Formal analysis of privacy in Direct Anonymous Attestation schemes

Ben Smyth[1], Mark D. Ryan[2], and Liqun Chen[3]

[1]Mathematical and Algorithmic Sciences Lab, France Research Center, Huawei Technologies Co. Ltd., France
[2]School of Computer Science, University of Birmingham, UK
[3]HP Laboratories, Bristol, UK

March 4, 2015

Abstract

This article introduces a definition of privacy for Direct Anonymous Attestation schemes. The definition is expressed as an equivalence property which is suited to automated reasoning using Blanchet's ProVerif. The practicality of the definition is demonstrated by analysing the RSA-based Direct Anonymous Attestation protocol by Brickell, Camenisch & Chen. The analysis discovers a vulnerability in the RSA-based scheme which can be exploited by a passive adversary and, under weaker assumptions, corrupt issuers and verifiers. A security fix is identified and the revised protocol is shown to satisfy our definition of privacy.

1 Introduction

Trusted computing allows commodity computers to provide cryptographic assurances about their behaviour. At the core of the architecture is a hardware device called the Trusted Platform Module (TPM). The TPM uses shielded memory to store cryptographic keys, and other sensitive data, which can be used to achieve security objectives, in particular, the chip can measure and report its state, and authenticate. These security objectives assume that a TPM's shielded memory protects keys and TPMs are said to be compromised if this assumption does not hold (see Tarnovsky [Tar10] for a hardware attack that successful extracts keys from shielded memory).

[†] This article is based upon [SRC11, SRC07] and an earlier version appeared in [Smy11, Chapter 4]. The ProVerif scripts which support this article are available at the following URL: `http://www.bensmyth.com/publications/2012-Direct-Anonymous-Attestation-anonymity-definition/`.

Cryptographic operations, by their nature, may reveal a platform's identity. As a consequence, the TPM has been perceived as a threat to privacy by some users, for example, see Stallman [Sta02, Sta10] and Anderson [And03, And04]. In an attempt to overcome these privacy concerns, Brickell, Camenisch & Chen [BCC04] have introduced Direct Anonymous Attestation (DAA), a historical account of DAA's development is presented by Brickell, Camenisch & Chen [BCC05].

Direct Anonymous Attestation enables a platform to authenticate in a manner that provides privacy and accountability. The concept is based upon group signatures with stronger anonymity guarantees, in particular, the identity of a signer can never be revealed, but certain signatures can be linked (as discussed below) and signatures produced by compromised platforms can be identified. A DAA scheme considers a set of *hosts*, *issuers*, *TPMs*, and *verifiers*; the host and TPM together form a *trusted platform* or *signer*. DAA protocols proceed as follows. A host requests membership to a *group of signers* managed by an issuer. The issuer authenticates the host as a trusted platform and grants an *attestation identity credential* (occasionally abbreviated *credential*). A verifier can authenticate trusted platforms using signatures produced from such credentials.

Brickell, Chen & Li [BCL08b, BCL09] and Chen [Che10a, Che11] characterise the following security properties[1] for Direct Anonymous Attestation schemes:

- *Anonymity.* The identity of a signer cannot be revealed from a signature.
- *Non-frameability.* An adversary cannot produce a signature associated with an honest TPM.
- *Unforgeability.* Signatures cannot be produced without a TPM.
- *User-controlled linkability.* A signer can control whether her signatures can be detected as being from the same signer.

A signer defines whether her signatures are *linkable* (that is, can be detected as being from the same signer) or *unlinkable* (that is, cannot be detected as being from the same signer) at the time of construction.

Our security properties aim to balance the privacy (anonymity and unlinkability properties) demands of users with the accountability (linkability, non-frameability and unforgeability properties) needs of issuers and verifiers. The distinction between privacy and accountability properties is reflected in our trust model: anonymity and unlinkability assume that two signers are honest, whereas, linkability, non-frameability and unforgeability assume that an issuer is honest. (The issuer must be honest for linkability, since a dishonest issuer can provide an adversary with a new credential for every signature, thereby ensuring that two signatures are never linked.) In addition, DAA schemes must be *correct*: valid signatures can be verified and, where applicable, linked.

[1] The necessity for non-frameability was highlighted by Backes, Maffei & Unruh [BMU08] and formalised by Chen [Che10a, Che11], the remaining properties were formalised by Brickell, Chen & Li [BCL08b, BCL09].

