Security of the TCG Privacy-CA Solution

Liqun Chen
HP Labs

Bogdan Warinschi
University of Bristol

***Abstract*—The privacy-CA solution (PCAS) is a protocol designed by the Trusted Computing Group (TCG) as an alternative to the Direct Anonymous Attestation scheme for anonymous authentication of Trusted Platform Module (TPM). The protocol has been specified in TPM Specification Version 1.2. In this paper we offer a rigorous security analysis of the protocol. We first design an appropriate security model that captures the level of security offered by PCAS. The model is justified via the expected uses of the protocol in real applications. We then prove, assuming standard security notions for the underlying primitives that the protocol indeed meets the security notion we design. Our analysis sheds some light on the design of the protocol. Finally, we propose a strengthened protocol that meets a stronger notion of security where the adversary is allowed to adaptively corrupt TPMs.**

Keywords: trusted platform module, privacy-CA, certification.

I. Introduction

A Trusted Platform Module (TPM) introduced by the industrial standard body Trusted Computing Group (TCG) [9], is a small tamper-resistant cryptographic chip embedded in a computer platform (e.g. on a PC motherboard). The key function of the TPM is to attest to certain information, e.g. the configuration of the platform, to a verifier, who may be a remote communication partner. For the purpose of the attestation application, each TPM holds a unique long-term asymmetric key pair, namely an Endorsement Key (EK), which is generated and certified by the TPM manufacturer.

Because of the uniqueness and long-term-ness properties of EK, if this key is directly used in an attestation process it may be possible for any third party to identify the communication as belonging to the particular TPM that owns EK. This may be undesirable since it raises serious privacy issues to the owner of the platform. In order to achieve platform attestation while preserving user privacy (i.e. anonymity and unlinkability), TCG designed a privacy-CA solution (PCAS for short), specified in the TCG TPM specification version 1.2 [8]. This specification has been adopted by ISO/IEC as an international standard [7].

PCAS is a special type of public key certification services, which enables a TPM to create an arbitrary number of signature/verification key pairs, called Attestation Identity Keys (AIKs). These keys can then be used to attest to the platform information instead of EK. Since each AIK is only intended for short-term use (could be once only), user privacy can be maintained. Informally, PCAS provides the verifier an authentic but invisible link between EK and AIK. This is achieved by using a privacy-preserving Certificate Authority (privacy-CA or CA for short), who issues a certificate on an AIK provided by a TPM as long as the TPM is in possession of a valid EK. Different from a conventional public key certificate, such as a PKI certificate, the certificate on AIK only provides the statement that this key belongs to a genuine TPM but does not reveal which TPM.

The CA is required to maintain the link between the EK and AIK private, and PCAS does not provide any cryptographic evidence, which enables the CA to convince a third party that a certain pair (EK,AIK) belongs to the same TPM. In order to ensure this, the EK is restricted to be an encryption and decryption key pair rather than a signature and verification key pair. This feature brings another interesting difference between PCAS and conventional certification services. In a typical conventional service, a CA issues a certificate on a new public key after checking a signature of the key under an already known public key and also checking a possession proof of the corresponding secret key. In PCAS, the CA is given neither a cryptographic binding between the EK and AIK nor the standard proof of possession of the secret part of the AIK. However, by using an encryption/decryption algorithm and a challenge-response protocol, the CA authenticates the TPM. A genuine TPM is responsible for using its AIKs correctly.

PCAS is not the only solution provided by TCG for privacy-preserving platform attestation. Another solution is Direct Anonymous Attestation (DAA) [4], which is an anonymous digital signature scheme. A DAA scheme is often seen as a special type of a group signature scheme. It allows TPMs to sign AIKs in a way that hides any individual TPM's identity. In case that a compromised TPM's DAA private key is published, then anonymity can be revoked by any verifier. Optionally two signatures signed by the same TPM can be linked, whilst still maintaining their anonymity. Although PCAS cannot achieve the same anonymous level as a DAA scheme does, PCAS is suitable for many applications, particularly those that naturally involve privacy-CAs. One important merit of PCAS, in contrast with DAA, is that its design idea is simple and easily understandable by non-cryptographers.

Due to their relevance to practice many of the protocols used in the context of TPMs have been thoroughly analyzed using the methods of modern provable security approach. For example, the standard security notions for group signatures were developed in [2], [3], the standard security notions and analysis for DAA schemes were addressed in multiple papers, e.g. [4], [5], [6]. To the best of our knowledge however, the security of the PCAS protocol had never been rigourously analyzed. The main result of this paper remedies this deficiency.

Our first contribution is a formal security model that clar-

ifies the level of security offered by PCAS. We consider an adversary that interacts with multiple TPMs and CAs and has absolute control of the communication. We model in this way the conservative assumption that the hosts of the TPAs may act completely malicious. Furthermore, we allow the adversary to corrupt some of the CAs that it chooses. We do not allow the adversary to corrupt any TPM, assumption that is consistent with the tamper proof nature of these devices. Our second contribution is a rigorous analysis of the PCAS protocol in the TCG TPM specification Version 1.2 [7], [8]. Our results show that the TCG PCAS protocol meets our security notion if the underlying asymmetric encryption algorithm and signature algorithm meet standard notions of security.

Our third contribution is a modification of the TCG PCAS protocol. We observe that the security model designed for the purpose of analyzing the TCG PCAS protocol does not capture the attack where an adversary with a broken genuine TPM claims to have a key belonging to another TPM. The problem is that our model does not allow for corruption of TPMs. In order to enhance security of the TCG PCAS we propose a new PCAS protocol. Security of this protocol can be proved under a stronger model which allows for the corruption of TPMs. Furthermore, the protocol that we propose has the implementation advantage that it does not require any change in the existing TPM functionality. Due to limited space we present the details of the stronger model and the analysis of the strengthened protocol in the full version of this paper.

In the rest of the paper, an overview of the TCG PCAS is given in Section II; the formal security model is introduced in Section III and followed by a formal security analysis of TCG PCAS in Section IV. We conclude with the improved PCAS protocol in Section V.

II. Overview of TCG PCAS

NOTATION: Throughout the paper, we will use some standard notation as follows. If S is a set, we denote the act of sampling from S uniformly at random and assigning the result to the variable x by $x \leftarrow S$. We let $\{0,1\}^*$ and $\{0,1\}^t$ denote the set of binary strings of arbitrary length and length t respectively. If A is an algorithm, we write $x \leftarrow A(y_1, \ldots, y_n)$ to indicate that x is obtained by invoking A on inputs $y_1, \ldots, y_n$. The algorithms that we consider may have access to oracles. We write $\mathcal{A}^{\mathcal{O}}$ to indicate that adversary $\mathcal{A}$ has access to oracle $\mathcal{O}$. We denote concatenation of two date strings x and y as $x \| y$.

