JOHANNES KEPLER
UNIVERSITY LINZ

Michael Preisach,
Michael Roland
Institute of
Networks and Security

michael.roland@ins.jku.at
https://www.digidow.eu/

June 2021

Group Signature Applications: Direct Anonymous Attestation

Technical Report

Christian Doppler Laboratory for
Private Digital Authentication in the Physical World

This work has been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World, funded by the Christian Doppler Forschungsgesellschaft, 3 Banken IT GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH, and Österreichische Staatsdruckerei GmbH.

INSTITUTE
OF NETWORKS
AND SECURITY

DIGiDOW

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Contents

1. Introduction

A *group signature* is a special class of signatures schemes. It allows multiple associated signers to form a group and to sign a message on behalf of the group. There is—as in common signature schemes—only one public key for signature verification. Ideally, from the verifier point of view, there is no procedural difference in message verification compared to a common signature scheme.

The feature of having multiple secret keys signing against a single public key introduces however a number of additional requirements to the underlying cryptosystem. Accordingly, verifying and proving functional correctness becomes more complex as well as implementing the algorithms. This fact may reason why there are only few group signature schemes available and not yet widespread in use—unlike common signature schemes which is an essential part to build trust in digital communication.

Direct Anonymous Attestation (DAA) is an application of group signatures. Although there exist some implementations of DAA, e.g. in *Trusted Platform Modules version 1.2* (TPM 1.2), there exists no standardized DAA scheme for general purpose use yet.

This report will explain the basic concept of group signatures along with its cryptographic requirements. With this foundation, the report will provide a description of two different variants of *Direct Anonymous Attestation* (DAA) as implementation of a group signature scheme. Finally, the report discusses the impact to real world issues and why we are still lacking an affordable group signature scheme.

2. Definition of Group Signatures

A common signature scheme consists of an asymmetric encryption algorithm and a hashing algorithm. There are furthermore two parties involved, a signer and a verifier. Three algorithmic parts fulfill the task of signing and trusting a certain message. The first part is called *Setup* and includes all preparation steps for creating a signature. That is selecting an asymmetric encryption scheme—like RSA or *Elliptic Curve Cryptography* (ECC)—and generating the signer's key pair *sk* and *pk*. The public part *pk* is then published to all verifiers.

In the *Sign* step, the signer is given a message which will be hashed to have a representation of the it with a fixed length. This hash—together with optional metadata—will be encrypted with the signer's secret key *sk* and appended to the message itself. Finally, the message will be sent to the verifier.

The verifier has to check whether the message was not tampered with and the creator is indeed the signer. It uses the hash of the cleartext part of the message to hash it again. The signer's public key *pk* is used to decrypt the appended signature. If the decrypted hash in the signature is equal to the hash computed by the verifier itself, the message is not modified and the signer is indeed the owner of *sk*. Trust is created between signer and verifier.

Unlike the common signature scheme, group signatures consist of more than one signer. This requires a *Group Manager* as additional party to organize the membership in the corresponding group. Bellare et al. [2] introduced as one of the first a formal definition and a list of requirements for the general concept of group signatures. First, the paper gives a definition of the cryptographic algorithms, every scheme should consist of:

- *Group Key Generation* (GKg): The algorithm should randomly, based on the security number k, generate the group public key *gpk*, the group manager's secret key *gsk* and n group member secret keys msk_i.
- *Group Signing* (GSig): Given one *msk* and a message m, a randomly generated signature σ will be returned.
- *Group Signature Verification* (GVf): This deterministic algorithm takes a message m and a signature σ and the public key *gpk* and checks whether the signature is valid or not.
- *Opening* (Open): This algorithm returns the signer's identity, given the manager private key *gmsk* and a signature σ.

Given a message with a signature from a group member, a verifier can only check with the group public key *gpk* that the signature is created by one of the members. Although this anonymity is a reasonable feature for parties not involved in the group, it might be necessary for the manager to unveil the identity of a signer in case of a dispute or a fraudulent use is detected. Consequently, the Opening algorithm is only used for handling corner cases of the scheme.

Furthermore, Bellare et al. [2] introduce three classes of group management. The easiest of them is of course a static subscription scheme. In this case only a predefined set of members can sign a message and modifying this set requires a complete new cryptographic group. A *partial dynamic group* is defined as a group where members can be added during lifetime without changing existing keys and signatures. Finally, a *full dynamic group* includes also revocation of members during lifetime.

