Property-Based Attestation without a Trusted Third Party

Liqun Chen[1], Hans Löhr[2], Mark Manulis[3], and Ahmad-Reza Sadeghi[2]

[1] HP Laboratories, Bristol, UK; liqun.chen@hp.com
[2] Horst Görtz Institute for IT Security, Ruhr-University of Bochum, Germany
{hans.loehr|ahmad.sadeghi}@trust.rub.de
[3] UCL Crypto Group, Université Catholique de Louvain, Belgium
mark.manulis@uclouvain.be

Abstract. The Trusted Computing Group (TCG) has proposed the binary attestation mechanism that enables a computing platform with a dedicated security chip, the Trusted Platform Module (TPM), to report its state to remote parties. The concept of property-based attestation (PBA) improves the binary attestation and compensates for some of its main deficiencies. In particular, PBA enhances user privacy by allowing the trusted platform to prove to a remote entity that it has certain properties without revealing its own configuration.
The existing PBA solutions, however, require a Trusted Third Party (TTP) to provide a reliable link of configurations to properties, e.g., by means of certificates. We present a new privacy-preserving PBA approach that avoids such a TTP. We define a formal model, propose an efficient protocol based on the ideas of ring signatures, and prove its security. The cryptographic technique deployed in our protocol is of independent interest, as it shows how ring signatures can be used to efficiently prove the knowledge of an element in a list without disclosing it.

Keywords. Property-based attestation, user privacy, ring signatures, proof of membership, configuration anonymity.

1 Introduction and Background

A fundamental issue in interaction between computing platforms is "trust" or "trustworthiness" — whether a remote platform behaves in a reliable and predictable manner, or will be (or already has been) subject to subversion. Cryptographic mechanisms support the establishment of secure channels and authorized access, but without assurance about the integrity of the communication endpoints. Commodity computing platforms suffer from inherent vulnerabilities due to high complexity, and lack of efficient protection against tampering or malware. Hence, an important subject of current research is to develop mechanisms for gaining assurance about the trustworthiness of remote peers regarding their integrity, platform configuration, and security policies. The concept of Trusted Computing aims at resolving such issues.

The TCG approach and binary attestation. An industrial approach towards the realization of the Trusted Computing functionality within the computing platforms is the initiative of the Trusted Computing Group (TCG). The

TCG has published many specifications amongst which the most important one is that of the Trusted Platform Module (TPM) [25]. Currently, TPMs are implemented as small, tamper-evident hardware modules embedded in commodity platforms, providing (i) a set of cryptographic functionalities, (ii) the protection of cryptographic keys, (iii) the authentication of platform configuration (attestation), and (iv) cryptographic sealing of sensitive data to particular system configurations. However, the TCG defines only a limited set of commands, and the firmware cannot be programmed by end-users to execute arbitrary functions. Millions of platforms (PCs, notebooks, and servers) being sold today are equipped with TPMs.

One of the main features supported by the TPM is the so-called trusted integrity measurement: a hash value of the platform state is computed during the boot process and stored in specific registers of the TPM, the *Platform Configuration Registers* (PCRs),those state is also called the platform's *configuration*. Of potential interest is the offered functionality called *binary attestation*, which allows a remote party (verifier) to get an authentic report about the binary configuration of another platform (prover), given by the prover's TPM signature on the configuration.

Deficiencies of TCG binary attestation. TCG binary attestation suffers from several shortcomings: The slightest change in the measured software or configuration files — whether security-relevant or not — will lead to a changed binary configuration. In general, it is not clear, how a verifier should derive the trustworthiness of a platform from such a binary value. System updates and backups are highly non-trivial; the multitude of different versions of many pieces of software cause serious manageability problems.

From the privacy point of view, binary attestation bears several risks: (1) The TPM's public key needed to verify an attestation could be used to identify a TPM and trace a platform. To solve this problem, Brickell et al. [3] introduced the *Direct Anonymous Attestation* (DAA) protocol. Improvements of DAA and alternative DAA schemes (e.g., [5,4,6]) are orthogonal to our work and could be used as a building block for our protocol. (2) Typically the information about the configuration of a computing platform or application is revealed to a remote party requesting the state of a platform. This information can be misused to discriminate against certain configurations (for example, operating systems) and even vendors, or may be exploited to mount attacks.

Property-based attestation. One general concept to overcome shortcomings of the TCG's binary attestation is to transform the binary attestation into the *property-based attestation* (PBA), as described by Sadeghi and Stüble [21], and by Poritz et al. [19]. The basic idea of PBA requires a computing platform to attest that it fulfills the desired (security) requirements, so-called 'properties', without revealing a respective software or/and hardware configuration. The formal definition of properties as well as the development of various practical solutions for PBA are still active areas of ongoing research.

One concrete solution for PBA was proposed by Chen et al. in 2006 [11]. Their protocol requires an off-line Trusted Third Party (TTP) to publish a list

of trusted configurations and respective certificates which attest that the configurations provide specific properties. A prover can use the signed configurations and certificates to prove to a verifier that it has appropriate configurations associated with the certified properties, without disclosing the specific configurations, which the platform holds.

Another solution for PBA is proposed by Kühn et al. [14]. In their work, the authors suggest a modified system boot architecture, such that not binary hash values of files are stored by the TPM, but instead abstract values representing properties, e.g., a public key associated with a property certificate. However, this approach also requires a TTP to issue certificates for properties and the bootloader must be binary-attested.

The drawback of these solutions is that such a TTP might not be available or/and desirable in many real applications, for example if two entities/users want to have a private communication with each other. They have their own understanding of the relation between various configurations and security properties. They do not need (and do not want) to ask any kind of TTPs to certify a correlation between the configurations and properties. However, they still want to keep their platform configuration information secret from each other.

Our contribution. In this paper, we propose a protocol for PBA that does not require the involvement of a TTP to certify properties, where a platform (equipped with a TPM) convinces a remote party that its configuration satisfies a given property. For this, the two parties first agree on a set of trusted configuration specifications, which they both consider to be trustworthy, i.e., associated with a well-defined security property or properties. The platform then proves that its configuration specification is in this set. In our protocol, TPM and the host software compute the proof jointly.

