Revocation of Direct Anonymous Attestation

Liqun Chen[1] and Jiangtao Li[2]

[1] Hewlett-Packard Laboratories, UK
liqun.chen@hp.com
[2] Intel Labs, USA
jiangtao.li@intel.com

Abstract. Direct Anonymous Attestation (DAA) is a special type of anonymous digital signatures, used by the Trusted Computing Group (TCG) for the purpose of computer platform attestation whilst preserving platform anonymity. Like any other anonymous cryptographic primitives, how to efficiently revoke an existing member who is no longer legitimate, is an important and challenging subject for DAA. In this paper, we first explain two general DAA revocation approaches and a number of different DAA revocation degrees. We then present a variety of revocation mechanisms, which enable us to achieve these approaches and degrees in the existing three types of DAA schemes. Some of these mechanisms have already been shown in the literature and others are newly proposed in this paper.

Keywords: direct anonymous attestation (DAA), revocation, rekey-based revocation, verifier-local revocation.

1 Introduction

Direct Anonymous Attestation (DAA) is a special type of digital signatures that preserve signer's privacy (i.e., DAA signatures are anonymous and optionally unlinkable). Here anonymity means that the identity of the signer is not revealed from his signatures and unlinkability means that whether two signatures were produced by the same signer cannot be determined from the signatures. A DAA scheme is often seen as a variant of a group signature scheme. A DAA signature, similar to a group signature, allows the signer to demonstrate knowledge of a signer's individual private key with respect to a public key belonging to a group, and the signature is publicly verifiable by anybody who has access to the group public key. Unlike group signatures, DAA signatures are anonymous not only to signature verifiers, but also to the group manager who is in the possession of the associated group secret key. DAA allows a signature verifier to learn the signer's identity if the verifier knows the signer's private key, and DAA also allows a verifier to learn whether two signatures were produced by the same signer if the signer includes a special proof of such a link in both of the signatures.

These features make DAA signatures attractive for many applications, one of which is remote anonymous authentication of trusted computing platforms, in which a platform attests certain information, such as the configuration of the platform, to a remote verifier whilst maintaining anonymity of the platform. The concept and the first concrete scheme of DAA were introduced by Brickell, Camenisch, and Chen [4]. Their DAA scheme was adopted by the industry standardization body Trusted Computing Group (TCG) in 2003, and specified in the TCG TPM Specification version 1.2 [26] that has recently been adopted by ISO/IEC as an international standard [1]. A number of DAA schemes have been developed since then, e.g. [23,7,5,15,16,19,13,18,9,14], and most of them have been proved secure under either the security model of [4] or the model of [6].

Observe that to be truly useful in practice, a DAA scheme must support dynamic group membership, which means supporting growing membership, i.e. new members can join without precipitating changes in the group public key or re-issuing group membership credentials for existing members, and as well as supporting shrinking group membership, i.e. illegitimate members can be revoked in an efficient way. The property of growing membership has been well-supported by all the existing DAA schemes. But shrinking group membership has not been given the same attention. For a variety of reasons, an existing group member can be no longer legitimate, e.g., the member's private key has been hacked, the member might leave the group voluntarily, or he might be excluded by the group membership issuer. A typical example of using a DAA scheme is that in the TCG environment the DAA signer is a Trusted Platform Module (TPM) embedded in a computer platform. If the TPM has been corrupted and its DAA private key is publicly revealed, the TPM is considered to be an invalid group member and needs to be revoked from the group. Consider another example, suppose a DAA scheme is implemented in identity cards for college students. Each student can use his or her card to access college resources. If the identity card of a student is stolen or lost, or if the student has graduated from the college, the issuer (the college) may want to revoke the DAA private key of the student in the identity card. In any case, after a DAA signer is revoked, DAA verifiers should reject any signatures created by the signer.

Although there have been a number of DAA signer revocation mechanisms presented in the literature, such as [4,7,16], to the best of our knowledge, this important topic has not been fully explored. This is the motivation of this paper. There are a number of contributions in this paper. We describe each contribution briefly as follows.

Classification of DAA revocation techniques. We categorize a variety of DAA revocation solutions into two general approaches, and name them *rekey-based revocation* (rekey for short) and *verifier-local revocation* (VLR for short). We also introduce a number of different revocation degrees for DAA, namely *key revocation*, *key+credential revocation*, *key+verifier revocation*, and *key+credential+verifier revocation*; the details will be given in Section 2.2.

New DAA revocation mechanisms. We review the existing DAA schemes within three categories, namely RSA-DAA, LRSW-DAA, and SDH-DAA, and then review the existing four DAA revocation solutions. After that, we propose a number of new DAA revocation mechanisms, covering both the rekey and VLR approaches. We demonstrate how these revocation mechanisms can be used with the three types of the existing DAA schemes.

A comparison of all the DAA revocation solutions. We provide a comprehensive survey of all the DAA revocation mechanisms, including the already existing mechanisms and the new ones. We show comparisons of all the VLR methods, and comparisons between the rekey and VLR approaches. We also discuss how these two approaches can be used together in one application.

The main purpose of this work is to introduce the concept of the two DAA revocation approaches and a number of practical revocation mechanisms. This paper provides neither a formal security model for DAA revocation nor a rigorous security proof of the proposed revocation mechanisms. We will leave the formalization of the security model and analysis to the future work.

The rest of this paper is organized as follows. We first introduce in the next section the general concept of DAA revocation, including the meanings of the two general revocation approaches and security properties. We then review three types of the existing DAA schemes in Section 3 and review four existing DAA revocation solutions in Section 4. In the following two sections, we present the rekey and VLR revocation approaches, respectively, in details. We compare different revocation mechanisms and discuss how they can be used together in Section 7. We conclude the paper with the summary and future work in Section 8.

