On the Design and Implementation of an Efficient DAA Scheme*

Liqun Chen[1], Dan Page[2], and Nigel P. Smart[2]

[1] Hewlett-Packard Laboratories,
Long Down Avenue, Stoke Gifford,
Bristol, BS34 8QZ,
United Kingdom
liqun.chen@hp.com
[2] Computer Science Department,
Woodland Road, University of Bristol,
Bristol, BS8 1UB,
United Kingdom
{page,nigel}@cs.bris.ac.uk

Abstract. Direct Anonymous Attestation (DAA) is an anonymous digital signature scheme that aims to provide both signer authentication and privacy. One of the properties that makes DAA an attractive choice in practice is the split signer role. In short, a principal signer (a Trusted Platform Module (TPM)) signs messages in collaboration with an assistant signer (the HOST, a standard computing platform into which the TPM is embedded). This split aims to harness the high level of security offered by the TPM, and augment it using the high level of computational and storage ability offered by the HOST. Our contribution in this paper is a modification to an existing pairing-based DAA scheme that significantly improves efficiency, and a comparison with the original RSA-based DAA scheme via a concrete implementation.

1 Introduction

An anonymous signature scheme is a special type of digital signature. In common with a conventional signature scheme, an anonymous signature scheme must ensure that only an authorised signer can produce a valid signature. However, given a signature it must also ensure that no unauthorised entity should be able to identify the signer. Another difference is highlighted by the nature of the public keys used to perform signature verification. To verify a conventional signature, the verifier makes use of a single public verification key which is bound to the identity of the signer. In contrast, to verify an anonymous signature the verifier makes use of either a group public key or multiple public keys. In either case the keys are not bound to an individual signer, and the level of anonymity

* The second and third author would like to thank the EU funded project eCrypt-2 for partially supporting the work in this paper. The third author was supported by a Royal Society Wolfson Merit Award.

D. Gollmann, J.-L. Lanet, J. Iguchi-Cartigny (Eds.): CARDIS 2010, LNCS 6035, pp. 223–237, 2010.

provided depends upon the size of the group or the number of public keys. The first anonymous signature schemes were group signatures, introduced in 1991 by Chaum and van Heyst [12].

In this paper we shall concentrate on a specific form of anonymous signature that uses a group public key, namely a signature provided by a Direct Anonymous Attestation (DAA) protocol. The security notions of DAA are different from group signatures, e.g., there is no group manager-based traceability in DAA. For the details of the security model of DAA and differentiation between the group signatures and DAA, we direct the reader to [5,7]. In addition to a number of interesting security and privacy features, DAA has a unique property that makes it an attractive choice. In short, the signer role in DAA is split between

1. a principal signer with limited computational and storage capability but high security assurance, and
2. an assistant signer with greater computational and storage capability, but lesser security.

Concrete realisation of these entities is provided by a TPM and a standard computing platform, termed the HOST, into which the TPM is embedded. The TPM represents the principle signer and holds the secret signing key; the HOST assists the TPM in computing a signature, and satisfying the privacy requirement. Note that the HOST is prevented from learning the secret signing key, and hence from producing a valid signature without interaction with the TPM.

The TPM is a physically secure hardware device designed to enable higher levels of security than are possible using software alone. One can view the TPM as a form of coprocessor capable of storing cryptographic keys and performing limited computational tasks in a secure manner; communication with the TPM is typically performed via a low-bandwidth Low Pin Count (LPC) bus interface. From a functional perspective, the TPM is specified by the Trusted Computing Group (TCG); estimates suggest over 100 million standardised TPM devices are currently in existence, mostly within high-end laptops. This deployment is intended to support a wide variety of applications including full disk encryption, Digital Rights Management (DRM) and, crucially for this work, anonymous digital signatures. Crucially, said applications must be designed with great care so as not to expose the constrained nature of both TPM and LPC bus as a bottleneck.

The concept of DAA, and a concrete scheme, were first introduced by Brickell, Camenisch, and Chen [5]; for a historical perspective, we direct the reader to [6]. This RSA-based scheme, which we term RSA-DAA from here on, was adopted by the TCG and included in version 1.2 of the TPM specification [26]; this TPM specification was recently adopted by ISO/IEC as an international standard [23]. Support for RSA-DAA alone represents around 10% of the TPM resources and as such, schemes that retain the same functionality but do so more efficiently remain an interesting and challenging open problem. One option, which has formed the focus of much recent work, is the development of DAA schemes based on elliptic curves and pairings; we generically term such schemes ECC-DAA from

here on. The advantage of using these underlying building blocks is obvious: both the key and signature length can be much shorter, and computational load placed on the TPM less severe. As a result, ECC-DAA is typically more efficient in computation, storage and communication cost than RSA-DAA.

Main Contributions: To the best of our knowledge, there are six existing ECC-DAA schemes. Brickell, Chen and Li [7,8] proposed the first such scheme, basing it on symmetric pairings. In order to improve flexibility and efficiency, Chen, Morrissey and Smart proposed two extensions [15,16,17] based instead on asymmetric pairings. Although the security of all three schemes is based on both the LRSW [24] and DDH problems, a flaw in the first extension was discovered by Li and further discussed in [14,17]. Three further schemes were proposed by Chen and Feng [19], Brickell and Li [10], and Chen [13] respectively. The security of these schemes is based on the q-SDH [4] and DDH problems. Using previous work as a starting point, this paper makes two main contributions.