For some applications, it might be unrealistic to assume that the parties in the attestation protocol can decide themselves which configurations are trustworthy and which are not, and thus they still have to rely on third parties in practice. Our protocol has the advantage that even in this case no global trusted party is necessary: both participants can choose independently how to agree on trustworthy configurations or they can delegate this decision to other parties.

Further, we define a formal security model for PBA, which we also use in our proofs, and where the main security requirements are evidence authentication and configuration privacy. While the former guarantees an unforgeable binding between the platform and its configuration specification, the latter provides the non-disclosure of the configuration specification. In our PBA protocol, these requirements are achieved through the use of a ring signature (cf. Section 4.3), i.e. configuration privacy results from the anonymity of the signer whereas evidence authentication is based on the unforgeability of the signature.

Moreover, the cryptographic technique employed in our protocol may be of independent interest: We show how ring signatures can be used for efficiently proving the knowledge of an element in a list without disclosing it.

Outline. In Section 2, we introduce the system model of property-based attestation. In Section 3, we sketch different solutions on a high level. In Section 4,

we set up notation and explain some building blocks which will be used in our concrete PBA scheme. In Section 5, we present and discuss a new PBA scheme, and in Section 6 we define a formal security model and state theorems about the security of the scheme. In Section 7, we conclude the paper by mentioning some unsolved problems and future work.

2 System Model for PBA

The following system model for PBA will serve as the basis for the security model in Section 6.

Involved parties. A PBA protocol involves two participants: a *prover* $\mathcal{P}$ and a *verifier* $\mathcal{V}$. The prover is a platform consisting of a host $\mathcal{H}$ and a trusted module TPM $\mathcal{M}$ (see Figure 1). To cover multiple executions of the protocol we consider multiple instances and use indices to distinguish among their participants, i.e., $\mathcal{P}_i$, $\mathcal{V}_i$. Each instance includes a single protocol execution with some *unique session identity* (SID) and two participants $\mathcal{P}_i$ and $\mathcal{V}_j$ are treated as communication partners (in the same instance) if they share the same SID.

Assumptions. It is assumed that the communication between a host $\mathcal{H}_i$ and its TPM $\mathcal{M}_i$ is through a secure channel (private and authentic), and that $\mathcal{M}_i$ and $\mathcal{V}_i$ communicate via $\mathcal{H}_i$. We omit the indices i and j of the participants in an instance when no risk of confusion exists. Moreover, the TPM is trusted by all parties and possesses a secret (signing) key $sk_{\mathcal{M}}$ which is unknown to the host. The corresponding public (verification) key is available to both $\mathcal{P}$ and $\mathcal{V}$; see also "trust relations" in Section 6.1.

Properties and configurations. Each prover $\mathcal{P}$ has a configuration value denoted $cs_{\mathcal{P}}$, which is an authenticated record about its platform's configuration. The value $cs_{\mathcal{P}}$ is known to both the host $\mathcal{H}$ and TPM $\mathcal{M}$, and it is computed by $\mathcal{M}$ from correctly measured configuration information, stored securely in special-purpose registers — the platform configuration registers (PCRs). As a result, $\mathcal{H}$ cannot modify this value without being detected. This is guaranteed by the properties of secure measurement and reporting based on the trusted computing technology [25]. It is assumed that before running the PBA protocol, $\mathcal{P}$ and $\mathcal{V}$ have already agreed on a set of configuration values denoted $CS = \{cs_1, ..., cs_n\}$ that satisfy the same property. So, we say that a configuration value cs satisfies a given property associated with CS, if and only if $cs \in CS$.

Definition of PBA. A property-based attestation (PBA) scheme consists of the following three polynomial-time algorithms:

– Setup: Given the security parameter 1^κ, this probabilistic algorithm selects a set of public parameters that are necessary to run the PBA protocol, and produces a private/public key pair for each TPM.

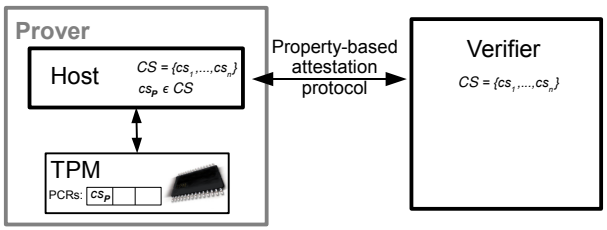

Fig. 1. PBA system model.

- PBA-Sign: On input a configuration value $cs_{\mathcal{P}}$, a list of admissible configurations CS, and a nonce N_v, this (distributed) randomized algorithm outputs a signature σ on $cs_{\mathcal{P}}$.
- PBA-Verify: On input a candidate signature σ and CS, this deterministic algorithm outputs 1 (accept) if σ is a valid signature on a value from CS, or 0 (reject) otherwise.

3 Solutions

In this section, we sketch two high-level solutions for PBA without relying on trusted parties to certify the link between configurations and properties.

Basically, $\mathcal{P}$ has to prove that its configuration value $cs_{\mathcal{P}}$ belongs to the agreed set $CS = \{cs_1, \ldots, cs_n\}$. More precisely, $\mathcal{V}$ would accept a proof if and only if: (i) The proof is created by a *valid* TPM. If TPM anonymity is required, the DAA scheme [3] can be used to provide this feature. (ii) The proof is a fresh response to a specific challenge from $\mathcal{V}$. (iii) The proof ensures that $cs_{\mathcal{P}} = cs_j$ for an index $j \in \{1, 2, \ldots, n\}$, but does not reveal the value of j.

Such a proof implements PBA-Sign, whereas PBA-Verify is the verification of the proof. In Setup, the keys for the TPM and system parameters are generated.

Solution 1: *TPM as single signer.* The proof can be achieved by a new TPM command defined as follows:

1. TPM takes as input a list of configurations CS and a nonce N. The nonce is assumed to be chosen by the verifier $\mathcal{V}$.
2. TPM checks for each $cs_j \in CS$ if $cs_{\mathcal{P}} = cs_j$, until either a match is found, or the entire list has been checked.
3. If $cs_{\mathcal{P}}$ is in the list, the TPM generates a signature on $(1, N, CS)$; otherwise, the TPM generates a signature on $(0, N, CS)$, which is then forwarded to $\mathcal{V}$.