2 General Concept of DAA Revocation

2.1 Two Approaches

In the literature, a DAA scheme involves four types of players: a set of DAA issuers $\mathfrak{i}_k \in \mathfrak{I}$, TPMs $\mathfrak{m}_i \in \mathfrak{M}$, hosts $\mathfrak{h}_i \in \mathfrak{H}$ and verifiers $\mathfrak{v}_j \in \mathfrak{V}$. The index values, k, i, j, are polynomial. Each pair of $\mathfrak{m}_i$ and $\mathfrak{h}_i$ form a computer platform in the trusted computing environment and share the role of a DAA signer. Throughout the paper, for the purpose of simplicity, we may omit some of the index values, i.e., we make use of $\mathfrak{i}$, $\mathfrak{m}$, $\mathfrak{h}$ and $\mathfrak{v}$ instead of $\mathfrak{i}_k$, $\mathfrak{m}_i$, $\mathfrak{h}_i$ and $\mathfrak{v}_j$. A DAA scheme $\mathcal{DAA} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Link})$ consists of the following five polynomial-time algorithms and protocols:

- Setup: On input of a security parameter 1^t, an issuer $\mathfrak{i}$ uses this randomized algorithm to produce its secret key isk, and public key ipk (for simplicity we let ipk include all the system public parameters). We will assume that ipk is publicly known so that we do not need to explicitly provide it as input to other algorithms.
- Join: This protocol, run between a signer $(\mathfrak{m}, \mathfrak{h})$ and an issuer $\mathfrak{i}$, creates the TPM's secret key $\mathtt{tsk}$ and its DAA credential cre. The value $\mathtt{tsk}$ is generated (and then secretly kept) by $\mathfrak{m}$, and the value cre is generated by $\mathfrak{i}$ based on isk and sent to $\mathfrak{h}$.
- Sign: On input of $\mathtt{tsk}$, cre, a basename bsn (either the name string of the verifier $\mathfrak{v}$ or a special symbol $\perp$), a message m to be signed and $\mathfrak{v}$'s nonce n_V for freshness, $\mathfrak{m}$ and $\mathfrak{h}$ run this protocol to produce a signature σ.

– Verify: On input of m, bsn, a candidate signature σ for m, and a set of rogue signers' secret keys, denoted by $\mathsf{RogueList}$, $\mathfrak{v}$ uses this deterministic algorithm to return either 1 (`accept`) or 0 (`reject`). Note that how to build $\mathsf{RogueList}$ is out the scope of a DAA scheme.
– Link: On input of two signatures σ_0 and σ_1, $\mathfrak{v}$ uses this deterministic algorithm to return 1 (linked), 0 (unlinked) or $\perp$ (invalid signatures). Note that, unlike Verify, the result of Link is not relied on $\mathsf{RogueList}$.

Generally speaking, DAA revocation can be achieved with either or both of the following two approaches:

1. Rekey-based Revocation. To revoke any illegitimate DAA signer(s), an issuer $\mathfrak{i}$ updates its public key ipk (which might or might not involve updating the corresponding secret key isk) and then allows only the credentials of all legitimate signers to be updated accordingly. In the algorithms of Sign, Verify and Link, the newly updated keys and credentials will replace the old keys and credentials. To implement this revocation approach in a DAA scheme, we add a new algorithm called Rekey in $\mathcal{DAA}$ as follows:
   – Rekey: On input of the current values of ipk, isk, all cre_i for the legitimate signer $(\mathfrak{m}_i, \mathfrak{h}_i)$, and all cre_l for the illegitimate signer $(\mathfrak{m}_l, \mathfrak{h}_l)$, this randomized algorithm outputs the updated values of ipk, isk and cre_i. Note that the value isk may be the same after running Rekey. Note also that this algorithm may not require interaction between $\mathfrak{i}$ and $\mathfrak{m}_i/\mathfrak{h}_i$.
2. Verifier-Local Revocation (VLR). To revoke a set of illegitimate DAA signers, a verifier $\mathfrak{v}$ takes as an extra input a revocation list, say RL, to Verify and then only accepts any signatures from those signers who are not listed in RL. To implement this revocation approach in a DAA scheme, we slightly modify the above Sign and Verify algorithms in $\mathcal{DAA}$ to the following:
   – Sign: On input of $\mathtt{tsk}$, cre, a basename bsn, (optionally) a revocation list RL[1] along with an indication about which type of the proof is required in the signature, a message m to be signed and $\mathfrak{v}$'s nonce n_V for freshness, $\mathfrak{m}$ and $\mathfrak{h}$ run this protocol to produce a signature σ.
   – Verify: On input of m, bsn, a candidate signature σ for m, and a revocation list RL, $\mathfrak{v}$ uses this deterministic algorithm to return either 1 (`accept`) or 0 (`reject`). Note that a rogue signer must be included in RL so the set of rogue signers' secret $\mathsf{RogueList}$ is covered by RL. Note also that how to build $\mathsf{RogueList}$ and RL is out the scope of this paper.

   The concept of VLR is a modification of the concept of VLR in group signatures [3]. The difference is that the DAA VLR is under the observation and cooperation of a signer, i.e., a signer and a verifier agree on a linking method to enable the verifier to build his own revocation list. Whereas in group signatures, VLR is a delegating function of the opening authority, and therefore is achieved transparently from signers.

[1] Whether such a revocation list is needed or not is dependent on the specific VLR mechanism used.

Remark 1. Users can choose to use any of the above two approaches in a DAA scheme, and can alternatively use both of them together in one DAA scheme.

2.2 Security Properties

In the literature, there are two security models used to define security of a DAA scheme. The first DAA security model introduced in [4] is presented using the ideal-system/real-system model and the second one introduced in [6] is presented with three notions: *correctness*, *user-controlled-anonymity* and *user-controlled-traceability*. The last two notions in the second model are each described by a game run between an adversary and a simulator. Due to limited space, we do not recall these two models here. Informally, these security definitions require the following four properties to be held in a DAA scheme:

1. *Correctness.* It ensures that a signature signed under a valid pair of tsk and cre verifies correctly.
2. *Anonymity.* An adversary not in possession of a tsk cannot trace the identity of the signer from its signature.
3. *Unforgeability.* An adversary, which has corrupted a set of tsk's and their cre's, finds it hard to forge a valid signature under a pair of tsk and cre, which are not in the set.
4. *User-controlled traceability.* On the one hand, given a single basename bsn ($\neq \perp$), an adversary is not able to create two different signatures associated with bsn signed under the same tsk, but the output of the algorithm Link is 0 (unlinked). On the other hand, given two signatures σ_0 and σ_1 associated with two different basenames bsn_0 and bsn_1 respectively, an adversary not in possession of the relevant tsk(s) finds it hard to tell whether or not the two signatures are signed by the same signer.

Broadly speaking, DAA revocation requires if a DAA signer has been revoked, the revoked signer is no longer able to create any valid DAA signatures. A valid signature means the verification algorithm with this signature outputs accept. But what do we mean that a signer is revoked? In order to answer this question, we first consider a few simple facts in use of a DAA scheme:

– To create a signature, a signer needs a pair of tsk and cre. If tsk is compromised, the signer should be revoked, no matter whether the associated cre is compromised or not.
– A signer might have two credentials cre_0 and cre_1, both associated with a single tsk. The pair (tsk, cre_0) might be revoked, but the pair (tsk, cre_1) might still be valid.
– Using a pair of tsk and cre, a signer might create signatures for different verifiers. The signer may be revoked or black-listed by one verifier, but may not be revoked by another verifier.