PARTIES: The TCG PCAS is a protocol that involves a TPM, the host platform (host for short) that contains the TPM, and a privacy-CA (CA for short). The general setting that we analyze in this paper considers sets $\mathcal{C}$, $\mathcal{T}$ and $\mathcal{H}$ of CAs, TPMs, and Host entities. The latter set is in some sense irrelevant; it plays no role in our analysis as we assume that of all hosts are compromised. So, strictly speaking, PCAS is a two party protocol between a CA and a TPM.

CRYPTOGRAPHIC PRIMITIVES: Each CA $c_j \in \mathcal{C}$ has a public/private key pair $(\mathsf{cpk}_j, \mathsf{csk}_j)$ for a digital signature scheme $\mathsf{DSIG} = (\mathsf{KG}, \mathsf{SIG}, \mathsf{VER})$. Each TPM $t_l \in \mathcal{T}$ has a encryption/decryption key pair $(\mathsf{epk}_l, \mathsf{esk}_l)$ for an asymmetric encryption scheme $\mathsf{AENC} = (\mathsf{A.KG}, \mathsf{A.ENC}, \mathsf{A.DEC})$. Recall that the key pair $(\mathsf{epk}_l, \mathsf{esk}_l)$ is the TPM unique and long-term Endorsement Key (EK). Often in this paper, we identify CA c_j and TPM t_l via their public keys, cpk_j and epk_l, respectively. The protocol also uses a symmetric encryption scheme $\mathsf{SENC} = (\mathsf{S.KG}, \mathsf{S.ENC}, \mathsf{S.DEC})$.

EXECUTION: The goal of the PCAS protocol is two-fold. On the one hand the CA wants to ensure that it interacts with a legitimate TPM, and on the other hand the TPM wants to obtain a certificate (i.e. a signature of the CA) on some short-term public keys (i.e. Attestation Identity Keys (AIKs)) known by the TPM. Notice that for the purposes of this paper it is irrelevant how these keys are generated and for what particular communication applications they are intended. All that matters is that the TPM knows somehow these keys (and the associated secret keys) and that the TPM is able to verify its ownership of these keys. To have an easy mnemonic, we assume that $(\mathsf{pk}_{l1}, \mathsf{sk}_{l1}), (\mathsf{pk}_{l2}, \mathsf{sk}_{l2}), \ldots$ are the keys known by the TPM t_l. We also reflect this association via a list Known that maintains pairs of the form $(\mathsf{epk}, \mathsf{pk})$ with the meaning that a TPM with the public key epk had generated the short-term key pk. In addition, each CA knows the public key of some of the TPMs. We model this knowledge through a list Valid that maintains pairs of the form $(\mathsf{cpk}_j, \mathsf{epk}_l)$ to indicate that CA c_j knows the long-term public key of TPM t_l, that enables c_j to verify the legitimacy and attendance of t_l.

The PCAS protocol as shown in Figure 1, proceeds the following steps amongst the TPM t_l, the corresponding host h_l and the CA c_j:

1) h_l initiates the protocol by sending t_l a data pack denoted by $(\mathsf{pk}_{li})^*$ that asks t_l to load the key pair $(\mathsf{pk}_{li}, \mathsf{sk}_{li})$, and meanwhile, h_l sends $(\mathsf{pk}_{li}, \mathsf{epk}_l)$ to c_j.
2) c_j checks whether epk_l is in the list Valid. If the entry is found, c_j chooses a random challenge nonce c and encrypts $c \| \mathsf{pk}_{li}$ under epk_l by using AENC, and sends the ciphertext a back to h_l, who then forwards it to t_l.
3) t_l decrypts a using esk_l to get $c \| \mathsf{pk}_{li}$, and checks that pk_{li} is in the list Known and that the key pair $(\mathsf{pk}_{li}, \mathsf{sk}_{li})$ has been loaded; if the check passes, t_l releases c to h_l, who then forwards it to c_j.
4) c_j checks whether the received c matches with his challenge. If it does, c_j creates a fresh symmetric key k, a certificate on pk_{li}, say cer_{li}, and encrypts the value $k \| \mathsf{pk}_{li}$ again under epk_l by using AENC to get the ciphertext d. Meanwhile c_j encrypts cer_{li} under k by using SENC to get the ciphertext b, and then sends both d and b to h_l. In the end, c_j records the triple $(\mathsf{epk}_l, \mathsf{pk}_{li}, \mathsf{cer}_{li})$ into a list Record to maintain certificated keys with epk's.
5) After receiving the value d from h_l, t_l takes the same action as in the item 3), checks pk_{li} is in the list Known and $(\mathsf{pk}_{li}, \mathsf{sk}_{li})$ has been loaded and then, if the checks succeed, releases the value k to h_l.
6) h_l decrypts b under k to obtain cer_{li}.

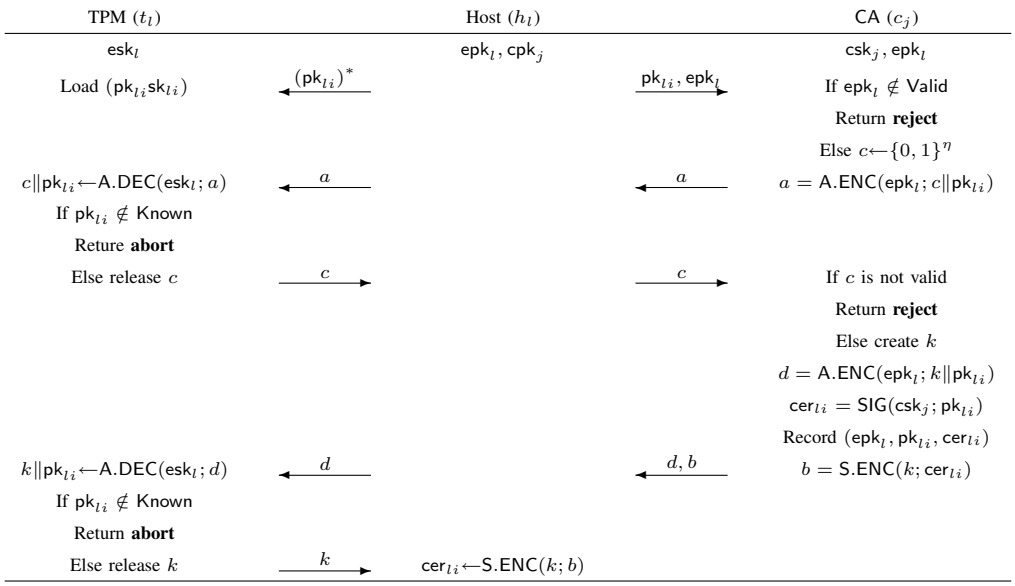

Fig. 1. The TCG PCAS protocol.

Remark 1: The TPM functionality in the PCAS protocol is achieved by running two TPM commands: the item 1) is done with the command TPM_LoadKey2; the items 3) and 5) are done with the command TPM_ActivateIdentity. The details of these two commands are specified in [7], [8].