The obvious drawbacks of this approach are: TPM operations depend on the size n of CS ($\mathcal{O}(n)$ in a straightforward implementation, and $\mathcal{O}(\log n)$ if CS is a sorted list). As the TPM's memory is very limited, this would either impose a severe restriction on the size of CS, or the transfer of the list would have to be split up, causing further complexity of the TPM-command and slowing down the communication between host and TPM, due to the overhead.

Solution 2: *TPM shares signer role with host.* In this solution, the TPM signs a hidden version — a commitment — of the configuration $cs_{\mathcal{P}}$, and the host completes the proof that the hidden configuration is in the set CS. A similar approach is used in the DAA protocol [3].

Our PBA protocol proposed in Section 5 is an elegant and efficient example of this solution. It makes use of ring signatures in that the host computes n public keys for a ring signature scheme from the configurations in CS and the commitment to $cs_{\mathcal{P}}$ (which was signed by the TPM), and determines the secret key that corresponds to $cs_{\mathcal{P}}$. The signer anonymity of the ring signature scheme ensures that the verifier does not learn which key has been used for signing, thus $cs_{\mathcal{P}}$ is not disclosed. Our construction guarantees that the prover succeeds only if the hidden configuration $cs_{\mathcal{P}}$ is indeed in CS.

Current TPMs support all operations (random number generation, modular exponentiation, and signature generation) needed by our protocol. However, the TCG currently does not specify a command to create and sign a commitment to a configuration which is stored inside the TPM. To implement such a command, only firmware changes would be required.

Other protocols for similar solutions could be developed, for instance based on existing zero-knowledge proofs (e.g., [8,13,7]) or zero-knowledge sets [15].

4 Preliminaries

4.1 TPM Signatures

The existing TCG technology defines two ways for a TPM to create own digital signature $\sigma_{\mathcal{M}}$. The first way is to use DAA [3]. With a DAA signature, a verifier is convinced that a TPM has signed a given message, but the verifier cannot learn the identity of the TPM. The message to be signed can be either an Attestation Identity Key (AIK), or an arbitrary data string, The second way is to use an ordinary signature scheme. A TPM generates a signature using an AIK as signing key, which could either be certified by a Privacy-CA, or it could be introduced by the TPM itself using a DAA signature. For simplicity, we do not distinguish these two cases, and denote by $\sigma_{\mathcal{M}} := \mathtt{SignM}(sk_{\mathcal{M}}; m)$ the output of TPM's signing algorithm on input the TPM's signing key $sk_{\mathcal{M}}$ and a message m, and by $\mathtt{VerM}(vk_{\mathcal{M}}; \sigma_{\mathcal{M}}, m)$ the corresponding verification algorithm, which on input the TPM's verification key $vk_{\mathcal{M}}$ outputs 1 if $\sigma_{\mathcal{M}}$ is valid and 0 otherwise.

4.2 Commitment Scheme

We apply the commitment scheme by Pedersen [18]: Let $sk^m_{\mathtt{com}}$ be the secret commitment key. A commitment on a message m is computed as $C_m := g^m h^{sk^m_{\mathtt{com}}} \bmod P$. P is a large prime, h is a generator of a cyclic subgroup $G_Q \subseteq \mathbb{Z}^*_P$ of prime order Q and $Q|P-1$. g is chosen randomly from $\langle h \rangle$; furthermore, $\log_h(g)$ is unknown to the committing party. Both the message m and $sk^m_{\mathtt{com}}$ are taken from $\mathbb{Z}_Q$. The Pedersen commitment scheme as described above is perfectly hiding and computationally binding, assuming the hardness of the discrete logarithm problem in a subgroup of $\mathbb{Z}^*_P$ of prime order (for P prime).

4.3 Ring Signatures

The notion of a ring signature was first introduced by Rivest et al. [20]. It allows a signer to create a signature with respect to a set of public keys. Successful verification convinces a verifier that a private key corresponding to one of the public keys was used, without disclosing which one. In contrast to group signatures, no group manager is needed.

For various security definitions for ring signatures see [2]. Recent efficient ring signature schemes which are provably secure in the standard model (i.e., without using random oracles) are proposed in [23,9], where in [9] a signature with size only $\mathcal{O}(\sqrt{n})$ is proposed. Dodis et al. [12] showed that ring signatures with constant size in the number of public keys can be achieved in the random oracle model.

Unfortunately, none of these schemes can be used easily for our purposes: In our protocol, we employ a construction, where the public keys for the ring signature are computed from commitments formed by the TPM. We show how this can be done efficiently for Pedersen commitments (cf. Section 4.2) and public keys of the form $y = g^x \bmod P$, where x is the corresponding secret key. However, the schemes above use keys of different types.

In Figure 2, we recall an efficient ring signature scheme from [1], which we propose to use for our PBA solution. The scheme is a generalization of the Schnorr signature scheme [22]: Intuitively, the product in step 2(b) corresponds to combined commitments for individual Schnorr signatures, in step 2(c) and 2(d), the challenges for the individual Schnorr signatures are derived from a single challenge, and in step 2(e), the secret key is used to compute s. The verification equation, where the sum of the challenges is compared to a hash value, ensures that a valid signature cannot be created without a secret key x_j. The scheme is provably secure in the random oracle model, under the discrete logarithm assumption.

We denote the generation of a ring signature σ_r on message m with respect to the public key ring $\{y_i\}_{1 \leq i \leq n}$ and with private signing key x by $\sigma_r := \texttt{SigRing}(x; \{y_i\}; m)$. Signature verification is denoted by $\texttt{VerRing}(\{y_i\}; \sigma_r, m)$. For simplicity, we omit the public parameters g, P, Q and the range of the index i in our notation.

5 Ring Signature-Based PBA without TTP

In this section, we propose a protocol for PBA, which is based on ring signatures. The TPM generates a signature on a commitment to the configuration $cs_{\mathcal{P}}$. Then the host $\mathcal{H}$ creates a proof, using a ring signature, that $cs_{\mathcal{P}}$ is in the agreed set CS of configurations with the given property. The verifier $\mathcal{V}$ verifies the TPM signature and the ring signature.