Besides the mentioned building blocks, Bellare et al. [2] discuss relevant security notions for a sound implementation. Similar to the extension of the algorithms, the security properties need to be extended with a subscription feature. Common signature schemes include:

- *Correctness*: The scheme must create valid signatures under all possible and valid public/secret key pairs and for all possible messages.
- *Collision resistance*: It is not feasible to compute two equal hashes with different content in messages.
- *Unforgeability*: Valid pairs of messages and signatures can only be produced with a valid secret key.

The above list of security parameters has to be extended for the generic group signature scheme:

- *Anonymity*: It is computationally hard to recover the identity of a signer without the group manager's secret key (*gmsk*).
- *Unlinkability*: Given two different signatures, an adversary cannot determine whether those relate to the same signer identity or not.
- *Exculpability*: No member of the group and no group manager is able to sign messages on behalf of other members.
- *Traceability*: The group manager is able to unveil the identity of a signer, given the signature of a message, the secret key *gmsk* and the opening algorithm. Therefore, it should not be possible that group members collude to create a signature that cannot be traced (coalition resistance). Furthermore, the scheme should prevent to create a signature in a way that the opening algorithm returns the identity of another group member.

With these algorithmic functions and security parameters a basic group signature scheme is defined. The group manager possesses some identifying information about the members (usually the member's public key) and knows with the group manager's secret key *gsk* which signature corresponds to which member. On the other hand, the verifier has no information at all about the signer's identity. The only known property is a valid subscription to the group with the corresponding public key *gpk*.

The next section will introduce an application of Group Signatures called Direct Anonymous Attestation which extends above described security parameters to fit the needs for attestation.

3. Direct Anonymous Attestation (DAA)

DAA is an application of the above defined group signature scheme. An early implementation of DAA was standardized with version 1.2 of the *Trusted Platform Module* (TPM 1.2) in 2004. Although this implementation was secure at that point in time, meaning that no weaknesses or bugs were known at the point of standardization, some serious concept and protocol flaws were found in the years after that.

TPMs are small crypto coprocesseors with very limited computation power and functionality. The goal of the *Trusted Computing Group* (TCG), an international consortium maintaining the TPM standard, is to provide well defined crypto functions mainly for the PC platform that work independently from the hosting system. Therefore, a TPM is a dedicated chip which communicates with the main processor via a slow bus interface.

Unfortunately, when the TPM 1.2 DAA protocol was proven broken, the required changes to fix all problems were too complex. Consequently, using this protocol became discouraged and TCG focused on a secure implementation in the newer standard TPM 2.0. The new version was released in 2014. However, at that point, no DAA protocol fulfilled the security requirements to be implemented. So the TCG decided to provide only the required functions as building blocks. Several scientific groups developed then different DAA protocols based on the generic TPM 2.0 functions.

This paper discusses mainly one approach of Camenisch et al. [6] which was also implemented and published by Xaptum[1]. The following section will describe the mathematical foundations as well as the resulting protocol in further detail.

3.1 Mathematical Foundations

The following definitions form the mathematical building blocks for DAA. It is noteworthy that these definitions work with RSA encryption as well as with *Elliptic Curve Cryptography* (ECC).

3.1.1 Discrete Logarithm Problem

Given a cyclic group $G = \langle g \rangle$ of order n, the discrete logarithm of $y \in G$ to the base g is the smallest positive integer α satisfying $g^{\alpha} = y$ if this x exists. For

[1]https://github.com/xaptum/ecdaa

sufficiently large n and properly chosen G and g, it is infeasible to compute the reverse $\alpha = \log_g y$. This problem is known as *Discrete Logarithm Problem* and is the basis for the following cryptographic algorithms.

3.1.2 Signature Proof of Knowledge (SPK)

A SPK is a signature of a message which proves that the creator of this signature is in possession of a certain secret. The secret itself is never revealed to any other party. Thus, this algorithm is a *Zero Knowledge Proof of Knowledge* (ZPK).

Camenisch and Stadler [8] introduced the algorithm based on the Schnorr Signature Scheme. It only assumes a collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ for signature creation. For instance,

$$SPK\{(\alpha) : y = g^\alpha\}(m)$$

denotes a proof of knowledge of the secret α, which is embedded in the signature of message m. The one-way protocol consists of three procedures:

1. *Setup.* Let m be a message to be signed, α be a secret and $y := g^\alpha$ be the corresponding public representation.
2. *Sign.* Choose a random number r and create the signature tuple (c, s) as
$$c := \mathcal{H}(m \,||\, y \,||\, g \,||\, g^r) \quad \text{and} \quad s := r - c\alpha \pmod{n}.$$
3. *Verify.* The verifier knows the values of y and g, as they are usually public. The message m comes with the signature values c and s. She computes the value
$$c' := \mathcal{H}(m \,||\, y \,||\, g \,||\, g^s y^c) \quad \text{and verifies, that} \quad c' = c\,.$$
The verification holds since
$$g^s y^c = g^r g^{-c\alpha} g^{c\alpha} = g^r.$$

This scheme is extensible to prove knowledge of an arbitrary number of secrets as well as more complex relations between secret and public values.