III. Security Model

In this section we detail the security model that we use to study the PCAS protocol. As explained before, we consider a setting where an adversary controls all of the hosts and thus controls all of the communication between TPMs and CAs. Each TPM and each CA can engage in multiple runs of the protocol. We model this interaction as a polynomial time adversary $\mathcal{A}$ that has access to a set of oracles that represent the execution of the protocol by either a TPM or a CA. Each TPM oracle is parametrized by a pair of encryption/decryption keys. All of these oracles share a global variable ValidKey. This variable maintains a list of pairs $(\mathsf{epk}, \mathsf{pk})$ with the meaning that the key pk is valid for the TPM with the public key epk. We also write $\mathsf{ValidKey}(\mathsf{epk}, \mathsf{pk}) = 1$ to indicate that $(\mathsf{epk}, \mathsf{pk})$ occurs in ValidKey. The oracle expects to receive as input a key pk which should be one of the short-term keys which the TPM knows. If this is not the case, the execution is aborted. Otherwise, the TPM expects to receive a challenge ciphertext a with underlying plaintext of the form $c||\mathsf{pk}$ for some c. The oracle returns c. Later the oracle expects to receive the encryption of a plaintext $k||\mathsf{pk}$ under epk, in which case it returns k; otherwise the oracle aborts the execution.

The CA oracle is parametrized by verification and signing keys $(\mathsf{cpk}, \mathsf{csk})$. All of the CA oracles share a global variable ValidTPM. This variable maintains a list of pairs $(\mathsf{cpk}, \mathsf{epk})$; each such pair indicates that CA with the public key cpk knows the TPM with the public key epk. As above, we also write $\mathsf{ValidTPM}(\mathsf{cpk}, \mathsf{epk}) = 1$ to indicate that $(\mathsf{cpk}, \mathsf{epk})$ occurs in ValidTPM.

The CA oracle is activated by a pair of keys $(\mathsf{epk}, \mathsf{pk})$. If epk is not the public key of a valid TPM the oracle aborts its execution. Otherwise, the oracle selects a random nonce c and prepares and then outputs an encryption of $c||\mathsf{pk}$ under epk. Next, the oracle expects to receive c, upon which the CA creates a certificate cer on pk. The oracle then sends out a symmetric encryption of cer under a freshly generated key k, together with an asymmetric encryption of $k||\mathsf{pk}$ under epk.

The behaviour of the two types of oracles is summarized in Figure 2 and Figure 3, respectively.

TPM$(\mathsf{epk}, \mathsf{esk})$
1 receive pk
  if $\mathsf{ValidKey}(\mathsf{epk}, \mathsf{pk}) = 0$ then **abort**
2 receive a
  if $\mathsf{A.DEC}(\mathsf{esk}, a) \neq c||\mathsf{pk}$ for some c then **abort**
  send c
3 receive d
  if $\mathsf{A.DEC}(\mathsf{esk}, d) \neq k||\mathsf{pk}$ for some k then **abort**
  send k

Fig. 2. The TPM oracle.

We model the security of the protocol through experiment $\mathbf{Exp}_{\mathsf{PCAS},\mathcal{A}}(\eta)$ that involves an adversary $\mathcal{A}$ working against the protocol defined by TPM and CA.

To distinguish between the different sessions we write TPM_l^n for the n'th instance of the protocol executed by TPM t_l, and we write CA_j^m for the m'th instance of the protocol executed by CA c_j. To avoid multiple parameters, we assume that for any fixed adversary the number of TPMs, CAs is bounded by some polynomial $p(\eta)$. We also assume that the number of sessions for each party is also bounded by the same

$\mathrm{CA}(\mathsf{cpk}, \mathsf{csk})$

1 receive $(\mathsf{epk}, \mathsf{pk})$
  if $\mathsf{ValidTPM}(\mathsf{cpk}, \mathsf{epk}) = 0$ then **abort**
  select $c \leftarrow \{0,1\}^\eta$
  $a \leftarrow \mathsf{A.ENC}(\mathsf{epk}, c||\mathsf{pk})$
  send a
2 receive c'
  if $c \neq c'$ then **abort**
  $\mathsf{cer} \leftarrow \mathsf{Sign}(\mathsf{csk}, \mathsf{pk})$
  $k \leftarrow \mathsf{A.KG}(\eta)$; $b \leftarrow \mathsf{S.ENC}(k, \mathsf{cer})$
  $d \leftarrow \mathsf{A.ENC}(\mathsf{epk}, k||\mathsf{pk})$
  append $(\mathsf{epk}, \mathsf{cpk}, \mathsf{pk}, \mathsf{cer})$ to RegList
  send b, d

Fig. 3. The CA oracle.

$\mathbf{Exp}_{\mathsf{PCAS},\mathcal{A}}(\eta)$
  for each CA c_j do $(\mathsf{cpk}_j, \mathsf{csk}_j) \leftarrow \mathsf{KG}(\eta)$
  for each TPM t_l do $(\mathsf{epk}_l, \mathsf{esk}_l) \leftarrow \mathsf{A.KG}(\eta)$
  $\mathsf{ValidTPM}, \mathsf{ValidKey} \leftarrow \mathcal{A}(\vec{\mathsf{cpk}}, \vec{\mathsf{epk}})$
  $\mathsf{RegList} \leftarrow \emptyset$
  $(\mathsf{epk}_{l^*}, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*) \leftarrow \mathcal{A}^{\mathrm{TPM}_l^n, \mathrm{CA}_j^m, \mathsf{CorrCA}}$
  if $\mathsf{VER}(\mathsf{cpk}_{j^*}, (\mathsf{pk}^*, \mathsf{cer}^*)) = 0$ or csk_{j^*} is corrupt return 0
  if $(*, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*) \notin \mathsf{RegList}$ return 1
  if $(\mathsf{epk}_{l^*}, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*) \in \mathsf{RegList}$ and
    $\mathsf{ValidKey}(\mathsf{epk}_{l^*}, \mathsf{pk}^*) = 0$ return 1
  return 0

Fig. 4. Experiment that defines the security of the protocol. The adversary has access to $p(\eta)$ sessions for each of the $p(\eta)$ parties, i.e. CAs and TPMs.

polynomial. Strictly speaking, this polynomial depends on the adversary (adversary with more runing time may interact with more oracles) but we omit to explicitly show this dependency. Instead, we quantify over all polynomials when we define the advantage of the adversary.

