{\ell'_e - 1}]$, and an appropriate and unique k_t , computes $v'' := \hat{v} + 2^{\ell_v - 1}$ and $A := \left(\frac{Z}{UR_2^{k_t} S^{v''}}\right)^{1/e} \bmod n$, and sends (A, e, v'') and k_t to the host.
6. To convince the platform that A was correctly computed, the attester as prover runs the proof signature

$$\text{SPK}\{(d) : A \equiv \pm \left(\frac{Z}{US^{v''} R_2^{k_t}}\right)^d \pmod{n}\}$$

with the host as verifier.

7. The host verifies whether e is a prime and lies in $[2^{\ell_e - 1}, 2^{\ell_e - 1} + 2^{\ell'_e - 1}]$. The host sends v'' to the TPM. The host stores A , e , and k_t .

The only change we have made to this protocol is that A is computed differently. That is, the term $R_2^{k_t}$ does not appear in the original DAA protocol in any of the computations of A . In particular, we did not modify anything that involves the TPM, so our modified protocol is compatible with the TCG TPM v1.2 specification [18, 4].

Obtaining a One-Time Certificate from the Privacy-CA. In this section we describe how the TPM/host convince the Privacy CA that they got an attestation certificate, how they obtain a frequency certificate from it, and how it is ensured that this certificate will also contain the value k_t that is contained on

the attestation certificate. The protocol for this can be seen as a combination of the DAA-join protocol and the DAA-sign and DAA-verify algorithms.

Let $PK_P := (n, g', g, h, S, Z, R_0, R_1, R_2, R_3)$ be the public key of the Privacy CA. Let $n_p \in \{0, 1\}^{\ell_{\mathcal{N}}}$ be a nonce and bsn_P a base name value provided by the Privacy CA.

The input to the protocol for the TPM is $(n, R_0, R_1, R_2, S = S^{2^{\ell_s}}, \Gamma, \rho)$ and (f_0, f_1, v_1, v_2) , and the host's input to the protocol is the certificate (A, e) and, $(n, g, g', h, R_0, R_1, R_2, S, Z, \gamma, \Gamma, \rho)$ and $(n, g', g, h, S, Z, R_0, R_1, R_2, R_3)$. The frequency certification protocol is as follows:

1. (a) The host computes $\zeta := (H_{\Gamma}(1\|\text{bsn}_P))^{(\Gamma-1)/\rho} \pmod{\Gamma}$ and sends ζ to the TPM.
2. (a) The host picks random integers $w, r \in [1, \lfloor \frac{n}{4} \rfloor]$ and computes $T_1 := Ah^w \pmod{n}$ and $T_2 := g^w h^e (g')^r \pmod{n}$.
 (b) The TPM computes $N_P := \zeta^{f_0+f_1 2^{\ell_f}} \pmod{\Gamma}$ and sends N_P to the host.
 (c) Let ran be a sufficiently large random string chosen by the host, e.g., $\text{ran} \in_R \{0, 1\}^{\ell_{\mathcal{N}}}$. Let $k_0 := \text{LSB}_{\ell_f}(H(\text{verifier-name}\|\text{ran}))$ and $k_1 := \text{CAR}_{\ell_f}(H(\text{verifier-name}\|\text{ran}))$. The host picks a random integer $v' \in_R [0, \lfloor \frac{n}{4} \rfloor]$ and computes $U := R_0^{k_0} R_1^{k_1} R_2^{k_t} S^{v'} \pmod{n}$.
3. To prove that U was computed correctly, that the TPM/host has obtained attestation, and that it contains the same value of k_t as does U , the host/TPM jointly perform the following proof signature

$$\begin{aligned}
 & SPK\{(f_0, f_1, v, e, w, r, k_t, k_0, k_1, v') : \\
 & Z \equiv T_1^e R_0^{f_0} R_1^{f_1} R_2^{k_t} S^v h^{-ew} \pmod{n} \wedge T_2 \equiv g^w h^e g'^r \pmod{n} \wedge \\
 & 1 \equiv T_2^{-e} g^{ew} h^{ee} g'^{er} \pmod{n} \wedge N_P \equiv \zeta^{f_0+f_1 2^{\ell_f}} \pmod{\Gamma} \wedge \\
 & U \equiv R_0^{k_0} R_1^{k_1} R_2^{k_t} S^{v'} \pmod{n} \wedge \\
 & f_0, f_1, k_t, k_0, k_1 \in \{0, 1\}^{\ell_f+\ell_{\emptyset}+\ell_{\mathcal{N}}+2} \wedge (e - 2^{\ell_e}) \in \{0, 1\}^{\ell'_e+\ell_{\emptyset}+\ell_{\mathcal{N}}+1} \}
 \end{aligned}$$

as prover with the Privacy CA as verifier. We show in the full version of this paper how this can be done.

