2011 Ninth Annual International Conference on Privacy, Security and Trust

UPBA: User-Authenticated Property-Based Attestation

Mark Manulis
Cryptographic Protocols Group
TU Darmstadt & CASED, Germany
Email: mark@manulis.eu

Marion Steiner
Cryptographic Protocols Group
TU Darmstadt & CASED, Germany
Email: marion.steiner@cased.de

***Abstract*—Remote attestation of computing platforms, using trusted hardware, guarantees the integrity, and by this the trustworthiness of a host to remote parties. While classical binary attestation attests the configuration itself, property-based attestation (PBA) attests properties and thus offers higher privacy guarantees to the host and its user. Nonetheless, both techniques are free from any user authentication mechanisms. Especially in distributed applications involving user interactions, the remote party may require assurance for the trustworthiness of the host and the authenticity of its user. Independence of user authentication from platform attestation may become an obstacle due to potential relay attacks. The *User-Authenticated Property-Based Attestation (UPBA)*, introduced in this work, can assure a remote party that some computing platform is trustworthy, and that it is used at that very moment by some particular user. Our basic protocol is secure and practical. We prove its security formally, discuss its compatibility with current trusted computing technology, and illustrate several nice enhancements.**

I. Introduction and Background

In modern computer systems secure interaction amongst different system components is inherent for ensuring the trustworthiness of the computing processes and distributed applications. The establishment of secure communication channels over possibly insecure networks can be easily done with modern technology based on traditional cryptographic authentication and key establishment techniques [7]. However, while securing the communication is feasible many modern threats in distributed systems and applications come from the insufficient protection of the communication end-points (hosts). Problems arise when two or more potentially distrusting communication partners, prior to their interaction and possible exchange of application information, wish to be convinced about the integrity or, more general, the trustworthiness of each other's computing platforms.

These problems can be addressed with the help of trusted computing technologies [26], [11] and, in particular, their platform attestation techniques. Such technology, as specified, e.g., in [28], uses trusted, tamper-proof hardware to measure and attest the entire configuration of one platform to its communication partner, who in turn can decide whether this configuration is admissible for the computing task or application, and either accept or refuse to communicate.

Available platform attestation techniques, see e.g. [15] for a recent survey (and also Section I-A), differ in offered privacy guarantees: for example, *binary attestation* [28] attests some specific configuration of the platform (e.g. its operation system), whereas *property-based attestation* (PBA) [20], [22] assumes mappings from concrete configurations to some more general properties (e.g. availability of a VPN connection) that can be satisfied by multiple configurations. This latter attestation type is more privacy-friendly with respect to the owner of the platform as it hides the actual configuration of the platform (behind the property), and can, thus, prevent discrimination of certain configurations.

Nonetheless, even PBA may have its deficiencies, when it comes to a deployment in practice, due to the following observation: In modern distributed systems and, especially, in distributed applications, the attestor (verifier) should be convinced not only about the trustworthiness of the host, but also about the trustworthiness of the user, who controls the host, and, more importantly, that this user is indeed using that particular host. However, current attestation techniques do not include user authentication mechanisms, which are usually treated independently at some higher level subsequent to the attestation process. While this is satisfying for inter-machine communications, problems may arise when user interaction is involved, as the relay attack suggests [27]: A session is established between a verifier and a trustworthy host after the successful platform attestation process. This session is then hijacked and redirected at application level to some non-trustworthy host, where the user authentication is performed. From this point, communication and user interaction take place between the non-trustworthy host and the verifier, therefore the attested properties do not apply anymore. This attack can be avoided by binding user authentication to the platform attestation process and, possibly, enriching the latter with the establishment of secure session keys for subsequent communication between the host and its verifier.

In this paper we show, how to achieve such binding with current technologies by extending the concept of PBA from [20], [22] towards *user-authenticated PBA (UPBA)*. We address the problem from both the practical and formal perspectives: First, we design an UPBA protocol maintaining a high level of modularity, that implicitly authenticates users as part of the platform attestation step, and, in addition, establishes a secure session key between the host and the verifier. We also discuss the compatibility of the protocol with available trusted computing technology. Second, we define an appropriate syn-

978-1-4577-0584-7/11/$26.00

tax for such UPBA protocols, specify their security model, and give a formal proof of security for our solution. Our UPBA model and protocol can be seen as modular extensions of the state-of-the-art PBA solution from [9]. For simplicity, our UPBA protocol and its security definitions assume a *single-user ownership* setting, where a particular device is used and controlled by at most one user. We discuss the limitations of this setting and highlight main challenges and obstacles that arise for UPBA in a *multi-user ownership* model, where devices can be shared amongst users.

A. Related Work on Platform Attestation

Binary attestation. The Trusted Platform Module (TPM) [28], specified by the Trusted Computing Group (TCG), provides functionality for trusted computing and platform attestation. This module, which is built into some common available computing platforms, can measure the software configuration of a platform on binary level. In particular, a hash value of the platform's state is computed by the TPM during the boot process and stored in its tamper-proof Platform Configuration Registers (PCR). This hash value represents thus the *configuration* of the platform. Of special interest for integrity checks and reports is the functionality called *binary attestation*, which allows some remote party (verifier) to obtain an authentic report about the configuration of the proving platform (prover), whereby the authenticity is provided by the TPM on the prover's side.

Binary attestation, however, is known to have some deficiencies with regard to the privacy of attested platforms: First, the verifier or any other party observing the attestation process can link different attestation attempts of the same platform (as the configuration is signed with a platform specific attestation identity key), and thus misuse this information, e.g. for profiling. One solution is given by Privacy CA [28], which resembles a trusted third party certifying new attestation keys chosen by the TPM. A relief from requiring a trusted third party for this process has been given by Brickell et al. [3] in form of the Direct Anonymous Attestation (DAA) protocol based on techniques that are similar to group signatures [8], [2]. DAA has been subsequently improved and extended in [4], [16], [6]. While our work does not primarily deal with the unlinkability of platform attestations, the DAA scheme can be used in our approach for the very same task. The second deficiency, observed e.g. in [20], [22] is that the verifier typically learns all information about the prover's platform configuration, not only which software is used, but possibly in which version, since each particular hash code references to a certain piece of software. This is more information than one may be willing to reveal, and can, in fact, lead to discrimination of certain platforms, e.g. by imposing different pricing policies or by simply blocking access. The third shortcoming is that the attestation protocol bears a risk of relay attacks, by which a third party acting as man-in-the-middle can forward messages between the verifier and the measured platform. This limitation has been informally addressed in [27] with the establishing of session keys within the attestation protocol.