The experiment, which we specify in detail in Figure 4, proceeds as follows. First, keys are generated for the TMPs and for the CAs. The adversary is given all of the public keys. We write $\vec{\mathsf{cpk}}$ for the set $\{\mathsf{cpk}_1, \mathsf{cpk}_2, \ldots\}$ of all public keys of the CAs, and $\vec{\mathsf{epk}}$ for the set $\{\mathsf{epk}_1, \mathsf{epk}_2, \ldots\}$ of all public keys of the TPMs. The adversary then specifies some arbitrary initial configurations. In particular, the adversary decides which CAs know which TPMs, by specifying a list $\mathsf{ValidTPM} \subset \vec{\mathsf{cpk}} \times \vec{\mathsf{epk}}$. The adversary also decides for each TPM what are the list of short-term keys that the TPM knows by specifying the list $\mathsf{ValidKey} \subset \{\mathsf{epk}_1, \mathsf{epk}_2, \ldots, \} \times \{0,1\}^*$. We require that ValidKey satisfies that for any $\mathsf{pk} \in \{0,1\}^*$ $\mathsf{ValidKey}(\mathsf{epk}_i, \mathsf{pk}) = 1$ then $\mathsf{ValidKey}(\mathsf{epk}_j, \mathsf{pk}) = 0$ for any $j \neq i$. This reflects the idea that the short-term keys valid for the TPM are disjoint. Notice that we do not impose any conditions on how the short-term keys are generated; the adversary is allowed to select arbitrary bit-strings for these keys. The adversary is then given access to multiple instances of the TPM and CA oracles. In addition, the adversary is given access to a CA corruption oracle. This oracle, CorrCA has access to the secret signing keys of the CAs. On an input $1 \leq j \leq p(\eta)$ the oracle returns csk_j. If the adversary sends j to this corruption oracle, then we say that its secret key csk_j is corrupt.

We write TPM_l^n for the n-th instance of the TPM oracle initialized with keys $\mathsf{epk}_l, \mathsf{esk}_l$. Similarly, we write CA_j^m for the m-th instance of the CA oracle initialized with keys $\mathsf{cpk}_j, \mathsf{csk}_j$. At the end of the execution, the adversary outputs a tuple $(\mathsf{epk}_{i^*}, \mathsf{cpk}_{j*}, \mathsf{pk}^*, \mathsf{cer}^*)$ with $\mathsf{epk}_{l^*} \in \vec{\mathsf{epk}}$, $\mathsf{cpk}_{j^*} \in \vec{\mathsf{cpk}}$ and $\mathsf{pk}^*, \mathsf{cer}^* \in \{0,1\}^*$. The outcome of the experiment is determined as follows.

If cer^* is not a valid signature on pk^* under cpk_{j^*} or the key csk_{j^*} has been corrupt then the experiment returns 0. Otherwise, if either c_{j^*} (with associated verification key cpk_{j^*}) never certified key pk so a tuple of the form $(\mathsf{epk}, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer})$ does not occur in RegList (in the formulation of the experiment we indicate using a "don't care" on the first position of the tuple), or if a tuple of the form $(\mathsf{epk}_l, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*)$ does occur in RegList (for some TPM public key TPM_l), then the key pk^* does not belong to TPM t_l, so $\mathsf{ValidKey}(\mathsf{epk}_l, \mathsf{pk}^*) = 0$ the experiment returns 1. If neither condition is satisfied, then the experiment returns 0.

Definition 1: For any adversary $\mathcal{A}$ we define its advantage as the probability that the experiment outputs 1, $\mathbf{Adv}_{\mathsf{PCAS},\mathcal{A}}(\eta) = \Pr[\mathbf{Exp}_{\mathsf{PCAS},\mathcal{A}}(\eta) = 1]$. The protocol is deemed secure, if for any probabilistic polynomial time adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathsf{PCAS},\mathcal{A}}(\eta)$ is a negligible function of the security parameter. Recall that in the above definition we quantify universally over the polynomial $p(\eta)$ that bounds the number of parties and the number of sessions of each party.

IV. Security Analysis

In this section we give and prove our main result. We prove that the protocol PCAS is secure, provided that the signature scheme used to produce certificates and the asymmetric encryption scheme used are secure. More precisely we prove the following theorem.

Theorem 1: If the signature scheme DSIG is EF-CMA secure, and the asymmetric encrytpion scheme AENC is IND-CCA secure, then the PCAS protocol is secure.

Proof: The idea of the proof is to change the execution in such a way that the adversary does not obtain any information about the random nonces that the certification authorities use in their challenges. Then, if the adversary produces a certificate on some key that had not been involved in an execution with the CA, then either the adversary managed to obtain the random nonce in the challenge (hence he breaks encryption) or he produced the certificate on his own (hence he breaks the signature scheme).

Technically, we prove that for any adversary $\mathcal{A}$ against the PCAS protocol, there exists adversaries $\mathcal{B}$ against the AENC encryption scheme and $\mathcal{C}$ against the signature scheme such

1 receive $(\mathsf{epk}, \mathsf{pk})$
  if $\mathsf{ValidTPM}(\mathsf{cpk}, \mathsf{epk}) = 0$ then **abort**
  select $c \leftarrow \{0,1\}^\eta$
  a$\leftarrow \mathsf{A.ENC}(\mathsf{epk}, 0^{|c||\mathsf{pk}|})$
  add $(\mathsf{epk}, a, c||\mathsf{pk})$ to list FakeEnc
 send a

Fig. 5. The $\mathrm{CA}'(\mathsf{cpk}, \mathsf{csk})$ oracle is obtained by replacing step 1 of the $\mathrm{CA}(\mathsf{cpk}, \mathsf{csk})$ with the above.

2 receive a
  if $(\mathsf{epk}, a, p||\mathsf{pk})$ in FakeEnc (for some p)
   then send $c = p$
   else if $\mathsf{A.DEC}(\mathsf{esk}, a) \neq c||\mathsf{pk}$ for some c
   then **abort**
 send c
3 receive d
  if $(\mathsf{epk}, d, p||\mathsf{pk})$ in FakeEnc (for some p)
   then send $k = p$
   else if $\mathsf{A.DEC}(\mathsf{esk}, d) \neq k||\mathsf{pk}$ for some k
   then **abort**
 send k

Fig. 6. The $\mathrm{TPM}'(\mathsf{epk}, \mathsf{esk})$ oracle is obtaine by replacing steps 2 and 3 of TPM with the above.

that:

$$\mathbf{Adv}_{\mathsf{PCAS},\mathcal{A}}(\eta) \leq \mathbf{Adv}^{\mathsf{indcca}}_{\mathsf{AENC},\mathcal{B}}(\eta) + \frac{1}{p(\eta)} \cdot \mathbf{Adv}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{C}}(\eta) + \frac{1}{2^\eta} \quad (1)$$

We consider a modified experiment where the challenge encryptions produced by the certification authorities are "fake", in that the underlying plaintext is $0^{|c||\mathsf{pk}_{in}|}$ (so only contains the encryption of 0). To ensure that the TPMs can decrypt fake ciphertexts appropriately the experiment maintains a list FakeEnc. An entry in the list is of the form (epk, c, p) to signify that c is a fake encryption of p under the key epk. More specifically, modify the oracles to which $\mathcal{A}$ has access as follows. In the CA_j oracle we modify by replacing the first part of the protocol with the process shown in Figure 5.

The protocol run by the TPM is modified accordingly: before decrypting a ciphertext, the TPM verifies that the encryption is not a fake one. If it is not the case, then the TPM works as before. If the encryption is fake then the underlying plaintext is recovered from the FakeEnc list. In the changed oracle we replace steps 2 and 3 with the new steps as shown in Figure 6.

