Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators (Extended version) ⋆ ⋆⋆

Last updated: July 5, 2007

Ben Smyth[1], Mark Ryan[1], and Liqun Chen[2]

[1] School of Computer Science,
University of Birmingham, UK
{B.A.Smyth, M.D.Ryan}@cs.bham.ac.uk
[2] HP Laboratories,
Bristol, UK
liqun.chen@hp.com

Abstract. The Direct Anonymous Attestation (DAA) scheme provides a means for remotely authenticating a trusted platform whilst preserving the user's privacy. The protocol has been adopted by the Trusted Computing Group (TCG) in the latest version of its Trusted Platform Module (TPM) specification. In this paper we show DAA places an unnecessarily large burden on the TPM host. We demonstrate how corrupt administrators can exploit this weakness to violate privacy. The paper provides a fix for the vulnerability. Further privacy issues concerning linkability are identified and a framework for their resolution is developed. In addition an optimisation to reduce the number of messages exchanged is proposed.

Key words: cryptographic protocol, trusted computing, privacy, anonymity

1 Introduction

1.1 Trusted Computing

Trusted computing is a mechanism by which a server can obtain cryptographically-strong guarantees about the state of a remote platform. Such guarantees can include information about the platform's configuration, the software it is running, the identity of its users and its geographical location. Once in possession of such information the server can make an informed decision as to whether to trust the

⋆ This is an extended version of the paper that appeared in proceedings of the Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks, Lecture Notes in Computer Science (LNCS), volume 4572, pp. 218-231, Springer-Verlag.

⋆⋆ This research was partially funded by the Engineering and Physical Sciences Research Council (EPSRC) under the WINES initiative as part of the UbiVal project.

platform. At the core of the architecture is a hardware device called a Trusted Platform Module (TPM). This chip provides the cryptographic guarantee that the reported data is indeed correct.

Applications for trusted computing include *ad hoc* networks, grid computing and corporate digital rights management (DRM). A mobile *ad hoc* network consists of a number of mobile nodes. Unlike traditional network topologies, *ad hoc* networks do not rely upon a fixed infrastructure. Instead, hosts rely upon each other to become and remain connected. Such technology could be deployed to support a campus network. However nodes may cheat: a selfish user may refuse to forward messages from others, thus becoming a 'freeloader.' Trusted computing can force each node to act in a fair manner. In the Grid Computing application, the resources of a large number of systems are used to tackle computationally expensive problems. The *M4 Message Breaking Project* is an example, and has recently deciphered two of the three previously unsolved German ciphers used during World War II. All Grid Computing projects share a similar impediment. The client may abuse the system by running modified software or may simply return fictitious values. Trusted computing addresses this problem by providing a guarantee that the client is running the legitimate program in the correct manner. In the corporate DRM setting, organisations can be assured that machines are running only authorised software which is capable of enforcing strict policies for the control of documents and electronic mail. Restrictions may prevent printing sensitive corporate data, or forwarding it to external sources.

1.2 Privacy concerns with trusted computing

The aforementioned grid computing example relies upon the ability of a trusted platform to provide a remote attestation. In a similar scenario a situation could exist where the user demands that their identity be protected. The server must therefore only learn that a platform is trusted and not which particular one. Cryptographers and privacy advocates have voiced concerns. The Trusted Computing Group (TCG) has addressed the issue.

The concept of privacy has been widely debated and several taxonomies have been formally proposed [1–3]. For the purposes of this document a privacy preserving protocol is one that satisfies anonymity and unlinkability, the definitions of which have been adopted from Pfitzmann & Köhntopp [2]. *Anonymity* is the state of not being identifiable within a set of agents with the same attributes. The set of agents consists of all those who might cause an action and anonymity becomes stronger as the size of the set increases. Reiter & Rubin [3] liken the notion to "blending into a crowd." In the presence of a large crowd, each member of which is equally likely to have performed an action, it is impossible to establish from whom the action originated. *Unlinkability* (also called *relationship anonymity*) specifies that given two or more items originating from the same agent it is not possible to link them. As a counterexample, two documents bearing the handwritten signature of an individual allow the items to be linked. Unlinkability only has meaning once anonymity has been achieved, since actions can always be linked if the identity of the agent is known. Of course,

privacy is only achievable in a communications protocol if the channel supports anonymity [3, 4].

1.3 Addressing privacy concerns

The solution first adopted by the TCG [5] required a trusted third party, namely a *privacy certification authority* (privacy CA). Each TPM has an embedded RSA key pair called an Endorsement Key (EK) which the privacy CA is assumed to know. In order to attest the TPM generates a second RSA key pair called an Attestation Identity Key (AIK). It sends the AIK, signed by EK, to the privacy CA who checks its validity and issues a certificate for the AIK. The host/TPM is now able to authenticate itself with respect to the certificate. This approach permits two possibilities for the detection of rogue TPMs: firstly the privacy CA should maintain a list of EKs known to be rogue and reject requests from them, secondly if a privacy CA receives too many requests from a particular EK it may reject them. The number of permitted requests should be subject to a risk management exercise and goes beyond the scope of this paper. This solution is problematic since the privacy CA must take part in every transaction which makes use of a new AIK, and thus must provide high availability whilst remaining secure. Furthermore privacy requirements may be violated if the privacy CA and verifier collude.

The Direct Anonymous Attestation (DAA) [6] scheme draws upon techniques developed for group signatures, identity escrow and credential systems. The protocol allows the remote authentication of a trusted platform whilst preserving the privacy of the system's user. It eliminates the need for a trusted third party and has been adopted by the TCG in the current TPM specification [7]. The approach can be seen as a group signature scheme without the ability to revoke anonymity, with an additional mechanism to detect rogue members. In broad terms the *host* contacts an *issuer* and requests membership to a group. If the issuer wishes to accept the request, it grants the host/TPM an *attestation identity credential*. The terms *credential* and *certificate* will be used interchangeably hereafter to mean attestation identity credential. The host is now able to anonymously authenticate itself as a group member to a *verifier* with respect to the certificate. The platform need only contact the issuer once, alleviating the previously discussed bottleneck. Note that if the host wishes to use multiple DAA keys associated with the same issuer then multiple interactions with the issuer will be required.

1.4 Contribution

This paper shows a weakness of the DAA protocol which allows an adversarial issuer and verifier to collude in order to violate the user's privacy. Subsequently, the paper describes how the vulnerability can be fixed. Further privacy issues with regards verifier-linkability are identified and a framework for their resolution is developed. In addition, an optimisation to the protocol is proposed. The paper

presents the DAA protocol in an accessible format which we believe is easier to understand than the original paper.

Structure of paper. The remainder of this paper is structured as follows. Section 2 introduces the cryptographic primitives used by this work. The DAA protocol is explained in Section 3. In Section 4 an informal security analysis of the protocol is conducted, as a result of which a vulnerability is discovered and subsequently corrected. In Section 5 the privacy problems concerning verifier-linkability are identified and a solution is presented. In Section 6 optimisations are proposed to reduce the number of messages exchanged and to improve the efficiency of rogue tagging. An appraisal of the work is presented in Section 7 and future research is considered in Section 8. Finally for completion, the DAA protocol is provided in its entirety, including the security fixes discussed, in the appendices.

2 Preliminaries

2.1 Protocols to prove knowledge

Various protocols which prove knowledge of and relations among discrete logarithms are used by DAA. These protocols will be described using the notation introduced by Camenisch & Stadler [8]. The example below has been adapted from Camenisch *et al.* [6]:

$$PK\{(\alpha, \beta, \gamma) : y = g^{\alpha} h^{\beta} \wedge \tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\gamma} \wedge \alpha \in [u, v]\}$$

It denotes a *"zero knowledge* P*roof of* K*nowledge of integers* α, β, γ *such that* $y = g^{\alpha} h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\gamma}$ *holds, where* $\alpha \in [u, v]$*."* The values $y, g, h, \tilde{y}, \tilde{g}$ and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. Greek letters are used for quantities of the knowledge that is being proved and values kept secret by the prover, while all other values are known to the verifier.

The Fiat-Shamir heuristic [9] allows an interactive zero knowledge scheme to be converted into a signature scheme. A signature acquired in this way is termed a S*ignature* P*roof of* K*nowledge* and is denoted, for example, as $SPK\{(\alpha) : y = g^{\alpha}\}(m)$.

2.2 Cryptographic assumptions

Assumption 1 (RSA assumption). *Given an RSA public key* (n, e) *and a random ciphertext* c *it is hard to compute* m *such that* $m^e = c \pmod{n}$.

Assumption 2 (Strong RSA assumption). *The strong RSA assumption allows the attacker to select the public exponent* e*. Thus the strong RSA assumption states, given* n *and a random ciphertext* c *it is hard to compute* m *and* e *such that* $m^e = c \pmod{n}$*, for an odd public exponent* $e \geq 3$.

Assumption 3 (Decisional Diffie-Hellman (DDH) assumption). *Let* p, q *be primes such that* $q \nmid p - 1$*. Let* g *be a generator of* $\mathbb{Z}_p^*$ *of order* q*. Then for sufficiently large values of* p, q *the tuple* (g, g^a, g^b, b^{ab}) *is computationally indistinguishable from* (g, g^a, g^b, b^c) *where* $a, b, c \in_R [0, q - 1]$.

3 High level overview

This section describes the DAA protocol at a high level. For simplicity in presentation, when the TPM is said to have sent or received a value, the message should be assumed to have been delivered by way of the host. The scheme requires that each issuer and verifier has a unique name, termed a basename, denoted bsn_I and bsn_V respectively.