Firstly, we make three modifications to the ECC-DAA scheme described in [17]. In summary, these modifications are:

1. The first modification is purely syntactic and implies no change to the security proof from [17]: we simply move computations which could be performed by the TPM to the Host, and vice versa. This has the effect of balancing the workloads of TPM and Host, ultimately reducing the computational load placed on the TPM.
2. Next we replace the public key signature based endorsement key from [17] with a public key encryption based endorsement key, combined with a Message Authentication Code (MAC). This mirrors more closely how the authentic channel is created in the currently deployed RSA-DAA scheme.
3. Finally we replace the root key, used by the Issuer in generation of the TPM DAA secret key, with a small subset of public system parameters. We also remove the requirement for the TPM to verify a certificate chain (from the root public key of the Issuer to the current public key) in every join process. This verification was required in all previous DAA schemes, including RSA-DAA, and has two main purposes: firstly to allow the TPM DAA secret key to have a different life-cycle from the Issuer public key, and secondly to to avoid the TPM accepting an arbitrary key that does not belong to a given issuer. Our modification is based on two facts:
   (a) in our new DAA scheme the TPM operations are strictly limited to the small subset of public system parameters and do not actually use the issuer current public key, and
   (b) the public system parameters can have a much longer life-cycle than the Issuer public key.

   The modification vastly reduces the TPM workload in the Join protocol.

From here on, and unless otherwise specified, one can read ECC-DAA as meaning our modified scheme as described in Section 2.

Secondly, we demonstrate how the ECC-DAA scheme can be implemented and evaluate it via a concrete comparison with the incumbent RSA-DAA. In particular, we present experimental results that illustrate various implementation

options and compare aspects of the schemes (e.g., efficiency and communication cost) using a commodity computing platform (that represents the HOST) and an embedded computing platform (that represents the TPM).

2 The Pairing-Based ECC-DAA Scheme

A DAA scheme involves a set of issuers, signers, and verifiers. An ISSUER is in charge of verifying the legitimacy of signers, and of issuing a DAA credential to each signer. A signer, which due to the split role is a pair of HOST and associated TPM, can prove membership to a VERIFIER by providing a DAA signature; this requires the signer holds a valid DAA credential. The VERIFIER can verify the membership credential from the signature, but it cannot learn the identity of the signer. Linkability of signatures issued by a HOST TPM pair is controlled by an input parameter bsn (standing for "base name") which is passed to the signing operation. There is assumed to be a list $\mathsf{RogueList}$ which contains a list of TPM secret keys which have been compromised. Based on these definitions, the rest of this section describes our ECC-DAA scheme, which is based on the scheme of [17] and relies on the use of asymmetric pairings.

NOTATION: Throughout the constituent protocols and algorithms, we let $\mathfrak{I}$, $\mathfrak{H}$ and $\mathcal{V}$ denote the set of all ISSUER, HOST and VERIFIER entities; the set of all TPM entities is denoted by $\mathfrak{M}$. The value of bsn will be used by the signer/verifier to link signatures, if $\mathsf{bsn} = \perp$ then this implies that signatures should be unlinkable.

If S is a set, we denote the act of sampling from S uniformly at random and assigning the result to the variable x by $x \leftarrow S$. We let $\{0,1\}^*$ and $\{0,1\}^t$ denote the set of binary strings of arbitrary length and length t respectively. If A is an algorithm, we denote the action of obtaining x by invoking A on inputs $y_1, \ldots, y_n$ by $x \leftarrow A(y_1, \ldots, y_n)$, where the probability distribution on x is determined by the internal coin tosses of A. Finally, we use $[x]P$ to denote the scalar multiplication of an elliptic curve point P by some integer x.

NOTE: Before proceeding with the description of our scheme, we note a general issue that needs to be considered throughout. Specifically, every group element received by any entity needs to be checked for validity, i.e., that it is within the correct group; in particular, it is important that the element does not lie in some larger group which contains the group in question. This strict stipulation avoids numerous attacks such as those related to small subgroups. When asymmetric pairings are used, as here, this is particularly important since $\mathbb{G}_1$ and $\mathbb{G}_2$ can be considered as distinct subgroups of a large group $\mathbb{G}$. If communicated group elements are actually in $\mathbb{G}$, as opposed to $\mathbb{G}_1$ and $\mathbb{G}_2$, then various properties such as anonymity and linkability break down. As a result, we implicitly assume that all transmitted group elements are elements of the specified groups: within our scheme, the use of Type-III pairings [20] allows efficient methods for checking subgroup membership as described by [18] and expanded upon in Section 3.

2.1 The Setup Algorithm

To initialise the system, one needs to select parameters for each protocol as well as the long term parameters for each Issuer. We assume that prior to initialisation each TPM has a private endorsement key $\mathcal{SK}$ embedded into it (e.g., in read-only memory) and that each Issuer has access to the corresponding public endorsement key $\mathcal{PK}$. We also assume a public key IND-CCA encryption/decryption scheme (ENC/DEC) has been selected for use with these keys, and a MAC algorithm (MAC) with key space $\mathcal{MK}$ has been selected in order to achieve authentication.