Let TPM' and CA' be the modified oracles and let $\mathbf{Exp}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$ be the resulting modified experiment (the corruption oracle is unchanged). We define $\mathbf{Adv}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$ to be the corresponding advantage. The following lemma says that if a polynomial time adversary sees a difference between the experiment $\mathbf{Exp}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$ and experiment $\mathbf{Exp}_{\mathsf{PCAS},\mathcal{A}}(\eta)$ (that is, if the output of the experiment changes noticeably

1 receive $(\mathsf{epk}, \mathsf{pk})$
  if $\mathsf{ValidTPM}(\mathsf{cpk}, \mathsf{epk}) = 0$ then **abort**
  select $c \leftarrow \{0,1\}^\eta$
  a$\leftarrow \mathsf{A.ENC}(\mathsf{epk}, c||\mathsf{pk})$
  add $(\mathsf{epk}, a, c||\mathsf{pk})$ to list FakeEnc
 send a

Fig. 7. The $\mathrm{CA}''(\mathsf{cpk}, \mathsf{csk})$ oracle is obtained by replacing step 1 of the $\mathrm{CA}(\mathsf{cpk}, \mathsf{csk})$ with the above.

between the two experiments) then the adversary $\mathcal{A}$ somehow breaks the asymmetric encryption scheme AENC.

Lemma 1: For any adversary $\mathcal{A}$ against PCAS there exists an adversary $\mathcal{B}$ against AENC such that:

$$\mathbf{Adv}_{\mathsf{PCAS},\mathcal{A}}(\eta) = \mathbf{Adv}_{\mathsf{PCAS}',A}(\eta) + \mathbf{Adv}^{\mathsf{indcca}}_{\mathsf{AENC},\mathcal{B}}(\eta) \quad (2)$$

Proof: We begin the proof of this lemma with the following observation. We first consider a further modification of the CA' oracle (recall that CA' oracle is obtained from the CA oracle via the modifications in Figure 5). The modification is that instead of encrypting $0^{|c||\mathsf{pk}_{in}|}$ the modified oracle actually encrypts $c||\mathsf{pk}_{in}$, i.e. the message that the original CA oracle encrypts. Let CA'' be the resulting oracle. It is immediate that the set of oracles $\{\mathrm{CA}''^m_j, \mathrm{TPM}'^n_l \mid 1 \leq j, l, m, n \leq p(\eta)\}$ behaves precisely as the set of oracles $\{\mathrm{CA}^m_j, \mathrm{TPM}^n_l \mid 1 \leq j, l, m, n \leq p(\eta)\}$.

The only difference between the two sets of oracles is that decryption of ciphertexts produced by CA oracles is done by the decryption algorithm for the original oracles and by table lookup for the modified ones. So, if we write $\mathbf{Exp}_{\mathsf{PCAS}'',\mathcal{A}}(\eta)$ for the experiment where the adversary is given access to oracles CA' and TPM'' then:

$$\Pr[\mathbf{Exp}_{\mathsf{PCAS},\mathcal{A}}(\eta) = 1] = \Pr[\mathbf{Exp}_{\mathsf{PCAS}'',\mathcal{A}}(\eta) = 1] \quad (3)$$

To prove the lemma we show that given an adversary $\mathcal{A}$ against PCAS we construct an adversary $\mathcal{B}$ against AENC such that:

$$\Pr[\mathbf{Exp}^{\mathsf{indcca\text{-}0}}_{\mathsf{AENC},\mathcal{B}}(\eta) = 1] = \Pr[\mathbf{Exp}_{\mathsf{PCAS}'',\mathcal{A}}(\eta) = 1] \quad (4)$$

and

$$\Pr[\mathbf{Exp}^{\mathsf{indcca\text{-}1}}_{\mathsf{AENC},\mathcal{B}}(\eta) = 1] = \Pr[\mathbf{Exp}_{\mathsf{PCAS}',\mathcal{A}}(\eta) = 1] \quad (5)$$

Then, the desired relation (Equation 2) follows by subtracting Equation 5 from Equation 4 and using Equation 3.

We now give the details of the adversary $\mathcal{B}$. This adversary is against encryption, so, by the notion of IND-CCA security of asymmetric encryption as shown in the appendix, it has access to $p(\eta)$ encryption oracles and their corresponding decryption oracles and receives as input the encryption keys for these oracles. Essentially, the adversary $\mathcal{B}$ simulates for $\mathcal{A}$ the environment of either $\mathbf{Exp}_{\mathsf{PCAS}'',\mathcal{A}}(\eta)$ or $\mathbf{Exp}_{\mathsf{PCAS},\mathcal{A}}(\eta)$, depending on the hidden bit in the oracles of $\mathcal{B}$. $\mathcal{B}$ generates signing keys for each CA. In the simulation, the TPM oracles use as encryption keys the encryption keys of the oracles to which $\mathcal{B}$ has access. The CA oracles are simulated by B on his

own: the corresponding signing keys are generated by $\mathcal{B}$ and the necessary decryptions are performed using the decryption oracles to which $\mathcal{B}$ has access or, as explained bellow, by doing look-ups in an appropriately constructed table.

Specifically, for each $c_j \in \mathcal{C}$ $\mathcal{B}$ runs $(\mathsf{csk}_j, \mathsf{cpk}_j) \leftarrow \mathsf{KG}(\eta)$. $\mathcal{B}$ is also given as input $\vec{\mathsf{epk}}$ the set of encryption keys $\mathsf{epk}_1, \ldots, \mathsf{epk}_2, \ldots$ used by the oracles to which it has access. Then $\mathcal{B}$ passes $\vec{\mathsf{epk}}$ and $\vec{\mathsf{cpk}}$ to $\mathcal{A}$ who starts making his queries. Adversary $\mathcal{B}$ simulates the oracles to which $\mathcal{A}$ has access, and the simulation is as follows. $\mathcal{B}$ maintains a variable FakeOrRealEnc similar to the one used by oracles CA' and CA''. To simulate a CA oracle, $\mathcal{B}$ executes the code of the CA' oracle, with the difference that ciphertext a is obtained as follows.

If the key epk that the adversary sends to the CA oracle is different from all keys epk_l in $\vec{\mathsf{epk}}$ then a is obtained by simply encrypting $c||pk$ under epk. Otherwise (i.e. $\mathsf{epk} = \mathsf{epk}_l$ for some l), $\mathcal{B}$ sends $c||\mathsf{pk}$ to the encryption oracle under epk_l and obtains in return some ciphertext a. $\mathcal{B}$ then adds $(\mathsf{epk}, a, c||\mathsf{pk})$ to the list FakeEnc. Notice that a is either a true encryption of $c||\mathsf{pk}$ if the bit of the encryption oracle is 0, or is the encryption of 0s if the hiddent bit is 1, so the simulation for CA oracles that $\mathcal{B}$ provides for $\mathcal{A}$ is either the execution of CA'' oracles if $b = 0$ or as the execution of CA' oracles if $b = 1$. The corruption queries (where $\mathcal{A}$ sends j to CorrCA are answered with csk_j).