Note that in our protocol, the TPM is trusted by all parties, but its resources are restricted, and it can execute only a very limited set of instructions. The host $\mathcal{H}$ is not trusted by the verifier $\mathcal{V}$, hence the protocol has to protect evidence

1. Key generation. Let κ be a security parameter. On input 1^κ, create g, P and Q. A signer S_i $(i = 1, ..., n)$ chooses $x_i \in_R \{0,1\}^{\ell_Q}$ and compute $y_i = g^{x_i} \bmod P$. Output its public key (g, P, Q, y_i) and the corresponding secret key x_i.
2. Signing algorithm $\texttt{SigRing}(x_j; \{y_i\}; m)$.
   A signer who owns secret key x_j generates a ring signature on a message m with public key list (g, P, Q, y_i) $(i = 1, ..., n)$, where $j \in \{1, ..., n\}$ as follows:
   (a) Choose $\alpha, c_i \in_R \{0,1\}^{\ell_Q}$ for $i = 1, ..., n$, $i \neq j$.
   (b) Compute $z = g^\alpha \prod_{i=1, i\neq j}^{n} y_i^{c_i} \bmod P$.
   (c) Compute $c = \texttt{Hash}(g \| P \| Q \| y_1 \| ... \| y_n \| m \| z)$.
   (d) Compute $c_j = c - (c_1 + ... + c_{j-1} + c_{j+1} + ... + c_n) \bmod Q$.
   (e) Compute $s = \alpha - c_j \cdot x_j \bmod Q$.
   (f) Output the signature $\sigma_r = (s, c_1, ..., c_n)$.
3. Verification algorithm $\texttt{VerRing}(\{y_i\}; \sigma_r, m)$.
   To verify that the tuple $\sigma_r = (s, c_1, ..., c_n)$ is a ring signature on message m, check that $\sum_{i=1}^{n} c_i \equiv \texttt{Hash}(g \| P \| Q \| y_1 \| ... \| y_n \| m \| g^s y_1^{c_1} ... y_n^{c_n} \bmod P)$.

Fig. 2. A Ring Signature Scheme [1]

authentication against a malicious host. $\mathcal{H}$ cannot be prevented from disclosing its own configuration $cs_\mathcal{P}$, thus for configuration privacy, we have to assume that $\mathcal{H}$ is honest.

5.1 Security Parameters

We suggest the following security parameters (values in parentheses indicate realistic values[4] for current TPMs):

- ℓ_{cs} (160): the size of the value of $cs_\mathcal{P}$.
- $\ell_\varnothing$ (160): the security parameter for the anti-replay value (nonce).
- ℓ_P (1024): the size of the modulus P.
- ℓ_Q (160): the size of the order Q of the subgroup of $\mathbb{Z}_P^*$.

The parameters ℓ_P and ℓ_Q should be chosen such that the discrete logarithm problem in the subgroup of $\mathbb{Z}_P^*$ of order Q with P and Q being primes such that $2^{\ell_Q} > Q > 2^{\ell_Q - 1}$ and $2^{\ell_P} > P > 2^{\ell_P - 1}$, is computationally hard.

5.2 Setup

We assume that $\mathcal{V}$ can verify TPM signatures (including revocation verification) and that $\mathcal{H}$ and $\mathcal{V}$ have agreed on a set of configurations CS.

Prior to the execution of the PBA protocol, the parties have to agree on the following parameters, which can be used for several protocol runs (potentially

[4] examples based on the use of SHA-1 [16] as a hash function (like in current TPMs), and recommendations of the US National Institute of Standards and Technology (NIST) for similar applications (see, for instance, [17]); changes corresponding to stronger hash-functions, such as SHA-256, can be made straightforwardly.

with different sets CS): primes P and Q, generators g and h of a subgroup of $\mathbb{Z}_P^*$ of order Q (i.e., the discrete logarithm problem is hard in $\langle g \rangle = \langle h \rangle$). The discrete logarithm $\log_g(h) \bmod P$ must be unknown to $\mathcal{H}$.

5.3 Signing and Verifying Protocol

The attestation procedure executed between a TPM ($\mathcal{M}$), its host ($\mathcal{H}$), and a verifier ($\mathcal{V}$) is described in Figure 3. As a result of the protocol, the host creates a ring signature σ_r, which is based on a TPM signature σ_M on the message C, which is a commitment to $cs_\mathcal{P}$. The TPM has to create and sign C, which it then opens towards $\mathcal{H}$. To create the ring signature, the host uses the value r as the secret key (if $cs_\mathcal{P} \in CS$, this works, because $y_j = h^r \bmod P$ for some j). From the ring signature, the verifier is convinced that the platform has been configured with one of the set of acceptable configuration specifications, $CS = \{cs_1, \cdots, cs_n\}$, without knowing which one.

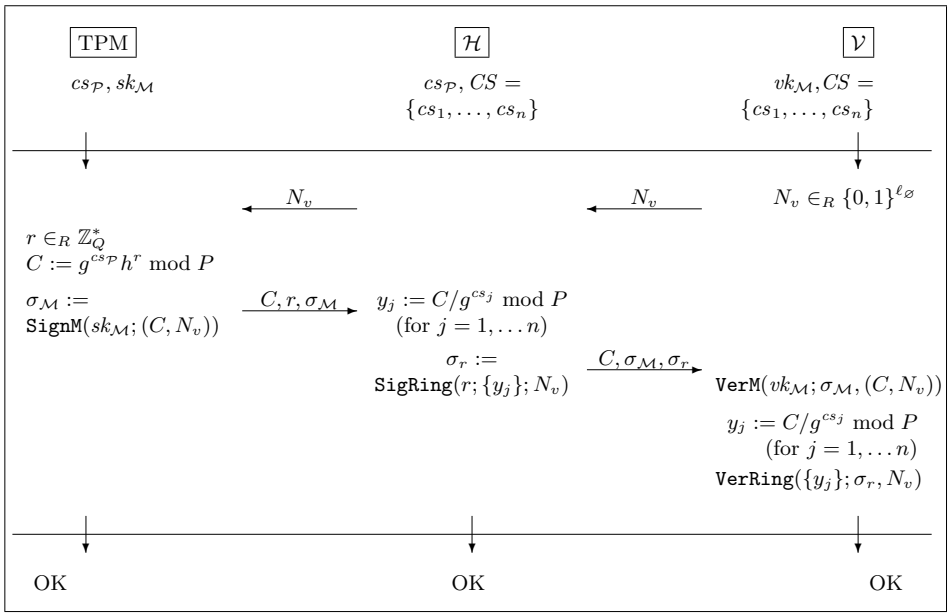

Fig. 3. The protocol of the PBA scheme. Common input: g, h, P, Q

5.4 Protocol Properties

Our protocol has some interesting properties:

First, no trusted third party is needed for this protocol. The only exception is the certification of TPM keys: The verifier may rely on a DAA issuer or a

Privacy-CA to ensure that the TPM key belongs to a valid TPM, depending on the TPM signature scheme (see Section 4.1). However, this is completely independent from the PBA protocol, and neither a DAA issuer nor a Privacy-CA could breach the configuration privacy of our protocol.