In order to address the security notion of DAA revocation to cover the above facts, we identify the following four different revocation degrees for DAA, each of which indicates one specific level what we mean by that a signer is revoked:

– *Key revocation.* We say that a `tsk` is revoked if any signature signed under this key is rejected no matter which cre is used and no matter which verifier the signature was signed for.
– *Key+credential revocation.* We say that a pair of `tsk` and cre are revoked if any signature signed under this key and credential pair is rejected no matter which verifier the signature was signed for. In that case, a signature signed under the same `tsk` but a different cre could still be valid.
– *Key+verifier revocation.* We say that a `tsk` is revoked with a certain verifier if any signature signed under this key for this verifier is rejected no matter which cre is used.
– *Key+credential+verifier revocation.* We say that a pair of `tsk` and cre are revoked with a certain verifier if any signature signed under this key and credential pair for this verifier is rejected. In this case, another signature signed under this key with a different credential or for a different verifier could still be valid.

Remark 2. With this notion of DAA revocation, we assume that each signer has a single `tsk` and each verifier has a single bsn. If a user (or a verifier) has two keys (or two bsn values), he/she plays the role of two signers (or verifiers).

Remark 3. We say a revocation mechanism holds the property of *backward validation* if a signature created before the signer is revoked remains valid even after the signer is revoked. This property is also called backward unlinkability in [25]. We notice that the property of backward validation is not always suitable for a DAA scheme, since it probably allows a revoked signer to create a signature and then to claim it was signed before revocation. Time stamps can sometimes avoid this drawback, but a time stamp is not usually included in the description of the DAA scheme. In our above notion of DAA revocation, in order to achieve backward validation, a certain time period can be included in the description of a verifier in the key+verifier revocation. However, there might be issues related to anonymity with time stamps; for example, the time-zone information contained by the time stamp results in a limitation of the anonymity protection as the signer can be related to a certain region. The users should take this concern into account when using time-stamps in any anonymous signatures, such as DAA.

3 Review Three Types of DAA Schemes

Since the original DAA scheme [4] was adopted by the Trusted Computing Group in 2003 [26], researchers have developed a large number of DAA schemes, such as [5,15,19,13,9,18,14], to list a few. For the purpose of this paper, we review the existing DAA schemes within three categories, as respectively shown in the following three subsections. The three categories cover most of the DAA schemes we know. Security of each category is based on a certain computationally-hard problem. Our review includes only the necessary information, which enables us to present our revocation mechanisms for these schemes in Sections 5 and 6, and we refer the details of these schemes to their original papers.

3.1 The RSA-DAA Scheme

The original DAA scheme proposed by Brickell, Camenisch, and Chen [4] is based on the strong RSA assumption [22]. We now give a brief review of this scheme using the same notation as in the paper [4], and we denote this scheme RSA-DAA. In this scheme, the issuer's private key is the factorization of a modulus n, and its public key includes $(R_0, R_1, S, Z, n, \Gamma)$. The signer's secret is $\texttt{tsk} = (f_0, f_1)$ and the corresponding credential is $\mathsf{cre} = (e, A, v)$, such that the triple (e, A, v) is a Camenisch-Lysyanskaya signature [11] on (f_0, f_1). See [4] for the sizes of these values. A DAA signature is a proof of knowledge of values f_0, f_1, A, e, and v under a given verifier's base name bsn, such that

$$A^e R_0^{f_0} R_1^{f_1} S^v \equiv Z \bmod n \text{ and } N_V \equiv \zeta^{f_0 + f_1 2^{\ell_f}} \bmod \Gamma,$$

where the value ζ is computed from bsn and $N_V = \zeta^{f_0 + f_1 2^{\ell_f}} \bmod \Gamma$ is computed by the signer. Both (ζ, N_V) are part of the signature.

3.2 The LRSW-DAA Schemes

Several pairing based DAA schemes [5,15,16,18,14] have recently been proposed using the LRSW assumption [24]. We now give a brief review of these schemes, and denote this type of DAA schemes LRSW-DAA. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a computable bilinear map function. Note that the scheme in [5] makes use of a symmetric pairing, i.e. $\mathbb{G}_1 = \mathbb{G}_2$. For simplicity, we do not distinguish between the LRSW-DAA schemes using symmetric or asymmetric pairings. In a LRSW-DAA scheme, the issuer's private key is $(x, y) \in \mathbb{Z}_p$ for a prime p and the public key includes $(p, P_1, P_2, X = [x]P_2, Y = [y]P_2)$, where $P_1 \in \mathbb{G}_1$ and $P_2, X, Y \in \mathbb{G}_2$. The signer's secret $\texttt{tsk}$ is $f \in \mathbb{Z}_p$ and the corresponding DAA credential cre is (A, B, C), which is a Camenisch-Lysyanskaya signature [12] of the value f. A DAA signature is a proof of knowledge of values f, A, B and C under a given base name bsn, such that

$$A = [r]P_1, B = [y]A, C = [x + xyf]A \text{ for a random } r \in \mathbb{Z}_p, \text{ and } K = [f]J,$$

where J is an element of $\mathbb{G}_1$, $\mathbb{G}_T$ or a different cyclic group, computed from bsn.

3.3 The SDH-DAA Schemes

A few pairing-based DAA schemes [19,13,9,8] were developed based on the Strong Diffie-Hellman (SDH) assumption [20]. We now give a brief review of these schemes, and denote this type of DAA schemes SDH-DAA. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a computable bilinear map function. In a SDH-DAA scheme, the issuer's private key is γ and its public key includes $(p, g_1, h_1, h_2, g_2, w = [\gamma]g_2)$, where p is a large prime, $g_1, h_1, h_2 \in \mathbb{G}_1$, and $g_2, w \in \mathbb{G}_2$. The signer's secret $\texttt{tsk}$ is $f \in \mathbb{Z}_q$ and the corresponding DAA credential is $\mathsf{cre} = (A, x)$ where x is randomly chosen from $\mathbb{Z}_p$ and $A = [1/(\gamma + x)](g_1 + [f]h_1)$; note that (A, x) is a BBS signature [2] on the value f. A DAA signature is a proof of knowledge of the values f, A, and x under a give base name bsn such that

$$A = [1/(\gamma + x)](g_1 + [f]h_1) \text{ and } K = [f]J,$$

where J is an element of $\mathbb{G}_1$ or a different cyclic group, computed from bsn.