Property-based attestation (PBA). PBA discovered in [20], [22] and improved and realized in [14], [9] (see [18] for a categorization and further challenges) addresses the second shortcoming of the binary attestation process, as mentioned above, by hiding particular configurations of attested platforms behind (certified) properties. The PBA approach assumes that each configuration (e.g. an operation system) may have several properties and that some particular property is likely to be satisfied by multiple configurations. Hence, attesting these properties, rather than concrete configurations, may suffice for many practical applications. Earlier PBA schemes such as [22], [14] required *property certifiers*, that is additional trusted parties (akin to Privacy CA) that were in charge of certifying mappings between admissible configurations and their properties. Later schemes, e.g. [9], do not have this limitation, and even allow hosts and verifiers to negotiate admissible configurations prior to the execution of the protocol. Nonetheless, existing PBA schemes, including [9], are loose from any user authentication processes, and are, thus, susceptible to the mentioned relay attacks.

Other attestation approaches. Sailer et al. [24] extended TCG binary attestation to allow the verifier to gain assurance about the integrity of Linux applications and system components at runtime. More fine-grained form of binary attestation for the portions of a code can be performed using the BIND approach [25]. Some attestation approaches abstract the attestation from physical platforms to the higher level of virtual machines [13], [23] or a more general notion of trusted virtual domains (TVD) [5], [12]. A comprehensive survey of attestation techniques, including shortcomings and potential solutions, can be found in the recent report by Lee-Thorp [15].

B. Organization

In Section II we define the system model and syntax for UPBA protocols. In Section III we present our UPBA protocol, analyze its complexity, and show how to realize the protocol using available technology. In Section IV we present our UPBA security model and define main security requirements. Finally, we conclude in Section VI.

II. UPBA System Model

In this section we describe the system model underlying UPBA protocols, which also serves as a basis for the UPBA security model we present in Section IV.

Protocol participants. As depicted on Figure 1, an UPBA scheme involves two participants: a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. The role of the prover is split into two parts: a host $\mathcal{H}$ and its built-in TPM $\mathcal{M}$. We assume that each $\mathcal{M}$ implicitly identifies its host and that there is at most one host per TPM. We work in the single-user ownership setting and thus assume that each $\mathcal{H}$ is controlled by at most one user $\mathcal{U}$. This means that we neither model nor consider $\mathcal{U}$ as a separate protocol participantas would be required in mult-users scenarios, as discussed in Section V.

Our model considers multiple UPBA sessions. We model them through multiple *instances* of participants and distinguish

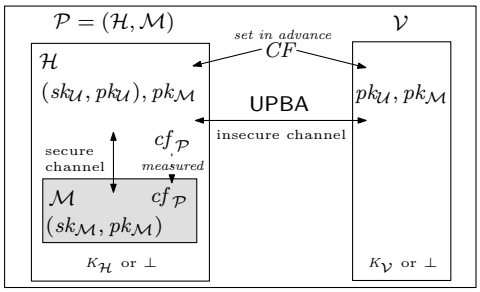

Fig. 1. UPBA system model

them using unique *session ids* sid. In particular, two instances of $\mathcal{P}$ and $\mathcal{V}$ are treated as participants of the same protocol session if their session ids match.

Communication model. For a given prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$, we assume that communication between $\mathcal{H}$ and $\mathcal{M}$ is performed over a secure channel. Messages generated by $\mathcal{M}$ for the corresponding verifier $\mathcal{V}$ and vice versa are delivered through the host $\mathcal{H}$. As becoming clear in our security model in Section IV, the only way for an adversary to directly communicate with the TPM is by corrupting the host.

Secret keys and trust assumptions. Since our goal is to bind user authentication to PBA we assume each user $\mathcal{U}$ is in possession of a private/public key pair $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$. Given that we work in a simplified single-user ownership setting, where hosts represent their (unique) users, the secret key $sk_{\mathcal{U}}$ is assumed to be stored at $\mathcal{H}$. The public key $pk_{\mathcal{U}}$ is assumed to be known to the corresponding verifier $\mathcal{V}$.

Additionally, UPBA protocols rely on TPMs, which are assumed to be trusted by all parties. Each TPM $\mathcal{M}$ is assumed to be in possession of a secret (signing) key $sk_{\mathcal{M}}$ which is internal to that TPM. In contrast, the corresponding public (verification) key $pk_{\mathcal{M}}$ is assumed to be known to its host $\mathcal{H}$ and the corresponding verifier $\mathcal{V}$.

Properties and configurations. Each prover $\mathcal{P}$ has a configuration value denoted $cf_{\mathcal{P}}$, which is an authenticated record of its host's configuration. The value $cf_{\mathcal{P}}$ is known to both the host $\mathcal{H}$ and the 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 technology. It is assumed that before running the UPBA protocol, $\mathcal{P}$ and $\mathcal{V}$ have already agreed on a set of admissible configuration values, denoted $CF = \{cf_1, \ldots, cf_n\}$, that satisfy some predefined property. So, we say that a configuration value $cf_{\mathcal{P}}$ satisfies a given property associated with CF, if and only if $cf_{\mathcal{P}} \in CF$.

DEFINITION 1 (USER-AUTHENTICATED PBA): A *user-authenticated property-based attestation (UPBA) scheme* $(\mathsf{Setup}, \mathsf{UPBA})$ consists of the following algorithms and protocols:

$\mathsf{Setup}(1^\kappa)$: On input a security parameter 1^κ this probabilistic algorithm outputs the list of admissible configuration values $CF = \{cf_1, \ldots, cf_n\}$ and generates public/private key pairs $(sk_{\mathcal{M}}, pk_{\mathcal{M}})$ for each TPM $\mathcal{M}$ and $(sk_{\mathcal{U}}, pk_{\mathcal{U}})$ for every user $\mathcal{U}$.

$\mathsf{UPBA}(\mathcal{P}, \mathcal{V})$: This is a probabilistic protocol between the instances of a prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ and a verifier $\mathcal{V}$ with matching session ids sid. The inputs of the prover consist of $(sk_{\mathcal{U}}, pk_{\mathcal{U}}, pk_{\mathcal{M}}, cf_{\mathcal{P}}, CF)$ for the host $\mathcal{H}$ and $(sk_{\mathcal{M}}, pk_{\mathcal{M}}, cf_{\mathcal{P}})$ for its built-in TPM $\mathcal{M}$. The input of the verifier contains $(pk_{\mathcal{M}}, pk_{\mathcal{U}}, CF)$. Finishing the execution UPBA outputs either the session keys $K_{\mathcal{H}}$ to $\mathcal{H}$ and $K_{\mathcal{V}}$ to $\mathcal{V}$ indicating the execution was successful or $\perp$.