The simulation of TPM oracles is precisely as for oracles TPM', with the difference that instead of FakeEnc the look-up is done in the list FakeOrRealEnc and that whenever the TPM oracle that corresponds to some key epk needs to peform a decryption, the decryption is done via decryption oracle under the corresponding esk. The simulation of TPM oracles that the adversary $\mathcal{B}$ provides to $\mathcal{A}$ is identical to the execution of the TPM' oracle. Furthermore, it is immediate that the adversary $\mathcal{B}$ that we construct is valid: he never sends to the decryption oracle a ciphertext obtained from the encryption oracle. This is true since all of these ciphertexts are maintained in FakeOrRealEnc list which is checked against prior to a decryption. If the ciphertexts do occur in the list, then underlying plaintext is recovered directly. To conclude, when the hidden bit b is 0, the view of $\mathcal{A}$ is as in $\mathbf{Exp}^{}_{\mathsf{PCAS}'',\mathcal{A}}(\eta)$ so Equation 4 holds. Similarly, when the hidden bit b is **1**, the view of $\mathcal{A}$ is as in $\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$ so Equation 5. ■

The next lemma states that an adversary that breaks the modified protocol actually breaks the digital signature scheme.

Lemma 2: For any adversary $\mathcal{A}$ against PCAS' there exists an adversary $\mathcal{B}$ against DSIG and a polynomial p such that:

$$\mathbf{Adv}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta) \leq \frac{1}{p(\eta)} \cdot \mathbf{Adv}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{B}}(\eta) + \frac{1}{2^\eta}$$

Proof: Fix an adversary $\mathcal{A}$ for the experiment $\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$. Let NoCert be the event that for the tuple $(\mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*)$ output by $\mathcal{A}$ there is no entry of the form $(\mathsf{epk}, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*)$ in RegList, and let CertInvKey be the event that $(\mathsf{epk}_l, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*)$ for some epk_l, and $\mathsf{ValidKey}(\mathsf{epk}_l, \mathsf{pk}^*) = 0$. Then we have that:

$$\begin{aligned}
&\Pr[\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta) = 1] = \\
&\quad \Pr[\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta) = 1 \wedge \mathsf{NoCert}] + \\
&\quad \Pr[\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta) = 1 \wedge \mathsf{CertInvKey}] \;\; = \\
&\quad \Pr[\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta) = 1 \wedge \mathsf{NoCert}] + \\
&\quad \Pr[\mathsf{CertInvKey}] \cdot \Pr[\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta) = 1 \mid \mathsf{CertInvKey}] \leq \\
&\quad \Pr[\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta) = 1 \wedge \mathsf{NoCert}] + \Pr[\mathsf{CertInvKey}]
\end{aligned}$$

Next we upper bound each of the terms above. The intuition is as follows. If event NoCert occurs, then cer^* is not produced by any oracle $\mathrm{CA}(\mathsf{cpk}_{j^*}, \mathsf{csk}_{j^*})$, so the adversary $\mathcal{A}$ must have produced this signature on his own, hence he forged a signature under csk^*. This intuition is behind the construction of the following adversary $\mathcal{B}$ for $\mathbf{Exp}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{B}}(\eta)$ out of an adversary $\mathcal{A}$ for $\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$.

Adversary $\mathcal{B}$ is for $\mathbf{Exp}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{B}}$ so, by the notion of EU-CMA signature schemes as shown in the appendix, $\mathcal{B}$ has as input some verification key pk and access to a signing oracle under the corresponding secret key sk. The adversary $\mathcal{B}$ run adversary $\mathcal{A}$ internally and simulates for $\mathcal{A}$ all of the oracles to which $\mathcal{A}$ has access in $\mathbf{Exp}^{}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$. The encryption keys for all of the TPM' oracles are obtained by running the key generation algorithm of the encryption scheme: $(\mathsf{epk}_l, \mathsf{esk}_l) \leftarrow \mathsf{A.KG}(\eta)$ (for all $1 \leq l \leq p(\eta)$). The keys for all but one randomly selected CAs are obtained using the key generation algorithm $(\mathsf{csk}_j, \mathsf{cpk}_j) \leftarrow \mathsf{KG}(\eta)$ (for all $1 \leq j \neq j_0 \leq p(\eta)$, where j_0 is selected uniformly at random from $1, 2, \ldots, p(\eta)$). The public key of CA_{j_0} is set to be the verification key pk that $\mathcal{B}$ has as input. Adversary $\mathcal{B}$ can therefore simulate perfectly all of the oracles: for TPM'^n_l oracles he simply executes the code defined by TPM' (Figures 2 and 6). For any oracle CA'^m_j with $j \neq j_0$ the adversary executes the code defined by CA' (Figures 3 and 5). For all oracles $\mathrm{CA}^m_{j_0}$ the adversary $\mathcal{B}$ executes the same code with the difference that any signature σ that such an oracle needs to produce on some message m is obtained from the signing oracle to which $\mathcal{B}$ has access. Adversary $\mathcal{B}$ can also answer corruption queries for all of the CAs except for c_{j_0}; if $\mathcal{A}$ ever queries j_0 to his corruption oracle, then $\mathcal{B}$ aborts its execution. Provided that j_0 is not submitted by $\mathcal{A}$ to his decryption oracle, the simulation that $\mathcal{B}$ offers to $\mathcal{A}$ is therefore perfect. When $\mathcal{A}$ produces his forgery $(\mathsf{cpk}^*, \mathsf{pk}^*, \mathsf{cer}^*)$ adversary $\mathcal{B}$ proceeds as follows. If $\mathsf{cpk}^* \neq \mathsf{pk}$ then $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ outputs $(\mathsf{pk}^*, \mathsf{cer}^*)$ as his forgery.

Since the simulation that $\mathcal{A}$ provides is perfect (if $\mathcal{B}$ does not abort), cpk_{j^*} in the output of $\mathcal{A}$ equals cpk_{j_0} (i.e. pk) with probability $\frac{1}{p(\eta)}$. It remains to determine when the forgery that is output by $\mathcal{B}$ is valid (if the guess j_0 is correct, that is, it does correspond to the forgery). This is the case when the message/signature pair $(\mathsf{pk}^*, \mathsf{cer}^*)$ is valid under pk, the signature cer^* had not been produced by the oracle to which $\mathcal{B}$ has access in response to a query pk^* by $\mathcal{B}$, and $\mathcal{B}$ did not abort due to a CAcorruption query. Whenever adversary $\mathcal{A}$ wins the triple $(\mathsf{cpk}^*_j, \mathsf{pk}^*, \mathsf{cer}^*)$ is such that cer^* had not