Observe that in some SDH-DAA schemes [19,8], an extra value y is included, such that $A = [1/(\gamma+x)](g_1+[f]h_1+[y]h'_1)$ and $\mathsf{cre} = (A, x, y)$. As proved in [13], one can remove this value while maintaining the same security level. Therefore, in this paper, we do not distinguish between the SDH-DAA schemes with and without the y value.

4 Review of the Existing Revocation Solutions

Currently, there are four types of revocation solutions are known for DAA. The first two solutions were proposed in the original DAA paper [4], whilst the third and forth have recently been proposed in [7] and [16] respectively. The following is a brief summary of these four solutions:

1. Revocation is consequent upon a signer's DAA secret becoming known. Anybody believing they have come into possession of a signer's DAA secret (which, of course, should not happen) can put the secret into the rogue list $\mathsf{RogueList}$ and then check if this is truly the case by carrying out a check to verify whether a DAA signature from the signer was signed using the secret in $\mathsf{RogueList}$ or not - if yes, the signature is rejected as the signer's DAA secret has clearly been compromised.
2. A verifier builds his own black list of unwelcome signers. In order to find whether a DAA signature was signed by a black-listed signer, the Verifier must require the signer to use a specific basename in his DAA signature.
3. In each DAA signature, a signer is required to prove, in a zero-knowledge proof manner, that his private signing key is not listed in a black list maintained by a revocation manager.
4. A DAA issuer updates his key either in a fixed interval or a flexible period. At each update of his keys the issuer also correspondingly updates each DAA credential it holds unless, from the issuer's knowledge, a signer is no longer a legitimate DAA signer.

We organize these four solutions into the two approaches described in Section 2.1: rekey-based revocation covering the solution 4, and verifier-local revocation covering all the other three solutions. In the next two sections, we will detail these two approaches and introduce a variety of mechanisms in each approach.

5 Rekey-Based Revocation

5.1 An Overview

In a DAA scheme, a signer can create a valid signature if the signer holds a valid DAA credential, which is created by a DAA issuer using the issuer's private key. The signature can then be verified by using the issuer's public key.

A rekey-based DAA revocation solution is that the issuer updates his public key (optionally with the private key as well) when needed. At each update of his keys the issuer also correspondingly updates each DAA credential it holds unless, from the issuer's knowledge, a signer is no longer a legitimate DAA signer. Subsequent executions of the sign protocol are affected on the basis of the signer's updated credential and the subsequent verification algorithm are also affected using the updated public key of the issuer. If a signer's DAA credential has not been updated in correspondence with the updating of the issuer's public key, verification of the signer's DAA signature by the Verifier will fail thereby giving effect to revocation of the signer's privilege of a legitimate DAA signer.

A naive implementation of rekey is letting each legitimate signer rejoin the group by running the Join protocol. Obviously this is not an efficient solution. A better method is that the issuer can recompute each legitimate signer's credential without the signer's involvement. This type of solution is called non-interactive rekey. The general idea of this solution was introduced in [16]. In this paper, we will present some new implementations of this solution.

The process of rekey can be done in a fixed time interval. The advantage of this choice is that each entity knows when the rekey process happens. The drawback is that any illegitimate signer cannot be revoked within the interval. The process of rekey can also be done flexibly when the group shrinks with some members leaving. Obviously the advantage and disadvantage of this choice is opposite to the previous one. The selection of the fixed interval, the flexible interval, and the length of the interval are dependent on applications.

Observe that a DAA signature is not traceable from the corresponding DAA credential. Even if the credential is published, the DAA scheme can still hold the properties of user-controlled-anonymity and user-controlled-traceability. As a consequence, it is acceptable for the issuer to maintain a publicly available list of all signers' credentials, or at least of those credential elements subject to updating. A few options on how the updated credential can be made available to the host have been suggested [16], see the following for a summary:

- Each signer downloads its own credential value from the list. This assumes some appropriate identifier of each list entry is provided; alternatively the issuer can create an index, where each signer has an reference number, which is not linked to its secret or credential but uniquely indicates the signer - in this case, the signer may wish to check the message authentication of the published list before downloading the updated value and this can be done by using an ordinary signature scheme.
- Although revocation of a signer need not result in removal of the corresponding entry from the public list maintained by the issuer (the non-updating of the updatable elements being adequate to effect revocation), it is preferable to remove such entries to minimise the size of the list.
- If the issuer desires to make updated credential elements available only to their respective signers, he can encrypt each updated credential under a public key belonging to the signer (e.g., if the signer is a TPM and its host, this key could be the TPM's public endorsement key). The issuer can choose

either listing the cipher-text rather than the plain-text on the public list maintained by the issuer, or making this information publicly unavailable.

Details of how the issuer's key and a signer's credential are updated by the issuer will be given in the following three subsections, respectively for the RSA-DAA scheme, the LRSW-DAA schemes and the SDH-DAA schemes.

5.2 Rekey in the RSA-DAA Scheme

In the Join protocol of the RSA-DAA scheme reviewed in Subsection 3.1, the signer chooses secret keys f_0 and f_1, picks a random value v', and then sends the commitment $U = R_0^{f_0} R_1^{f_1} S^{v'} \bmod n$ to the issuer. The issuer then picks v'' and e, and computes $A = (Z/US^{v''})^{1/e} \bmod n$. Let $v = v' + v''$. The signing key (f_0, f_1, e, A, v) satisfies the equation $A^e R_0^{f_0} R_1^{f_1} S^v = Z \bmod n$. To support rekey, we assume the issuer stores at least the following items from the join protocol with each signer: the commitment U of the signer's secret keys, the quantity e, and the quantity v''. Alternatively to storing U and v'' individually, the quantity $US^{v''}$ can be stored. In a typical setting, the issuer stores $US^{v''}$ and the triple (e, A, v'') for each signer.

We now describe a variety of updating methods as follows. The first one is a recall of [16].