DEFINITION 2 (CORRECTNESS OF UPBA): An UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ is *correct* if for all outputs $(sk_{\mathcal{M}}, pk_{\mathcal{M}}, sk_{\mathcal{U}}, pk_{\mathcal{U}}, CF) \leftarrow \mathsf{Setup}(1^\kappa)$ the execution of $\mathsf{UPBA}(\mathcal{P}, \mathcal{V})$ for $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ on the corresponding inputs $(sk_{\mathcal{U}}, pk_{\mathcal{U}}, pk_{\mathcal{M}}, cf_{\mathcal{P}}, CF)$ to $\mathcal{H}$, $(sk_{\mathcal{M}}, pk_{\mathcal{M}}, cf_{\mathcal{P}})$ to $\mathcal{M}$, and $(pk_{\mathcal{M}}, pk_{\mathcal{U}}, CF)$ to $\mathcal{V}$ results in two instances of $\mathcal{P}$ and $\mathcal{V}$ with matching session ids such that if $cf_{\mathcal{P}} \in CF$ then $\mathcal{H}$ computes $K_{\mathcal{H}}$, $\mathcal{V}$ computes $K_{\mathcal{V}}$, and $K_{\mathcal{H}} = K_{\mathcal{V}}$.

III. OUR UPBA SCHEME

A. Notations and Building Blocks

By $\mathbb{G} := \langle g \rangle$ we denote a cyclic group of prime order Q of length κ and generator g. We assume that the classical *Decision Diffie-Hellman (DDH)* problem in $\mathbb{G}$ is hard and denote by $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(1^\kappa)$ the maximal advantage probability for distinguishing DDH tuples (g, g^a, g^b, g^{ab}) from random tuples (g, g^a, g^b, g^c) where $a, b, c \in \mathbb{Z}^*_Q$, which is assumed to be negligible.

TPM and Host Signatures. Let $\Sigma := (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ be a digital signature scheme, where $\mathsf{KGen}(1^\kappa)$ generates a key pair (sk, pk), $\mathsf{Sig}(sk, m)$ outputs a signature σ on a message m, and $\mathsf{Ver}(pk, \sigma, m)$ outputs 1 if σ is a valid signature on m and 0 otherwise. We assume that Σ is existentially unforgeable under chosen message attacks (EUF-CMA) and denote by $\mathsf{Succ}^{\mathsf{uf-cma}}_{\Sigma}(1^\kappa)$ the maximal success probability for forging signatures, which is assumed to be negligible.

In our UPBA scheme, Σ will be used by both hosts and their built-in TPMs for signing certain protocol messages. We do not distinguish between the signature schemes used by these entities since we only expect these schemes to satisfy the minimal security requirement of EUF-CMA security. In practice, however, existing TCG technology defines two options by which TPMs can sign, the DAA scheme [3], which ensures unforgeability of TPM signatures and hides the identity of the TPM, or some ordinary signature scheme (typically RSA signature) and an attestation identity key for signing, chosen by the TPM.

Commitment Scheme Our UPBA scheme uses the general version of Pedersen commitments [19], which we denote $\mathtt{COM}$. Working in the cyclic group $\mathbb{G} := \langle g \rangle$ of prime order Q we denote by $h \in_R \mathbb{G}$ an additional generator $\mathbb{G}$, chosen at random such that $\log_g(h)$ is unknown. Then, (g, h) constitutes the public key of the scheme. In order to commit to some message $m \in \mathbb{Z}_Q$, the committer sends $C := g^m h^r$ to

the verifier using some uniformly chosen $r \in_R \mathbb{Z}_Q$. By $\mathsf{Adv}^{\mathsf{hide}}_{\mathtt{COM}}(1^\kappa)$ we denote the probability for breaking the hiding property of $\mathtt{COM}$ and by $\mathsf{Succ}^{\mathsf{bind}}_{\mathtt{COM}}(1^\kappa)$ for breaking its binding property. It is well known that Pedersen commitments provide perfectly hiding and computationally binding, assuming the hardness of the Discrete Logarithm problem in $\mathbb{G}$.

In our UPBA protocol, the TPM will commit to the configuration value $cf_{\mathcal{P}}$ and output the corresponding commitment C to the host together with the secret randomness r being used.

Ring Signature. A ring signature scheme $\mathtt{RS} := (\mathsf{RGen}, \mathsf{RSig}, \mathsf{RVer})$ has been introduced by Rivest et al. [21] as a privacy-preserving primitive that allows the signer to create a signature with respect to a set of public keys such that its successful verification convinces a verifier that a private key corresponding to one of the public keys has been used, yet without disclosing which one.

Similar to [9], our UPBA scheme cannot use an arbitrary ring signature scheme. Due to the design of our protocol we need a ring signature scheme that allows to extract a set of valid public keys out of the generalized Pedersen commitment C and use the corresponding randomness r as a secret signing key. Fortunately, the efficient scheme by Abe et al. [1], which we recall in a generalized version, satisfies our goals. The scheme works in the same group $\mathbb{G} := \langle g \rangle$ of primer order Q as the commitment scheme $\mathtt{COM}$ and uses a hash function $H : \{0,1\}^* \mapsto \mathbb{Z}_Q$ modeled as a random oracle.

$\mathsf{RGen}(1^\kappa)$: On input 1^κ this key generation algorithm outputs a private/public key pair (x_i, y_i) with $x_i \in_R \mathbb{Z}^*_Q$ and $y_i := g^{x_i}$ for each signer $\mathcal{U}_i$, $i = 1, \ldots, n$.

$\mathsf{RSig}(x_i, Y, m)$: On input the secret key x_i, the set of public keys $Y = (y_1, ..., y_n)$, and a message $m \in \{0,1\}^*$ the ring signature σ_R is computed as follows:

- $\alpha, c_j \in_R \mathbb{Z}^*_Q$ for $j = 1, \ldots, n$, $i \neq j$.
- $z := g^\alpha \prod_{j=1, j\neq i}^{n} y_j^{c_j}$.
- $c := H(g, Q, Y, z, m)$.
- $c_j := c - (c_1 + ... + c_{i-1} + c_{i+1} + ... + c_n) \bmod Q$.
- $s := \alpha - c_i \cdot x_i \bmod Q$.

The algorithm outputs $\sigma_R := (s, c_1, ..., c_n)$.

$\mathsf{RVer}(Y, \sigma_R, m)$: On input a set of public keys $Y = (y_1, \ldots, y_n)$, a candidate ring signature $\sigma_R = (s, c_1, ..., c_n)$, and a message m this algorithm outputs 1 if $\sum_{i=1}^{n} c_i \equiv H(g, Q, Y, g^s y_1^{c_1} \cdots y_n^{c_n}, m) \bmod Q$ and 0 otherwise.