As explained previously, this latter point is both a minor departure from the ECC-DAA scheme of [17] (in that there, the TPM endorsement key was a signature/verification key pair) and a minor departure from the TCG developed TPM specification [26] (in that there, message integrity was "achieved" by using SHA-1 in [26] instead of a MAC function[1]).

On input of the security parameter 1^t, the Setup algorithm executes the following steps:

1. *Generate the Commitment Parameters* $\mathsf{par}_{\mathrm{C}}$. In this step, three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, of sufficiently large prime order q, are selected. Two random generators are then selected such that $\mathbb{G}_1 = \langle P_1 \rangle$ and $\mathbb{G}_2 = \langle P_2 \rangle$ along with a pairing $\hat{h} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$. Next, two hash functions $H_1 : \{0,1\}^* \mapsto \mathbb{G}_1$ and $H_2 : \{0,1\}^* \mapsto \mathbb{Z}_q$ are selected and $\mathsf{par}_{\mathrm{C}}$ is set to $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{h}, P_1, P_2, q, H_1, H_2)$. Note that in our scheme, and in contrast with existing ECC-DAA schemes, the TPM operations are strictly limited to $\mathbb{G}_1$. This allows a subset of $\mathsf{par}_{\mathrm{C}}$, namely $\mathsf{par}_{\mathrm{T}}$, to be set to $(\mathbb{G}_1, P_1, q)$ and installed on the TPM in preference to $\mathsf{par}_{\mathrm{C}}$.
2. *Generate Signature and Verification Parameters* $\mathsf{par}_{\mathrm{S}}$. Two additional hash functions are selected, namely $H_3 : \{0,1\}^* \mapsto \mathbb{Z}_q$ and $H_4 : \{0,1\}^* \mapsto \mathbb{Z}_q$, and $\mathsf{par}_{\mathrm{S}}$ is set to (H_3, H_4).
3. *Generate the* Issuer *Parameters* $\mathsf{par}_{\mathrm{I}}$. For each $\mathfrak{i}_k \in \mathfrak{I}$, the following steps are performed. Two integers $x, y \leftarrow \mathbb{Z}_q$ are selected, and the Issuer private key isk_k is set to (x, y). Next, the values $X = [x]P_2 \in \mathbb{G}_2$ and $Y = [y]P_2 \in \mathbb{G}_2$ are computed; the Issuer public key ipk_k is set to (X, Y). Then an Issuer value K_k is derived from the Issuer public values. Finally, $\mathsf{par}_{\mathrm{I}}$ is set to $(\{\mathsf{ipk}_k, \mathrm{K}_k\})$ for each Issuer $\mathfrak{i}_k \in \mathfrak{I}$.
4. *Generate* TPM *Parameters.* The TPM generates a public/private key pair $(\mathcal{PK}, \mathcal{SK})$ for the associated endorsement key. In addition, it generates the private secret value DAAseed. Finally, $\mathsf{par}_{\mathrm{T}}$ is set to $(\mathcal{PK}_{\mathfrak{h}})$ for each TPM embedded in some host $\mathfrak{h} \in \mathfrak{H}$.
5. *Publish Public Parameters.* Finally, the public system parameters par are set to $(\mathsf{par}_{\mathrm{C}}, \mathsf{par}_{\mathrm{S}}, \mathsf{par}_{\mathrm{I}}, \mathsf{par}_{\mathrm{T}})$ and published.

[1] We place "achieved" in quotes, since it is a well known that a Merkle–Damgård style hash function, when applied to a concatenation of the message and a key, cannot provide secure message authentication.

NOTE 1: In our scheme, the ISSUER value K_k is derived from a representation of $\mathsf{par}_{\mathrm{T}}$; if the same $\mathsf{par}_{\mathrm{T}}$ is used by multiple issuers, in order to limit K_k to a single issuer, the issuer value K_k can be set by using both $\mathsf{par}_{\mathrm{T}}$ and a unique issuer name. This is an important difference from other existing DAA schemes, including RSA-DAA. In all the previous DAA schemes, K_k is computed to be a representation of the ISSUER root public key. This is used to certify the ISSUER's public key ipk_k so that the ISSUER and TPM can update their keys without synchronising with each other.

However, there may be a long certificate chain between K_k and ipk_k that could result in the Join protocol very inefficient; specifically, the TPM needs to verify said certificate chain. Our modification is based on the fact that the parameter $\mathsf{par}_{\mathrm{T}}$ could be used over a longer timescale than the the ISSUER public/private key pair $(\mathsf{isk}_k, \mathsf{ipk}_k)$. Therefore, the ISSUER could update his key without changing $\mathsf{par}_{\mathrm{T}}$ and without requiring each TPM to update its private key synchronously.

NOTE 2: Each TPM has a single DAAseed, but can create multiple DAA secret keys, even associated with a single issuer. To allow this, a number cnt is used as an additional input to DAA secret key generation: the TPM DAA secret key is generated by using DAAseed, K_k and cnt as input.