Second, the configuration set CS is created flexibly, dependent on the agreement between prover $\mathcal{P}$ and verifier $\mathcal{V}$. One approach to negotiate the set of acceptable configurations could be analogous to the SSL/TLS handshake: The prover sends a proposal for CS to $\mathcal{V}$, who can then select an appropriate subset. However, our protocol allows for different ways to agree on CS; the particular method can be chosen according to a concrete application scenario.

Third, the size n of the set CS affects the configuration privacy. If n is small, $\mathcal{V}$ might have a high probability in guessing the configuration $cs_{\mathcal{P}}$. Therefore, to keep $cs_{\mathcal{P}}$ private, $\mathcal{P}$ should execute the protocol only if CS is of acceptable size. Moreover, $\mathcal{P}$ has to ensure that $\mathcal{V}$ cannot learn $cs_{\mathcal{P}}$ by running the PBA protocol multiple times with different configuration sets, because in the case of several successful attestations, $\mathcal{V}$ would know that $cs_{\mathcal{P}}$ is in the (possibly small) intersection of the sets used in the protocol executions. This example shows that $\mathcal{P}$ should install a privacy policy which prevents such abuses of the PBA protocol.

Fourth, note that the overhead of the TPM compared to binary attestation is small. Additionally, the TPM has to form the commitment C, which must be signed instead of $cs_{\mathcal{P}}$. So the overhead is just choosing a random number r and performing a modular multi-exponentiation modulo P (with two exponents). As with binary attestation, the TPM has to generate one digital signature (e.g., 2048 bit RSA). The TPM's computation does not depend on the size of CS.

6 Security of our PBA Scheme

Here, we define a formal (game-based) security model based on the system model from Section 2, and state theorems about the security of our PBA scheme.

6.1 Security Model

Adversary model. The adversary $\mathcal{A}$ is a PPT algorithm and an active adversary that has full control over the communication channel between $\mathcal{H}$ and $\mathcal{V}$. This is modeled by the query of the form $\texttt{send}(E, m)$ which allows $\mathcal{A}$ to address a message m to an entity $E \in \{\mathcal{H}, \mathcal{V}\}$. In response, A receives a message which would be generated by E according to the protocol execution. In the definition of entity authentication, in which malicious hosts should also be considered, $\mathcal{A}$ is also given access to another query $\texttt{sendTPM}(m)$ by which it can communicate with $\mathcal{M}$. We assume that m contains the identity of the sender (as chosen by $\mathcal{A}$). Moreover, when considering evidence authentication, the adversary may corrupt the host via the query $\texttt{corrupt}_{\mathcal{H}}$, which returns the configuration $cs_{\mathcal{P}}$ to $\mathcal{A}$ ($cs_{\mathcal{P}}$ is $\mathcal{H}$'s only secret).

We assume that $\mathcal{A}$ cannot corrupt the TPM. In reality, a hardware attack would be necessary to corrupt a TPM, i.e., we limit the adversary to software-only attacks, which is the assumption of the TCG [25]. In case a real-world adversary succeeds in attacking the TPM, our protocol has to rely on the revocation mechanisms for TPM signatures.

Evidence authentication. We formalize the intuitive security requirement that $\mathcal{A}$ should not be able to pretend that $\mathcal{P}$ has a configuration $cs_{\mathcal{P}}$ satisfying the property that has to be attested (i.e., $cs_{\mathcal{P}} \in CS$), when in fact the property is not fulfilled (i.e., $cs_{\mathcal{P}} \notin CS$).

Let $\mathsf{Game}_{\mathcal{A}}^{\text{ev-auth}}(1^\kappa)$ be the following interaction between $\mathcal{P}$, $\mathcal{V}$, and $\mathcal{A}$. Before the interaction, $\mathcal{A}$ chooses a platform with a valid TPM $\mathcal{M}$ and with a configuration $cs_{\mathcal{P}} \notin CS$. Then $\mathcal{A}$ is given access to $\texttt{send}(E, m)$, $\texttt{sendTPM}(m)$, and $\texttt{corrupt}_{\mathcal{H}}$ queries to any $\mathcal{P}$ chosen by $\mathcal{A}$. Uncorrupted parties behave as specified by the protocol. $\mathcal{A}$ wins, if it outputs a PBA signature σ, such that $\mathsf{PBA\text{-}Verify}$ accepts σ. We denote the success probability of $\mathcal{A}$ by $\mathsf{Succ}_{\mathcal{A}}^{\text{cf-priv}}(1^\kappa) := \Pr[\mathsf{Game}_{\mathcal{A}}^{\text{ev-auth}}(1^\kappa) = \mathsf{win}]$, and its maximum over all PPT adversaries $\mathcal{A}$ (running in time κ) as $\mathsf{Succ}^{\text{cf-priv}}(1^\kappa)$.
A PBA protocol provides evidence authentication if $\mathsf{Succ}^{\text{cf-priv}}(1^\kappa)$ is negligible in κ.

Configuration privacy. The security requirement that the configuration $cs_{\mathcal{P}}$ of $\mathcal{P}$ should be kept private is captured by the following game. For this requirement, host $\mathcal{H}$ and TPM $\mathcal{M}$ of $\mathcal{P}$ have to be honest because $\mathcal{P}$ could always send $cs_{\mathcal{P}}$ to $\mathcal{A}$.