By $\mathsf{Succ}^{\mathsf{uf-cma}}_{\mathtt{RS}}(1^\kappa)$ we denote the maximal success probability for forging the scheme under chosen message attacks that has been proven negligible in [1]. By $\mathsf{Succ}^{\mathsf{uf-cma}}_{\mathtt{RS}}(1^\kappa)$ we denote the maximal advantage probability for breaking the anonymity property of the scheme, which has been proven to be 0 (i.e. unconditional) in [1].

Intuitively, the ring signature scheme will be used in our UPBA protocol for ensuring the property attestation and configuration privacy requirements.

Key derivation function. In our UPBA protocol session keys will be derived from Diffie-Hellman keys, which are elements of $\mathbb{G}$, using some appropriate key derivation function $\mathsf{KDF} : \{0,1\}^* \mapsto \{0,1\}^\kappa$, which can be realized using some hash function or a randomness extractor [10]. By $\mathsf{Adv}_{\mathtt{KDF}}(1^\kappa)$ we denote the maximum advantage probability of distinguishing the outputs of KDF from uniformly sampled values in $\{0,1\}^\kappa$ and assume that this probability is negligible.

B. Basic UPBA Scheme

Our basic UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ consists of the following algorithms and protocols. We assume that the description of a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order Q as well as the public key (g, h) used in the Pedersen commitment scheme is an implicit input to all algorithms and entities participating in the protocol.

$\mathsf{Setup}(1^\kappa)$: This algorithm outputs for every TPM $\mathcal{M}$ a signature/verification key pair $(sk_{\mathcal{M}}, pk_{\mathcal{M}}) \leftarrow \mathsf{KGen}(1^\kappa)$, for every user $\mathcal{U}$ a signature/verification key pair $(sk_{\mathcal{U}}, pk_{\mathcal{U}}) \leftarrow \mathsf{KGen}(1^\kappa)$, which is stored in the corresponding host $\mathcal{H}$, and a set of admissible configuration values $CF = (cf_1, \ldots, cf_n) \in \mathbb{Z}^n_Q$.

Note that configuration values are assumed to be elements of $\mathbb{Z}_Q$, which can be realized by hashing the configurations using an appropriate hash function.

$\mathsf{UPBA}(\mathcal{P}, \mathcal{V})$: The description of the UPBA protocol is given in Figure 2. We assume that for a given prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ the corresponding configuration value $cf_{\mathcal{P}}$ has been measured by $\mathcal{M}$.

The verifier $\mathcal{V}$ picks a random exponent $x_{\mathcal{V}}$ which serves two purposes: on the one hand it is seen as an exponent for the computation of the Diffie-Hellman key with the host; on the other hand, its public version $g^{x_{\mathcal{V}}}$ also becomes part of the session id sid, that is then signed by the parties and is thus, seen as a nonce protecting against replay attacks. Similar purpose have also the exponent $x_{\mathcal{H}}$ and its public version $g^{x_{\mathcal{H}}}$, chosen and communicated by the host.

The TPM $\mathcal{M}$ generates a signature $\sigma_{\mathcal{M}}$ on a Pedersen commitment C to the configuration value $cf_{\mathcal{P}}$, whereby the session id sid is included into the signed message. $\mathcal{M}$ then discloses C and used randomness r. By dividing C with g^{cf_j} for each $cf_j \in CF$, $j = 1, \ldots, n$, the host obtains a set of public keys $Y = (y_1, \ldots, y_n)$ that will be used to build the ring signature σ_R. Note that one public key in Y, namely the one which corresponds to the operation $C/g^{cf_{\mathcal{P}}}$, has the form h^r, whereby all other public keys are randomly distributed in $\mathbb{G}$. That is, the host can effectively compute a ring signature σ_R using the received randomness r as the secret signing key. In order to authenticate the user $\mathcal{U}$, the host also computes a digital signature $\sigma_{\mathcal{U}}$ on the session id and outputs the session key $K_{\mathcal{H}}$ derived using the key derivation function KDF, after sending $(g^{x_{\mathcal{H}}}, C, \sigma_{\mathcal{M}}, \sigma_{\mathcal{U}}, \sigma_R)$ to the verifier.

$\mathcal{V}$ checks whether all received signatures are valid. In particular, for the verification of the ring signature σ_R, $\mathcal{V}$ has to extract public keys Y using the set CF of admissible configuration values. If at least one signature verification fails then $\mathcal{V}$ aborts the protocol execution and outputs $\perp$.

Otherwise, it computes the session key $K_{\mathcal{V}}$ derived using the key derivation function KDF. Observe that $K_{\mathcal{H}} = K_{\mathcal{V}}$ holds if both $\mathcal{H}$ and $\mathcal{V}$ compute the same Diffie-Hellman key $g^{x_{\mathcal{H}} x_{\mathcal{V}}}$ and use matching session ids $sid_{\mathcal{H}} = sid_{\mathcal{V}}$.

UPBA in its basic form is not concerned with authentication of verifiers. Such extensions could be useful in practice, and are discussed in the full version [17].

C. Properties of the Basic Scheme

Efficiency and comparison. Concerning efficiency, our solution requires almost no additional costs on TPM side, i.e. one multiexponentiation more compared to [28]. Additional costs at $\mathcal{H}$ are dominated by the extraction of public keys $(y_1, \ldots, y_n)$, which involves n modular exponentiations and inversions. Note that the actual computation of the ring signature σ_R, which is due to [1], requires a linear amount of single-exponentiations in n, but can be computed more efficiently through one multi-exponentiation with $n+1$ exponents. The verifier has essentially the same costs as the prover. Further, we observe that in comparison to the PBA scheme from [9], which serves as a basis for our UPBA, the overhead at the host's side amounts to two modular exponentiations for the computation of the Diffie-Hellman value $g^{x_{\mathcal{V}} x_{\mathcal{P}}}$ and the generation of $\sigma_{\mathcal{H}}$, whereas at the verifier's side to the additional verification of $\sigma_{\mathcal{H}}$. The costs for KDF are negligible. Finally, notice that in practice Q can be a prime number of 160 to 180 bits, which implies the same size for the configuration values in CF.

No trusted third parties. Similar to the PBA approach from [9], and unlike earlier ones in [22], [14], our UPBA protocol does not require any trusted property certifier for the set CF, although may have one, if needed.