2.2 The Join Protocol

This is a protocol between a given TPM $\mathfrak{m} \in \mathfrak{M}$, the corresponding HOST $\mathfrak{h} \in \mathfrak{H}$ and an ISSUER $\mathfrak{i} \in \mathfrak{I}$. The protocol proceeds as shown in Figure 1, and it is virtually identical to that of [17]. The difference is the way that the ISSUER and TPM establish an authentic channel between themselves. For simplicity of analysis, in [17] this mechanism was provided by a digital signature algorithm. However, in practice the TPM will not use a signature algorithm, but an encryption algorithm and an integrity check function. For the sake of privacy, the TCG does not want a TPM to provide a piece of evidence for each transaction. Since each endorsement key is bound to a particular TPM, a signature signed under the endorsement key can be used as such evidence in a public manner, (although the public key encryption mechanism still provides this evidence to an issuer). In this paper we follow this approach; the minor difference from the TPM specification [26] is, as mentioned, that we make use of a MAC function to achieve authentication rather than a hash function as in [26].

2.3 The Sign/Verify Protocols

This is a protocol between a given TPM $\mathfrak{m} \in \mathfrak{M}$, HOST $\mathfrak{h} \in \mathfrak{H}$ and VERIFIER $\mathfrak{v} \in \mathcal{V}$ as described in Figure 2. The main difference between this version and the protocol in [17] is the computation of $W \leftarrow [t]D$ by the HOST, as opposed to the computation of $\beta \leftarrow \hat{h}(S, X)$. This avoids a costly pairing operation by the HOST, and an expensive $\mathbb{G}_T$ operation by the TPM. This advantage comes at the expense of the VERIFIER being required to compute more operations in $\mathbb{G}_1$, but less pairing operations and operations in $\mathbb{G}_T$.

TPM (m)		HOST (h)		Initiator (I)
				ISSUER **Request**
				$k_M \leftarrow \mathcal{MK}$
				$c_I \leftarrow \mathsf{ENC}_{\mathcal{PK}}(k_M)$
				$n_I \leftarrow \{0,1\}^t$
TPM **Response**	$\xleftarrow{\mathsf{comm}_{\mathrm{req}}}$		$\xleftarrow{\mathsf{comm}_{\mathrm{req}}}$	$\mathsf{comm}_{\mathrm{req}} \leftarrow (c_I, n_I)$
$\mathsf{sk}_T \leftarrow \mathsf{PRF}(\mathsf{DAAseed} \Vert \mathsf{K}_I \Vert \mathsf{cnt})$				
$k_M \leftarrow \mathsf{DEC}_{\mathcal{SK}}(c_I)$				
If $k_M = \perp$ then abort				
$\mathsf{str} \leftarrow X \Vert Y \Vert n_I$				$\mathsf{str} \leftarrow X \Vert Y \Vert n_I$
$Q_2 \leftarrow [\mathsf{sk}_T] P_1$				
$u \leftarrow \mathbb{Z}_q; U \leftarrow [u] P_1$				
$v \leftarrow H_2(P_1 \Vert Q_2 \Vert U \Vert \mathsf{str})$				
$w \leftarrow u + v \cdot \mathsf{sk}_T \pmod q$				
$\gamma \leftarrow \mathsf{MAC}_{k_M}(P_1 \Vert Q_2 \Vert v \Vert w)$				
$\mathsf{comm} \leftarrow (Q_2, v, w, \gamma, n_I)$	$\xrightarrow{\mathsf{comm}}$		$\xrightarrow{\mathsf{comm}}$	ISSUER **Response**
				If $n_I \notin \{\mathsf{comm}_{\mathrm{req}}\}$ then abort
				$\gamma' \leftarrow \mathsf{MAC}_{k_M}(P_1 \Vert Q_2 \Vert v \Vert w)$
				If $\gamma \neq \gamma'$ then abort
				$U' \leftarrow [w] P_1 - [v] Q_2$
				$v' \leftarrow H_2(P_1 \Vert Q_2 \Vert U' \Vert \mathsf{str})$
				If $v \neq v'$ then abort
				$\forall \mathsf{sk}'_T \in \mathsf{RogueList}$
				if $Q_2 = [\mathsf{sk}'_T] P_1$ then abort
				$r \leftarrow \mathbb{Z}_q$
				$A \leftarrow [r] P_1$; $B \leftarrow [y] A$
				$C \leftarrow [x] A + [rxy] Q_2$
	$\xleftarrow{B}$		$\xleftarrow{\mathsf{cre}}$	$\mathsf{cre} \leftarrow (A, B, C)$
TPM **Open**				
$D \leftarrow [\mathsf{sk}_T] B$	$\xrightarrow{D}$	**Host Verify**		
		If $\hat{h}(A, Y)$		
		$\neq \hat{h}(B, P_2)$		
		or $\hat{h}(A + D, X)$		
		$\neq \hat{h}(C, P_2)$		
		then abort		

Fig. 1. The Join protocol

It is easy to see that these minor modifications to the Sign and Verify protocols have no affect on the security proof from [17]. Indeed, the VERIFIER still verifies a Camensich-Lysyanskaya credential [11] and a proof of equality of two discrete