Let $\mathsf{Game}_{\mathcal{A}}^{\text{cf-priv}}(1^\kappa)$ be the following interaction between $\mathcal{P}$, $\mathcal{V}$ and $\mathcal{A}$. $\mathcal{A}$ is given access to $\texttt{send}(E, m)$ queries. Moreover, $\mathcal{A}$ may access $\texttt{sendTPM}(m)$ and $\texttt{corrupt}_{\mathcal{H}}$ queries for all but one prover $\mathcal{P}$ chosen adaptively by $\mathcal{A}$, which has to remain honest. At the end of the interaction, $\mathcal{A}$ outputs an index i. $\mathcal{A}$ wins if i is the index of $\mathcal{P}$'s configuration in the set $CS = \{cs_1, \ldots, cs_n\}$, i.e., if $cs_{\mathcal{P}} = cs_i$. We denote the advantage of $\mathcal{A}$ (over a random guess) with $\mathsf{Adv}_{\mathcal{A}}^{\text{cf-priv}}(1^\kappa, n) := |\Pr[\mathsf{Game}_{\mathcal{A}}^{\text{cf-priv}}(1^\kappa) = \mathsf{win}] - 1/n|$, and its maximum over all PPT adversaries $\mathcal{A}$ (running in time κ) as $\mathsf{Adv}^{\text{cf-priv}}(1^\kappa, n)$.
A PBA protocol provides configuration privacy if $\mathsf{Adv}^{\text{cf-priv}}(1^\kappa, n)$ is negligible in κ.

Security of PBA. A PBA scheme is *secure*, if and only if it provides both evidence authentication and configuration privacy.

Trust relations. The TPM is assumed to be trusted by both host and verifier. For evidence authentication, a PBA protocol must ensure that a malicious host cannot cheat an honest verifier, whereas for configuration privacy, it must prevent a verifier controlled by $\mathcal{A}$ from determining the configuration of an honest host.

6.2 Security Analysis

The following theorems demonstrate the security of our PBA scheme. For the proofs, see Appendix A.

Theorem 1 (Evidence Authentication). *The PBA protocol presented in Section 5 provides evidence authentication (in the random oracle model), assuming the security of the ring signature scheme, the security of TPM signatures, and the hardness of the discrete logarithm assumption. In more detail:*

$$\mathsf{Succ}^{\mathsf{cf\text{-}priv}}(1^\kappa) \quad \leq \quad q^2/2^{\ell_\varnothing} + \varepsilon_{\mathsf{TPM}} + \varepsilon_{\mathsf{ring}} + \varepsilon_{\mathsf{dlog}},$$

where q is the number of protocol runs, $\ell_\varnothing$ is polynomial in the security parameter κ, $\varepsilon_{\mathsf{TPM}}$ is the probability of an adversary to forge a TPM signature, $\varepsilon_{\mathsf{ring}}$ is the probability to forge a ring signature, and $\varepsilon_{\mathsf{dlog}}$ is the probability to solve the underlying discrete logarithm problem.

Remark. Our proof does not directly use the random oracle model, however, it is required by the ring signature scheme we use.

Theorem 2 (Configuration Privacy). *The PBA protocol presented in Section 5 provides configuration privacy against computationally unbounded adversaries, due to the unconditional signer anonymity of the ring signature scheme and perfect hiding of the commitment scheme.*

Remark. Although our definition of configuration privacy assumes a PPT adversary (which would be reasonable for practical purposes), our protocol offers even unconditional security, because we use a perfectly hiding commitment scheme and an unconditionally signer-anonymous ring signature scheme.

7 Conclusion and Future Work

The concept of property-based attestation (PBA) has been proposed to overcome several deficiencies of the (binary) attestation scheme proposed by the Trusted Computing Group (TCG). Amongst others, the TCG attestation reveals the system configuration to third parties that could misuse it for privacy violations and product discrimination.

In this paper, we proposed the first cryptographic protocol for PBA which, in contrast to the previous solutions, does not require a Trusted Third Party to certify properties. In our protocol, the TPM has to compute only one commitment and one signature.

Furthermore, the cryptographic technique used here might be of independent interest: We demonstrate how a ring signature can be employed to prove membership in a list.

Future work may include the investigation of how to determine meaningful properties. Moreover, a generic approach based on any ring signature scheme, an efficient scheme with a security proof in the standard model, and the design of a PBA protocol with sub-linear communication and computation complexity in the size of the configuration set CS are still open problems.

References

1. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT '02*, LNCS vol. 2501, pp. 415-432. Springer, 2002.
2. A. Bender, J. Katz, R. Morselli. Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. In *TCC '06*, LNCS vol. 3876, pp. 60–79, Springer, 2006.
3. E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In B. Pfitzmann, and P. Liu, editor, *Proceedings of ACM CCS '04*, pp. 132–145, ACM Press, Oct. 2004.
4. E. Brickell, L. Chen, and J. Li. A new direct anonymous attestation scheme from bilinear maps. In *Proceedings of the 1st International Conference on Trust (TRUST 2008)*, LNCS, Springer Verlag, March 2008.
5. E. Brickell and J. Li. Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 6th Workshop on Privacy in the Electronic Society (WPES'07)*, pp. 21–30, ACM Press, 2007.
6. J. Camenisch. Better privacy for trusted computing platforms. In *Proceedings of 9th European Symposium On Research in Computer Security (ESORICS 2004)*, LNCS vol. 3193, pp. 73–88, Springer Verlag, 2004.
7. J. Camenisch and M. Michels. Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes. In *EUROCRYPT '99*, LNCS vol. 1592, pp. 107–122, Springer, 1999.
8. J. Camenisch and M. Stadler. Proof Systems for General Statements about Discrete Logarithms. Technical Report TR 260, Dep. of Computer Science, ETH Zürich, March 1997.
9. N. Chandran, J. Groth, and A. Sahai. Ring Signatures of Sub-linear Size Without Random Oracles. In *Proceedings of ICALP '07*, LNCS vol. 4596, pp. 423–434, Springer, 2007.
10. D. Chaum and H. van Antwerpen. Undeniable signatures. In *CRYPTO '89*, LNCS vol. 435, pp. 212–216, Springer, 1990.
11. L. Chen, R. Landfermann, H. Löhr, M. Rohe, A. Sadeghi and C. Stüble. A Protocol for Property-Based Attestation. In *Proceedings of ACM STC '06*, pp. 7–16, ACM Press, 2006.
12. Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT '04*, LNCS vol. 3027, pp. 609–626, Springer, 2004.
13. E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *CRYPTO '97*, LNCS vol. 1294, pp. 16–30, Springer, 1997.
14. U. Kühn, M. Selhorst, and C. Stüble. Realizing Property-Based Attestation and Sealing on Commonly Available Hard- and Software. In *ACM STC '07*, pp. 50–57, ACM Press, 2007.
15. S. Micali, M. O. Rabin, and J Kilian. Zero-Knowledge Sets. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS'03)*, pp. 80–91, IEEE Computer Society, 2003.
16. National Institute of Standards and Technology (NIST). Secure Hash Standard (SHS). FIPS PUB 180-2, August 2002.
17. National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). FIPS PUB 186-3 (Draft), March 2006.
18. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *CRYPTO '91*, LNCS vol. 576, pp. 129–140. Springer, 1992.