Compatibility with TCG technology. Our proposed UPBA protocol is practical and realizable with current TPM standards and functionality [28]. In particular, TPMs are already equipped with random number generators and have the ability to perform modular exponentiations, as well as generate signatures. Regarding the computation of $\sigma_{\mathcal{M}} := \mathsf{Sig}(sk_{\mathcal{M}}, C|sid_{\mathcal{H}})$ observe, that in practice, the TPM will rather sign the hash of $C|sid_{\mathcal{H}}$. We omit this implicit hashing in our protocol, but remark that this corresponds to the standard hash-and-sign approach, which is known to offer sufficient security.

IV. Security Model and Analysis

A. UPBA Security Model

Adversary model. The adversary $\mathcal{A}$, modeled as a PPT machine, has control over different communication links between the parties involved in the attestation protocol and is also able to corrupt selected parties and reveal their secrets. In particular, $\mathcal{A}$ interacts with the parties via following set of queries:

- $\texttt{send}(E, sid, m)$ with $E \in \{\mathcal{H}, \mathcal{V}\}$: $\mathcal{A}$ can send a direct message m to an entity E in session sid. In response, $\mathcal{A}$ is given the message generated by E (if any).
- $\texttt{sendTPM}(\mathcal{M}, m)$: $\mathcal{A}$ can send direct messages to a TPM $\mathcal{M}$. This query can only be asked after $\mathcal{A}$ queried $\texttt{corrupt}(\mathcal{H})$ for the host $\mathcal{H}$ of that TPM; note that each $\mathcal{M}$ implicitly identifies its host. With this restriction we model the fact that in order to communicate with the built-in TPM directly, the adversary must take control over the host.
- $\texttt{corrupt}(E)$ with $E \in \{\mathcal{H}, \mathcal{V}\}$: $\mathcal{A}$ is able to compromise some entity E and in response obtain all secrets stored in E. This query also gives $\mathcal{A}$ the ability to impersonate E. Note that we do not allow corruptions of TPMs due to their tamper-resistance property [28].

An UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ should satisfy four security requirements that address various intuitive goals regarding property-based attestation, privacy of configurations, authentication of the users, and the establishment of secure session keys, specified in the following.

Property Attestation. This requirement aims at attesting that the configuration $cf_{\mathcal{P}}$ of prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ satisfies some predefined property. Informally, an adversary $\mathcal{A}$ should not be able to maliciously convince an honest verifier $\mathcal{V}$ that the prover's configuration $cf_{\mathcal{P}}$ is amongst the admissible set CF when in fact $cf_{\mathcal{P}} \notin CF$. We formalize this requirement in a setting, where $\mathcal{A}$ can corrupt the host and change its configuration, which captures potential malware attacks as well as malicious user activities.

Definition 3 (UPBA Property Attestation): Let $(\mathsf{Setup}, \mathsf{UPBA})$ be an UPBA scheme. Let $\mathsf{Game}_{\mathcal{A}}^{\mathsf{pr-att}}(1^\kappa)$ be the following attack game with an adversary $\mathcal{A}$: After execution of $\mathsf{Setup}(1^\kappa)$, $\mathcal{A}$ first picks prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ and some configuration $cf_{\mathcal{P}} \notin CF$, which then serves as input to $\mathcal{M}$ in all UPBA sessions involving chosen $\mathcal{P}$. Then, $\mathcal{A}$ is given access to the queries of the form $\texttt{send}$, $\texttt{sendTPM}$, and $\texttt{corrupt}$. $\mathcal{A}$ interacts with the parties using these queries until it stops.

$\mathcal{A}$ wins, denoted by $\mathsf{Game}_{\mathcal{A}}^{\mathsf{pr-att}}(1^\kappa) = 1$, if at the end of the game all of the following holds:

- There exists a verifier $\mathcal{V}$ that on input CF has computed the key $K_{\mathcal{V}}$ in the execution of some UPBA session sid with $\mathcal{P}$.
- There has been no $\texttt{corrupt}(\mathcal{V})$ query, i.e. the mentioned verifier remained uncorrupted.

Let $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{pr-att}}(1^\kappa) := \Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{pr-att}}(1^\kappa) = 1]$ denote the success probability of $\mathcal{A}$ and $\mathsf{Succ}^{\mathsf{pr-att}}(1^\kappa)$ its maximum over all PPT adversaries $\mathcal{A}$ (running in time polynomial in κ). The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides *property attestation* if $\mathsf{Succ}^{\mathsf{pr-att}}(1^\kappa)$ is negligible in κ.

Configuration privacy. The requirement of configuration privacy aims at hiding the actual configuration $cf_{\mathcal{P}}$ of the prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ from malicious verifiers. The underlying assumption is that some property of the platform can be satisfied by multiple, different configurations, which are all part of the admissible set CF. Hence, the goal of configuration privacy is to prevent the adversary from being able to decide, which configuration $cf_{\mathcal{P}} \in CF$ has been used by $\mathcal{P}$ in a successful execution of the protocol. We model this property using the standard indistinguishability approach, by consid-