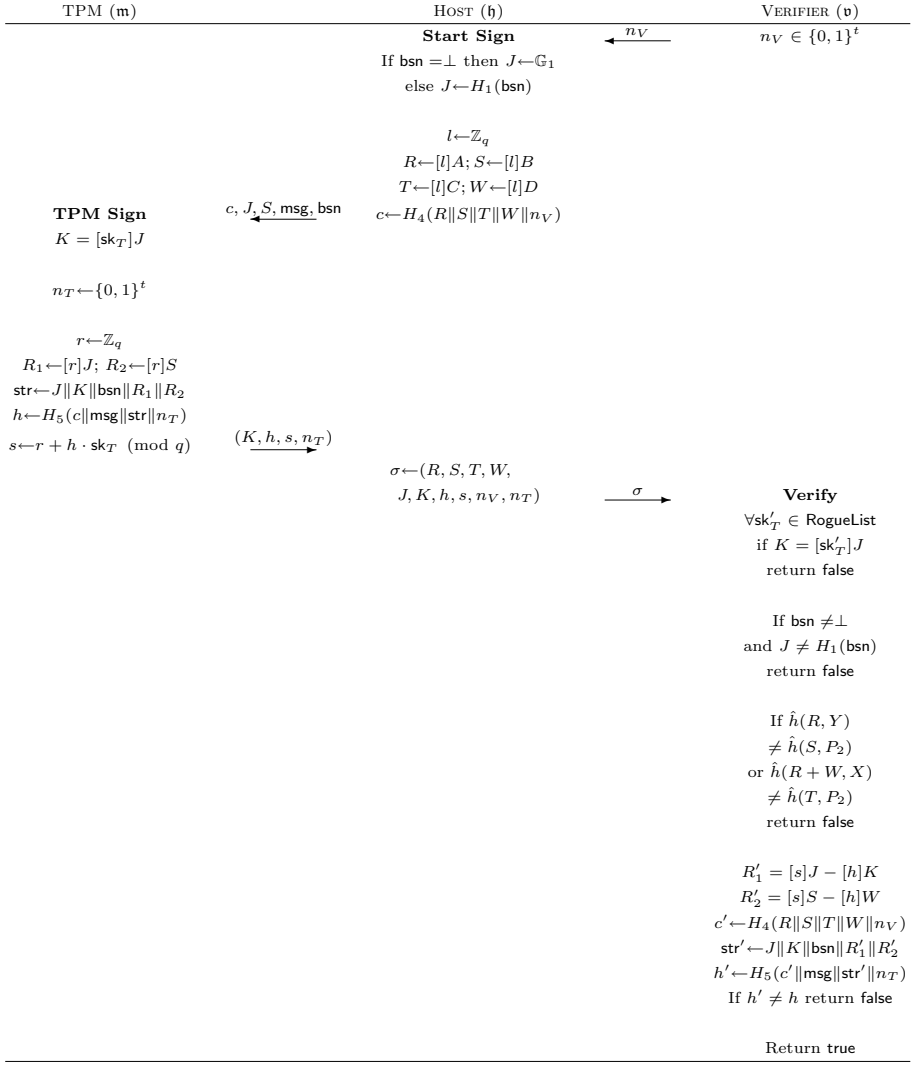

Fig. 2. The Sign/Verify protocol

logarithms. However, now these two verifications are performed in distinct steps rather than being mixed together; we suggested this makes the overall protocol structure simpler to understand. In addition, the modifications produces a more efficient protocol for the Host, TPM and Verifier. In addition, by splitting the proof of equality of discrete logarithms from the credential verification step, we enable the use of batch pairing verification techniques as expanded upon in Section 3.

3 Implementation Details

3.1 RSA-DAA Scheme

Rather than replicate the detail here, we refer the reader to [5] for much of the notation and the RSA-DAA scheme itself. Several implementation details demand some discussion and clarification.

Choice of Parameters: The RSA-DAA scheme was instantiated using the security parameters defined in [5]. That is, the security parameters in [5, Section 4.2] were used to generate public and private Issuer keys as described by [5, Section 4.3]; pertinent features include the 2048-bit n and 1632-bit Γ.

Endorsement Key Algorithm: As described in [5, Appendix B], we altered the Join protocol to include the notion of TPM endorsement. Effectively this means the TPM holds a 2048-bit RSA key, and the PKCS#1 RSAES-OAEP primitive is used as the endorsement key algorithm within Step 4 of the Join protocol; we assume the public key of each TPM utilises a "small" exponent (i.e., $e = 65537$) and interleave additional messages to avoid extra communication steps.

Implementation Options: A central part of each protocol in the scheme is computation of k-term (multi-)exponentiations (i.e., the product of k single-exponentiations) modulo an n and Γ.