The TPM is a small chip with limited resources. DAA therefore aims to minimise the operations that it must perform. This is achieved by outsourcing computation to the host whilst maintaining security. A corrupt host should not of course be able to authenticate without the TPM. However, privacy properties need only be guaranteed if the host is not corrupt. Since a corrupted host can always reveal its identity as it controls all external communication. The low level distinction between computation conducted by the host and TPM are described in the appendices.

The protocol is initiated when a host wishes to obtain a credential. This is known as the join protocol and is shown in Figure 1. The TPM creates a secret f value and a blinding factor v'. It then constructs the blind message $U := blind(f, v')$ and $N_I := \zeta_I^f$, where $\zeta_I := (hash(1\|bsn_I))^{(\Gamma-1)/\rho} \pmod{\Gamma}$ and Γ, ρ are components of the issuer's public key. The U and N_I values are submitted to the issuer I. The issuer creates a random nonce value n_e, encrypts it with the public key PK_{EK} of the host's TPM and returns the encrypted value. The TPM decrypts the message, revealing n_e, and returns $hash(U\|n_e)$. The issuer confirms that the hash is correctly formed and is convinced that it is communicating with a valid host/TPM. The issuer checks whether the N_I value stems from a rogue TPM or if it has been seen previously (the issuer might chose to reissue the credential in this case). Rogue tagging will be detailed later. The issuer generates a nonce n_i and sends it to the host. The host/TPM constructs a signature proof of knowledge that the messages U and N_I are correctly formed. The issuer verifies the proof and generates a blind signature on the message U. It returns the signature along with a proof that a covert channel, which could violate privacy, has not been used (for more detail seeAppendix B.4). The host verifies the signature and proof and the TPM unblinds the signature revealing a secret credential v (the signed f).

Once the host has obtained an anonymous attestation credential from the issuer it is able to produce a signature proof of knowledge of attestation on a message m. This is known as the sign/verify protocol and is shown in Figure 2. Intuitively if a verifier is presented with such a proof it is convinced that it is communicating with a trusted platform and the message is genuine. The message m may be either a public part of an Attestation Identity Key (AIK) produced by the TPM or an arbitrary message. If m is an AIK, the key can later be used to sign PCR data or to certify a non-migratable key. Where m is arbitrary its purpose is application dependent. It may for example be a session key. To distinguish between these two modes of operation a variable b is defined. When $b = 0$ the message was generated by the TPM and when $b = 1$ the message was input to the TPM. The process of convincing a verifier that a host has obtained attes-

tation will now be more precisely described. The host engages in communication with the verifier, during which the verifier requires the host to demonstrate that it is indeed a trusted platform. The host and verifier negotiate whether the verifier is able to link transactions and the verifier sends nonce n_v to the host. The host/TPM produce a signature proof of knowledge of attestation on the message $(n_t \| n_v \| b \| m)$, where n_t is a nonce defined by the TPM and m is a message. In addition the host computes $N_V := \zeta^f$, where $\zeta := (hash(1 \| bsn_V))^{(\Gamma-1)/\rho} \pmod{\Gamma}$ or ζ is chosen randomly. The value N_V allows for rogue tagging. In addition, if ζ is not random the N_V value can be used to link different transaction made by the same TPM while not identifying it. Where linkable transactions are used it is also possible to reject an N_V where it has appeared too often (e.g. the verifier may only accept a given value ten times a day).

3.1 Rogue tagging

The DAA protocol is designed so that a known rogue TPM can be prevented from obtaining certification or making a successful claim of attestation to a verifier. A rogue TPM is defined as an entity which has obtained an attestation identity credential and the associated f. Once a rogue TPM is discovered, the attestation identity credential and the f value are distributed to all potential issuers/verifiers who add the value to their rogue list. Note that this does not involve a certificate revocation authority since anybody can verify that the credential is indeed a signature on the f value. On receipt of N_I and N_V values the issuer/verifier can check if the originating TPM is rogue by ensuring $N_I \stackrel{?}{\not\equiv} \zeta_I^{\tilde{f}} \pmod{\Gamma}$ and $N_V \stackrel{?}{\not\equiv} \zeta^{\tilde{f}} \pmod{\Gamma}$ for all values $\tilde{f}$ that are known to stem from rogue TPMs. This check can be done efficiently since the rogue list can be expected to be short and the exponents are relatively small [6].

4 Security analysis

4.1 DAA security properties

The objective of DAA is to provide a mechanism for the remote authentication of a trusted platform whilst preserving the privacy of the system's user. The DAA protocol [6] defines the following security properties:

1. Only a trusted platform is able to authenticate.
2. Privacy of non-corrupt host is guaranteed by the sign/verify protocol:
   (a) Interactions are anonymous.
   (b) Linkability (of transactions) is controlled by the user.
3. Privacy is restored to a corrupted host if malicious software is removed.

Brickell, Camenisch & Chen [6] have shown DAA to be secure in the provable security model under the decisional Diffie-Hellman and strong RSA assumption in the random oracle model. Such proofs are an important part of protocol analysis,

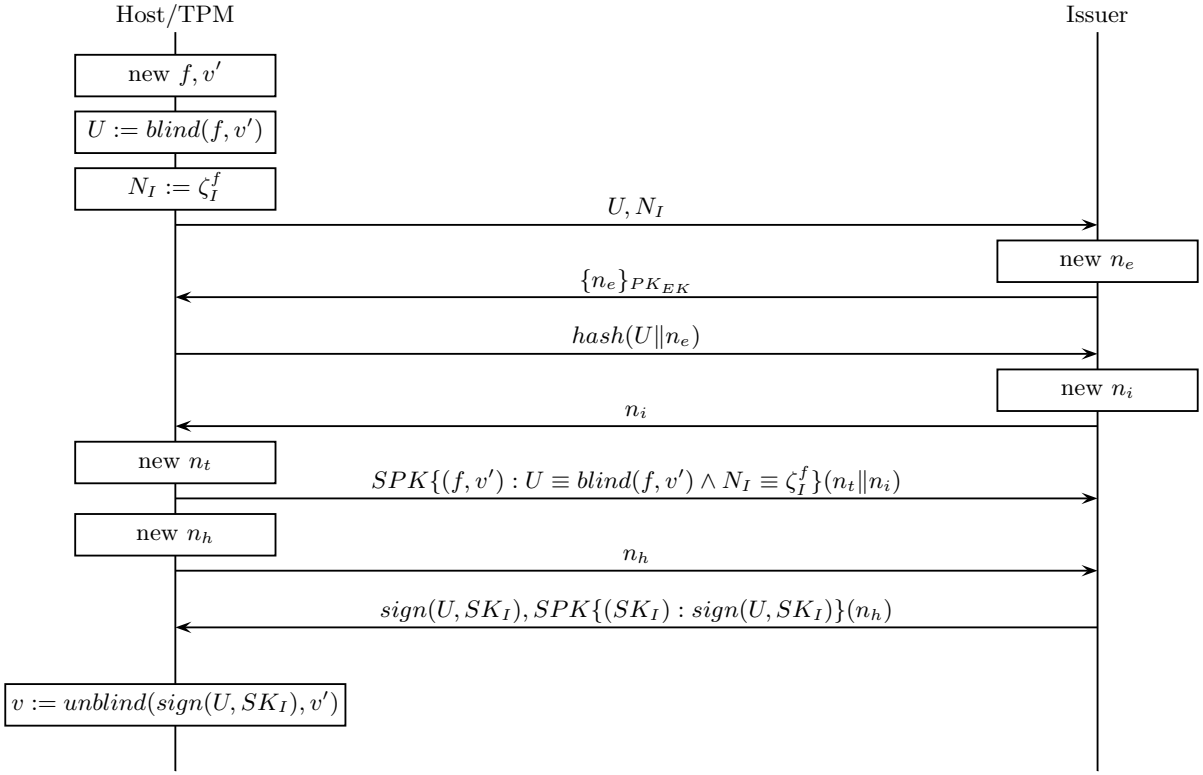


Fig. 1. Join Protocol

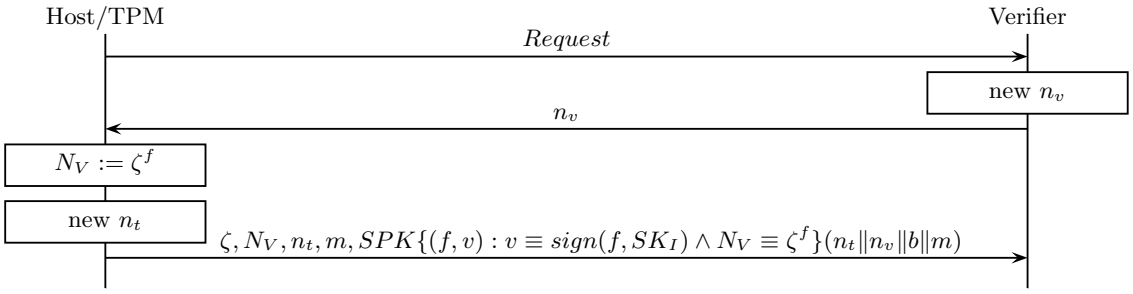


Fig. 2. Sign/Verify Protocol

but they are insufficient. Showing that breaking the scheme is *"essentially as difficult as solving a well-known and supposedly difficult problem"* [10] is a limited view of security and fails to anticipate the majority of attacks on cryptographic systems [11, 12]. Koblitz & Menezes [12] argue that *"throughout the history of public-key cryptography almost all of the effective attacks on the most popular systems have not* [been solving difficult problems (for example integer factorisation)]*, but rather by finding a weakness in the protocol."* Koblitz & Menezes go on to suggest that *"formalistic proofs* [are] *so turgid that other specialists don't even read* [them]*. As a result, proof-checking* [is] *a largely unmet security objective, leaving* [protocols] *vulnerable to attack."* This forms the motivation for an informal security analysis of the DAA scheme.