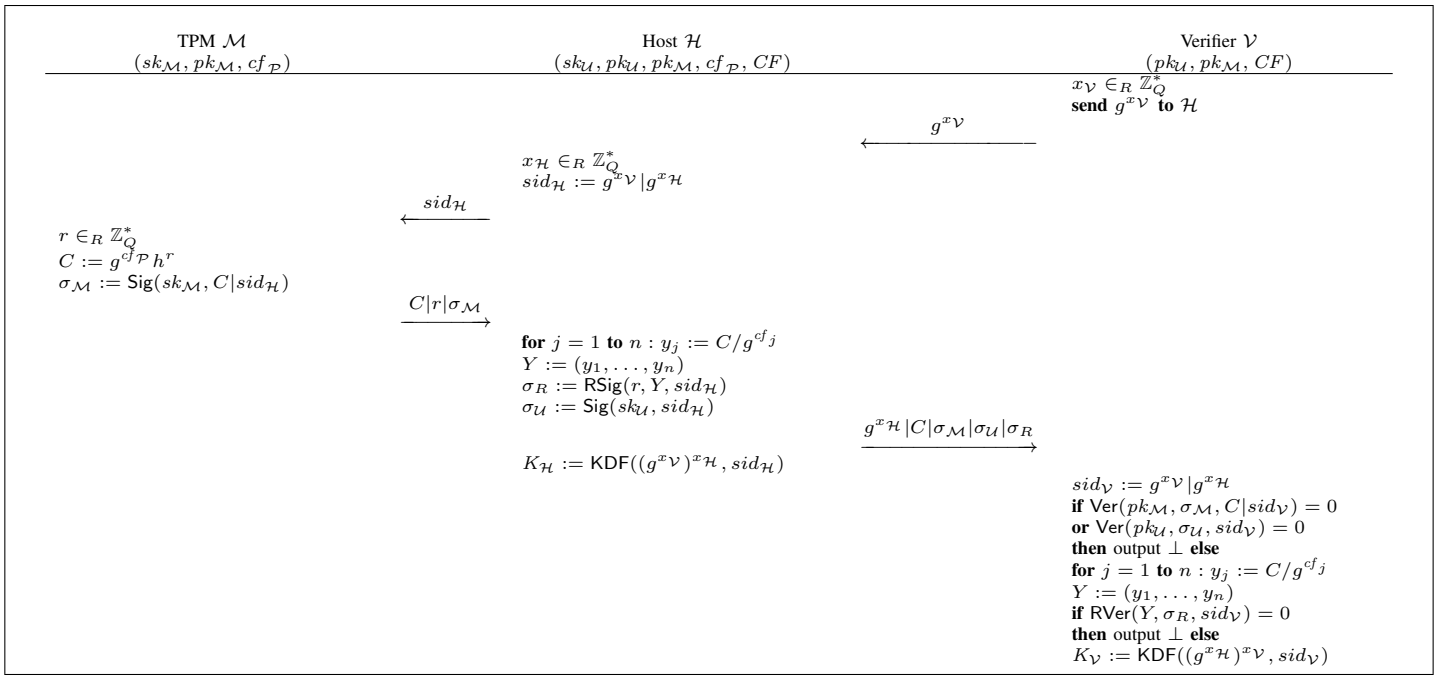

Fig. 2. UPBA protocol session between prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ and verifier $\mathcal{V}$ with $CF = \{cf_1, \ldots, cf_n\}$

ering two different configurations $cf_0, cf_1 \in CF$ chosen by $\mathcal{A}$. The goal of $\mathcal{A}$ is to decide, which configuration has been used in an execution of the UPBA protocol, in which $\mathcal{A}$ is allowed to play the role of the malicious verifier. Since $cf_{\mathcal{P}}$ is implicitly known to both the corresponding TPM $\mathcal{M}$ and the host $\mathcal{H}$, the definition prevents $\mathcal{A}$ from corrupting that host.

DEFINITION 4 (UPBA CONFIGURATION PRIVACY): Let $(\mathsf{Setup}, \mathsf{UPBA})$ be an UPBA scheme. Let $\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf-pri}}(1^\kappa)$ be the following attack game with an adversary $\mathcal{A}$: After execution of $\mathsf{Setup}(1^\kappa)$ $\mathcal{A}$ is given access to the queries of the form `send`, `sendTPM`, and `corrupt`. At some stage $\mathcal{A}$ picks $(\mathcal{P}, cf_0, cf_1)$ such that $cf_0, cf_1 \in CF$ (where CF has been returned by Setup). Then, a bit $b \in_R \{0,1\}$ is chosen uniformly and an UPBA session sid is executed on behalf of selected $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ with $cf_{\mathcal{P}} := cf_b$ being the configuration value input to $\mathcal{H}$ and $\mathcal{M}$. Finally, $\mathcal{A}$ outputs b'.

$\mathcal{A}$ wins, denoted $\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf-pri}}(1^\kappa) = 1$, if at the end of the game all of the following holds:

- $b = b'$.
- There has been no $\texttt{corrupt}(\mathcal{H})$ query, i.e. the target host remained uncorrupted.

Let $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{cf-pri}}(1^\kappa) := |\Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{cf-pri}}(1^\kappa) = 1] - \frac{1}{2}|$ denote the advantage probability of $\mathcal{A}$ (over a random guess) and $\mathsf{Adv}^{\mathsf{cf-pri}}(1^\kappa)$ its maximum over all PPT adversaries $\mathcal{A}$ (running in time polynomial in κ). The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides *configuration privacy* if $\mathsf{Adv}^{\mathsf{cf-pri}}(1^\kappa)$ is negligible in κ.

User Authentication. The requirement of user authentication aims at ensuring that some particular user $\mathcal{U}$ is involved in the attestation protocol. It can be seen as dual to the property attestation, which attests the configuration of the platform but fails to provide the binding to a particular user. A protocol which achieves both requirements effectively binds session establishment to some particular device and its user. Informally, an adversary $\mathcal{A}$ should not be able to maliciously convince an honest verifier $\mathcal{V}$ that some UPBA protocol session between $\mathcal{P}$ and $\mathcal{V}$ is executed on behalf of some user $\mathcal{U}$. That is user authentication prevents $\mathcal{A}$ from impersonating the user. Since we work in the single-user ownership setting where user's authentication keys are typically stored on the corresponding host it is necessary to prevent $\mathcal{A}$ from obtaining these keys, modeled by prohibiting corruption of the host. We still allow $\mathcal{A}$ to initiate protocol sessions with $\mathcal{V}$ using other hosts and possibly knowing secret authentication keys of users behind those hosts.

DEFINITION 5 (UPBA USER AUTHENTICATION): Let $(\mathsf{Setup}, \mathsf{UPBA})$ be an UPBA scheme. Let $\mathsf{Game}_{\mathcal{A}}^{\mathsf{us-aut}}(1^\kappa)$ be the following attack game with an adversary $\mathcal{A}$: After execution of $\mathsf{Setup}(1^\kappa)$ $\mathcal{A}$ is given access to the queries of the form `send`, `sendTPM`, and `corrupt`. $\mathcal{A}$ interacts with the parties using these queries until it stops.

$\mathcal{A}$ wins, denoted by $\mathsf{Game}_{\mathcal{A}}^{\mathsf{us-aut}}(1^\kappa) = 1$, if at the end of the game all of the following holds:

- There exists a verifier $\mathcal{V}$ that has computed a key $K_{\mathcal{V}}$ in the execution of some UPBA session sid with prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ for which no matching session exists.
- There have been neither $\texttt{corrupt}(\mathcal{V})$ nor $\texttt{corrupt}(\mathcal{H})$ queries for the mentioned $\mathcal{V}$ and $\mathcal{H}$.

Let $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{us-aut}}(1^\kappa) := \Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{us-aut}}(1^\kappa) = 1]$ denote the success probability of $\mathcal{A}$ and $\mathsf{Succ}^{\mathsf{us-aut}}(1^\kappa)$ its maximum over all PPT adversaries $\mathcal{A}$ (running in time polynomial in κ). The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides *user authentication* if $\mathsf{Succ}^{\mathsf{us-aut}}(1^\kappa)$ is negligible in κ.

An UPBA protocol $(\mathsf{Setup}, \mathsf{UPBA})$, which simultaneously provides property attestation and user authentication, guarantees to an honest verifier $\mathcal{V}$, upon the successful computation of the session key $K_{\mathcal{V}}$, that the protocol session has been executed on a platform with an admissible configuration (which satisfies some predefined property) and that the platform belongs to a legitimate user.