Brickell, Camenisch & Chen [BCC04] propose the first concrete instance of a Direct Anonymous Attestation scheme. Their scheme is based upon RSA and support for this scheme is mandated by the TPM specification version 1.2 [TCG07], which has been defined as an ISO/IEC international standard [Int09]. Moreover, TPM version 1.2 is estimated to have been embedded in over 500 million computers [Tru11] (however, the Trusted Computer Group acknowledges that the opt-in policy – whereby, users must choose to enable the TPM – has hindered development [Tru12], moreover, Martin claims that only 5% of these TPMs have been turned on [Mar08, §6] and we suspect significantly fewer are in active use). Furthermore, the RSA-based DAA scheme has also been included in the ISO/IEC anonymous digital signature standard [Int11]. A brief review of other DAA schemes appears in A.

1.1 Contribution

We formalise Direct Anonymous Attestation protocols in the applied pi calculus and present a definition of privacy as an equivalence property which is suited to automated reasoning using ProVerif (Section 4). Informally, the security definition asserts that an adversary cannot distinguish between signatures produced by two distinct signers, even when the adversary controls the issuer and has observed signatures produced by each signer. The application of the definition is demonstrated by analysing privacy in the RSA-based DAA protocol (Section 5). The analysis discovers a vulnerability in the protocol which allows privacy to be violated by a passive adversary and, under weaker assumptions, corrupt issuers and verifiers. A fix is identified, and the revised RSA-based DAA protocol is shown to be secure in the symbolic model. We examine the balance between privacy and accountability offered by DAA and propose extensions to DAA (Section 6): we propose a stronger notion of privacy which is intuitively satisfied by the fixed RSA-based scheme, address an issue which can prevent linkability, and provide some practical guidelines for basenames to help resolve a flaw in unlinkability.

1.2 Related work

In the computational model, Brickell, Camenisch & Chen [BCC04] introduce simulation-based models of security and Brickell, Chen & Li [BCL08b, BCL09] propose a game-based security definition; the relationship between simulation-based models and game-based definition is unknown [CMS08a, pp158]. Bernhard *et al.* [BFG^{+}11] argue that the simulation-based definitions and the game-based definition are insufficient for accountability due to informal handling of identities and propose an alternative game-based security definition, moreover, Bernhard *et al.* show that the simulation-based model by Chen, Morrissey & Smart [CMS09] is unsatisfiable (for all protocols there trivially exists a distinction between the ideal- and real-world). We consider a symbolic definition for privacy, based upon the game-based definition by Brickell, Chen & Li (we stress that the criticisms from Bernhard *et al.* relate to the accountability game and not the privacy game, hence, their concerns are not relevant to our work). Backes, Maffei & Unruh [BMU08] formalise an earlier notion of privacy (informally described in [BCC04]) for the RSA-based DAA protocol. This formalisation is tightly coupled with their model of the RSA-based protocol and it is unclear whether other DAA schemes can be analysed or, indeed, how to analyse alternative models of the RSA-based protocol. In addition, their formalisation pre-dates the privacy definitions by Brickell, Chen & Li and considers a conceptually weaker adversary, for example, signers are only permitted one credential and signatures can only be produced after both signers have credentials. Finally, our definition is intuitively simpler, which should aid analysis and, in particular, be better suited to automated reasoning.