been produced by some oracle cpk_j^* as a signature on pk^*, since otherwise, a tuple of the form $\mathsf{epk}_l, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*$ would appear in RegList since RegList contains all tuples $(\mathsf{epk}, \mathsf{pk}, m, \mathsf{cer})$ where pk is the verification key that $\mathcal{B}$ has as input and cer had been obtained from the signing oracle of $\mathcal{B}$. In this case, j^* had also not been queried by $\mathcal{A}$ to its corruption oracle (as otherwise the output of $\mathcal{A}$ would not lead to a successful break), so $\mathcal{B}$ does not abort. To conclude, when $\mathcal{A}$ produces his output $(\mathsf{cpk}^*, \mathsf{pk}^*, \mathsf{cer}^*)$, with probability $\frac{1}{p(\eta)}$ the forgery is with respect to the public key of CA_{j_0} so $\mathsf{pk}^* = \mathsf{pk}$. Furthermore, if event NoCert occurs in the simulation of $\mathbf{Exp}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$, then $(*, \mathsf{pk}, \mathsf{pk}^*, \mathsf{cer}^*)$ does not appear in RegList. According to the observation above cer^* has not been obtained by sending pk^* to the signing oracle under sk, hence $(\mathsf{pk}^*, \mathsf{cer}^*)$ is a successful forgery for $\mathbf{Exp}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{B}}(\eta)$. We therefore obtain that

$$\Pr[\mathbf{Exp}_{\mathsf{PCAS}',\mathcal{A}}(\eta) \wedge \mathsf{NoCert}] \leq \frac{1}{p(\eta)} \cdot \Pr[\mathbf{Exp}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{B}}(\eta) = 1] \tag{6}$$

Next we bound the probability of event $\Pr[\mathsf{CertInvKey}]$. An important observation is that in the interaction of adversary $\mathcal{A}$ in $\mathbf{Exp}_{\mathsf{PCAS}',\mathcal{A}}(\eta)$ all information about the nonces that are used by the CA' oracles in their challenge/response phase is completely hidden from the adversary. Indeed, the ciphertexts that the adversary receives are only encryptions of 0s.

Assume that event CertInvKey occurs so for some epk_l the tuple $(\mathsf{epk}_l, \mathsf{cpk}_{j^*}, \mathsf{pk}^*, \mathsf{cer}^*)$ appears in RegList and $\mathsf{ValidKey}(\mathsf{epk}_l, \mathsf{pk}^*) = 0$. Then, there exists some oracle $\mathcal{O} = \mathrm{CA}'(\mathsf{cpk}_{j^*}, \mathsf{csk}_{j^*})$ such that $\mathsf{epk}_l, \mathsf{pk}^*$ is the first message received by $\mathcal{O}$. The oracle then outputs an encryption a^* of $0^{|c^*||pk^*|}$ under epk_l and records $(\mathsf{epk}_l, a, c^*||\mathsf{pk}^*)$ in FakeEnc. The oracle receives as its next message from the adversary c^* (as otherwise the oracle would abort its execution).

However, the only way for $\mathcal{A}$ to obtain any information about c^* is to submit a^* to some TPM' oracle for which the public encryption key is epk_l. By the definition of TPM' the only case when upon query a^* the oracle would return c^* is if pk^* is such that $\mathsf{ValidKey}(\mathsf{epk}_l, \mathsf{pk}^*) = 1$, but this contradicts that $\mathsf{ValidKey}(\mathsf{epk}_l, \mathsf{pk}^*) = 0$. In conclusion, c^* is informationally theortic hidden from $\mathcal{A}$ so the probability that the second message of $\mathcal{A}$ to oracle $\mathcal{O}$ is c^* is at most $\frac{1}{2^\eta}$.

Together with the previous bound, we obtain that:

$$\mathbf{Adv}_{\mathsf{PCAS}',\mathcal{A}}(\eta) \leq \frac{1}{p(\eta)} \mathbf{Adv}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{B}} + \frac{1}{2^\eta} \tag{7}$$

■

The proof of the main theorem follows immediately from the above two lemmas. ■

Observe that security of the symmetric encryption scheme SENC is not required for the security proof. So considering the security strength, the PCAS protocol in Section II is not different from the modification with $k = \mathsf{cer}$ and ignoring SENC. However, the usage of the symmetric encryption scheme provides some implementation benefit: since cer could be longer than one block of SENC, and therefore, using SENC can reduce the TPM workload.

V. Improvement of TCG PCAS

In the TCG PCAS protocol, from the CA's point of view, the possession of esk_l is proved by using a challenge-response protocol under the asymmetric encryption scheme AENC. The possession of pk_{li} is not proved. The CA accepts the statement that the owner of epk_l also owns pk_{li} relying on the fact that the TPM with the key esk_l should follow the protocol properly, since the TPM is a tamper-resistant chip. However, if any TPM is compromised, this fact will no longer be true. If a compromised TPM's epk is still in the list Valid, then the TPM can obtain a certificate on any honest TPM's pk, which will be listed in Record together with the *compromised but still valid* epk. For this reason, our security model for PCAS in Section III does not allow the adversary to corrupt any TPM.

In this section, we proposes a modified PCAS protocol in order to enhance security, with which we mean that if an adversary has a broken genuine TPM, he is only able to claim a key belonging to this TPM, but not any key belonging to any honest TPM. This protocol is secure under a stronger model against the adoptive chosen TPM attack, i.e., the adversary can adaptively corrupt an arbitrary number of TPMs in his choice.

The enhanced PCAS protocol as shown in Figure 8, proceeds the following steps among the TPM t_l, the corresponding host h_l and the CA c_j: h_l initiates the protocol by sending t_l a data pack $(\mathsf{pk}_{li})^*$. t_l loads the key pair $(\mathsf{pk}_{li}, \mathsf{sk}_{li})$ and verifies that the key is in the list Known. Recall that the list Known maintains these keys created by t_l and has been loaded to t_l. If the verification passes, t_l signs its epk_l under sk_{li} and returns the signature scer_{li} back to h_l, who then sends pk_{li}, epk_l and scer_{li} to c_j. c_j checks whether epk_l is in the list Valid. Recall that the list Valid maintains these epk's the CA already know. c_j also checks whether scer_{li} is a valid signature of epk_l. If both checks pass, the rest of the protocol is identical to the second phase of the TCG PCAS protocol in Figure 1.

Remark 2: The TPM functionality in this enhanced PCAS protocol is achieved by two respective TPM commands, namely TPM_CertifyKey and TPM_ActivateIdentity.

As mentioned earlier, for the reason of the limited space, we leave a formal proof of security of the new PCAS protocol in the full version of this paper. Here we only give an intuitive analysis. The signature scer_{li} on epk_l under sk_{li} shows that someone knowing sk_{li} has claimed that he is the owner of epk_l. The possession of esk_l is proved in the PCAS protocol by using a challenge-response protocol under the asymmetric encryption scheme AENC. The new PCAS protocol binds these two functions together to convince the CA that the owner of epk_l and the owner of pk_{li} are essentially the same entity. However, the CA is not able to convince a third part of this knowledge, since the CA can simply create the key pair $(\mathsf{pk}_{li}, \mathsf{sk}_{li})$ and sign epk_l under the key by himself.

References

[1] M. Bellare, A. Boldyreva and S. Micali. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In *Advances in Cryptology — EUROCRYPT '00*, Springer-Verlag LNCS 1807, 259–274, 2003.