1. To update the issuer's public key and the DAA credential of each currently-legitimate signer, the issuer simply chooses a new value for Z, here called Z'', and, for each currently-legitimate signer computes a new value for A, here called A'', as:
$$A'' = (Z''/US^{v''})^{1/e} \bmod n.$$
The issuer then publishes Z'' to replace the value of Z in its previously published public key; the issuer also publishes a new non-interactive proof of the fact that R_0, R_1, S, Z, g, and h are computed correctly i.e. that g, $h \in \langle g' \rangle$, S, $Z \in \langle h \rangle$ and R_0, $R_1 \in \langle S \rangle$. For each updated credential, the issuer makes the newly computed value A'' available to the corresponding signer to enable the signer to update its credential by replacing the previous value of A with A''.
2. When the size of the group with all signers is small, the above updating process can be simplified as follows. Let $\mathtt{G}$ be the group, and suppose the value of e_i for each $i \in \mathtt{G}$ is known to all the members of $\mathtt{G}$. Let $\mathtt{H}$ be a subgroup of $\mathtt{G}$, i.e. $\mathtt{H} \subset \mathtt{G}$. If the issuer wants to revoke all the members in $\mathtt{H}$, he randomly chooses an integer $r \in \phi(n)$, computes $Z' = Z^r \bmod n$ and $Z'' = Z \cdot Z' \bmod n$, and publishes
$$a = (Z')^{1/\prod_{i \in \mathtt{G} \setminus \mathtt{H}} e_i} \bmod n,$$
with the announcement that the value of e_k for each $k \in \mathtt{H}$ has been revoked and the value Z in the public key has been updated to Z''. Each member

who is still legitimate, say $j \notin \mathtt{H}$, can update his own membership credential as

$$A'' = A \cdot a^{\prod_{i \in \mathtt{G}, i \notin \mathtt{H}, i \neq j} e_i} \bmod n = A \cdot (Z'')^{1/e_j}.$$

Each revoked member $k \in \mathtt{H}$ is not able to update her membership credential.

3. Another alternative of rekey-based revocation for the RSA-DAA scheme is to use a dynamic accumulator mechanism, such as the one described in [10]. However, the extra proof of accumulator is required in both the signing algorithm and verification algorithm. We omit the details of the modification of the RSA-DAA signature and refer readers to [10].

5.3 Rekey in the LRSW-DAA Schemes

In the Join protocol of a LRSW-DAA scheme reviewed in Subsection 3.2, the signer chooses a secret key $\mathtt{tsk} = f$, and sends the commitment $F = [f]P_1$ to the issuer. The issuer then computes the credential $\mathsf{cre} = (A, B, C)$ for the signer, such that $A = [r]P_1, B = [y]A, C = [x]A + [xyr]F = [x + xyf]A$. The issuer stores at least element C of the newly-computed credential cre for each signer; typically, the issuer will store the complete credential cre along with F. This method is a recall of [16].

To effect the rekey process, the issuer first updates its private key by picking a new x value (here called x''), and replaces the previous value x in its private key. Let $\beta = x''/x$. The issuer updates its public key by computing $X'' = [\beta]X$. The issuer then publishes the new public key (X'', Y).

For each currently legitimate signer, the issuer updates the signer's credential $\mathsf{cre} = (A, B, C)$ by replacing the the credential element C with $C'' = [\beta]C$. The issuer then makes C'' available to the corresponding signer to enable the signer to update its copy of the new credential as $\mathsf{cre}'' = (A, B, C'')$. The signer may optionally wish to verify that the updated credential is an CL signature on f associated with the issuer's new public key; if so, the signer carries out the CL signature verification in respect of signature (A, B, C'') on the signed message f. Note that since neither the value of the private signing key f, the value of the key commitment F, nor the values A and B of the credential are changed during the updating, so this updating process does not require the TPM to be involved.

5.4 Rekey in the SDH-DAA Schemes

The Join protocol in a SDH-DAA scheme reviewed in Subsection 3.3 is as follows. The signer chooses a secret key $\mathtt{tsk} = f$, and sends the commitment $F = [f]h_1$ to the issuer. The issuer then computes the credential $\mathsf{cre} = (A, x)$ for the signer, such that $A = [1/(\gamma + x)](g_1 + F)$. During the join protocol for each signer, typically, the issuer will store the complete credential cre along with F. To update the issuer's key and the DAA credential of each currently-legitimate signer, there are at least three simple methods as follows:

1. The issuer chooses a new value for $g_1 \in \mathbb{G}_1$ (here called g_1''), and for each currently-legitimate signer computes a new value for A, here called A'', as:
$$A'' = [1/(\gamma + x)](g_1'' + F).$$
The issuer then publishes g_1'' to replace the value of g_1 in its previously published public key. For each updated credential, the issuer makes the newly computed value A'' available to the corresponding signer to enable the signer to update its credential by replacing the previous value of A with A''.
2. Alternatively, the issuer does not compute and publish the value A''. The issuer chooses a random $g_1' \in G_1$ and sets $g_1'' = g_1 + g_1'$ as part of the new public key. The issuer computes and publishes an updating value A' for each legitimate signer, where
$$A' = [1/(\gamma + x)]g_1'.$$
Each legitimate member than updates his own membership credential by computing
$$A'' = A + A' \in \mathbb{G}_1.$$
Note that $A'' = [1/(\gamma + x)](g_1 + F) + [1/(\gamma + x)]g_1' = [1/(\gamma + x)](g_1'' + F)$.
3. The issuer first updates its private key by deriving a new value for γ (here called γ''), and replaces the previous value for γ in its private key. For each currently legitimate signer, the issuer updates the signer's credential cre by replacing the previous value of the credential element A with
$$A'' = [1/(\gamma'' + x)](g_1 + F).$$
The issuer then publishes $w'' = [w'']g_2 \in \mathbb{G}_2$ to replace the previous value of w in its public key, and for each updated credential, makes A'' available to the corresponding signer to enable the signer to update its copy of the credential as (A'', x). The signer may optionally wish to verify that the updated credential is a signature on f associated with the issuer's new public key; if so, the signer checks whether $e(A'', w'' + [x]g_2) = e(g_1 + F, g_2)$ holds.

For any of the above revocation methods introduced in this section, we do not require the TPM to be involved in the updating process.

6 Verifier-Local Revocation

6.1 An Overview

In an ordinary digital signature scheme based on public key infrastructure (PKI), if a verifier holds a certificate revocation list (CRL), the verifier is allowed to distinguish whether or not each received signature is signed under a key in the list. This idea cannot directly be used in an anonymous digital signature scheme, such as a group signature scheme or a DAA scheme, since the verification process in these schemes makes use of the group public key instead of a signer's membership certificate. Thus from the verifier's view, the signature is not connected to the signer's membership certificate.