4.2 Violation of privacy in the presence of corrupt administrators

It is now shown that a colluding issuer and verifier can conspire to break anonymity when linkable transactions are used, violating security properties 2a and 2b. The verifier and issuer conspire to use the same basename, i.e. $bsn_V = bsn_I$. This will result in the host computing $\zeta = \zeta_I$. Recall that $\zeta_I = (hash(1\|bsn_I))^{(\Gamma-1)/\rho} \pmod{\Gamma}$ and $\zeta = (hash(1\|bsn_V))^{(\Gamma-1)/\rho} \pmod{\Gamma}$. The issuer learnt the identity of the host and which N_I value the host used during the join protocol. The verifier receives N_V during the execution of the sign protocol. The host identity is revealed, since $N_I = N_V = \zeta_I^{f_0+f_1 2^{l_f}} = \zeta^{f_0+f_1 2^{l_f}} \pmod{\Gamma}$ and the issuer is able to link the hosts identity with N_I.

The privacy violation relies upon the assumption that an issuer and verifier share the same basename (i.e. $bsn_I = bsn_V$). For example, this assumption holds in the following scenario. An online service provider could act as an issuer during the registration process and a verifier during service usage. This use case is in fact presented[3] by Camenisch *et al.* in earlier work on the idemix (identity mixer) system [13, 14] which forms the basis of the DAA protocol. Under these conditions the issuer and verifier are the same entity and thus it makes logical sense for them to share a single basename. In fact, not doing so could cause confusion. Requiring the user to distinguish between bsn_I and bsn_V values places unnecessary burden on the user and will inevitably lead to their incorrect use. Furthermore, putting in place a procedure for obtaining a unique basename would ultimately require a worldwide governing body. At best this is undesirable since interaction with an authority reintroduces the bottleneck DAA aims to avoid. At worst, such a body is infeasible. It is simply not economic to setup an organisation for the sole purpose of issuing basenames. In addition such a body is likely to charge for its services.

4.3 Fix

The values ζ_I and ζ need not be computed in such a similar manner. It is therefore proposed that the join protocol uses $\zeta_I := (hash(0\|bsn_V))^{(\Gamma-1)/\rho} \pmod{\Gamma}$

[3] See http://www.zurich.ibm.com/security/idemix/idemix-slides.pdf (slide 10).

and the sign/verify protocol uses $\zeta := (hash(1\|bsn_V))^{(\Gamma-1)/\rho} \pmod{\Gamma}$. The collusion between issuer and verifier to break privacy is no longer possible, regardless of whether $bsn_V = bsn_I$. Basenames may now be selected from a single name space as the distinction between issuer and verifier is no longer required.

4.4 Revised DAA protocol

The appendicespresent the complete DAA protocol. The presentation attempts to provide clarity to the reader, incorporates the security fix (Section 4.3) and includes the observation made by Camenisch & Groth [15] for increased efficiency [16]. We believe our presentation is in a more accessible format which is easier to understand than the original paper. To avoid over-complication the optimisations described in Section 6.1 and the construction/use of basenames (Section 5) are not shown; making these changes is trivial.

5 Overcoming problems with DAA basenames

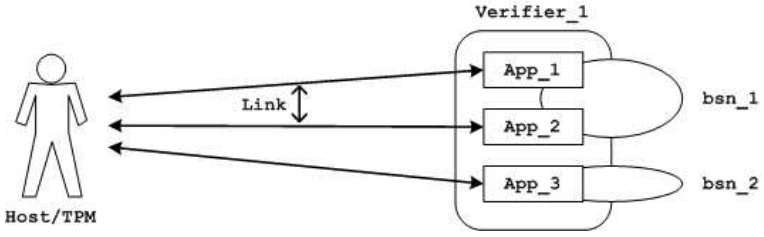
The DAA protocol provides user controlled linkability (security property 2b, Section 4.1). More precisely two modes of operation are defined: verifier-linkable and verifier-unlinkable. Verifier-linkability is controlled by the construction of $N_V := \zeta^f$, where ζ is either derived from a basename or selected randomly (see Section 3). The former construction allows linkability, whereas the latter prevents it. By design DAA therefore provides provisions to link transactions which use the same basename. There are three types of linkable transactions:

1. **Single application linkability** A verifier providing a single application is able to link transactions.
2. **Cross application linkability** A verifier providing multiple applications which share the same basename is able to link transactions between different applications.
3. **Cross verifier linkability** Different verifiers offering several applications which share the same basename are able to link transactions.

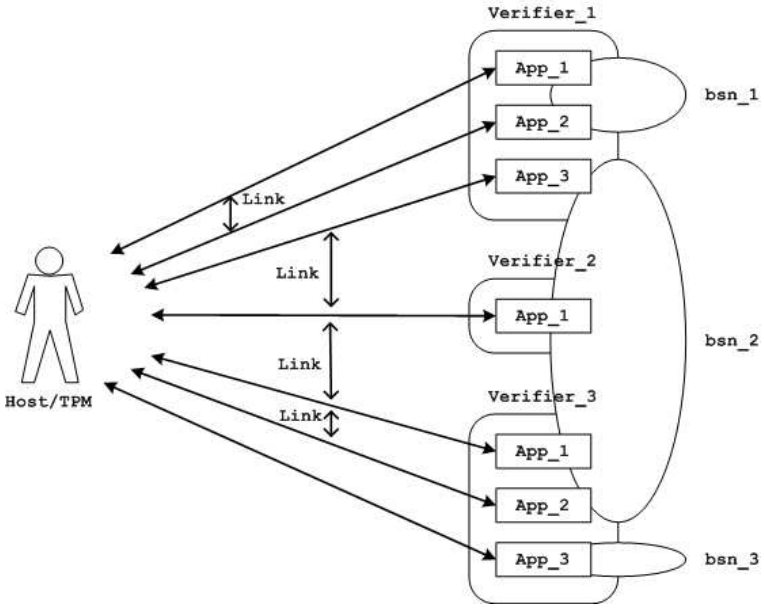
These forms of linkability are shown under various operating conditions in Figure 3. We note that cross issuer linkability - that is linkability between applications with different issuers - is not possible. Since the construction of N_V contains the TPM's secret f value, which in turn incorporates the issuer's public key. Different issuers must use different public keys, thus cross issuer linkability is not possible.

The DAA protocol does not define the security requirements of basenames nor does it specify how basenames should be implemented. This presents two potential problems:

1. **Security properties.** In order to ensure the user controlled linkability, the user must be assured as to which verifier(s) will use a basename and for what application(s). DAA does not provide adequate provisions for this. Thus the host may inadvertently allow linkability between verifiers and/or applications, violating user controlled linkability.



(a) Single verifier



(b) Multiple verifiers

Fig. 3. Linkability in various scenarios

2. **Implementation.** The protocol does not specify how to implement user controlled linkability. A naïve solution is that the host maintains a list of basenames associated with its communicating partners, including DAA issuers and a DAA verifiers, who have been associated with a basename. However, if a DAA key is used for a long time and for many different applications, which is the DAA scheme designed for, maintaining such a list is infeasible for most ordinary users.

Subsection 5.1 defines a technique which will resolve these two issues and Section 5.2 will discuss its use in practice.

5.1 Constructing a basename

The host must be able to uniquely identify with whom a basename should be used and for what application. It is therefore proposed that the basename is constructed from application, verifier and issuer specific data. An example of such information is shown in Table 1. The host is then able to check a basename prior to its use, thus preserving user controlled linkability. The construction of the basename may be undertaken by either the verifier or the host. Alternatively it could be created through negotiation. This decision is left to application developers. When the host is responsible for construction, it may be pre-programmed in the host's software, or determined by the user at run-time for example.

5.2 Using a basename

The host will be required to maintain the information used for constructing basenames as shown in Table 1 and a blacklist of basenames which the host does not want to be used any more. When a new basename is required, the host (and the verifier) will create it based on the particular application. When an existing basename is given it is selected from the list and the host checks that it matches the application specification. The host's blacklist will then be consulted to ensure that the basename has not previously been blacklisted. If desired the

Table 1. Information to be used for computing a basename.

Application	DAA operation	Issuer/verifier data	Date	Other
1. Specification	1. DAA key issuing	1. Issuer identity	1. Start date	1. Random data string∗
2. URL	2. PCR signing	2. Issuer public key	2. Expiry date	2. Policy
3. User ID	3. AIK signing	3. Verifier identity	3. Other	3. Terms & conditions
4. Password	4. External input signing	4. Verifier public key		4. Other
5. Shared key	5. System input signing	5. Auth request		
6. Other	6. Other	6. Auth algorithm		
		7. Other		

∗ This item is listed in the table for completion. The data string must be freshly created by the host and it should only be used for the construction of random basenames.

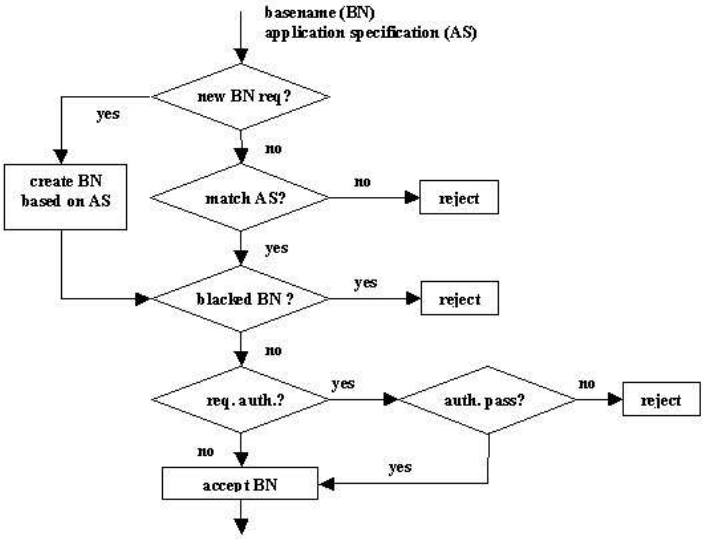


Fig. 4. The proposed solution.

verifier will be asked to authenticate to the host. This process is presented in Figure 4.