3.1.3 Bilinear Maps

Bilinear Maps define a special property for mathematical groups which form the basis for verifying the signatures in DAA. Consider three mathematical groups $\mathbb{G}_1$, $\mathbb{G}_2$, with their corresponding base points g_1, g_2, and $\mathbb{G}_T$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that satisfies three properties [7, 6]:

- *Bilinearity.* For all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, for all $a, b \in \mathbb{Z} : e(P^a, Q^b) = e(P, Q)^{ab}$.
- *Non-degeneracy.* For all generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2 : e(g_1, g_2)$ generates $\mathbb{G}_T$.
- *Efficiency.* There exists an efficient algorithm that outputs the bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ and an efficient algorithm for computing e.

3.1.4 Camenisch-Lysyanskaya Signature Scheme

The Camenisch-Lysyanskaya (CL) Signature Scheme [7] is based on the LRSW assumption and allows efficient proofs for signature posession and is the basis for the DAA scheme discussed below. It is based on a bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ that is available to all steps in the protocol.

- *Setup.* Choose $x \leftarrow \mathbb{Z}_q$ and $y \leftarrow \mathbb{Z}_q$ at random. Set the secret key $sk \leftarrow (x, y)$ and the public key $pk \leftarrow (g_2^x, g_2^y) = (X, Y)$.
- *Sign.* Given a message m, and the secret key sk, choose a at random and output the signature $\sigma \leftarrow (a, a^y, a^{x+xym}) = (a, b, c)$.
- *Verify.* Given message m, signature σ and public key pk, verify, that $a \neq 1_{\mathbb{G}_1}$, $e(a, Y) = e(b, g_2)$ and $e(a, X) \cdot e(b, X)^m = e(c, g_2)$.

Camenisch et al. stated in section 4.2 of their paper [6] that one has to verify the equation against $e(g_1, b)$ and $e(g_1, c)$ which is not correct.

3.2 DAA Protocol on LRSW Assumption

DAA is a group signature protocol, which aims with a supporting TPM to reveal no additional information about the signing host besides content and validity of the signed message m. According to Camenisch et al.[6], the DAA protocol consists of three parties:

- *Issuer $\mathcal{I}$.* The issuer maintains a group and has evidence of hosts that are members in this group. This role equals the group manager of Bellare's generic definition.
- *Host $\mathcal{H}$.* The Host creates a platform with the corresponding TPM $\mathcal{M}$. Membership of groups are maintained by the TPM. Compared to Bellare et al. [2], the role of a member is split into two cooperating parties, the key owner (TPM, passive) and the message author (Host, active).
- *Verifier $\mathcal{V}$.* A verifier can check whether a Host with its TPM is in a group or not. Besides the group membership, no additional information is provided.

A certificate authority $\mathcal{F}_{ca}$ is providing a certificate for the issuer itself. The basename `bsn` is some clear text string, whereas `nym` represent the encrypted basename $\texttt{bsn}^{gsk}$. $\mathcal{L}$ is the list of registered group members which is maintained by $\mathcal{I}$. The paper of Camenisch et al. [6] introduces further variables that are necessary for their proof of correctness. These extensions were omitted in the following to understand the protocol more easily.

- *Setup.* During Setup $\mathcal{I}$ is generating the issuer secret key *isk* and the corresponding issuer public key *ipk*. The public key is published and assumed to be known to everyone.
  1. On input SETUP $\mathcal{I}$
     - generates $x, y \leftarrow \mathbb{Z}_q$ and sets $isk = (x.y)$ and $ipk \leftarrow (g_2^x, g_2^y) = (X, Y)$. Initialize $\mathcal{L} \leftarrow \emptyset$,
     - generates a proof $\pi \overset{\$}{\leftarrow} SPK\{(x, y) : X = g_2^x \wedge Y = g_2^y\}$ that the key pair is well formed,
     - registers the public key (X, Y, π) at $\mathcal{F}_{ca}$ and stores the secret key,
     - outputs SETUPDONE.