However, a similar idea can also be used in these two types of anonymous signatures. Instead of using a CRL, if a verifier holds a list of special information for each revoked signer, and it allows the verifier to distinguish whether or not each received signature is signed under a key corresponding to this information, then the revocation process can be done during the verification as well. In group signatures, this type of solution is called Verifier-Local Revocation (VLR) and this type of list is called Revocation List (RL). Boneh and Shacham [3] formalized the concept of a group signature scheme with VLR in which the RL is distributed only to the verifiers, and the revocation check process is transparent to signers.

We adapt the same names of VLR and RL for DAA. However, actually, the concept of VLR in DAA is quite different from the one in group signatures. In a group signature scheme, the content of a RL is based on the property of traceability and provided by the opening authority, which is part of the group manager functionality. By holding the revocation list, the verifier plays the role of the delegatee of the opening authority. In the other words, the opening authority delegates his opening ability to these specific list of signers to the verifier. But in DAA, there is no such an opening authority. Finding the identity of a DAA signer is not a required property for a DAA scheme. However, a verifier playing the role of a service provider can create a shadow identifier for his client, the DAA signer. He can revoke such a DAA signer by revoking its shadow identifier. Unlike the VLR in group signatures, the DAA verifier must enforce a signer to cooperate in a VLR process by including a link proof associated with such a shadow identifier in his signature; that fortunately is a designed feature of DAA.

Each verifier can build his own revocation list based on his own experiences with each signer. More specifically, in the literature, there are the following types of Revocation Lists (RLs) in a DAA scheme:

`RL1` The first type of RLs is a DAA rogue list, denoted by RogueList. The content of RogueList is the DAA secret key `tsk` of each corrupted TPM. The DAA verification process includes the rogue list check. This idea is a basic DAA functionality as introduced in [4].

`RL2` The second type of RLs is a DAA basename related DAA secret key `tsk` link list. The content of the RL is a set of signatures signed with a special basename. This is a black list maintained by the verifier. If a signer has agreed to use the same basename, the verifier can reject any signature created under a key listed in the RL. This is another basic DAA functionality in [4].

A basename related VLR solution can also be seen as a behavior-based revocation method. A behavior-based revocation solution is an application oriented solution. Some application requires a signer to create a certain number of signatures to a particular event; for example, [17] shows such an application of using a DAA scheme to build a threshold anonymous announcement solution for vehicular ad-hoc networks (VANETs).

RL3 The third type of RLs is a list of all unwelcome DAA signatures. If a verifier maintains a black list with all the signatures which upset the verifier, the verifier can use this list as his RL. Alternatively, the verifier can receive a black list of signatures from the issuer. The verifier has to force a signer to proof his DAA signing key is not the same as anyone in the list. This proof is added to an ordinary DAA signature as a new part of the modified DAA signature and its existence is mandatory for such a DAA signature to be accepted. This idea was introduced in [7] and formalized in [8].

In this paper, we provide a new DAA VLR method, which uses the following type of RLs:

RL4 The forth type of RLs is a DAA basename related DAA membership credential cre link list. The content of the RL is a set of signatures signed with a special basename. This is a black list maintained by the verifier. If a signer is agreed to use the same basename, the verifier is able to reject any signature signed under a credential cre listed in the RL. The different from RL2 is that it provides the *key+credential+verifier revocation* rather than *key+verifier revocation*, as defined in Subsection 2.2.

The details of this method used in the RSA-DAA scheme and the SDH-DAA scheme will respectively be explained in the following two subsections. We have not figured out how this idea can be used in the LRSW-DAA schemes. Finding a similar linking proof for the LRSW-DAA scheme is an interesting open issue.

6.2 New VLR in the RSA-DAA Scheme

Recall that in the RSA-DAA scheme as reviewed in Subsection 3.1, a DAA signature is a proof of knowledge of values f_0, f_1, A, e, and v such that

$$A^e R_0^{f_0} R_1^{f_1} S^v \equiv Z \bmod n \text{ and } N_V \equiv \zeta^{f_0 + f_1 2^{\ell_f}} \bmod \Gamma.$$

The value ζ is based on a given verifier's base name bsn. Let G_Γ be the subgroup of $\mathbb{Z}_\Gamma^*$ of order ρ and H_Γ be a collision-resistent hash function $H_\Gamma : \{0,1\}^* \rightarrow G_\Gamma$. There are two options: if $\mathsf{bsn} = \perp$, $\zeta \in_R G_\Gamma$ created by randomly choosing, and if $\mathsf{bsn} \neq \perp, \zeta = H_\Gamma(\mathsf{bsn})$. The value N_V enables the verifier to check the pair of f_0 and f_1 is not a rogue pair and also to link two signatures if $\mathsf{bsn} \neq \perp$ is used in both the signatures.

We now propose a new linkable proof based on both the values e and $f = f_0 + f_1 2^{\ell_f}$, instead of f only. It is associated with a given verifier's basename bsn represented as a pair bsn_f and bsn_e, which may or may not be the same. Let H_f and H_e be two different hash-functions $H_f : \{0,1\}^* \rightarrow G_\Gamma$, $H_e : \{0,1\}^* \rightarrow G_\Gamma$. The modification to the original RSA-DAA scheme is as follows:

– Compute ζ_f and ζ_e in one of the following three cases:
  1. If $\mathsf{bsn}_f = \perp$, choose ζ_f and ζ_e randomly from G_Γ.
  2. If $\mathsf{bsn}_f \neq \perp$ and $\mathsf{bsn}_e = \perp$, set $\zeta_e = 1$ and compute $\zeta_f = H_f(\mathsf{bsn}_f)$.

   3. If $\mathsf{bsn}_f \neq \perp$ and $\mathsf{bsn}_e \neq \perp$, compute $\zeta_f = H_f(\mathsf{bsn}_f)$ and $\zeta_e = H_e(\mathsf{bsn}_e)$.
- Compute $N_V = \zeta_f^f \cdot \zeta_e^e \bmod \Gamma$,
- Compute $\tilde{N}_V = \zeta_f^{r_f} \cdot \zeta_e^{r_e} \bmod \Gamma$, where r_f and r_e are the same values as in the original RSA-DAA scheme (refer to Section 3.1).
- Replace the values N_V and $\tilde{N}_V$ in the original RSA-DAA scheme with these new values.

What is in the revocation list is the value N_V. Obviously Case 1 and Case 2 play the same roles as the random name mode and the base name mode of an ordinary DAA scheme, respectively. In Case 3, when two signatures include the same e value and f value and use the same base name bsn, they are linked. This case can be used in the situation that a signer does not want to provide the link based on the (f_0, f_1) values, since they might be used for different applications, but does not mind to provide the link based on the e and f values, since that is used for a particular application.