4. The Privacy CA checks whether $\zeta \stackrel{?}{\equiv} (H_{\Gamma}(1\|\text{bsn}_P))^{(\Gamma-1)/\rho} \pmod{\Gamma}$.
5. For all (f_0, f_1) on the revocation list, check if $N_P \stackrel{?}{\not\equiv} (\zeta^{f_0+f_1 2^{\ell_f}}) \pmod{\Gamma}$.
6. The Privacy CA checks whether N_P has appeared too often lately, i.e., performs the frequency checks on N_P .
7. Let $\text{exp-date} \in \{0, 1\}^{\ell_f}$ encode the certificate's expiration date. The Privacy CA chooses $\hat{v} \in_R \{0, 1\}^{\ell_v-1}$ and a prime $e \in_R [2^{\ell_e-1}, 2^{\ell_e-1} + 2^{\ell_e-1}]$ and computes $v'' := \hat{v} + 2^{\ell_v-1}$ and $A := \left(\frac{Z}{\text{UR}_3^{\text{exp-date}} S^{v''}}\right)^{1/e} \pmod{n}$, and sends (A, e, v'') and exp-date to the host.
8. To convince the platform that A was correctly computed, the attester as prover runs the proof signature

$$SPK\{(d) : A \equiv \pm \left(\frac{Z}{\text{UR}_3^{\text{exp-date}} S^{v''}}\right)^d \pmod{n}\}$$

with the host as verifier.

9. The host verifies whether e is a prime and lies in $[2^{\ell_e-1}, 2^{\ell_e-1} + 2^{\ell_e'-1}]$. The host computes $v := v'' + v'$, verifies $A^e R_0^{k_0} R_1^{k_1} R_2^{k_2} R_3^{\text{exp-date}} S^v \equiv Z \pmod{n}$ and stores (A, e, v) together with **verifier-name**, **ran**, and **exp-date**.

As mentioned this protocol can be seen as a merge of the DAA-Join and the DAA-Sign protocols. However, as far as the TPM is concerned it is basically the DAA-Sign protocol, all other operations are solely carried out on the host. This can be easily be seen for all steps of the protocol, apart from the step 3 where the TPM and host proof that they got an attestation certificate, that the value U was computed correctly, and that the attestation certificate and U both contained the same (secret) value k_t . How this proof is exactly done is described in the full version of this paper. The idea of it is that the TPM executes all its operations of the DAA-sign protocol, i.e., it basically executes the proof signature $SPK\{(f_0, f_1, v) : (Z/A^e) \equiv R_0^{f_0} R_1^{f_1} S^v \pmod{n} \wedge N_V \equiv \zeta^{f_0+f_1 2^{\ell_f}} \pmod{\Gamma}\}$, which the host then extends to the full proof of step 3. Note that this proof does not involve any terms related to the Privacy CA. Thus, our protocol is compatible with the DAA-Sign protocol as in the TCG TPM v1.2 specification [18, 4].

We finally remark that the expiration date selected by the Privacy CA will later also be shown to the verifier. Thus it, if it is chosen too fine-grained, the Privacy CA and the verifier could link the transactions by the expiration date. Specifying the expiration date in days should be fine for most applications. Moreover, as the certificate can be used only one, there seems to be no strong reason to even have an expiration date at all. If no expiration date is required one can just set **exp-date** = 0, or just drop all terms $R_3^{\text{exp-date}}$.

Using the Certificate from the Privacy CA. In this section we finally provide the procedure for the TPM/host to convince that they got an attestation certificate and a frequency certificate. The procedure can be seen as running two instances of the DAA-sign protocol in parallel, one w.r.t. the attestation certificate and one w.r.t. the frequency certificate. In the latter instance, however, the TPM is not involved at all – here the host plays also the role of the TPM.