Our earlier work also merits comparison. The attack and fix presented in this article originally appeared in Smyth, Ryan & Chen [SRC07]. Delaune, Ryan & Smyth [DRS08] demonstrate a technique to extend the class of equivalences that can be checked using diff-equivalence and present a high-level description of how their technique can be used to analyse the fixed DAA scheme. In addition, Delaune, Ryan & Smyth formalise a notion of privacy for the RSA-based DAA protocol in an unpublished technical report [DRS07]. This formalisation is subject to the issues highlighted with respect to Backes, Maffei & Unruh, moreover, signers may only output one signature, the issuer's key is assumed to be honestly generated, and subtle properties of the underlying CL signature scheme are omitted; we overcome these limitations in this article. Furthermore, as witnessed by Backes, Maffei & Unruh, privacy can be checked using the standard class of equivalences that can be checked using diff-equivalence, rather than the extended class proposed by Delaune, Ryan & Smyth. Smyth presented a version of this article in his thesis [Smy11, Chapter 4], in particular, Smyth's thesis contains a formalisation of Direct Anonymous Attestation schemes as processes in the applied pi calculus, a privacy definition (based upon the cryptographic game proposed by Brickell, Chen & Li), an analysis of the fixed RSA-based DAA protocol, and a special case of the Smyth, Ryan & Chen attack which can be exploited by passive adversaries. The formalisation of DAA schemes as processes and the privacy definition have been developed by Smyth, Ryan & Chen [SRC11], in addition, Smyth, Ryan & Chen analyse the ECC-based DAA protocol by Brickell, Chen & Li [BCL08a, BCL09]. This article collates Smyth, Ryan & Chen [SRC07], Smyth [Smy11, Chapter 4] and, Smyth, Ryan & Chen [SRC11]. Smyth's thesis has not been previously published, hence the analysis of the fixed RSA-based DAA protocol and a special case of the attack which allows a passive adversary to violate privacy are new. In addition, this article provides a more detailed discussion of our results, highlights the limitations of our model, notes some ambiguities in the cryptographic game by Brickell, Chen & Li, and proposes a refinement to the RSA-based DAA protocol to help balance privacy and accountability.

2 Calculus of ProVerif

We adopt a dialect [Bla04, BAF08] of the applied pi calculus [AF01, RS11] which is suited to automated reasoning using Blanchet's ProVerif [BS11]. In this setting, we acknowledge that security claims do not imply the absence of real-world attacks (see [RS98, AR00, AR02, War03, War05], for example), due to the abstract representation of cryptographic primitives. Nonetheless, we believe symbolic analysis is useful because computational security methodologies have provided limited success in the analysis of complex cryptographic protocols [Moo88, Mea03, KM06, KM07]. Indeed, Chen, Morrissey & Smart [CMS08a, pp157] attribute flaws in the security proof of RSA-based DAA [BCC04] to the highly complex nature of DAA schemes and the inherent difficulties of composing correct provable security proofs.

2.1 Syntax and semantics

The calculus assumes an infinite set of names, an infinite set of variables, and a signature Σ consisting of a finite set of function symbols (constructors and destructors), each with an associated arity. A function symbol with arity 0 is a constant. We write f for a constructor, g for a destructor, and h for either a constructor or destructor. Terms are defined over names, variables, and constructors applied to other terms (Figure 1). A substitution, denoted $\{^M/_x\}$, replaces the variable x with the term M and we let the letters σ and τ range over substitutions. We write $N\sigma$ for the result of applying σ to the free variables of N.

The signature Σ is equipped with a finite set of equations of the form $M = N$ and we derive an equational theory from this set by reflexive, symmetric and transitive closure, closure under the application of constructors, closure under substitution of terms for variables, and closure under bijective renaming of names. We write $\Sigma \vdash M = N$ for an equality modulo the equational theory and $\Sigma \vdash M \neq N$ for an inequality modulo the equational theory. (We write $M = N$ and $M \neq N$ for syntactic equality and inequality, respectively.)

The semantics of a destructor g of arity l is given by a finite set $\mathrm{def}_\Sigma(g)$ of rewrite rules $g(M'_1, \ldots, M'_l) \to M'$, where $M'_1, \ldots, M'_l, M'$ are terms containing only constructors and variables, the variables of M' are bound in $M'_1, \ldots, M'_l$, and variables are subject to renaming. The term evaluation $g(M_1, \ldots, M_l)$ is defined if and only if there exists a substitution σ and a rewrite rule $g(M'_1, \ldots, M'_l) \to M'$ in $\mathrm{def}_\Sigma(g)$ such that $M_i = M'_i\sigma$ for all $i \in \{1, \ldots, l\}$, and in this case $g(M_1, \ldots, M_l)$ is $M'\sigma$. In order to avoid distinguishing constructors and destructors in the definition of term evaluation, we let $\mathrm{def}_\Sigma(f)$ be $\{f(x_1, \ldots, x_l) \to f(x_1, \ldots, x_l)\}$ when f is a constructor of arity l.