19. J. Poritz, M. Schunter, E. van Herreweghen, and M. Waidner. Property Attestation – Scalable and Privacy-friendly Security Assessment of Peer Computers. IBM Research Report RZ 3548 (# 99559), Oct. 2004.
20. R. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In C. Boyd, editor, *ASIACRYPT '01*, LNCS vol. 2248, pp. 552–565, Springer, 2001.
21. A. Sadeghi and C. Stüble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. In *Proceedings of NSPW '04*, pp. 67–77, ACM Press, Sep. 2004.
22. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *J. Cryptology*, 4(3):161–174, Springer, 1991.
23. H. Shacham and B. Waters. Efficient Ring Signatures without Random Oracles. In *Proceedings of PKC '07*, LNCS vol. 4450, pp. 166–180, Springer, 2007.
24. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *Cryptology ePrint Archive*, Report 2004/332 `http://eprint.iacr.org/2004/332`, 2004.
25. Trusted Computing Group. TCG TPM Specification, Version 1.2. Available at https://www.trustedcomputinggroup.org/.

A Security Proofs

Proof (Evidence Authentication). We structure the proof as a sequence of games [24], where a PPT adversary $\mathcal{A}$ (see Section 2 for the adversary model) interacts with a simulator $\mathcal{S}$. The first game is $\mathsf{Game}_{\mathcal{A}}^{\mathsf{ev\text{-}auth}}$. In each subsequent game, a new "event" is introduced. $\mathcal{S}$ aborts, whenever this event occurs. We show that each event can only happen with negligible probability for any PPT adversary, hence the probability for $\mathcal{A}$ to win game $\mathbf{G}_{i+1}$, denoted by $\Pr[\mathsf{win}_{i+1}]$, differs only by a negligible amount from its probability $\Pr[\mathsf{win}_i]$ to win game $\mathbf{G}_i$.

$\mathbf{G}_0$ The initial game is $\mathsf{Game}_{\mathcal{A}}^{\mathsf{ev\text{-}auth}}$, where $\mathcal{S}$ plays the game with $\mathcal{A}$ by simulating the honest parties as specified by the protocol. $\mathcal{A}$ chooses a platform with a configuration $cs_{\mathcal{P}} \notin CS$ of his choice (as specified in Section 6.1), and $\mathcal{S}$ simulates the honest TPM $\mathcal{M}$ of this platform. $\mathcal{A}$ wins $\mathsf{Game}_{\mathcal{A}}^{\mathsf{ev\text{-}auth}}$, and hence $\mathbf{G}_0$, if it manages to output $\sigma = (C, \sigma_{\mathcal{M}}, \sigma_r)$ such that $\mathcal{S}$ (acting as an honest verifier) accepts σ as a proof that $cs_{\mathcal{P}} \in CS$, although actually $cs_{\mathcal{P}} \notin CS$. Because $\mathbf{G}_0$ is $\mathsf{Game}_{\mathcal{A}}^{\mathsf{ev\text{-}auth}}$, we have $\Pr[\mathsf{win}_0] = \mathsf{Succ}^{\mathsf{cf\text{-}priv}}(1^\kappa)$.

$\mathbf{G}_1$ In the event that $\mathcal{S}$, acting as a verifier, chooses a nonce N_v that already occurred in a previous protocol run, $\mathcal{S}$ aborts the simulation. For this comparison, $\mathcal{S}$ records all nonces. As N_v is chosen randomly by $\mathcal{S}$, the probability ε_1 of this is $\leq q^2/2^{\ell_{\mathcal{O}}}$ (which is negligible in the security parameter), where q denotes the number of protocol runs. Hence, $\mathsf{Succ}^{\mathsf{cf\text{-}priv}}(1^\kappa) \leq \Pr[\mathsf{win}_1] + \varepsilon_1$.

$\mathbf{G}_2$ $\mathcal{S}$ simulates protocol execution as before, with the difference that all TPM signatures are obtained from the corresponding signing oracle. In the event that $\mathcal{S}$ receives an output $(C, \sigma_{\mathcal{M}}, \sigma_r)$ from $\mathcal{A}$, where $\sigma_{\mathcal{M}}$ was not created previously by $\mathcal{S}$, the simulation is aborted. In this case, $\mathcal{A}$ provided $\mathcal{S}$ with a forgery of a TPM signature. The probability $\varepsilon_{\mathsf{TPM}}$ of this event is the probability of a forgery of a TPM signature. Thus, $\mathsf{Succ}^{\mathsf{cf\text{-}priv}}(1^\kappa) \leq \Pr[\mathsf{win}_2] + \varepsilon_1 + \varepsilon_{\mathsf{TPM}}$.

$\mathbf{G}_1$ covers replay attacks by estimating the probability that the same nonce occurs twice, and $\mathbf{G}_2$ covers forgeries of TPM signatures. It remains to estimate the probability $\Pr[\mathsf{win}_2]$. We consider two cases: either $\mathcal{A}$ wins in $\mathbf{G}_2$ by forging the ring signature (with probability $\varepsilon_{\mathsf{ring}}$), or without it. Since we are interested in the overall probability of $\mathcal{A}$ winning in $\mathbf{G}_2$, we do not require from $\mathcal{S}$ to detect which of these distinct cases occurs.