Three issues related to this type of computation are worthy of note. Firstly, there is scope for pre-computation based on the "somewhat fixed" bases taken from the per-Issuer public key (e.g., g', g and h). The value of such an approach relates to the frequency of interaction with a given Issuer, and therefore we do not pursue it in our implementation. Secondly, there exists a subtle overhead associated with the fact that various exponents for exponentiation modulo n are larger than the group order; no entity other than the Issuer has knowledge of $\Phi(n)$ and hence they must be used, as presented, in their larger form. Thirdly, we suggest that the diverse range of operand lengths used leads to some natural inefficiency: in short, one must either employ a general-purpose implementation strategy and pay a penalty in terms of performance, or employ a special-purpose implementation strategy and pay a penalty in terms of memory footprint. Depending on the entity (i.e., Issuer, Host, Verifier or TPM) one or other may be unattractive. Examples of this issue include:

- In various steps, entities perform computation modulo n and Γ which are of significantly different lengths: either one adopts a general-purpose strategy that supports all moduli, or a special-purpose strategy for each.
- In various steps, entities compute the result of multi-exponentiation with 2, 3, 4 and 6 terms with varying length exponents. Either one adopts a general-purpose strategy for any number of terms and any exponent length, or a special-purpose strategy or each. As an example, consider that in step 3.a.i of the Sign protocol, the TPM is required to compute

$$R_0^{r_{f0}} R_1^{r_{f1}} S^{r_v} \pmod{n}$$

where r_{f_0} and r_{f_1} are both 688-bit integers and r_v is a significantly larger 2776-bit integer. One could view the resulting mismatch as disadvantageous in the sense that some effort is being "wasted" during a multi-exponentiation,

We opted for general-purpose strategy throughout our implementation, reasoning that this provides a fair comparison given the similar approach in the ECC-DAA case.

3.2 ECC-DAA Scheme

The ECC-DAA under consideration is that described in Section 2; several implementation details demand some discussion and clarification.

Choice of Parameters: To instantiate the ECC-DAA scheme, we use pairing groups based on Barreto-Naehrig (BN) curves [2]. These are elliptic curves of the form

$$E : y^2 = x^3 + b,$$

for $b \neq 0$, where the curve order and the finite field are defined by the polynomials

$$q(s) = 36s^4 - 36s^3 + 18s^2 - 6s + 1,$$
$$p(s) = 36s^4 - 36s^3 + 24s^2 - 6s + 1.$$

To generate such curves, one searches random values of s of the correct form until $q(s)$ and $p(s)$ are both prime; searching for a b that produces a valid elliptic curve over $\mathbb{F}_p$, of order q, is then a simple task. In our implementation we selected $s = -7493989779944505618$ and defined a curve using $b = 18$. This yields roughly 256-bit values for $q(s)$ and $p(s)$ that are hence compatible with AES-128 bit key sizes; alternatively, one could consider the security level as comparable with 3000-bit RSA. Based on this curve, we select

1. The rational points on the curve $E(\mathbb{F}_p)$ as $\mathbb{G}_1$.
2. The order-q subgroup of $\hat{E}(\mathbb{F}_{p^2})$, where $\hat{E}$ is the sextic twist of E available due to the form of BN-curves, as $\mathbb{G}_2$.
3. The order-q subgroup of the finite field $\mathbb{F}_{p^{12}}$, available due to the embedding degree of BN-curves being 12, as $\mathbb{G}_T$.

Elements of the finite field $\mathbb{F}_{p^{12}}$ are represented by a polynomial basis with respect to the polynomial $\chi^{12} + 6$. Using these groups, we implemented the Ate pairing [22], which runs a short full-Miller loop followed by an exponentiation in $\mathbb{G}_T$. We note that more efficient pairing algorithms exist (e.g., R-ate), but made this selection partly based on possible inclusion in the IEEE P1363.3 Identity-Based Public Key Cryptography standard.

Endorsement Key Algorithm: The choice of TPM endorsement key algorithm was selected to be ECIES defined over the group $\mathbb{G}_1$. The KEM component consists of a single element in $\mathbb{G}_1$. For encryption two point multiplications are required, whilst for decryption one point multiplication is required. The DEM

component consisted of AES-128, combined with a CBC-MAC, (actually a variant of CBC-MAC called EMAC was used, which corresponds to MAC algorithm two in the ISO 9797-1 standard).

Credential Verification: In both the Join and Sign/Verify protocols, verification of a blinded Camenisch-Lysyanskaya signature is required. Namely, given $A, B, C, D \in \mathbb{G}_1$ we need to verify whether both

$$\hat{h}(A, Y) = \hat{h}(B, P_2)$$

and

$$\hat{h}(A + D, X) = \hat{h}(C, P_2).$$

To optimise this operation, we use an analogue of the small-exponent batch verification techniques from [3]. Specifically, we select two small exponents $e_1, e_2 \in \mathbb{Z}_q$ whose bit length is half that of q; to verify the two pairing equations we then verify whether

$$\hat{h}([e_1]A, Y) \cdot ([-e_1]B, P_2) \cdot \hat{h}([e_2](A + D), X) \cdot \hat{h}([-e_2]C, P_2) = 1.$$

Thus the verification involving four pairing computations is replaced by one product of four pairings, plus four (relatively short) multiplications in $\mathbb{G}_1$. As surveyed in [21], computing a "product of pairings" is less expensive than computing the pairings independently; the methods improves verification of a blinded Camenisch-Lysyanskaya signature by around 40%.