Session key (SK) security. The requirement of session key (SK) security for an UPBA scheme is motivated by the fact that, in practice, users authenticate themselves to verifiers and then continue communicating with them. It is desirable to protect this later communication as well. The goal of SK security is to establish a secure session key between the protocol participants. We model SK-Security akin to [7] by providing the adversary $\mathcal{A}$ with two additional queries (see [17] for the detailed description):

- $\texttt{reveal}(E, sid)$ with $E \in \{\mathcal{H}, \mathcal{V}\}$: $\mathcal{A}$ can reveal session key K_E (if existing) computed by entity E in session sid.
- $\texttt{test}(E, sid)$ with $E \in \{\mathcal{H}, \mathcal{V}\}$: This query can be asked only once and only if entity E has computed key K_E in session sid. In this case a uniform bit $b \in_R \{0,1\}$ is flipped and if $b = 1$ then $\mathcal{A}$ receives K_E; otherwise$\mathcal{A}$ receives a uniformly chosen string $K \in_R \{0,1\}^\kappa$.

Taking into account that basic UPBA scheme ensures attestation and authentication of provers (and not of verifiers), we exclude trivial impersonation attacks on unauthenticated verifiers by restricting the $\texttt{send}(\mathcal{H}, sid, m)$ query: For UPBA sessions invoked between $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ and $\mathcal{V}$ the adversary $\mathcal{A}$ can query $\texttt{send}(\mathcal{H}, sid, m)$ only after having compromised $\mathcal{V}$ (by querying $\texttt{corrupt}(\mathcal{V})$ at some earlier stage).

DEFINITION 6 (UPBA SESSION KEY SECURITY): Let $(\mathsf{Setup}, \mathsf{UPBA})$ be an UPBA scheme. Let $\mathsf{Game}_{\mathcal{A}}^{\mathsf{sk-sec}}(1^\kappa)$ be the following attack game with an adversary $\mathcal{A}$: After execution of $\mathsf{Setup}(1^\kappa)$ $\mathcal{A}$ is given access to the queries of the form `send`, `sendTPM`, `corrupt`, `reveal`, and `test`. At some point $\mathcal{A}$ queries $\texttt{test}(E, sid)$ to an entity E that has computed the session key K_E and receives either K_E or some random string K (depending on the bit b). Then, $\mathcal{A}$ continues interacting with parties using the above queries until it outputs some bit b'.

$\mathcal{A}$ wins, denoted $\mathsf{Game}_{\mathcal{A}}^{\mathsf{sk-sec}}(1^\kappa) = 1$, if at the end of the game all of the following holds:

- $b' = b$ (i.e. $\mathcal{A}$ successfully guessed the bit chosen in response to the `test` query).
- There has been no $\texttt{reveal}(E, sid)$ query and no $\texttt{corrupt}(E)$ query prior to the computation of K_E.
- If $E = \mathcal{H}$ and there is a matching session sid for some verifier $\mathcal{V}$ then: There has been no $\texttt{reveal}(\mathcal{V})$ query and there has been no $\texttt{corrupt}(\mathcal{V})$ query prior to the computation of K_E.
- If $E = \mathcal{V}$ and there is a matching session sid for some prover $\mathcal{P} = (\mathcal{H}, \mathcal{M})$ then: There has been no $\texttt{reveal}(\mathcal{H})$ query and there has been no $\texttt{corrupt}(\mathcal{H})$ query prior to the computation of K_E.

Let $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{sk-sec}}(1^\kappa) := |\Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{sk-sec}}(1^\kappa) = 1] - \frac{1}{2}|$ denote the advantage probability of $\mathcal{A}$ (over a random guess) and $\mathsf{Adv}^{\mathsf{sk-sec}}(1^\kappa)$ its maximum over all PPT adversaries $\mathcal{A}$ (running in time polynomial in κ). The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides *SK-security* if $\mathsf{Adv}^{\mathsf{sk-sec}}(1^\kappa)$ is negligible in κ.

B. Security Analysis of Our UPBA Scheme

The following theorems show that our UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ introduced in Section III satisfies the defined security requirements of property attestation, configuration privacy, user authentication, and SK-security in the single-user ownership setting (proofs of these theorems are delegated to the full version [17]).

Theorem 1 (Property Attestation): The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides property attestation, assuming unforgeability of the signature scheme Σ, unforgeability of the ring signature scheme `RS`, and the binding property of the commitment scheme `COM`. Moreover,

$$\mathsf{Succ}^{\mathsf{pr-att}}(1^\kappa) \leq \frac{q^2}{2^\kappa} + \mathsf{Succ}_{\Sigma}^{\mathsf{uf-cma}}(1^\kappa) + \mathsf{Succ}_{\mathtt{RS}}^{\mathsf{uf-cma}}(1^\kappa) + \mathsf{Succ}_{\mathtt{COM}}^{\mathsf{bind}}(1^\kappa),$$

where q is the number of executed UPBA sessions.

Remark. Although not explicitly stated in Theorem 1, our proof inherits the random oracle assumption needed for the unforgeability of ring signatures scheme from [1].

Theorem 2 (Configuration Privacy): The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides configuration privacy against computationally unbounded adversaries, due to the unconditional anonymity of the ring signature scheme `RS` and perfect hiding of the commitment scheme `COM`, i.e. $\mathsf{Adv}^{\mathsf{cf-pri}}(1^\kappa) = 0$.

Theorem 3 (User Authentication): The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides user authentication, assuming unforgeability of the signature scheme Σ, i.e.

$$\mathsf{Succ}^{\mathsf{us-aut}}(1^\kappa) \leq \frac{q^2}{2^\kappa} + \mathsf{Succ}_{\Sigma}^{\mathsf{uf-cma}}(1^\kappa),$$

where q is the number of executed UPBA sessions.

Remark. The security of commitment scheme `COM` and ring signature scheme `RS` is not required to provide user authentication since authentication of the user is performed in a modular way via a separate signature scheme. However, since our UPBA scheme also provides property attestation, both these requirements together guarantee that the protocol was executed on a platform with an admissible property owned by a legitimate user.

Theorem 4 (SK-Security): The UPBA scheme $(\mathsf{Setup}, \mathsf{UPBA})$ provides SK-security, assuming the unforgeability of the signature scheme Σ, the hardness of the DDH problem in $\mathbb{G}$, and the pseudorandomness of the key derivation function KDF. Moreover,

$$\mathsf{Adv}^{\mathsf{sk-sec}}(1^\kappa) \leq \frac{q^2}{2^\kappa} + \mathsf{Succ}_{\Sigma}^{\mathsf{uf-cma}}(1^\kappa) + q\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(1^\kappa) + q\mathsf{Adv}_{\mathtt{KDF}}(1^\kappa),$$

where q is the number of executed UPBA sessions.