- *Join.* When a platform, consisting of host $\mathcal{H}_j$ and TPM $\mathcal{M}_i$, wants to become a member of the issuer's group, it joins the group by authenticating to the issuer $\mathcal{I}$.
  1. On input JOIN, host $\mathcal{H}_j$ sends the message JOIN to $\mathcal{I}$.
  2. $\mathcal{I}$ upon receiving JOIN from $\mathcal{H}_j$, chooses a fresh nonce $n \leftarrow \{0,1\}^\tau$ and sends it back to $\mathcal{H}_j$.
  3. $\mathcal{H}_j$ upon receiving n from $\mathcal{I}$, forwards n to $\mathcal{M}_i$.
  4. $\mathcal{M}_i$ generates the secret key:
     - Check, that no completed key record exists. Otherwise, it is already a member of that group.
     - Choose $gsk \overset{\$}{\leftarrow} \mathbb{Z}_q$ and store the key as $(gsk, \perp)$.
     - Set $Q \leftarrow g_1^{gsk}$ and compute $\pi_1 \overset{\$}{\leftarrow} SPK\{(gsk) : Q = g_1^{gsk}\}(n)$.
     - Return (Q, π_1) to $\mathcal{H}_j$.
  5. $\mathcal{H}_j$ forwards JOINPROCEED(Q, π_1) to $\mathcal{I}$.
  6. $\mathcal{I}$ upon input JOINPROCEED(Q, π_1) creates the CL-credential:
     - Verify that π_1 is correct.
     - Add $\mathcal{M}_i$ to $\mathcal{L}$.
     - Choose $r \overset{\$}{\leftarrow} \mathbb{Z}_q$ and compute $a \leftarrow g_1^r$, $b \leftarrow a^y$, $c \leftarrow a^x \cdot Q^{rxy}$, $d \leftarrow Q^{ry}$.
     - Create the prove $\pi_2 \overset{\$}{\leftarrow} SPK\{(t) : b = g_1^t \wedge d = Q^t\}$.
     - Send APPEND(a, b, c, d, π_2) to $\mathcal{H}_j$
  7. $\mathcal{H}_j$ upon receiving APPEND(a, b, c, d, π_2)
     - verifies, that $a \neq 1$, $e(a, Y) = e(b, g_2)$ and $e(c, g_2) = e(a \cdot d, X)$.
     - forwards (b, d, π_2) to $\mathcal{M}_i$.
  8. $\mathcal{M}_i$ receives (b, d, π_2) and verifies π_2. The join is completed after the record is extended to $(gsk, (b, d))$. $\mathcal{M}_i$ returns JOINED to $\mathcal{H}_j$.
  9. $\mathcal{H}_j$ stores (a, b, c, d) and outputs JOINED.
- *Sign.* After joining the group, a host $\mathcal{H}_j$ and TPM $\mathcal{M}_i$ can sign a message m with respect to basename bsn.
  1. $\mathcal{H}_j$ upon input SIGN(m, bsn) re-randomizes the CL credential:
     - Retrieve the join record (a, b, c, d) and choose $r \overset{\$}{\leftarrow} \mathbb{Z}_q$. Set $(a', b', c', d') \leftarrow (a^r, b^r, c^r, d^r)$.
     - Send (m, bsn, r) to $\mathcal{M}_i$ and store (a', b', c', d').
  2. $\mathcal{M}_i$ upon receiving (m, bsn, r)
     - checks, that a complete join record $(gsk, (b, d))$ exists, and
     - stores (m, bsn, r).
  3. $\mathcal{M}_i$ completes the signature after it gets permission to do so.
     - Retrieve group record $(gsk, (b, d))$ and message record (m, bsn, r).

  - Compute $b' \leftarrow b^r, d' \leftarrow d^r$.
  - If $\mathsf{bsn} = \perp$ set $\mathsf{nym} \leftarrow \perp$ and compute $\pi \xleftarrow{\$} SPK\{(gsk) : d' = b'^{gsk}\}(m, \mathsf{bsn})$.
  - If $\mathsf{bsn} \neq \perp$ set $\mathsf{nym} \leftarrow H_1(\mathsf{bsn})^{gsk}$ and compute $\pi \xleftarrow{\$} SPK\{(gsk) : \mathsf{nym} = H_1(\mathsf{bsn})^{gsk} \wedge d' = b'^{gsk}\}(m, \mathsf{bsn})$.
  - Send (π, nym) to $\mathcal{H}_j$.