This new mechanism enables us to achieve the *key+credential+verifier revocation* as well as the *key+verifier revocation*. The extra operation (computing ζ_e, ζ_e^e and $\zeta_e^{r_e}$) in this modification can be done by the host, and the TPM's workload does not increase. For a platform host, this extra cost is trivial.

6.3 New VLR in the SDH-DAA Schemes

Recall that in a SDH-DAA scheme as reviewed in Section 3.3, a DAA signature is a proof of knowledge of the values f, A and x such that

$$A = [1/(\gamma + x)](g_1 + [f]h_1) \text{ and } K = [f]J.$$

The value J is computed based on a basename of a given verifier. There are two options: if $\mathsf{bsn} = \perp$, J is chosen at random in G_1 (or another suitable cyclic group; for simplicity we omit difference from this); if $\mathsf{bsn} \neq \perp$, $J = H(\mathsf{bsn})$ with a hash-function $H : \{0,1\}^* \to \mathbb{G}_1$.

We propose a new linkable proof based on both the values x and f. Let H_f be a hash-function $H_f : \{0,1\}^* \to \mathbb{G}_1$ and H_x be a hash-function $H_x : \{0,1\}^* \to \mathbb{G}_1$. Given a basename $\mathsf{bsn} = (\mathsf{bsn}_f, \mathsf{bsn}_x)$, the modification to the original SDH-DAA scheme is as follows:

- Compute J_f and J_x in one of the following three cases:
   1. If $\mathsf{bsn}_f = \perp$, choose J_f and J_x randomly from G_1.
   2. If $\mathsf{bsn}_f \neq \perp$ and $\mathsf{bsn}_x = \perp$, set $J_x = O$ and compute $J_f = H_f(\mathsf{bsn}_f)$, where O is point at infinity.
   3. If $\mathsf{bsn}_f \neq \perp$ and $\mathsf{bsn}_x \neq \perp$, compute $J_f = H_f(\mathsf{bsn}_f)$ and $J_x = H_x(\mathsf{bsn}_x)$.
- Compute $K = [f]J_f + [x]J_x \in \mathbb{G}_1$,
- Compute $\tilde{K} = [r_f]J_f + [r_x]J_x \in \mathbb{G}_1$, where r_f and r_x are the same values as in the original SDH-DAA scheme (refer to Section 3.3).
- Replace the values K and $\tilde{K}$ in the original SDH-DAA scheme with these new values.

What in the revocation list is the value K. The discussion on the previous linking proof for the RSA-DAA scheme is also suitable for this scheme. In summary, the first two cases are the same as the random base mode and name base mode of the original DAA scheme; the third case provides the *key+credential+verifier revocation.* The computation required in this modification can be done by the host, rather than the TPM.

7 Comparisons and Discussions

In this section, we compare different revocation methods and discuss how they can be used together. We start with performance measurements of a DAA revocation mechanism.

7.1 Performance Measurement

We identify the following performance measures of a DAA revocation mechanism:

1. Computational cost of Rekey by an issuer.
2. Computational cost of Rekey by a TPM.
3. Computational cost of Rekey by a host.
4. Increased computational cost of Sign by a TPM.
5. Increased computational cost of Sign by a host.
6. Increased size of a DAA signature σ.
7. Increased computational cost of Verify by a verifier.
8. Increased size of an issuer's public key ipk.

In these measures, the increased computational cost covers the extra computational operation for the purpose of revocation, and the increased size covers the extra length again for the purpose of revocation. Note that items 1-3 above apply for rekey-based revocation mechanisms whereas items 4-8 above are related to VLR mechanisms.

7.2 Comparisons of VLR Methods

We first compare the four VLR methods that are described in Section 6. The comparison is summarized in Table 1. As for the performance for a revocation process, RL2 has the best efficiency, whereas RL3 is least efficient. None of the four VLR revocation methods requires changing the size of a group public key. For RL1, neither the TPM nor the host needs to perform additional computation on the top of the ordinary DAA signing operation; the verifier needs to compute one exponentiation (denote as E in Table 1) per item in the revocation list. For RL2, again the signer does not need to do additional work besides computing an ordinary DAA signature; the verifier only needs to check the pseudonym in the signature (the N_V value in RSA-DAA or the K value in LRSW-DAA and SDH-DAA) against its revocation list. For RL4, the ordinary Sign algorithm is slightly modified by adding a credential linking proof into the secret key linking proof, so the signer's operation is slightly more expensive than the original one, but this extra job is constant and not linear to the size of the revocation list, and it can be done by the host rather than the TPM; therefore we consider this extra cost is trivial. The verifier's operation is the same as in RL2. For RL3, both the TPM and the verifier need to provide extra operation in the revocation check, i.e., the TPM uses zero-knowledge proof to prove that it did not create those signatures in RL3, and the verifier validates the proof. Therefore RL3 is more expensive.

Table 1. A comparison of different DAA VLR methods

	RL1	RL2	RL3	RL4
TPM sign	No impact	No impact	3 Es per item	No impact
Host sign	No impact	No impact	No impact	2 Es per sign
Verify	1 E per item	Linear lookup	2 Es per item	Linear lookup
Signature size	No impact	No impact	Grows linearly	No impact
Public key size	No impact	No impact	No impact	No impact
Unknown tsk	Non-revocable	Revocable	Revocable	Revocable
Random base mode	Revocable	Non-revocable	Revocable	Non-revocable
Name base mode	Revocable	Revocable	Revocable	Revocable
Sharing RL	Yes	No	Yes	No
Multi cred. per tsk	Linkable	Linkable	Linkable	Not linkable

We now compare the revocation capabilities of the VLR methods in DAA. For RL1, a DAA signer is revoked if its signing key is revealed. If the adversary has a DAA signature key that is unknown to the verifier, then the verifier cannot revoke the adversary. For RL2 and RL4, the black list is built up based on the name base mode of DAA signing (i.e., $\mathsf{bsn} \neq \perp$). In other words, if a signer chooses to create the DAA signatures in the random base mode (i.e., $\mathsf{bsn} = \perp$), her signatures are unlinkable and cannot be revoked in RL2 and RL4; unless a verifier decides to reject all signatures in the random base mode. The RL3 method has better revocation capability, as it can revoke a corrupted DAA signer even if his private key is unknown and if he uses the random base mode. The drawback of RL3 is the efficiency.