Let $n_v \in \{0, 1\}^{\ell_{n_v}}$ be a nonce provided by the verifier. Let b be a byte describing the mode of the protocol, i.e., $b = 0$ means that the message m is an AIK generated by the TPM and $b = 1$ means that the message m was input to the TPM.

The input to the protocol of for the TPM is m , $(n, R_0, R_1, S, S' = S^{2^{\ell_s}}, \Gamma, \rho)$ and (f_0, f_1, v_1, v_2) , and the host's input to the protocol is m , the certificate (A, e) and, $(n, g, g', h, R_0, R_1, S, Z, \gamma, \Gamma, \rho)$;

Let $PK_P := (n, g', g, h, S, Z, R_0, R_1, R_2)$ be the public key of the Privacy CA.

The signing algorithm is as follows.

1. The host sends the verifier **verifier-name**, **ran**, and **exp-date** who checks whether **verifier-name** is correct, whether **ran** is a value that he has not seen so far, and whether **exp-date** is still valid.
2. (a) The host chooses a random $\zeta \in_R \langle \gamma \rangle$ and sends ζ to the TPM.

- (b) The host receives N_V from the TPM (however, we don't use N_V in the following).
3. (a) The host picks random integers $w, r \in [1, \lfloor \frac{n}{4} \rfloor]$ and computes $T_1 := Ah^w \bmod n$ and $T_2 := g^w h^e (g')^r \bmod n$.
- (b) The host picks random integers $w, r \in [1, \lfloor \frac{n}{4} \rfloor]$ and computes $\mathbb{T}_1 := Ah^w \bmod n$ and $\mathbb{T}_2 := \mathbf{g}^w h^e (g')^r \bmod n$.
4. The host sends $T_1, T_2, \mathbb{T}_1, \mathbb{T}_2, \text{exp-date}, \text{verifier-name}$, and ran to the verifier.
5. The verifier checks whether it has seen ran before. If that is the case, the verifier aborts. Otherwise, it computes

$$k_0 := \text{LSB}_{\ell_f}(H(\text{verifier-name} \parallel \text{ran})) \quad (1)$$

$$k_1 := \text{CAR}_{\ell_f}(H(\text{verifier-name} \parallel \text{ran})) \quad (2)$$

6. To prove that the TPM/host has obtained attestation, that it has obtained a one-time certificate from the Privacy CA that contains the values k_0, k_1 and exp-date as well some value (k_t) that is also contained in the attestation, the host/TPM jointly perform the following proof signature

$$\begin{aligned} SPK\{(f_0, f_1, v, e, w, r, k_t, v, e, w, r) : & T_2 \equiv g^w h^e g'^r \pmod{n} \wedge \\ Z \equiv T_1^e R_0^{f_0} R_1^{f_1} R_2^{k_t} S^v h^{-ew} \pmod{n} \wedge 1 \equiv T_2^{-e} g^{ew} h^{ee} g'^{er} \pmod{n} \wedge \\ T_2 \equiv \mathbf{g}^w h^e \mathbf{g}'^r \pmod{n} \wedge \frac{Z}{R_0^{k_0} R_1^{k_1} R_3^{\text{exp-date}}} \equiv \mathbb{T}_1^e R_2^{k_t} S^v h^{-ew} \pmod{n} \wedge \\ 1 \equiv \mathbb{T}_2^{-e} \mathbf{g}^{ew} h^{ee} \mathbf{g}'^{er} \pmod{n} \wedge f_0, f_1, k_t \in \{0, 1\}^{\ell_f + \ell_\varnothing + \ell_{\mathcal{H}} + 2} \wedge \\ (e - 2^{\ell_e}), (e - 2^{\ell_e}) \in \{0, 1\}^{\ell_e + \ell_\varnothing + \ell_{\mathcal{H}} + 1}\}(n_t \parallel n_p \parallel b \parallel m) \end{aligned}$$

as prover with the Privacy CA as verifier, where n_t is a nonce generated by the TPM. We refer to the full version of this paper for the details on how the TPM and the host jointly compute this proof signature.

Apart from the proof signature in the last step of the protocol, it is clear that the TPM performs only the operations as it would in the DAA-sign protocol. In the full version of this paper, we show that also in order to generate this proof signature, it is sufficient if the TPM only executes the steps from the DAA-sign protocol. Thus, also this protocol does not requires any change to the TPM as specified [18].