Motivating authentication of the verifier. To ensure that a user's affiliations are not learnt by an adversary the host must authenticate the verifier. Although the DAA protocol does not require verifier authentication it is expected that this will be the case in real applications. Standard authentication techniques can be used.

Manageability of basename list. The framework makes basenames more manageable. Basenames are constructed from application specific data and prior to use the host may authenticate the verifier. This means that the host need not maintain a complete list of basenames, since checks can be made to ensure that the basename is suitable for use with a specific application/verifier. This will ensure the list is relatively short. The host need only keep a blacklist if it wishes to avoid certain basenames. Expired basenames can be removed from either list.

6 Optimisations

6.1 Reduction in messages

An optimisation of the join protocol, which reduces the number of messages exchanged from seven to four, is shown in Figure 5. A formal analysis of the

optimisation is beyond the scope of this paper, but an informal discussion is given. The optimisation allows the host to learn n_i earlier than the original protocol. Since this value provides the host with no advantage the protocol is believed to remain secure. The three subsequent messages are all passed from the host to the issuer in succession, it therefore makes no difference to the security of the protocol to concatenate these messages into a single message. It is claimed the optimisation reduces the number of messages whilst maintaining security.

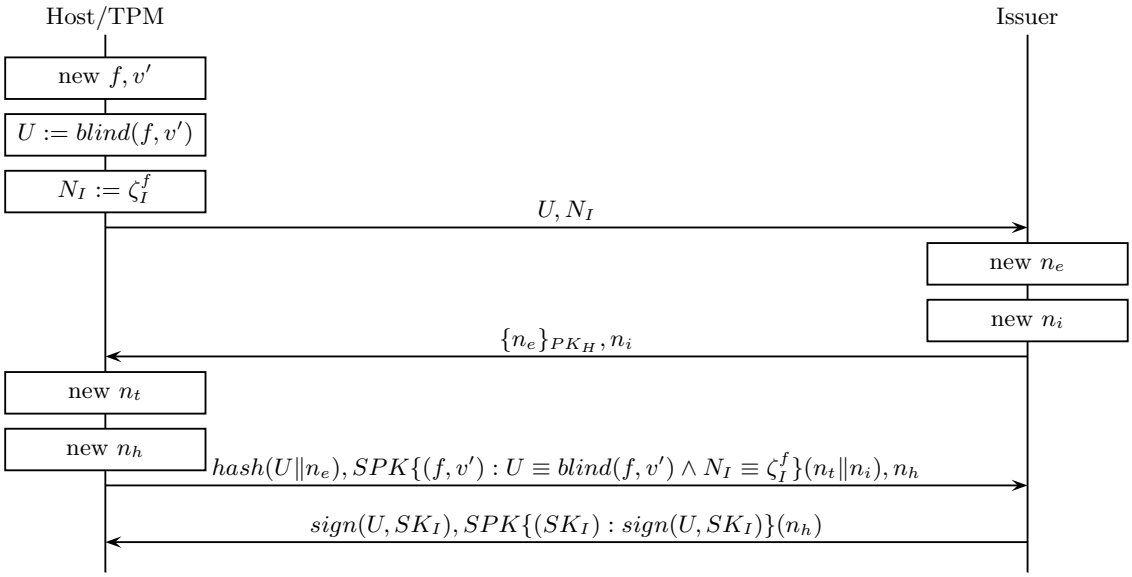


Fig. 5. Optimised Join Protocol

6.2 Rogue tagging

The rogue tagging checks can be optimised. Since ζ_I is a constant in the join protocol the issuer is able to precompute $\zeta_I^{\tilde{f}_0+\tilde{f}_1 2^{l_f}} \pmod{\Gamma}$ for all $(\tilde{f}_0, \tilde{f}_1)$ on the rogue list. This technique can also be applied to the sign/verify protocol when ζ is constant. In the case where ζ is random Brickell, Camenisch & Chen [6] propose that a considerable speedup can be achieved using the batch verification techniques defined by Bellare, Garay & Rabin [17, 18].

7 Conclusion

In this paper a weakness of the Direct Anonymous Attestation protocol is presented. The weakness allows an issuer and verifier to collude to violate the privacy of the host. The vulnerability is fixed by making a minor alteration to the scheme. It is noted that the modification only affects the host part of the protocol (i.e. no modifications need be made to the hardware TPM). The fix is believed to be safe. Proving this formally is the topic of current research. Further privacy issues surround verifier-linkability. The DAA protocol provides inadequate provisions to enable the host to identify with whom, and for which application, a basename may be used. This may result in a privacy violation. The problem is resolved by the development of a framework which facilitates the correct construction/use of basenames. In addition, optimisations to reduce the number of messages exchanged and to improve the efficiency of rogue tagging are presented.

8 Further work

This paper used informal techniques to identify an inadequacy of the DAA scheme. Such methods are not complete and thus formal verification techniques must be applied to give assurance that the protocol is indeed secure. The applied pi calculus is a formalism suitable for modelling DAA which allows us to verify properties using automatic tools. The verification of the scheme remains the topic of future research.

The strength of a security system is inversely proportional to its complexity. DAA provides a esoteric solution to a seemingly simply problem. This work has discovered a vulnerability in its design. Inevitably, implementation will result in intrinsic weaknesses. Further research should aim to establish simpler solutions, ultimately producing systems with greater security and efficiency.

Cryptographers can create secure systems which deliver provably strong security properties. Society, however, is unwilling to accept such systems. Chaum introduced digital cash in the 1980s offering powerful properties including anonymity and unlinkability. Digital cash attracted little attention and was essentially rejected by society over concerns of *"taxation* [evasion] *and money laundering, instability of the exchange rate, disturbance of the money supply, and the possibility of a Black Monday in cyberspace"* [19]. DAA addresses society's concerns using linkability, an impurity which appears undesirable, but is demanded by the real world. Further research should look to enable a more fine-grained approach to the level of privacy provided to the user. Revocable unlinkability could for example be provided. This would provide absolute privacy in normal operation but would allow linkability to be revoked by the collaboration of the issuer and n verifiers.

APPENDICES: The DAA Protocol

A Preliminaries

The DAA protocol draws upon a large combination of primitives from mathematics, which in turn form the building blocks for cryptographic techniques. This section provides a basic overview, for a more complete coverage please refer to [20].

A.1 Notation

The binary string of length l is denoted $\{0,1\}^l$. Concatenation of binary strings α and β is shown as $\alpha\|\beta$. The u least significant bits of the binary string α is shown using $LSB_u(\alpha) := \alpha - 2^u\lfloor\frac{\alpha}{2^u}\rfloor$ and the u most significant bits of the binary string α is denoted $MSB_u(\alpha) := \lfloor\frac{\alpha}{2^u}\rfloor$. It should be noted that $\alpha = MSB_u(\alpha)\|LSB_u(\alpha) = 2^u MSB_u(\alpha) + LSB_u(\alpha)$.

A.2 Group theory

Definition 1 (Cyclic group). *A group G is* cyclic *if there exists an element $g \in G$ such that for every $y \in G$ there exists an integer i where $y = g^i$. Such an element g is called a* generator *of G.*

Fact 1 (Subgroup generator). *If G is a group and $g \in G$ then the set of all the powers of g forms a cyclic subgroup of G. The element g is called a* subgroup generator *and the generated subgroup is denoted $\langle g\rangle$.*

A.3 Protocols to prove knowledge

The DAA protocol applies proofs of knowledge to the group of quadratic residues modulo a safe prime product. As a consequence the prover must demonstrate that elements are indeed quadratic residues since the verifier is unable to do so. The prover is therefore required to show that the square root of the element exists, this can be achieved by executing $PK\{(\alpha) : y^2 = (g^2)^\alpha\}$ or $PK\{(\alpha) : y = \pm g^\alpha\}$ instead of $PK\{(\alpha) : y = g^\alpha\}$, where $\alpha = \log_{g^2} y^2$ which is equivalent to $\alpha = \log_g y$ in the case where $y \in QR_n$ [6].

This section only introduces the notation used to prove knowledge and relations among discrete logarithm. The reader is referred to [20] and the original sources for complete explanations of individual protocols.

Demonstrating possession of a discrete logarithm A proof of knowledge of an element $y \in G$ with respect to base $g \in G$ is denoted $PK\{(\alpha) : y = g^\alpha\}$ [21, 22]. Furthermore it can be generalised to prove knowledge of $y \in G$ with respect to several bases $g_0, \ldots, g_v \in G$, as denoted by $PK\{(\alpha_0, \ldots, \alpha_v) : y = g_0^{\alpha_0} \cdots g_v^{\alpha_v}\}$.

A proof of knowledge of a discreet logarithm $y \in G$ with respect to bases $g \in G$ such that $\alpha \in \pm\{0,1\}^l$ is denoted $PK\{(\alpha) : y = g^\alpha \wedge (-2^l < \alpha < 2^l)\}$ [23–25]. To enable the prover to successfully complete the protocol it is necessary to use a tighter bound $\alpha \in \pm\{0,1\}^{(l-2)/l_\phi}$, where l_ϕ controls the statistical zero knowledge property. Since the protocol uses bit challenges it is not very efficient. Boudot presents an enhanced solution [26] and Camenisch & Michels [23] provide a modification which allows a proof that $(b - 2^l < \alpha < b + 2^l)$ for a fixed offset b.