V. On UPBA in a Multi-User Ownership Model

Our UPBA security model and protocol are designed in a simplified ownership model, where each device (host) is controlled by at most one user and the secret key $sk_{\mathcal{U}}$ can be stored safely at the host. In a multi-user ownership model where one device is shared amongst several users this definition is not sufficient as some user may attempt to impersonate another user while using the device. A meaningful definition of user authentication for such multi-user setting would have to consider users as separate entities in addition to hosts, and allow adversary corruptions of hosts and all users except for the future victim. Clearly, our protocol would not fulfil this requirement as the adversary would be able to obtain $sk_{\mathcal{U}}$ upon corruption of $\mathcal{H}$. Therefore, one necessary condition in a multi-user setting is not to store secret authentication material persistently on the device, unless the device storage is trusted to restrict access to these secrets to the corresponding user $\mathcal{U}$ when using the device. An alternative would be to store secret keys at some other location (e.g. in an user's smart card or some electronic identity document) or to resort to password-based authentication techniques. Unfortunately, both approaches are still insufficient since corruption of hosts implicitly allows $\mathcal{A}$ to install malware which seems to be the main threat in the multi-user setting to capture authentication secrets related to other users. On the other hand, assuming a device is malware-free, even if users might be malicious, seems a too strong assumption.

Bottom line, while our security model can be easily extended to capture the multi-user ownership setting, the actual design of such UPBA protocols remains an open problem.

VI. Conclusion

Modern platform attestation techniques, aiming at assuring the trustworthiness of computing platforms, have shortcomings, when user interaction is involved, and trustworthiness of the user, in particular authenticity, should be guaranteed as well. Without a binding between attestation and authentication of users, relay attacks are possible. Focusing on the property-based attestation (PBA), we designed a user-authenticated version of PBA in a single-user ownership setting. In addition to giving a concrete UPBA protocol and evaluating its efficiency, we gave a formal security model for such schemes and used it to analyze our own construction. We suggested several extensions, aiming at stronger security and privacy properties. Note that ideas behind our UPBA scheme can also be applied to the traditional TCG's binary attestation process.

References

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n Signatures from a Variety of Keys. In *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 415–432. Springer, 2002.

[2] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.

[3] E. F. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *ACM CCS 2004*, pages 132–145. ACM, 2004.

[4] E. F. Brickell and J. Li. Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. In *ACM WPES 2007*, pages 21–30. ACM, 2007.

[5] A. Bussani, J. L. Grifin, B. Jasen, K. Julish, G. Karjot, H. Maruyama, M. Nakamura, R. Perez, M. Shunter, A. Tanner, L. V. Doorn, E. V. Herreweghen, M. Waidner, and S. Yoshihama. Trusted Virtual Domains: Secure Foundation for Business and IT services. Technical Report RC23793, IBM Research, 2005.

[6] J. Camenisch. Better Privacy for Trusted Computing Platforms. In *ESORICS 2004*, volume 3193 of *LNCS*, pages 72–88. Springer, 2004.

[7] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.

[8] D. Chaum and E. van Heyst. Group Signatures. In *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

[9] L. Chen, H. Löhr, M. Manulis, and A.-R. Sadeghi. Property-Based Attestation without a Trusted Third Party. In *ISC 2008*, volume 5222 of *LNCS*, pages 31–46. Springer, 2008.

[10] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin. Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 494–510. Springer, 2004.

[11] E. Gallery. An Overview of Trusted Computing Technology. In *Trusted Computing (Professional Applications of Computing)*. Chapter 3, IEEE Press, 2005.

[12] J. L. Griffin, T. Jaeger, R. Perez, R. Sailer, and L. van Doorn Ramón Cáceres and. Trusted Virtual Domains: Toward Secure Distributed Services. In *Hot Topics in System Dependability 2005*, 2005.

[13] V. Haldar, D. Chandra, and M. Franz. Semantic Remote Attestation – Virtual Machine Directed Approach to Trusted Computing. In *Virtual Machine Research and Technology Symposium*, pages 29–41. USENIX, 2004.

[14] U. Kühn, M. Selhorst, and C. Stüble. Realizing Property-Based Attestation and Sealing with Commonly Available Hard- and Software. In *ACM STC 2007*, pages 50–57. ACM, 2007.

[15] A. Lee-Thorp. Attestation in Trusted Computing: Challenges and Potential Solutions. Technical Report RHUL-MA-2010-09, Royal Holloway, University of London, 2010.

[16] A. Leung, L. Chen, and C. J. Mitchell. On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA). In *TRUST 2008*, volume 4968 of *LNCS*, pages 179–190. Springer, 2008.

[17] M. Manulis and M. Steiner. UPBA: User-Authenticated Property-Based Attestation, PST 2011. Full Version.

[18] A. Nagarajan, V. Varadharajan, M. Hitchens, and E. Gallery. Property Based Attestation and Trusted Computing: Analysis and Challenges. In *NSS 2009*, pages 278–285. IEEE CS, 2009.

[19] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO 1991*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

[20] J. Poritz, M. Schunter, E. V. Herreweghen, and M. Waidner. Property Attestation—Scalable and Privacy-friendly Security Assessment of Peer Computers. Technical Report 3548, IBM Research Zurich, 2004.

[21] R. L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.

[22] A.-R. Sadeghi and C. Stüble. Property-Based Attestation for Computing Platforms: Caring about Properties, not Mechanisms. In *ACM NSPW 2004*, pages 67–77. ACM, 2004.

[23] A.-R. Sadeghi, C. Stüble, and M. Winandy. Property-Based TPM Virtualization. In *ISC 2008*, volume 5222 of *LNCS*, pages 1–16. Springer, 2008.

[24] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *USENIX Security Symposium*, pages 223–238. USENIX, 2004.

[25] E. Shi, A. Perrig, and L. van Doorn. BIND: A Fine-Grained Attestation Service for Secure Distributed Systems. In *IEEE Symposium on Security and Privacy*, pages 154–168. IEEE CS, 2005.

[26] S. W. Smith. *Trusted Computing Platforms: Design and Applications*. Springer, 2004.

[27] F. Stumpf, O. Tafreschi, P. Röder, and C. Eckert. A Robust Integrity Reporting Protocol for Remote Attestation. In *WATC 2006*, 2006.

[28] Trusted Computing Group. *Trusted Platform Module Main Specification*, version 1.2, revision 62 edition, October 2003.