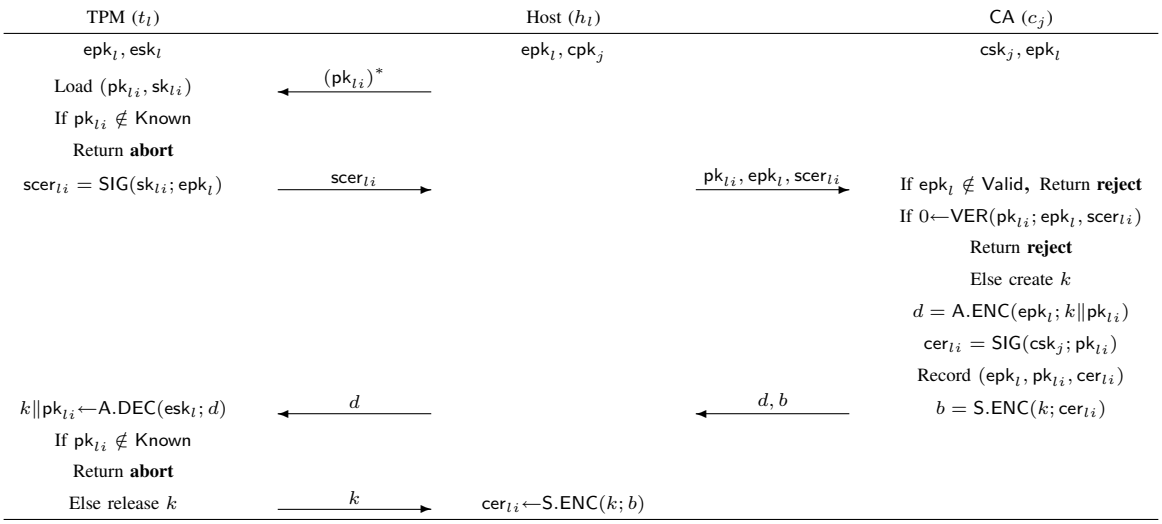

Fig. 8. The enhanced PCAS protocol.

[2] M. Bellare, D. Micciancio and B. Warinschi. Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology — EUROCRYPT '03*, Springer-Verlag LNCS 2656, 614–629, 2003.

[3] M. Bellare, H. Shi and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *Topics in Cryptology – CT-RSA 2005*, Springer-Verlag LNCS 3376, 136–153, 2005.

[4] E. Brickell, J. Camenisch and L. Chen. Direct anonymous attestation. In *the 11th ACM Conference on Computer and Communications Security*. ACM Press, 132–145, 2004.

[5] E. Brickell, L. Chen and J. Li. Simplified security notions for direct anonymous attestation and a concrete scheme from pairings. *Int. Journal of Information Security*, **8**, 315–330, 2009.

[6] L. Chen, P. Morrissey and N. P. Smart. DAA: Fixing the pairing based protocols. *Cryptology ePrint Archive*. Report 2009/198, available at http://eprint.iacr.org/2009/198.

[7] ISO/IEC 11889:2009 Information technology – Security techniques – Trusted Platform Module.

[8] Trusted Computing Group. TCG TPM specification 1.2. Available at http://www.trustedcomputinggroup.org, 2003.

[9] Trusted Computing Group. http://www.trustedcomputinggroup.org. Last accessed on June 30, 2010.

APPENDIX

IND-CCA SECURITY OF ASYMMETRIC ENCRYPTION: In this paper we use security of asymmetric encryption schemes defined in a multi-user environment [1]. The experiment that defines this notion uses a left-right function $\mathsf{LR}(\cdot, b)$: if b is 0 then the oracle on input m returns m. If the bit b is 1 then the oracle returns $0^{|m|}$, i.e. the all-zero string of length $|m|$.

The experiment $\mathbf{Exp}^{\mathsf{indcca}\text{-}b}_{\mathsf{AENC},\mathcal{A}}(\eta)$ that defines IND-CCA security for encryption scheme AENC involves an adversary $\mathcal{A}$ and is parametrized by a bit b. The experiment is the following one. Encryption/decryption key pairs $(ek_1, dk_1), (ek_2, dk_2), \ldots, p(\eta)$ are generated using the key generation algorithm of the encryption scheme, run on security parameter η. Here $p(\cdot)$ is some fixed polynomial. The adversary is then given access to a set of encryption oracles, oracle i being keyed with key ek_i. When the adversary sends message m to encryption oracle i, it receives in return a ciphertext under ek_i. If the bit b is 0 then the underlying plaintext is m, otherwise the underlying plaintext is $0^{|m|}$. The adversary is also given access to decryption oracles under keys dk_i. On a query c, decryption oracle i returns $\mathsf{A.DEC}(dk_i, c)$. Eventually the adversary outputs a guess bit d and terminates. If $d = b$, then the experiment returns 1, otherwise the experiment returns 0. In his interaction, the adversary is not allowed to submit a ciphertext c obtain from encryption oracle with key ek_i to the corresponding decryption oracle. The advantage of adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{indcca}}_{\mathsf{A.ENC},\mathcal{A}}(\eta) = \Pr[\mathbf{Exp}^{\mathsf{indcca}\text{-}0}_{\mathsf{AENC},\mathcal{A}}(\eta) = 1] - \Pr[\mathbf{Exp}^{\mathsf{indcca}\text{-}1}_{\mathsf{AENC},\mathcal{A}}(\eta) = 0].$$

We say that encryption scheme AENC is IND-CCA if for any probabilistic polynomial time adversary $\mathbf{Adv}^{\mathsf{indcca}}_{\mathsf{A.ENC},\mathcal{A}}(\eta)$ is a negligible function. In the above definition the polynomial p used to define the experiment is quantified universally.

EU-CMA FOR DIGITAL SIGNATURE SCHEMES: The security of a signature scheme $\mathsf{DSIG} = (\mathsf{KG}, \mathsf{SIG}, \mathsf{VER})$ is defined through game $\mathbf{Exp}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{A}}(\eta)$. In this game a pair of signing and verification keys (sk, pk) is generated by running the key generation algorithm of the digital signature scheme. Then, adversary $\mathcal{A}$ is given the verification key pk is provided with access to an oracle $\mathsf{SIG}(\mathsf{sk}, \cdot)$. For each message m that the adversary sends to the oracle, the oracle responds with a signature σ on m. Eventually, the adversary terminates its execution and outputs a message signature pair (m^*, σ^*). The experiment returns 1 if σ^* is a valid signature on m^* under pk^* and the message m^* was never queried to the oracle. The experiment returns 0 otherwise. The advantage of the adversary in breaking existential-unforgeability under chosen message attack for the scheme DSIG is defined as:

$$\mathbf{Adv}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{A}}(\eta) = \Pr[\mathbf{Exp}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{A}}(\eta) = 1]$$

The scheme DSIG is EU-CMA secure if for any probabilitisc polynomial time adversary $\mathcal{A}$, its advantage $\mathbf{Adv}^{\mathsf{eucma}}_{\mathsf{DSIG},\mathcal{A}}(\eta)$ is a negligible function.