The grammar for processes appears in Figure 1. The process let $x = D$ in P else Q tries to evaluate D; if this succeeds, then x is bound to the result and P is executed, otherwise, Q is executed. For convenience, the statement let $x = D$ in P else Q may be abbreviated as let $x = D$ in P when Q is the null process. The syntax does not include the conditional if $M = N$ then P else Q, but this

Figure 1 Syntax for terms and processes

$M, N ::=$	terms
$a, b, c, \ldots, k, \ldots, m, n, \ldots, s$	name
x, y, z	variable
$f(M_1, \ldots, M_l)$	constructor application
$D ::=$	term evaluations
M	term
eval $h(D_1, \ldots, D_l)$	function evaluation
$P, Q, R ::=$	processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu\, a.P$	name restriction
$M(x).P$	message input
$\overline{M}\langle N \rangle.P$	message output
let $x = D$ in P else Q	term evaluation

can be defined as let $x = \mathsf{eq}(M, N)$ in P else Q, where x is a fresh variable and eq is a binary destructor with the rewrite rule $\mathsf{eq}(x, x) \rightarrow x$. We always include this destructor in Σ. The rest of the syntax is standard (see Blanchet [Bla04, BAF08] for details).

The sets of free and bound names, respectively variables, in process P are denoted by $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$, respectively $\mathrm{fv}(P)$ and $\mathrm{bv}(P)$. We also write $\mathrm{fn}(M)$ and $\mathrm{fv}(M)$ for the sets of names and variables in term M. A process P is closed if it has no free variables. A context C is a process with a hole and we obtain $C[P]$ as the result of filling C's hole with P. An evaluation context is a context whose hole is not in the scope of a replication, an input, an output, or a term evaluation.

The operational semantics are defined by reduction ($\rightarrow_\Sigma$) in association with the auxiliary rules for term evaluation ($\Downarrow_\Sigma$) and structural equivalence ($\equiv$). Both $\equiv$ and $\rightarrow_\Sigma$ are defined only on closed processes. We write $\rightarrow^*_\Sigma$ for the reflexive and transitive closure of $\rightarrow_\Sigma$, and we write $\rightarrow^*_\Sigma\equiv$ for the union of $\rightarrow^*_\Sigma$ with $\equiv$. We occasionally abbreviate $\rightarrow_\Sigma$ as $\rightarrow$ and $\Downarrow_\Sigma$ as $\Downarrow$.

2.2 Biprocesses

The calculus provides a notation for modelling pairs of processes that have the same structure and differ only by the terms and term evaluations that they contain. We call such a pair of processes a *biprocess*. The grammar for the calculus with biprocesses is a simple extension of Figure 1, with additional cases so that $\mathrm{diff}[M, M']$ is a term and $\mathrm{diff}[D, D']$ is a term evaluation. The semantics for biprocesses include the rules in Figure 2, except for (Red I/O), (Red Fun 1), and (Red Fun 2), which are revised in Figure 3. We also extend

Figure 2 Semantics for terms and processes

$$
\begin{array}{l}
M \Downarrow M \\
\text{eval } h(D_1, \ldots, D_n) \Downarrow N\sigma \\
\quad \text{if } h(N_1, \ldots, N_n) \rightarrow N \in \text{def}_{\Sigma}(h) \text{ and } \sigma \text{ is such that} \\
\quad \text{for all } i,\ D_i \Downarrow M_i \text{ and } \Sigma \vdash M_i = N_i\sigma
\end{array}
$$

$$
\begin{array}{ll}
P \mid 0 \equiv P & P \equiv P \\
P \mid Q \equiv Q \mid P & Q \equiv P \ \Rightarrow\ P \equiv Q \\
(P \mid Q) \mid R \equiv P \mid (Q \mid R) & P \equiv Q,\, Q \equiv R \ \Rightarrow\ P \equiv R \\
\nu\, a.\nu\, b.P \equiv \nu\, b.\nu\, a.P & P \equiv Q \ \Rightarrow\ P \mid R \equiv Q \mid R \\
\nu\, a.(P \mid Q) \equiv P \mid \nu\, a.Q & P \equiv Q \ \Rightarrow\ \nu\, a.P \equiv \nu\, a.Q \\
\quad \text{if } a \notin \text{fn}(P) &
\end{array}
$$