Subgroup Checking: Recall from Section 2 that group element accepted by some entity within a protocol needs to be verified, i.e., checked to ensure it is within the correct subgroup. Such checking falls into one of three categories:

- Checking whether $X, Y \in \mathbb{G}_2$ can be done by first checking whether $X, Y \in \hat{E}(F_{p^2})$, and then verifying that $[q]X = [q]Y = \mathcal{O}$.
- Checking whether $x \in \mathbb{Z}_q$ or $Y \in \mathbb{G}_1$ is trivial. In the first case we simply need to check whether x is an integer in the range $[0, \ldots, q-1]$, and in the second case we check whether Y lies on the curve $E(\mathbb{F}_p)$. This simplicity is possible because there is no cofactor of $\mathbb{G}_1$ in $E(\mathbb{F}_p)$ as a result of the curve choice.
- We ignore the cost of checking whether $X, Y \in \mathbb{G}_2$ because this is performed once only, by a given entity, on receipt of a public key from some Issuer. That is, we expect the cost to be amortised across all interactions with the Issuer.

3.3 Experimental Results

To evaluate the proposed ECC-DAA scheme, and compare it with RSA-DAA, we present some concrete experimental results. We used two platforms where

1. the Issuer, Host and Verifier entities were represented by a 64-bit, 2.4 GHz Intel Core2 (6600) processor targeted with GCC 4.3.2, and
2. the TPM entity was represented by a (simulated) 32-bit, 33 MHz ARM7TDMI processor targeted with ARM ADS 1.2.

Table 1. Experimental results for RSA-DAA and ECC-DAA. Note that Steps 5 and 6.*c* of the RSA-DAAJoin protocol involve primality testing and, as such, the results have quite a high standard deviation; where operations relate to entries in a rogue list (e.g., Step 4 of the RSA-DAAVerify protocol), the quoted result is per-entry rather than based on an assumed list length.

Join			
Step	Issuer	Host	TPM
1	$--$	$13.38ms$	$--$
2	$--$	$--$	$25.73s$
3	$1.62ms$	$--$	$--$
4.*a*	$--$	$--$	$30.15s$
4.*b*	$0.32ms$	$--$	$--$
4.*c*	$--$	$< 0.01ms$	$--$
4.*d*	$--$	$--$	$< 0.01s$
4.*e*	$--$	$--$	$17.26s$
4.*f*	$--$	$--$	$--$
4.*g*	$46.15ms$	$--$	$--$
5	$138.08ms$	$--$	$--$
6.*a*	$--$	$< 0.01ms$	$--$
6.*b*	$30.61ms$	$--$	$--$
6.*c*	$--$	$72.99ms$	$--$
7	$--$	$--$	$--$
8	$--$	$--$	$< 0.01s$

(a) Performance of the RSA-DAAJoin protocol.

Sign				
Step	Host		TPM	
	bsn $=\perp$	bsn $\neq\perp$	bsn $=\perp$	bsn $\neq\perp$
1.*a*	$2.03ms$	$13.33ms$		
1.*b*	$--$		$1.28s$	
2.*a*	$71.69ms$		$--$	
2.*b*	$--$		$1.27s$	
3.*a.i*	$--$		$30.99s$	
3.*a.ii*	$144.19ms$		$--$	
3.*b.i*	$0.04ms$		$--$	
3.*b.ii*	$--$		$< 0.01s$	
3.*c.i*	$--$		$< 0.01s$	
3.*c.ii*	$< 0.01ms$		$--$	
4	$--$		$--$	

Verify		
Step	Verifier	
	bsn $=\perp$	bsn $\neq\perp$
1	$175.04ms$	$--$
2	$3.99ms$	$--$
3	$--$	$13.31ms$
4	$1.61ms$	$--$

(b) Performance of the RSA-DAASign/Verify protocols.

Join			
Step	Issuer	Host	TPM
Issuer Request	$1.14ms$	$--$	$--$
TPM Response	$--$	$--$	$2.77s$
Issuer Response	$4.16ms$	$--$	$--$
TPM Open	$--$	$--$	$1.12s$
Host Verify	$--$	$46.08ms$	$--$

(c) Performance of the ECC-DAAJoin protocol.

Sign				
Step	Host		TPM	
	bsn $=\perp$	bsn $\neq\perp$	bsn $=\perp$	bsn $\neq\perp$
Start Sign	$4.09ms$	$6.53ms$	$--$	
TPM Sign	$--$		$3.34s$	

Verify		
Step	Verifier	
	bsn $=\perp$	bsn $\neq\perp$
Verify	$47.19ms$	$48.31ms$

(d) Performance of the ECC-DAASign/Verify protocols.

One might argue this software-only approach makes little sense because a TPM will typically be equipped with hardware accelerators for primitives such as RSA, SHA1 and a PRNG. As a result, we stress that our results are indicative only: we attempt to model the asymmetry that exists between entities in terms of

computational ability, and give a relative comparison of the two schemes that relates move directly to a software-based TPM [27].

Our implementation of both schemes was constructed using vanilla C with assembly language fragments for performance-critical sections. Both use SHA-256 as an underlying hash function with a counter-based iteration extending the digest size where appropriate; rather than a cryptographically secure PRNG, both implementations use the C LCG-based PRNG. Within both schemes we take advantage of persistent state where possible: we avoid re-computation of common intermediate results (e.g., between Step 5 and 6.*b* of RSA-DAA) via caching. However, the clear advantage that exists in terms of computation is counter-balanced by an implication for memory footprint that we ignore somewhat.