Proving equality of discrete logarithms A proof of equality of discreet logarithms of group elements $y_0, y_1 \in G$ with respect to bases $g \in G$ and $h \in G$ (i.e. the prover knows α such that $\log_g y_0 \equiv \log_h y_1$), is denoted $PK\{(\alpha) : y_0 = g^\alpha \wedge y_0 = h^\alpha\}$ [27, 28]. Generalisations to prove equalities among $y_0, \ldots, y_v \in G$ to bases $g_0, \ldots, g_v \in G$ are trivial [8].

Proving equality of discrete logarithms in different groups A proof of equality of discrete logarithms $y_0, y_1 \in G_0$ to the bases $g_0 \in G_0$ and $g_1 \in G_1$, where G_0 and G_1 are *different* groups. The orders of the groups are q_0 and q_1 respectively. Let l be an integer such that $2^{l+1} < \min(q_0, q_1)$. The prover can convince the verifier that $\log_{g_0} y_0 \equiv \log_{g_1} y_1$ if α is an element in the tighter range $\{0,1\}^{(l-2)/l_\phi}$ by executing $PK\{(\alpha) : y_0 \overset{G_0}{=} g_0^\alpha \wedge y_1 \overset{G_1}{=} g_1^\alpha \wedge (-2^l < \alpha < 2^l)\}$. The prover and verifier engage in an initial setup during which the prover commits to $\tilde{y} := g^\beta h^\alpha \pmod n$, where $G = \langle g \rangle = \langle h \rangle$ the order of which are unknown (to the verifier) and $\beta \in_R G$. The prover then carries out $PK\{(\alpha, \beta) : y_0 \overset{G_0}{=} g_0^\alpha \wedge y_1 \overset{G_1}{=} g_1^\alpha \wedge \tilde{y} \overset{G}{=} g^\alpha h^\beta \wedge (-2^l < \alpha < 2^l)\}$ in collaboration with the verifier [24].

A.4 Camenisch-Lysyanskaya signature scheme

Direct Anonymous Attestation (DAA) is based upon the Camenisch-Lysyanskaya (CL) signature scheme. Brickell, Camenisch & Chen [6] claim the CL scheme is particularly suited to DAA since it allows *"efficient protocols to prove knowledge of a signature and to retrieve signatures on secret messages efficiently using discrete logarithm, based proofs of knowledge"*. The complete protocol is presented in [29, 30] and is secure under the strong RSA assumption [29, 30]. A summary of the scheme for signing blocks of L messages $m_0, \ldots, m_{L-1}$ and a variant for blind signatures is presented below.

Key generation On input length l_n of the special RSA modulus, choose safe primes p and q of length $\lceil \frac{l_n}{2} \rceil$. Let $n := pq$ and select $R_0, \ldots, R_{L-1}, S, Z \in_R QR_n$.

Message space The message space is the set $\{(m_0, \ldots, m_{L-1}) : m_i \in \pm\{0,1\}^{l_m}\}$, where l_m is a parameter and L is the number of blocks.

Scheme for signing blocks On input message blocks $m_0, \ldots, m_{L-1}$ choose a random prime e of length $l_e \geq l_m + 1$ and select a random number v of length $l_v > l_n + l_m + l_r$, where l_r is a security parameter. Compute A such that $Z \equiv R_0^{m_0} \cdots R_{L-1}^{m_{L-1}} S^v A^e \pmod{n}$. The signature on blocks $(m_0, \ldots, m_{L-1})$ is defined as (A, e, v).

Blind signatures on blocks On input message blocks $m_0, \ldots, m_{L-1}$ select $v' \in_R \{0,1\}^{l_n + l_\phi}$, where l_ϕ is a security parameter and compute $U := R_0^{m_0} \cdots R_{L-1}^{m_{L-1}} S^{v'} \pmod{n}$. Send the blinded message to the signer. On receipt of U the signer chooses a random prime e of length $l_e \geq l_m + 1$, selects $v'' \in_R [2^{l_v - 1}, 2^{l_v} - 1]$ and computes A such that $Z \equiv U S^{v''} A^e \pmod{n}$. The blinded signature (A, e, v'') is returned. The blind signature on the block of messages $m_0, \ldots, m_{L-1}$ is $(A, e, v := v' + v'')$. In order to keep $m_0, \ldots, m_{L-1}$ secret, v must remain secret, A and e can be public.

Verification algorithm To verify that the tuple (A, e, v) is indeed a signature on the block of messages $m_0, \ldots, m_{L-1}$ check that $Z \equiv R_0^{m_0} \cdots R_{L-1}^{m_{L-1}} S^v A^e \pmod{n}$ and ensure $2^{l_e} > e > 2^{l_e - 1}$.

B The Direct Anonymous Attestation (DAA) Scheme

B.1 Security parameters

The security parameters $l_n, l_f, l_v, l_e, l'_e, l_\phi, l_r, l_H, l_\Gamma, l_\rho$ are defined. The purpose of each will now be discussed. The number in parentheses represents the proposed values of these parameters and have been adopted from the original schema [6]. The parameter l_n (2048) is the size of the RSA modulus and l_f (104) is the size of the TPM's secret f_0 and f_1 values. The size of the random v part of the certificate is specified by l_v (2536) and the size of prime e is l_e (368); l'_e (120) is the size of the interval from which the e's are selected. l_ϕ (80) controls the statistical zero knowledge proof property, l_r (80) is needed for the reduction in the proof of security and l_H (160) is the length of the output from the hash function used for the Fiat-Shamir heuristic. The parameter l_Γ is the size of the modulus Γ and finally l_ρ is the size of the order ρ of the sub group of $\mathbb{Z}_\Gamma^*$ that is used for rogue tagging. The scheme requires that: $l_e > l_\phi + l_H + max(l_f + 4, l'_e + 2)$, $l_v > l_n + l_\phi + l_H + max(l_f + l_r + 3, l_\phi + 2)$ and $l_\rho = 2l_f$. Finally, let $H(\cdot)$ and $H_\Gamma(\cdot)$ be two collision resistant hash functions such that $H(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^{l_H}$ and $H_\Gamma(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^{l_\Gamma + l_\phi}$. It should be noted that collision resistant hash functions exist under the strong RSA assumption.

B.2 Setup for the Issuer

The description of how an issuer creates a key pair and a non-interactive proof that the key values are correctly formed is shown below (adapted from [6]). The latter provides an assurance to the host that privacy requirements will be preserved.

1. Choose a special RSA modulus $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$, where p, p', q, q' are all primes and n has l_n bits.
2. Select a random generator g' of QR_n.
3. Choose $x_g, x_h, x_s, x_z, x_0, x_1 \in_R [1, \phi(n)]$ and compute:

$$\begin{array}{lll} g := g'^{x_g} \bmod n & h := g'^{x_h} \bmod n & S := h^{x_s} \bmod n \\ Z := h^{x_z} \bmod n & R_0 := S^{x_0} \bmod n & R_1 := S^{x_1} \bmod n \end{array}$$

4. Generate a group of prime order. Pick random primes ρ and Γ such that $\Gamma = r\rho + 1$ for some r with $\rho \nmid r$, $2^{l_\Gamma - 1} < \Gamma < 2^{l_\Gamma}$ and $2^{l_\rho - 1} < \rho < 2^{l_\rho}$. Select $\gamma \in_R \mathbb{Z}_\Gamma^*$ where $\gamma^{(\Gamma-1)/\rho} \not\equiv 1 \pmod{\Gamma}$.
5. Produce a non-interactive *proof* that g, h, S, Z, R_0, R_1 are computed correctly, i.e. $g, h \in \langle g' \rangle$, $S, Z \in \langle h \rangle$ and $R_0, R_1 \in \langle S \rangle$.
   (a) Choose randoms

$$\begin{array}{ll} \tilde{x}_{(g,1)}, \ldots, \tilde{x}_{(g,l_H)} \in_R [1, \phi(n)] & \tilde{x}_{(h,1)}, \ldots, \tilde{x}_{(h,l_H)} \in_R [1, \phi(n)] \\ \tilde{x}_{(s,1)}, \ldots, \tilde{x}_{(s,l_H)} \in_R [1, \phi(n)] & \tilde{x}_{(z,1)}, \ldots, \tilde{x}_{(z,l_H)} \in_R [1, \phi(n)] \\ \tilde{x}_{(0,1)}, \ldots, \tilde{x}_{(0,l_H)} \in_R [1, \phi(n)] & \tilde{x}_{(1,1)}, \ldots, \tilde{x}_{(1,l_H)} \in_R [1, \phi(n)] \end{array}$$

   (b) Compute *for* $i = 1$ *to* l_H

$$\begin{array}{ll} \tilde{g}_{(g,i)} := g'^{\tilde{x}_{(g,i)}} \bmod n & \tilde{h}_{(h,i)} := g'^{\tilde{x}_{(h,i)}} \bmod n \\ \tilde{S}_{(s,i)} := h^{\tilde{x}_{(s,i)}} \bmod n & \tilde{Z}_{(z,i)} := h^{\tilde{x}_{(z,i)}} \bmod n \\ \tilde{R}_{(0,i)} := S^{\tilde{x}_{(0,i)}} \bmod n & \tilde{R}_{(1,i)} := S^{\tilde{x}_{(1,i)}} \bmod n \end{array}$$