$$
\begin{array}{lr}
\overline{N}\langle M\rangle.Q \mid N'(x).P \ \rightarrow\ Q \mid P\{^{M}/_{x}\} & (\textsc{Red I/O}) \\
\quad \text{if } \Sigma \vdash N = N' & \\
\text{let } x = D \text{ in } P \text{ else } Q \rightarrow P\{^{M}/_{x}\} & (\textsc{Red Fun 1}) \\
\quad \text{if } D \Downarrow M & \\
\text{let } x = D \text{ in } P \text{ else } Q \rightarrow Q & (\textsc{Red Fun 2}) \\
\quad \text{if there is no } M \text{ such that } D \Downarrow M & \\
!P \ \rightarrow\ P \mid !P & (\textsc{Red Repl}) \\
P \ \rightarrow\ Q \ \Rightarrow\ P \mid R \ \rightarrow\ Q \mid R & (\textsc{Red Par}) \\
P \ \rightarrow\ Q \ \Rightarrow\ \nu\, a.P \ \rightarrow\ \nu\, a.Q & (\textsc{Red Res}) \\
P' \equiv P,\, P \ \rightarrow\ Q,\, Q \equiv Q' \ \Rightarrow\ P' \ \rightarrow\ Q' & (\textsc{Red} \equiv)
\end{array}
$$

Figure 3 Generalised semantics for biprocesses

$$
\begin{array}{lr}
\overline{N}\langle M\rangle.Q \mid N'(x).P \ \rightarrow\ Q \mid P\{^{M}/_{x}\} & (\textsc{Red I/O}) \\
\quad \text{if } \Sigma \vdash \text{fst}(N) = \text{fst}(N') \text{ and } \Sigma \vdash \text{snd}(N) = \text{snd}(N') & \\
\text{let } x = D \text{ in } P \text{ else } Q \rightarrow P\{^{\text{diff}[M_1, M_2]}/_{x}\} & (\textsc{Red Fun 1}) \\
\quad \text{if fst}(D) \Downarrow M_1 \text{ and snd}(D) \Downarrow M_2 & \\
\text{let } x = D \text{ in } P \text{ else } Q \rightarrow Q & (\textsc{Red Fun 2}) \\
\quad \text{if there is no } M_1 \text{ such that fst}(D) \Downarrow M_1 \text{ and} & \\
\quad \text{there is no } M_2 \text{ such that snd}(D) \Downarrow M_2 &
\end{array}
$$

the definition of contexts to permit the use of diff.

Given a biprocess P, we define processes fst(P) and snd(P), as follows: fst(P) is obtained by replacing all occurrences of diff$[M, M']$ with M and diff$[D, D']$ with D in P; and, similarly, snd(P) is obtained by replacing diff$[M, M']$ with M' and diff$[D, D']$ with D' in P. We define fst(D), fst(M), snd(D), and snd(M) similarly.

2.3 Observational equivalence

Intuitively, processes P and Q are said to be observationally equivalent if they can output on the same channels, no matter what context they are placed inside. Formally, we write $P \downarrow_M$ when P can send a message on M, that is, when $P \equiv C[\overline{M'}\langle N\rangle.R]$ for some evaluation context $C[\_]$ such that $\mathrm{bn}(C) \cap \mathrm{fn}(M) = \emptyset$ and $\Sigma \vdash M = M'$. The definition of observational equivalence [Bla04, BAF08] follows.

Definition 1 (Observational equivalence)**.** Observational equivalence $\sim$ *is the largest symmetric relation* $\mathcal{R}$ *between closed processes such that* $P \; \mathcal{R} \; Q$ *implies:*

1. *if* $P \downarrow_M$*, then* $Q \downarrow_M$*;*
2. *if* $P \rightarrow P'$*, then* $Q \rightarrow Q'$ *and* $P' \; \mathcal{R} \; Q'$ *for some* Q'*;*
3. $C[P] \; \mathcal{R} \; C[Q]$ *for all evaluation contexts* $C[\_]$*.*

We additionally define observational equivalence as a property of biprocesses.

Definition 2. *The closed biprocess* P satisfies observational equivalence *if* $\mathrm{fst}(P) \sim \mathrm{snd}(P)$.