An important difference is the algorithms used for exponentiation (resp. scalar multiplication):

- Within RSA-DAA and with $k = 1$ term, the HOST, ISSUER and VERIFIER use sliding window (with $w = 4$) exponentiation; with $k > 1$ terms they use Strauss' method (or Shamir's trick) for multi-exponentiation. The more constrained TPM platform also uses sliding window (with $w = 4$) exponentiation when $k = 1$, but repeated single-exponentiation for $k > 1$.
- Within ECC-DAA, all entities use (signed) sliding window (with $w = 4$) scalar multiplication and group exponentiation; there are no instances of multi-exponentiation.

We concede that aggressive specialisation and efficient (multi-)exponentiation techniques (e.g., those described in detail by Möller [25] and Avanzi [1]) would yield improved results for both schemes.

The results are given in Tables 1. For each step in each protocol, we give timings in milli-seconds on the HOST, ISSUER, VERIFIER or (simulated) TPM platform as appropriate. The names for the different stages for the RSA-DAA protocol are taken from the description in [5]. We again stress that the results are indicative only, and they do not include hidden costs such as communication and random number generation. Even so, one can draw several concrete conclusions:

- Aside from performance, the results highlight that the RSA-DAA scheme is significantly more complicated in terms of the number of steps and interaction between entities; we suggest that this hints at a higher communication cost.
- The performance of RSA-DAA has some significant performance bottlenecks in our software-only approach, particularly in terms of computation on the TPM; the availability of hardware accelerators within makes this point moot however. In contrast, the results indicate that our ECC-DAA scheme *is* feasible using a software-only approach. As such, one can either view it as removing the need for dedicated hardware in the TPM (e.g., for modular arithmetic) *or* potentially being significantly faster should similarly dedicated hardware be available.

– Our ECC-DAA scheme not only provides performance advantages for the TPM, but also for the Issuer and Host. What is surprising is that even though the verification in our ECC-DAA scheme requires several pairing operations, it is still faster than the equivalent RSA-DAA verification operation by some margin. In part, this is due to our efficient batching technique, and the specific choice of pairings adopted.
– Finally, we note that the security parameters for our ECC-DAA scheme have been selected to be equivalent to a 128-bit AES security level. Thus, on paper at least, our ECC-DAA scheme has a higher security margin than the RSA-DAA scheme while still delivering the aforementioned performance advantages.

References

1. Avanzi, R.M.: The complexity of certain multi-exponentiation techniques in cryptography. Journal of Cryptology 18, 357–373 (2005)
2. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
3. Bellare, M., Garay, J., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
4. Boneh, D., Boyen, X.: Sort signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
5. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Computer and Communications Security – CCS 2004, pp. 132–145. ACM Press, New York (2004)
6. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation in context. In: Mitchell, C. (ed.) Trusted Computing, ch. 5, pp. 143–174. IEEE, London (2005)
7. Brickell, E., Chen, L., Li, J.: Simplified security notions for direct anonymous attestation and a concrete scheme from pairings. Int. Journal of Information Security 8, 315–330 (2009)
8. 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)
9. Brickell, E., Li, J.: Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In: Privacy in the Electronic Society – WPES 2007, pp. 21–30. ACM Press, New York (2007)
10. Brickell, E., Li, J.: Enhanced privacy ID from bilinear pairing. Cryptology ePrint Archive. Report 2009/095, `http://eprint.iacr.org/2009/095`
11. 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)
12. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
13. Chen, L.: A DAA scheme requiring less TPM resources. In: Int. Conference on Information Security and Cryptology - Inscrypt 2009 (2009) (to appear)

14. Chen, L., Li, J.: A note on the Chen-Morrissey-Smart direct anonymous attestation scheme (preprint)
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.: On proofs of security of DAA schemes. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 156–175. Springer, Heidelberg (2008)
17. Chen, L., Morrissey, P., Smart, N.P.: DAA: Fixing the pairing based protocols. Cryptology ePrint Archive. Report 2009/198, http://eprint.iacr.org/2009/198
18. Chen, L., Cheng, Z., Smart, N.P.: Identity-based key agreement protocols from pairings. Int. Journal of Information Security 6, 213–242 (2007)
19. Chen, X., Feng, D.: Direct anonymous attestation for next generation TPM. Journal of Computers 3, 43–50 (2008)
20. Galbraith, S., Paterson, K., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156, 3113–3121 (2008)
21. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive. Report 2006/172, http://eprint.iacr.org/2006/172
22. Hess, F., Smart, N.P., Vercauteren, F.: The Eta pairing revisited. IEEE Transactions on Information Theory 52, 4595–4602 (2006)
23. ISO/IEC 11889: 2009 Information technology – Security techniques – Trusted Platform Module (2009)
24. Lysyanskaya, A., Rivest, R., 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. Möller, B.: Algorithms for multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
26. Trusted Computing Group. TCG TPM specification 1.2 (2003), http://www.trustedcomputinggroup.org
27. Strasser, M., Stamer, H.: A software-based trusted platform module emulator. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 33–47. Springer, Heidelberg (2008)