   (c) Compute

$$\begin{aligned} c := H(n \| g' \| g \| h \| S \| Z \| R_0 \| R_1 \| \tilde{g}_{(g,1)} \| \ldots \| \tilde{g}_{(g,l_H)} \| \\ \tilde{h}_{(h,1)} \| \ldots \| \tilde{h}_{(h,l_H)} \| \tilde{S}_{(s,1)} \| \ldots \| \tilde{S}_{(s,l_H)} \| \tilde{Z}_{(z,1)} \| \ldots \| \tilde{Z}_{(z,l_H)} \| \\ \tilde{R}_{(0,1)} \| \ldots \| \tilde{R}_{(0,l_H)} \| \tilde{R}_{(1,1)} \| \ldots \| \tilde{R}_{(1,l_H)}) \end{aligned}$$

   (d) Compute *for* $i = 1$ *to* l_H, where c_i is the ith bit of c

$$\begin{array}{ll} \hat{x}_{(g,i)} := \tilde{x}_{(g,i)} - c_i x_g \bmod \phi(n) & \hat{x}_{(h,i)} := \tilde{x}_{(h,i)} - c_i x_h \bmod \phi(n) \\ \hat{x}_{(s,i)} := \tilde{x}_{(s,i)} - c_i x_s \bmod \phi(n) & \hat{x}_{(z,i)} := \tilde{x}_{(z,i)} - c_i x_z \bmod \phi(n) \\ \hat{x}_{(0,i)} := \tilde{x}_{(0,i)} - c_i x_0 \bmod \phi(n) & \hat{x}_{(1,i)} := \tilde{x}_{(1,i)} - c_i x_1 \bmod \phi(n) \end{array}$$

   (e) Let

$$\begin{aligned} \mathit{proof} := (c, \hat{x}_{(g,1)}, \ldots, \hat{x}_{(g,l_H)}, \hat{x}_{(h,1)}, \ldots, \hat{x}_{(h,l_H)}, \hat{x}_{(s,1)}, \ldots, \hat{x}_{(s,l_H)}, \\ \hat{x}_{(z,1)}, \ldots, \hat{x}_{(z,l_H)}, \hat{x}_{(0,1)}, \ldots, \hat{x}_{(0,l_H)}, \hat{x}_{(1,1)}, \ldots, \hat{x}_{(1,l_H)}) \end{aligned}$$

6. Finally the public key $(n, g', g, h, S, Z, R_0, R_1, \gamma, \Gamma, \rho)$ and *proof* are published. The private key values p', q' are stored by the issuer.

B.3 Verification of the Issuer's public key

A valid public key is essential to ensure the privacy of the host. Incorrectly formed g, h, S, Z, R_0, R_1 values could potential break this property. Furthermore if γ does not generate the subgroup $\mathbb{Z}_\Gamma^*$ the issuer can link transactions (signatures presented to the verifier). The requirement that n is a special RSA modulus ensures that Step 6 of the join protocol is zero knowledge. This is of concern to the issuer and is not important to the security of the host. Verification of an issuer's public key is shown below (adapted from [6]). Note that this verification need only be performed once and not necessarily by every user of a system (i.e. in a multi-user environment it is sufficient for a single user to perform the verification).

1. Verify the *proof*, that is check $g, h \in \langle g' \rangle$, $S, Z \in \langle h \rangle$ and $R_0, R_1 \in \langle S \rangle$. On input:

$$\begin{aligned} proof = (c, \hat{x}_{(g,1)}, \ldots, \hat{x}_{(g,l_H)}, \hat{x}_{(h,1)}, \ldots, \hat{x}_{(h,l_H)}, \hat{x}_{(s,1)}, \ldots, \hat{x}_{(s,l_H)}, \\ \hat{x}_{(z,1)}, \ldots, \hat{x}_{(z,l_H)}, \hat{x}_{(0,1)}, \ldots, \hat{x}_{(0,l_H)}, \hat{x}_{(1,1)}, \ldots, \hat{x}_{(1,l_H)}) \end{aligned}$$

   (a) Compute *for* $i = 1$ *to* l_H, where c_i is the ith bit of c

$$\begin{aligned} \hat{g}_{(g,i)} &:= g^{c_i} g'^{\hat{x}_{(g,i)}} \bmod n & \hat{h}_{(h,i)} &:= h^{c_i} g'^{\hat{x}_{(h,i)}} \bmod n \\ \hat{S}_{(s,i)} &:= S^{c_i} h^{\hat{x}_{(s,i)}} \bmod n & \hat{Z}_{(z,i)} &:= Z^{c_i} h^{\hat{x}_{(z,i)}} \bmod n \\ \hat{R}_{(0,i)} &:= R_0^{c_i} S^{\hat{x}_{(0,i)}} \bmod n & \hat{R}_{(1,i)} &:= R_1^{c_i} S^{\hat{x}_{(1,i)}} \bmod n \end{aligned}$$

   (b) Verify

$$\begin{aligned} c \stackrel{?}{=} H(n \| g' \| g \| h \| S \| Z \| R_0 \| R_1 \| \hat{g}_{(g,1)} \| \ldots \| \hat{g}_{(g,l_H)} \| \\ \hat{h}_{(h,1)} \| \ldots \| \hat{h}_{(h,l_H)} \| \hat{S}_{(s,1)} \| \ldots \| \hat{S}_{(s,l_H)} \| \hat{Z}_{(z,1)} \| \ldots \| \hat{Z}_{(z,l_H)} \| \\ \hat{R}_{(0,1)} \| \ldots \| \hat{R}_{(0,l_H)} \| \hat{R}_{(1,1)} \| \ldots \| \hat{R}_{(1,l_H)}) \end{aligned}$$

2. Check that Γ and ρ are primes. Ensure $\rho \mid (\Gamma - 1)$, $\rho \nmid \frac{\Gamma - 1}{\rho}$ and $\gamma^\rho \equiv 1 \pmod{\Gamma}$.
3. Check that all public key values have the required length.

B.4 Join protocol

The purpose of the join protocol is to enable a host/TPM to acquire a blind CL-signature on a secret f value which can later be used as an anonymous attestation identity credential. Let $PK_I := (n, g', g, h, S, Z, R_0, R_1, \gamma, \Gamma, \rho)$ be the public key of the issuer and PK_I' the long term public key of the issuer used to authenticate PK_I. The value bsn_I is a unique basename assigned to each issuer and *cnt* is an internal counter stored within the TPM. The counter records the number of times the TPM has executed the join protocol. Prior to executing the join protocol the host is assumed to verify that PK_I is authenticated by PK_I'.

The TPM computes f using the issuer's public key, its secret seed $DAAseed$ and counter value cnt. Computing f from the secret seed $DAAseed$ as opposed to a random nonce reduces the computational and storage requirements of the TPM. The counter value allows the TPM to obtain different DAA keys using the same $DAAseed$, alternatively the TPM is allowed to re-run the join protocol using the same cnt value. The TPM splits the f value into two l_f bit messages, the pair (f_0, f_1) allow the computation of smaller exponentials and permit the use of a smaller prime e. The TPM commits to the message pair (f_0, f_1) i.e. $U := R_0^{f_0} R_1^{f_1} S^{v'} \pmod{n}$, where v' is chosen randomly to blind the f_i's. The TPM also computes $N_I := \zeta_I^{f_0 + f_1 2^{l_f}} \pmod{\Gamma}$, for rogue tagging purposes. The TPM forwards U, N_I to the host who sends them to the issuer. The TPM and issuer establish a one-way authentic channel to assure the issuer of the origin of U. The issuer checks whether the f's stem from a rogue TPM or if N_I has been used too many times previously, in which case it aborts. The platform convinces the issuer that U, N_I are correctly formed and that the f_i's are of the appropriate lengths. To grant a certificate the issuer executes the blind CL-signature protocol and sends the platform (A, e, v''). The issuer also provides a signature proof of knowledge (Step 7) that $A \in \langle h \rangle$. The host verifies the proof and is assured that A can be statistically hidden in $\langle h \rangle$, preventing an adversarial issuer from violating privacy. Note that an adversarial issuer could compute $A := b \left(\frac{Z}{US^{v''}} \right)^{1/e} \pmod{n}$, where $b^e = 1$ and $b \notin \langle h \rangle$. Since the sign protocol contains $T = AS^w$ for some random w, the adversarial would be able to link T to A (by testing $T \in \langle h \rangle$) and thus violate privacy [6]. The pair (A, e) are stored by the host and can be publicly known. The host forwards v'' to the TPM, which unblinds the message revealing $v := v' + v''$, a signature on the message pair (f_0, f_1). The explicit details of the join protocol are provided below (adapted from [6]).

1. The host computes $\zeta_I := (H_\Gamma(0 \| bsn_I))^{(\Gamma-1)/\rho} \pmod{\Gamma}$ and sends ζ_I to the TPM.
2. The TPM checks whether $\zeta_I^\rho \stackrel{?}{\equiv} 1 \pmod{\Gamma}$. Let $i := \lfloor \frac{l_\rho + l_\phi}{l_H} \rfloor$ ($i = 1$ for the parameters specified in Section B.1). The TPM computes:

$$f := H\Big(H\big(DAAseed \| H(PK'_I)\big) \| cnt \| 0\Big) \| \\ \ldots \| H\Big(H\big(DAAseed \| H(PK'_I)\big) \| cnt \| i\Big) \pmod{\rho},$$

$$f_0 := LSB_{l_f}(f), \quad f_1 := MSB_{l_f}(f), \quad v' \in_R \{0,1\}^{l_n + l_\phi},$$
$$U := R_0^{f_0} R_1^{f_1} S^{v'} \pmod{n}, \quad N_I := \zeta_I^{f_0 + f_1 2^{l_f}} \pmod{\Gamma}$$

   The TPM forwards U and N_I to the host who sends them to the issuer.
3. The issuer checks that the U value stems from the TPM that owns a given public endorsement key (PK_{EK}):

(a) The issuer chooses $n_e \in_R \{0,1\}^{l_\phi}$, encrypts n_e with PK_{EK} and sends the encryption to the TPM.
(b) The TPM decrypts the value, revealing n_e, computes $a_U := H(U \| n_e)$ and returns a_U to the issuer.
(c) The issuer checks $a_U \stackrel{?}{=} H(U \| n_e)$.