If no forgery of the ring signature occurred, but $\mathcal{A}$ wins $\mathbf{G}_2$, $\mathcal{A}$ must know a secret key r' matching one of the public keys used to compute the ring signature. Hence, $\mathcal{A}$ must know r', such that $h^{r'} = C/g^{cs_j} = g^{cs_{\mathcal{P}} - cs_j} h^r \bmod P$ for some $j \in \{1, \ldots, n\}$. Because $cs_{\mathcal{P}} \neq cs_j$, we have $r \neq r'$, thus $\mathcal{A}$ could compute the discrete logarithm $\log_g(h) = (cs_{\mathcal{P}} - cs_j)/(r' - r) \bmod Q$. The probability of the adversary to win the last game is $\Pr[\mathsf{win}_2] = \varepsilon_{\mathsf{ring}} + (1 - \varepsilon_{\mathsf{ring}}) \cdot \varepsilon_{\mathsf{dlog}} \leq \varepsilon_{\mathsf{ring}} + \varepsilon_{\mathsf{dlog}}$, where $\varepsilon_{\mathsf{dlog}}$ is the probability to solve the underlying discrete logarithm problem.

Thus, in total, $\mathsf{Succ}^{\mathsf{cf\text{-}priv}}(1^\kappa) \leq \varepsilon_1 + \varepsilon_{\mathsf{TPM}} + \varepsilon_{\mathsf{ring}} + \varepsilon_{\mathsf{dlog}}$, which is negligible in κ if the TPM signature and ring signature schemes are secure and the underlying discrete logarithm problem is hard. □

Note that although our proof is in the standard model, the ring signature scheme in [1] requires the random oracle model.

Proof (Configuration Privacy). We demonstrate that $\mathsf{Adv}^{\mathsf{cf\text{-}priv}}(1^\kappa, n)$, the maximum advantage over all $\mathcal{A}$ in $\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf\text{-}priv}}$, is negligible in κ, even if the adversary is computationally unbounded. For this, we construct a simulator $\mathcal{S}$ that plays $\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf\text{-}priv}}$ with some $\mathcal{A}$, simulating the honest parties. The goal of $\mathcal{A}$ is to break the configuration privacy of the PBA scheme, and the simulator's goal is to break either the perfect hiding property of the commitment scheme or the unconditional signer ambiguity property of the ring signature scheme.

We play the game twice. In the first case, we assume that the ring signature is secure and show how $\mathcal{S}$ can break the commitment scheme. In the second case, we assume that the commitment scheme is secure, and hence, we show how $\mathcal{S}$ can break the ring signature scheme.

Case 1. In this case, $\mathcal{S}$ is given a commitment $C = g^{cs_{\mathcal{P}}} \cdot h^r \bmod P$ with $cs_{\mathcal{P}} \in CS$, and plays $\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf\text{-}priv}}$ with $\mathcal{A}$.

Once $\mathcal{S}$ receives a `send` query with a nonce N_v from $\mathcal{A}$, it uses C in the PBA protocol execution as the TPM's commitment (without knowing $cs_{\mathcal{P}}$ and r), and creates a TPM signature $\sigma_{\mathcal{M}} = \mathtt{SignM}(sk_{\mathcal{M}}; (C, N_v))$. The computationally unbounded simulator $\mathcal{S}$ can compute α, such that $h = g^\alpha \bmod P$, and k, such that $C = g^k = g^{cs_{\mathcal{P}} + \alpha r} \bmod P$. Although $\mathcal{S}$ knows neither $cs_{\mathcal{P}}$ nor r, it can establish n equations $k = cs_j + \alpha \cdot r_j$ (for $j = 1, \ldots, n$). Thus, $\mathcal{S}$ can compute n pairs (cs_j, r_j), and create the ring signature $\sigma_r = \mathtt{SigRing}(r_j; \{y_j\}; N_v)$, where $y_j = g^{\alpha r_j} = h^{r_j} \bmod P$, with any of these r_j as a signing key. Because of the signer ambiguity of the ring signature scheme, $\mathcal{S}$ can choose an arbitrary r_j (for $j \in_R \{1, \ldots, n\}$). $\mathcal{S}$ sends C, $\sigma_{\mathcal{M}}$, and σ_r to $\mathcal{A}$.

At the end of the game, $\mathcal{A}$ outputs an index i. $\mathcal{S}$ attacks the perfect hiding property of the commitment scheme by using the pairs (cs_j, r_j) computed above, and opening the commitment to (cs_i, r_i).

Because we assume that the ring signature is secure, the probability of $\mathcal{S}$ to break the commitment scheme successfully is the probability of $\mathcal{A}$ to determine i with $cs_i = cs_{\mathcal{P}}$. Thus, a non-negligible advantage $\mathsf{Adv}^{\mathsf{cf\text{-}priv}}$ implies that $\mathcal{S}$ can break the perfect hiding property.

Case 2. In this case, $\mathcal{S}$ is given public/private key pairs (y_j, x_j) $(j = 1, \ldots, n)$ for the ring signature scheme, and access to a signature oracle for ring signatures under this key ring. $\mathcal{S}$ can use the oracle to query ring signatures on arbitrary messages. The unconditional signer ambiguity states that $\mathcal{S}$ should not be able to find out which private key was used for signing (although $\mathcal{S}$ knows all public and private keys). $\mathcal{S}$ chooses $k \in_R \mathbb{Z}_Q$, and computes $cs_j = k - x_j \bmod Q$ for $j = 1, \ldots, n$. Then, $\mathcal{S}$ starts to play $\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf\text{-}priv}}$ with $\mathcal{A}$.

Once $\mathcal{S}$ receives a `send` query with a nonce N_v from $\mathcal{A}$, it computes $C := g^k \bmod P$ and $\sigma_{\mathcal{M}} := \mathtt{SignM}(sk_{\mathcal{M}}; (C, N_v))$. $\mathcal{S}$ uses the ring signature oracle to create a ring signature σ_r on the message N_v, and sends C, $\sigma_{\mathcal{M}}$, and σ_r to $\mathcal{A}$.

At the end of the game, $\mathcal{A}$ outputs an index i. Since the commitment C was chosen randomly, the only possibility of $\mathcal{A}$ to win $\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf\text{-}priv}}$ is to break the signer ambiguity of the ring signature. $\mathcal{S}$ also outputs i to indicate that x_i was used to generate the signature, thus breaking the unconditional signer ambiguity of the ring signature scheme. □