Observe that although a verifier can build his own RL2 and RL4, he cannot share the revocation lists with other verifiers, since different basenames are used. RL1 and RL3 can be shared among all verifiers. Also observe that RL2 and RL4 have similar functionality and efficiency. The main difference is that RL4 supports *key+credential+verifier revocation* rather than *key+verifier revocation* in RL2.

Note that, if RL4 is used, then we have to modify RL1 and RL3 methods slightly in order to support RL4. For example, in SDH-DAA, the revocation list of RL1 contains a set of revoked f value. To support RL4, the pseudonym K in the signature is computed based on both f and x, therefore, the revoked list contains a set of revoked (f, x) pair.

7.3 Comparisons between Rekey and VLR Methods

The rekey-based revocation solution provides the *key+credential* revocation, and holds the property of backward and forward unlinkability. Any revoked member, say `tsk` and `cre`, is not able to create a signature which can be accepted in the verification process under the updated group public key `ipk`. However, any signatures created by the member `tsk` before the updating should still be valid under the previous group public key before the updating process. If the member is allowed to join another group or to rejoin this group, the member can still use `tsk`. His signature will still be valid. As discussed in Remark 3 of Section 2.2, extra care may be required, since the verifier might not be able to recognize when a signature was created, although he can see when the signer is legitimate.

The significant advantages of rekey-based revocation, compared with the VLR methods, are the following:

– It does not rely on knowledge of a compromised signer's secret, unlike the `RL1` method.
– It does not require any extra proof or verification in the sign and verification process, unlike the `RL3` method, and so is more efficient.
– Since there is no need to update the TPM's secret, the TPM is not involved in the whole revocation process, if the issuer does not want to send the updated credential encrypted under the TPM's endorsement key. This is the same as in `RL1`, `RL2`, and `RL4` methods.

In general speaking, adding an elegant key updating scheme, as these described in Section 5, does neither bring an extra cost to the TPM in the key updating process, nor provide an extra computational cost in the signing process. The only downside is that the issuer needs to update his key to match with the group member shrinks.

7.4 Using Rekey and VLR Approaches Together

In a DAA scheme, we can support both rekey and VLR approaches together. For example, assume the issuer will perform credential update periodically using the rekey methods in Section 5, we can handle VLR methods with the following two options.

1. Between credential update, a verifier can build up his own revocation lists `RL1`, `RL2`, `RL3`, and `RL4`. After each credential update event, the verifier wipes out all the revocation lists and sets them to be empty sets.
2. Observe that `tsk` does not change during credential update. Furthermore, in the credential update methods in Section 5, the e value in RSA-DAA or the x value in SDH-DAA might not change either between credential update. Therefore, if a DAA signing key is revoked in one of the revocation lists (`RL1`, `RL2`, `RL3`, and `RL4`), then the DAA signer is still revoked in the revocation lists after credential update, unless the verifier resets the lists. This allows a verifier to build a different revocation list from the issuer. For example, for

a given TPM, the issuer may think it is valid, and continue updating the DAA credential for the TPM. However, a verifier puts the TPM in one of his revocation lists, then it is up to the verifier whether to reset his revocation lists after credential update.

8 Conclusions

We have discussed how to support revoking group membership in DAA and suggested two categories of DAA revocation approaches, namely rekey-based revocation and verifier-local revocation. We have reviewed the existing DAA revocation mechanisms, proposed a number of new mechanisms, and demonstrated how to use these revocation mechanisms with the existing three types of DAA schemes. Future work includes study on how to modify the DAA security model to incorporate the revocation functions described in this paper and how to make a rigorous security analysis of these DAA mechanisms.

Acknowledgement

We thank the anonymous INTRUST reviewers for their useful comments and suggestions to improve this paper.

References

1. ISO/IEC 11889: Information technology – Security techniques – Trusted platform module (2009)
2. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
3. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Proceedings of 11th ACM Conference on Computer and Communications Security, pp. 168–177 (October 2004)
4. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 132–145. ACM Press, New York (2004)
5. Brickell, E., Chen, L., Li, J.: A new direct anonymous attestation scheme from bilinear maps. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 166–178. Springer, Heidelberg (2008)
6. Brickell, E., Chen, L., Li, J.: Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. International Journal of Information Security 8(5), 315–330 (2009)
7. Brickell, E., Li, J.: Enhanced Privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In: Proceedings of the 6th ACM Workshop on Privacy in the Electronic Society, pp. 21–30 (October 2007)
8. Brickell, E., Li, J.: Enhanced Privacy ID from bilinear pairing for hardware authentication and attestation. In: Proceedings of 2nd IEEE International Conference on Information Privacy, Security, Risk and Trust, pp. 768–775 (2010)

9. Brickell, E., Li, J.: A pairing-based DAA scheme further reducing TPM resources. In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) TRUST 2010. LNCS, vol. 6101, pp. 181–195. Springer, Heidelberg (2010)
10. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
11. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
12. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
13. Chen, L.: A DAA scheme requiring less TPM resources. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Inscrypt 2009. LNCS, vol. 6151, pp. 350–365. Springer, Heidelberg (2010)
14. Chen, L.: A DAA scheme using batch proof and verification. In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) TRUST 2010. LNCS, vol. 6101, pp. 166–180. Springer, Heidelberg (2010)
15. Chen, L., Morrissey, P., Smart, N.P.: Pairings in trusted computing. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 1–17. Springer, Heidelberg (2008)
16. Chen, L., Morrissey, P., Smart, N.P.: DAA: Fixing the pairing based protocols. Cryptology ePrint Archive, Report 2009/198 (2009), `http://eprint.iacr.org/`
17. Chen, L., Ng, S.-L., Wang, G.: Threshold anonymous announcement in VANETs. IEEE Journal on Selected Areas in Communications, Special Issue on Vehicular Communications and Networks (2010)
18. Chen, L., Page, D., Smart, N.P.: On the design and implementation of an efficient DAA scheme. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 223–237. Springer, Heidelberg (2010)
19. Chen, X., Feng, D.: Direct anonymous attestation for next generation TPM. Journal of Computers 3(12), 43–50 (2008)
20. Cheon, J.H.: Security analysis of the strong Diffie-Hellman problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
21. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
22. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
23. Ge, H., Tate, S.R.: A direct anonymous attestation scheme for embedded devices. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 16–30. Springer, Heidelberg (2007)
24. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
25. Nakanishi, T., Funabiki, N.: Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 533–548. Springer, Heidelberg (2005)
26. Trusted Computing Group. TCG TPM specification 1.2 (2003), `http://www.trustedcomputinggroup.org`