4. The issuer ensures for all $(\tilde{f}_0, \tilde{f}_1)$ on the rogue list that $N_I \stackrel{?}{\neq} \zeta_I^{\tilde{f}_0 + \tilde{f}_1 2^{l_f}} \pmod{\Gamma}$. The issuer also checks that the N_I has not been used too many times. If the issuer finds the platform to be rogue it aborts.
5. The platform proves knowledge of f_0, f_1 and v'. It executes:

$$SPK\{(f_0, f_1, v') : U \equiv R_0^{f_0} R_1^{f_1} S^{v'} \pmod{n} \quad \wedge \quad N_I \equiv \zeta_I^{f_0 + f_1 2^{l_f}} \pmod{\Gamma} \quad \wedge \quad f_0, f_1 \in \{0,1\}^{l_f + l_\phi + l_H + 2} \quad \wedge \quad v' \in \{0,1\}^{l_n + l_\phi + l_h + 2}\}(n_t \| n_i)$$

(a) The TPM chooses $r_{f_0}, r_{f_1} \in_R \{0,1\}^{l_f + l_\phi + l_H}$ and $r_{v'} \in_R \{0,1\}^{l_n + l_\phi + l_H + 2}$. It computes $\tilde{U} := R_0^{r_{f_0}} R_1^{r_{f_1}} S^{r_{v'}} \pmod{n}$, $\tilde{N}_I := \zeta_I^{r_{f_0} + r_{f_1} + 2^{l_f}} \pmod{\Gamma}$ and sends $\tilde{U}, \tilde{N}_I$ to the host.
(b) The issuer selects $n_i \in_R \{0,1\}^{l_H}$ and sends it to the host.
(c) The host computes $c_h := H(n \| R_0 \| R_1 \| S \| U \| N_I \| \tilde{U} \| \tilde{N}_I \| n_i)$ and sends c_h to the TPM.
(d) The TPM picks $n_t \in_R \{0,1\}^{l_\phi}$, computes $c := H(c_h \| n_t)$, calculates $s_{f_0} := r_{f_0} + c \cdot f_0$, $s_{f_1} := r_{f_1} + c \cdot f_1$ and $s_{v'} := r_{v'} + c \cdot v'$. The TPM sends $(c, n_t, s_{f_0}, s_{f_1}, s_{v'})$ to the host, who forwards to the issuer.
(e) The issuer computes:

$$\hat{U} := \pm U^{-c} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_{v'}} \pmod{n} \text{ and } \hat{N}_I := N_I^{-c} \zeta_I^{s_{f_0} + s_{f_1} 2^{l_f}} \pmod{\Gamma}$$

and verifies the proof by checking:

$$c \stackrel{?}{=} H\big(H(n \| R_0 \| R_1 \| S \| U \| N_I \| \tilde{U} \| \tilde{N}_I \| n_i) \| n_t\big),$$
$$s_{f_0}, s_{f_1} \stackrel{?}{\in} \{0,1\}^{l_f + l_\phi + l_H + 1} \quad \text{and} \quad s_{v'} \stackrel{?}{\in} \{0,1\}^{l_n + 2l_\phi + l_H + 1}$$

6. The issuer chooses $v'' \in_R [2^{l_v}, 2^{l_v} - 1]$, a prime $e \in_R [2^{l_e - 1}, 2^{l_e - 1} + 2^{l'_e - 1}]$ and computes $Z := U S^{v''} A^e \pmod{n}$. That is, the issuer produces a blind CL-signature on U.
7. To convince the host that A was correctly formed, the issuer runs the protocol:

$$SPK\{(d) : A \equiv \pm \left(\frac{Z}{U S^{v''}} \right)^d \pmod{n}\}(n_h)$$

(a) The host selects $n_h \in_R \{0,1\}^{l_\phi}$ and sends n_h to the issuer.

(b) The issuer chooses $r_e \in_R [0, \phi(n)]$ and computes:

$$\tilde{A} := \left(\frac{Z}{US^{v''}}\right)^{r_e} \pmod{n},$$
$$c' := H(n\|Z\|S\|U\|v''\|A\|\tilde{A}\|n_h), \quad s_e := r_e - c'/e \mod \phi(n)$$

and sends c', s_e and (A, e, v'') to the host.

(c) The host verifies that e is prime and $e \in [2^{l_e-1}, 2^{l_e-1} + 2^{l'_e-1}]$, computes:

$$\hat{A} := A^{c'} \left(\frac{Z}{US^{v''}}\right)^{s_e} \pmod{n}$$

and checks that:

$$c' \stackrel{?}{=} H(n\|Z\|S\|U\|v''\|A\|\hat{A}\|n_h)$$

(d) The host forwards v'' to the TPM, which in turn stores $v := v'' + v'$.

B.5 Sign protocol

The sign protocol enables the host to generate a signature proof of knowledge of attestation on a message m. Intuitively if a verifier is presented with such a proof it is convinced that it is communicating with a trusted platform and the message is genuine. In addition the host must convince the verifier that it is not rogue. The message m may be either a public part of an Attestation Identity Key (AIK) produced by the TPM or an arbitrary message. If m is an AIK, the key can later be used to sign PCR data or to certify a non-migratable key. Where m is arbitrary its purpose is application dependent. It may for example be a session key. To distinguish between these two modes of operation a variable b is defined. When $b = 0$ the message was generated by the TPM and when $b = 1$ the message was input to the TPM. Let $n_v \in \{0,1\}^{l_H}$ be a nonce generated by the verifier and bsn_V the verifier's basename. The protocol is described below (adapted from [6, 16]), as a result of which the signature $\sigma = (\zeta, T, N_V, c, n_t, s_{\bar{v}}, s_{f_0}, s_{f_1}, s_e)$ will be produced. Many of the secrets involved in the process are actually known to the host. In fact only f_0, f_1, v need to remain secret to the TPM.

1. (a) Depending on whether linkability is desirable the host computes ζ as follows:

$$\zeta \in_R \langle\gamma\rangle \quad \text{or} \quad \zeta := (H_\Gamma(1\|bsn_I))^{(\Gamma-1)/\rho} \pmod{\Gamma}$$

and sends ζ to the TPM.

(b) The TPM checks $\zeta \stackrel{?}{\in} \langle\gamma\rangle$, that is it verifies $\zeta^\rho \stackrel{?}{\equiv} 1 \pmod{\Gamma}$.

2. (a) The host picks $w \in_R \{0,1\}^{l_n+l_\phi}$ and computes $T := AS^w \pmod{n}$.

(b) The TPM computes $N_V := \zeta^{f_0+f_1 2^{l_f}} \pmod{\Gamma}$ and sends N_V to the host.

3. The platform produces a signature of knowledge that T commits to an attestation certificate and N_V was computed using the same secret value f_0, f_1:

$$SPK\{(f_0, f_1, \bar{v}, e) : Z \equiv \pm T^e R_0^{f_0} R_1^{f_1} S^{\bar{v}} \pmod{n} \quad \wedge$$
$$N_V \equiv \pm \zeta^{f_0 + f_1 2^{l_f}} \pmod{\Gamma} \quad \wedge \quad f_0, f_1 \in \{0,1\}^{l_f + l_\phi + l_H + 2} \quad \wedge$$
$$(e - 2^{l_e}) \in \{0,1\}^{l'_e + l_\phi + l_H + 1}\}(n_t \| n_v \| b \| m)$$

(a) i. The TPM chooses random integers $r_v \in_R \{0,1\}^{l_v + l_\phi + l_H}$ and $r_{f_0}, r_{f_1} \in_R \{0,1\}^{l_f + l_\phi + l_H}$ and computes:

$$\tilde{T}_{1t} := R_0^{r_{f_0}} R_1^{r_{f_1}} S^{r_v} \pmod{n}$$
$$\tilde{r}_f := r_{f_0} + r_{f_1} 2^{l_f} \pmod{\rho} \quad \tilde{N}_V := \zeta^{\tilde{r}_f} \pmod{\Gamma}$$

The TPM sends $\tilde{T}_{1t}$ and $\tilde{N}_V$ to the host[4].

ii. The host selects random integers:

$$r_e \in_R \{0,1\}^{l'_e + l_\phi + l_H} \qquad r_{\bar{v}} \in_R \{0,1\}^{l_e + l_n + 2l_\phi + l_H + 1}$$

and computes $\tilde{T} := \tilde{T}_t T^{r_e} S^{r_{\bar{v}}} \pmod{n}$.

(b) i. The host computes:

$$c_h := H(n \| g \| g' \| h \| R_0 \| R_1 \| S \| Z \| \gamma \| \Gamma \| \rho \| \zeta \| T \| N_V \| \tilde{T} \| \tilde{N}_V \| n_v)$$

and sends c_h to the TPM.

ii. The TPM picks $n_t \in_R \{0,1\}^{l_\phi}$, computes $c := H\big(H(c_h \| n_t) \| b \| m\big)$ and sends c, n_t to the host.