4. $\mathcal{H}_j$ assembles the signature $\sigma \leftarrow (a', b', c', d', \pi, \mathsf{nym})$ and outputs SIGNATURE(σ).

- *Verify.* Given a signed message, everyone can check whether the signature with respect to bsn is valid and the signer is member of this group. Furthermore a revocation list RL holds the private keys of corrupted TPMs, whose signatures are no longer accepted.
  1. $\mathcal{V}$ upon input VERIFY(m, bsn, σ)
     - parses $\sigma \leftarrow (a, b, c, d, \pi, \mathsf{nym})$,
     - verifies π with respect to (m, bsn) and nym if $\mathsf{bsn} \neq \perp$.
     - checks, that $a \neq 1$, $b \neq 1$ $e(a, Y) = e(b, g_2)$ and $e(c, g_2) = e(a \cdot d, X)$,
     - checks, that for every $gsk_i \in \mathsf{RL} : b^{gsk_i} \neq d$,
     - sets $f \leftarrow 1$ if all test pass, otherwise $f \leftarrow 0$, and
     - outputs VERIFIED(f).
- *Link.* After proving validity of the signature, the verifier can test whether two different messages with the same basename $\mathsf{bsn} \neq \perp$ are generated from the same TPM.
  1. $\mathcal{V}$ on input LINK($\sigma, m, \sigma', m', bsn$) verifies the signatures and compares the pseudonyms contained in σ, σ':
     - Check, that $\mathsf{bsn} \neq \perp$ and that both signatures σ, σ' are valid.
     - Parse the signatures $\sigma \leftarrow (a, b, c, d, \pi, \mathsf{nym})$, $\sigma' \leftarrow (a', b', c', d', \pi', \mathsf{nym}')$.
     - If $\mathsf{nym} = \mathsf{nym}'$, set $f \leftarrow 1$, otherwise $f \leftarrow 0$.
     - Output LINK(f).

Camenisch et al. [6] extend with their concept the general group concept scheme. The feature of linking messages together requires further security features within the DAA scheme, which the authors also prove in their paper along with the other properties of the scheme:

- *Non-frameability*: No one can create signatures that the platform never signed, but that link to messages signed from that platform.
- *Correctness of link*: Two signatures will link when the honest platform signs it with the same basename.
- *Symmetry of Link*: It does not matter in which order the linked signatures will be proven. The link algorithm will always output the same result.

3.3 DAA Extensions and Variants

As previosly mentioned, the DAA scheme of Camenisch et al. [6] is just one which is supported by TPMs. It is a dynamic group signature scheme with a very basic revocation algorithm. Only by publishing the private key of the subscribed TPM, a revocation can be done, which effectively breaks the TPM.

Consequently, Camenisch et al. [4] introduced a concept for signature based revocation. This is an additional list of signatures which is managed by the issuer. The proposed solution defines that signer must provide a SPK to show that his signature is not in the list of revocations.

Furthermore, they include a larger discussion about proving security and correctness of other DAA schemes as well as the protocol interface between the TPM and the host. The authors found several errors or incomplete mathematical proofs which make many of the DAA concepts insecure or even easily breakable.

Additionally, the concept of DAA can build upon another cryptographic problem, namely the *q-strong Diffie Hellman Problem* (q-SDH). Camenisch et al. [5] developed a scheme for this problem as well and were able to prove its correctness and security. The authors use the signature scheme from Boneh, Boyen and Shacham (BBS+) [3] instead of the CL scheme. According to the different problem, BBS+ signatures have a different construction for the SPK and a modified protocol. The application for DAA is similar to that of CL signatures. The resulting q-SDH DAA scheme is slightly modified to work with BBS+ signatures and directly support signature revocations. Until now there were no papers published which could proof that one of the two DAA schemes of Camenisch et al. is wrong or incomplete.

3.4 DAA Implementation

Any TPM with the newer specification 2.0 will support all above mentioned DAA concepts. The group administrator can choose between RSA and ECC when using DAA. Unfortunately, TPM chips lack of computational power which makes RSA very inconvenient to use. Signing and key generation takes several seconds on the TPM.

All DAA schemes build upon the concept of Bilinear Maps as described in section 3.1.3. The function *e* defined as a bilinear non-degenerative map is, however, not provided by common ECC curves. This additional feature is called *cryptographic pairing*, which makes the ECC curve *pairing-friendly*. TCG standardized several pairing-friendly ECC curves, but only one is mandatory in the current specification. This is the ECC curve of Barreto and Naehrig [1]. It provides a key length of up to 256 bits which is sufficiently secure for implementing DAA.