Security Analysis. Providing a formal proof of security is beyond the scope of this extended abstract; we will only provide an informal discussion why the proposed protocol meet the requirements listed in Section 2. However, given the proof of security of the original direct anonymous attestation protocol [4], deriving a proof of security for the protocols described in this paper is not too hard.

We argue why the requirements from Section 2 are fulfilled.

Unforgeability. First note that due to the properties of the direct anonymous attestation scheme, a TPM/host pair can only generate a DAA-signature if it has obtained an attestation certificate. Furthermore, if the same base ζ is used in the signature, the pseudonym N_P (resp. N_V) contained in the signature will be the same for all DAA-signatures produced by the TPM/host. Thus a TPM-host pair can not have more than one pseudonym with the same Privacy CA and will only obtain a frequency certificate if it is indeed entitled to obtain one. Furthermore, a TPM-host pair can use a frequency certificate only once and only with a single verifier. This is ensured by deriving the k_0 and k_1 that are included into the frequency certificate from `verifier-name` and `ran`. Changing these values can only be done if one could forge frequency certificates or break the hash function. Thus, if the verifier only accepts certificates where k_0 and k_1 are derived from its name and a random string `ran` that it had not seen before, a frequency certificate can only be used with a single verifier and only once.

Anonymity. Apart from the zero-knowledge proofs and the pseudonyms N_P , all values that the TPM-host pair ever reveal are information theoretic commitments, or values that appear only in a single transaction. For instance, a value `ran` will appear only in a transaction with a verifier in the clear; in the corresponding transaction with the Privacy CA it appears only in a commitment. Furthermore, it is not hard to show that all the proofs protocols employed are statistically zero-knowledge. Thus, the pseudonyms N_P are the only information that could be used to link transactions. However, if it is ensured that the Privacy CAs each use a different base ζ , then the only transaction by the same TPM-host pair that can be linked are those made with the same Privacy CA, provided the decisional Diffie Hellman assumption is true. Ensuring that each Privacy CA indeed uses a different base could for instance be done by requiring that it is derived by the Privacy CA's name using a hash function. Thus anonymity is guaranteed even if the attester, all Privacy CAs, and all verifiers collude.

This assumes of course that the TPM-host pairs do choose the time at which to run the protocol to obtain a frequency certificate and the protocol to show it such that these instances become not linkable solely by the time they are executed.

We finally note that because of the anonymity property even holds if the verifiers and the Privacy CAs collude, the role of the Privacy CAs can assume by the verifiers. Thus our solution overcomes the main drawback of the original Privacy CA solution where the TPM/host needs to trust these parties not to collude and hence this separation was needed.

5 Conclusion

We have shown how to modify the direct anonymous attestation protocol as to provide the same level of privacy as the TCG's initial Privacy CA solution while avoiding all the drawbacks of it. In particular, in our solution the TPM-host pair no longer needs to trust the Privacy CA which drastically lowers the requirements

for running such a Privacy CA. In fact, in our solution the Privacy CA and the verifier can be the same party without losing any security properties.

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, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 236–250. Springer Verlag, 1998.
3. D. Boneh, X. Boyen, and H. Shacham. Short group signatures using strong Diffie Hellman. In *CRYPTO 2004*, *LNCS*, Springer Verlag, 2004.
4. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. Technical Report Research Report RZ 3450, IBM Research Division, Mar. 2004.
5. 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.
6. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 2002*, vol. 2576 of *LNCS*, pp. 268–289. Springer Verlag, 2003.
7. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, *LNCS*, Springer Verlag, 2004.
8. 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.
9. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, pp. 199–203. Plenum Press, 1983.
10. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.
11. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 257–265. Springer-Verlag, 1991.
12. 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, 1987.
13. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.
14. J. Kilian and E. Petrank. Identity escrow. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 169–185, Berlin, 1998. Springer Verlag.
15. A. K. Lenstra and E. K. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.
16. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT '96*, vol. 1070 of *LNCS*, pp. 387–398. Springer Verlag, 1996.
17. Trusted Computing Group. Trusted Computing Group (TCG) main specification, Version 1.1b, Available at www.trustedcomputinggroup.org, 2001.
18. Trusted Computing Group. TCG TPM specification 1.2. Available at www.trustedcomputinggroup.org, 2003.
19. Trusted Computing Group website. www.trustedcomputinggroup.org.