(c) i. The TPM computes

$$s_v := r_v + c \cdot v \qquad s_{f_0} := r_{f_0} + c \cdot f_0 \qquad s_{f_1} := r_{f_1} + c \cdot f_1$$

and sends s_v, s_{f_0}, s_{f_1} to the host.

ii. The host computes:

$$s_e := r_e + c \cdot (e - 2^{l_e - 1}) \qquad s_{\bar{v}} := s_v + r_v - c \cdot w \cdot e$$

4. The host outputs the signature $\sigma = (\zeta, T, N_V, c, n_t, s_{\bar{v}}, s_{f_0}, s_{f_1}, s_e)$.

The main difference between the DAA signing protocol and the signature generation of prior schemes [13, 29] is that DAA distributes the computation between the TPM and the host. The TPM only produces a signature proof of knowledge $SPK\{(f_0, f_1, v) : (Z/A^e) \equiv R_0^{f_0} R_1^{f_1} S^v \pmod{n} \wedge N_V \equiv \zeta^{f_0 + f_1 2^{l_f}} \pmod{\Gamma}\}(n_t \| n_v \| b \| m)$, which the host extends to a full DAA signature. Note that the signature produced by the TPM is not anonymous as the value (Z/A^e) would fully identify the host.

[4] Note that $\zeta^{\tilde{r}_f} \equiv \zeta^{f_0 + f_1 2^{l_f}} \pmod{\Gamma}$ and the order of $\rho < \Gamma$ the calculation $\zeta^{\tilde{r}_f} \pmod{\Gamma}$ will involve a smaller exponential thus providing a performance gain.

B.6 Verification protocol

The verification protocol defines the method by which a verifier checks a signature σ on message m with respect to the issuer's public key $(n, g', g, h, S, Z, R_0, R_1, \gamma, \Gamma, \rho)$. The protocol is described below (adapted from [6, 16]).

1. The verifier computes:

$$\hat{T} := Z^{-c} T^{s_e + c2^{l_e - 1}} R_0^{s_{f_0}} R_1^{s_{f_1}} S^{s_{\bar{v}}} \pmod{n} \quad \hat{N}_V := N_V^{-c} \zeta^{s_{f_0} + s_{f_1}} \pmod{\Gamma}$$

2. The verifier checks:

$$c \stackrel{?}{=} H\Big(H\big(H(n\|g\|g'\|h\|R_0\|R_1\|S\|Z\|\gamma\|\Gamma\|\rho\|\zeta\|T\|N_V\|\hat{T}\|\hat{N}_V\|n_v)\|n_t\big)\|b\|m\Big),$$

$$N_V, \zeta \stackrel{?}{\in} \langle\gamma\rangle, \qquad s_{f_0}, s_{f_1} \stackrel{?}{\in} \{0,1\}^{l_f + l_\phi + l_H + 1} \quad \text{and} \quad s_e \stackrel{?}{\in} \{0,1\}^{l'_e + l_\phi + l_H + 1}$$

   Note that the check $N_V, \zeta \stackrel{?}{\in} \langle\gamma\rangle$ can be done by raising N_V and ζ to the order of γ (which is ρ) and checking whether the result is one i.e. check that $N_V^\rho \stackrel{?}{\equiv} 1 \pmod{\Gamma}$ and $\zeta^\rho \stackrel{?}{\equiv} 1 \pmod{\Gamma}$.
3. If ζ was derived from the verifier's basename, check that $\zeta \stackrel{?}{\equiv} (H_\Gamma(1\|bsn_I))^{(\Gamma-1)/\rho} \pmod{\Gamma}$.
4. The verifier checks for all $(\tilde{f}_0, \tilde{f}_1)$ on the rogue list that $N_I \stackrel{?}{\neq} \zeta_I^{\tilde{f}_0 + \tilde{f}_1 2^{l_f}} \pmod{\Gamma}$.

B.7 Rogue tagging

The DAA protocol is designed so that a known rogue TPM can be prevented from obtaining certification or making a successful claim of attestation to a verifier. A rogue TPM is defined as an entity which has obtained an attestation identity credential and the associated f. Once a rogue TPM is discovered, the certificate (A, e, v) and the values f_0, f_1 are distributed to all potential issuers/verifiers who add the value to their rogue list. Note that this does not involve a certificate revocation authority.

References

1. Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, and pseudeonymity a proposal for terminology. In: International workshop on Designing privacy enhancing technologies, New York, USA, Springer-Verlag (2001) 1–9
2. Pfitzmann, A., Köhntopp, M.: Anonymity, unlinkability, unobservability, pseudonymity, and identity management a consolidated proposal for terminology. version 0.26. Technical report, Department of Computer Science, Technische Universität Dresden (2005) Available from http://dud.inf.tu-dresden.de/Anon_Terminology.shtml.
3. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Transactions on Information and System Security (TISSEC) **1**(1) (1998) 66–92

4. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Anonymous connections and onion routing. Selected Areas in Communications **16**(4) (1998) 482–494
5. TCG: Trusted Computing Platform Alliance (TCPA) Main Specification Version 1.1b. Technical report, Trusted Computing Group (2002) Previously published by the Trusted Computing Platform Alliance.
6. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: CCS '04: 11th ACM conference on Computer and communications security, New York, United States of America, ACM Press (October 2004) 132–145 See also [31, 32].
7. TCG: TCG TPM Specification Version 1.2 Revision 85. Technical report, Trusted Computing Group (2005)
8. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: CRYPTO '97: 17th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science (LNCS), London, UK, Springer-Verlag (1997) 410–424
9. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: CRYPTO '86: Advances in cryptology, Lecture Notes in Computer Science (LCNS), London, UK, Springer-Verlag (1987) 186–194
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. 5 edn. CRC Press (2001) Available from http://www.cacr.math.uwaterloo.ca/hac/.
11. Meadows, C.: Formal methods for cryptographic protocol analysis: emerging issues and trends. Selected Areas in Communications **21**(1) (2003) 44–54
12. Koblitz, N., Menezes, A.J.: Another look at "provable security". Cryptology ePrint Archive, Report 2004/152 (2004)
13. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, London, UK, Springer-Verlag (2001) 93–118
14. Camenisch, J., Herreweghen, E.V.: Design and implementation of the idemix anonymous credential system. In: CCS '02: Proceedings of the 9th ACM conference on Computer and communications security, New York, NY, USA, ACM Press (2002) 21–30
15. Camenisch, J., Groth, J.: Group signatures: better efficiency and new theoretical aspects. In: SCN '04: Fourth Conference on Security in Communication Networks, Lecture Notes in Computer Science (LNCS). Volume 3352., Springer-Verlag (2004) 120–133
16. Brickell, E., Camenisch, J., Chen, L.: The DAA Scheme in Context, in Chris Mitchell (eds) Trusted Computing. The Institute of Electrical Engineers (IEE) (2005)
17. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: EUROCRYPT '98: Advances in Cryptology, Lecture Notes in Computer Science (LNCS). Volume 1403., Springer-Verlag (1998) 236–250
18. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. Cryptology ePrint Archive, Report 1998/007 (1998) Full version.
19. Tanaka, T.: Possible economic consequences of digital cash. In: INET '96: Proceedings of the 6th Annual Internet Society Conference, ISOC (1996)
20. Smyth, B.: Direct Anonymous Attestation (DAA): An implementation and security analysis. Master's thesis, School of Computer Science, University of Birmingham, UK (2006) Available from http://www.cs.bham.ac.uk/~bas/.

21. Chaum, D., Evertse, J.H., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: EUROCRYPT '87: Advances in Cryptography, Lecture Notes in Computer Science (LNCS). Volume 304., Springer-Verlag (1988) 127–141
22. Chaum, D., Evertse, J.H., van de Graaf, J., Peralta, R.: Demonstrating possession of a discrete logarithm without revealing it. In: CRYPTO '86: Advances in Cryptology, Lecture Notes in Computer Science (LNCS), London, UK, Springer-Verlag (1987) 200–212
23. Camenisch, J., Michels, M.: A group signature scheme based on an RSA-variant. Technical report, Basic Research in Computer Science (BRICS) (RS-28-27) (November 1998) A preliminary version of this paper appeared in [33]. Available from http://www.brics.dk/RS/98/27/.
24. Camenisch, J., Michels, M.: Separability and efficiency for generic group signature schemes. In: CRYPTO '99: 19th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science (LCNS). Volume 1666., London, UK, Springer-Verlag (1999) 413–430
25. Brickell, E.F., Chaum, D., Damgård, I., van de Graaf, J.: Gradual and verifiable release of a secret. In: CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, Lecture Notes in Computer Science (LNCS), London, UK, Springer-Verlag (1988) 156–166 Available from http://dsns.csie.nctu.edu.tw/research/crypto/HTML/PDF/C87/156.PDF.
26. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: EUROCRYPT 2000: Advances in Cryptology, Lecture Notes in Computer Science (LCNS). Volume 1807., Springer-Verlag (2000) 431–444
27. Cramer, R.J.F., Pedersen, T.P.: Improved privacy in wallets with observers. In: EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology, Lecture Notes in Computer Science (LNCS), Secaucus, New Jersey, United States of America, Springer-Verlag (1994) 329–343
28. Chaum, D.: Zero-knowledge undeniable signatures (extended abstract). In: EUROCRYPT '90: Advances in Cryptography, Lecture Notes in Computer Science (LNCS). Volume 473., Springer-Verlag (1991) 458–464
29. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: SCN'02: Third Conference on Security in Communication Networks, Lecture Notes in Computer Science (LNCS). Volume 2576., Springer-Verlag (2002) 268–289
30. Lysyanskaya, A.: Signature schemes and applications to cryptographic protocol design. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, United States of America (2002)
31. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. Cryptology ePrint Archive, Report 2004/205 (February 2004) Full version of ACM CCS '04 paper.
32. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. Technical report, HP Labs (HPL-2004-93) (June 2004) Available from http://www.hpl.hp.com/techreports/2004/HPL-2004-93.html.
33. Camenisch, J., Michels, M.: A group signature scheme with improved efficiency. In: ASIACRYPT'98: Advances in Cryptology, Lecture Notes in Computer Science (LCNS). Volume 1998., Springer-Verlag (1998) 160–174