The only currently published implementation of DAA which uses TPM 2.0 is the ECDAA repository which is maintained by Xaptum ENF. It implements the LRSW variant of Camenisch's DAA and makes optionally use of the attached TPM. They also implemented a variant without TPM usage to show that the cryptographic algorithm works.

4. Conclusion

The concept of group signatures is very complex to define, to prove and to implement. This may be a reason for the fact that there exist only few applications and hardly any real-world implementations. TCG was one of the first adopter and failed unfortunately with their first approach in TPM 1.2 by implementing a DAA scheme which is nowadays declared broken. Consequently, for the recent TPM 2.0 specification, there exist only the building blocks for DAA and the scientific community developed several concepts against this specification. The benefit of supporting an eventually secure DAA scheme contrasts the drawback of a standardized implementation. Yet there is also no widespread application available.

Within the Digidow project, TPMs and DAA could be used to attest trustworthiness of participants in the Digidow transaction protocol. When a sensor provides measurement readings to the personal identity agent, we would also like to know if the sensor platform contains the expected software state. Only then, the sensor readings can be trusted. A personal identity agent might even want to refuse to communicate with sensors that cannot attest a certain software state based on a complete chain of trust starting at bootup. Similarly other parties in the Digidow transaction protocol might only want to communicate with personal identity agents that contain a certain software state considered trustworthy by them. In that particular case, it is essential that the authentication of the software state does not reveal any cross-session unique identifiers that would allow linking otherwise unlinkable transactions to a specific individual's personal identity agent. DAA can contribute to establishing such trust without deanonymizing the attested party.

References

[1] Paulo S. L. M. Barreto and Michael Naehrig. 2005. Pairing-Friendly Elliptic Curves of Prime Order. In *Selected Areas in Cryptography* (*LNCS*, volume 3897). 12th International Workshop on Selected Areas in Cryptography, SAC 2005 (Kingston, ON, Canada). Springer, Berlin, Heidelberg, (August 2005), 319–331. DOI: 10.1007/11693383_22.

[2] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. 2003. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *Advances in Cryptology — EUROCRYPT 2003* (*LNCS*, volume 2656). International Conference on the Theory and Applications of Cryptographic Techniques (Warsaw, Poland). Springer, Berlin, Heidelberg, 614–629. DOI: 10.1007/3-540-39200-9_38.

[3] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In *Advances in Cryptology – CRYPTO 2004* (*LNCS*, volume 3152). 24th Annual International Cryptology Conference (Santa Barbara, CA, USA). Springer, Berlin, Heidelberg, (August 2004), 41–55. DOI: 10.1007/978-3-540-28628-8_3.

[4] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. 2017. One TPM to Bind Them All: Fixing TPM 2.0 for Provably Secure Anonymous Attestation. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Jose, CA, USA, (May 2017), 901–920. DOI: 10.1109/SP.2017.22.

[5] Jan Camenisch, Manu Drijvers, and Anja Lehmann. 2016. Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited. In *Trust and Trustworthy Computing* (*LNCS*, volume 9824). 9th International Conference on Trust and Trustworthy Computing, TRUST 2016 (Vienna, Austria). Springer, Cham, 1–20. DOI: 10.1007/978-3-319-45572-3_1.

[6] Jan Camenisch, Manu Drijvers, and Anja Lehmann. 2016. Universally Composable Direct Anonymous Attestation. In *Public-Key Cryptography – PKC 2016* (*LNCS*, volume 9615). 19th IACR International Conference on Practice and Theory in Public-Key Cryptography (Taipei, Taiwan). Springer, Berlin, Heidelberg, (March 2016), 234–264. DOI: 10.1007/978-3-662-49387-8_10.

[7] Jan Camenisch and Anna Lysyanskaya. 2004. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Advances in Cryptology – CRYPTO 2004* (*LNCS*, volume 3152). 24th Annual International Cryptology Conference (Santa Barbara, CA, USA). Springer, Berlin, Heidelberg, (August 2004), 56–72. DOI: 10.1007/978-3-540-28628-8_4.

[8] Jan Camenisch and Markus Stadler. 1997. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO ’97* (*LNCS*, volume 1294). 17th Annual International Cryptology Conference (Santa Barbara, CA, USA). Springer, Berlin, Heidelberg, 410–424. DOI: 10.1007/BFb0052252.