Automated reasoning It follows from the semantics of biprocess that if $P \rightarrow Q$ for some biprocesses P and Q, then $\mathrm{fst}(P) \rightarrow \mathrm{fst}(Q)$ and $\mathrm{snd}(P) \rightarrow \mathrm{snd}(Q)$. However, reductions in $\mathrm{fst}(P)$ or $\mathrm{snd}(P)$ do not imply biprocess reductions in P, that is, there exist biprocesses P such that $\mathrm{fst}(P) \rightarrow \mathrm{fst}(Q)$, but there is no such reduction $P \rightarrow Q$, and symmetrically for $\mathrm{snd}(P)$. For example, consider the biprocess $P = \overline{\mathrm{diff}[a, c]}\langle n\rangle \mid a(x)$, we have $\mathrm{fst}(P) \rightarrow 0$, but there is no reduction $P \rightarrow 0$. Blanchet, Abadi & Fournet [Bla04, BAF08] have shown that a biprocess P satisfies observational equivalence when reductions in $\mathrm{fst}(P)$ or $\mathrm{snd}(P)$ imply reductions in P. Moreover, they propose an automated procedure for reasoning with observational equivalence of biprocesses and this procedure has been implemented in ProVerif.

2.4 Assumptions and notation

In this article, all signatures are tacitly assumed to include the constant $\varnothing$, unary destructors fst and snd, and the binary constructor pair. Furthermore, for all variables x and y we assume the rewrite rules

$$\mathsf{fst}(\mathsf{pair}(x, y)) \rightarrow x \qquad \mathsf{snd}(\mathsf{pair}(x, y)) \rightarrow y$$

For convenience, $\mathsf{pair}(M_1, \mathsf{pair}(\ldots, \mathsf{pair}(M_n, \varnothing)))$ is occasionally abbreviated as $(\!|M_1, \ldots, M_n|\!)$ and $\mathsf{fst}(\mathsf{snd}^{i-1}(M))$ is denoted $\pi_i(M)$.

3 DAA schemes

A Direct Anonymous Attestation scheme comprises of five algorithms [BCC04, BCL08b, BCL09], each of which will now be discussed.

Setup The setup algorithm is used by the issuer to construct a DAA key pair sk_I and $\mathsf{pk}(sk_I)$, the public part $\mathsf{pk}(sk_I)$ is published. In addition, the setup algorithm may define implementation-specific parameters.

Join The join algorithm is run between a trusted platform and an issuer for the purpose of obtaining group membership. The algorithm assumes that the trusted platform and issuer have established a one-way authenticated channel, that is, the issuer is assured to be communicating with a host and TPM. The definition of DAA does not mandate a particular authentication mechanism (the Trusted Computing Group recommend encrypting every message sent by the issuer under the TPM's endorsement key [TCG07]). On successful completion of the join algorithm, the issuer grants the trusted platform with an attestation identity credential $\mathtt{cre}$ based upon a secret $\mathtt{tsk}$ known only by the TPM.

Sign The sign algorithm is executed by a trusted platform to produce a signature σ, based upon an attestation identity credential $\mathtt{cre}$ and secret $\mathtt{tsk}$, which asserts group membership and, therefore, trusted platform status. In addition to $\mathtt{cre}$ and $\mathtt{tsk}$, the algorithm takes as input a message m and a basename $\mathtt{bsn}$. The basename is used to control linkability between signatures: if $\mathtt{bsn} = \bot$, then signatures should be unlinkable, otherwise, signatures produced by the same signer and based upon the same basename can be linked (see Section 6 for further discussion on linkability).

Verify The verification algorithm is used by a verifier to check the validity of a signature. The algorithm takes as input a set of secret keys $\mathtt{ROGUE}_{\mathtt{tsk}}$, which are known to have been successfully extracted from compromised TPMs, allowing the identification of rogue platforms. The methodology used to build $\mathtt{ROGUE}_{\mathtt{tsk}}$ is not defined by DAA, see Chen & Li [CL10b] for further discussion.

Link The link algorithm is used by a verifier to check if two valid signatures are linked, that is, signed using the same basename $\mathtt{bsn}$ and secret $\mathtt{tsk}$.

The inputs and outputs of these algorithms are explicitly summarised in Table 1.

4 Security definition: privacy

Informally, the notion of privacy asserts that given two honest signers $\mathcal{A}$ and $\mathcal{B}$, an adversary cannot distinguish between: a situation in which $\mathcal{A}$ signs a message and a situation in which $\mathcal{B}$ signs a message. Based upon the game-based definition by Brickell, Chen & Li [BCL08b, BCL09], we present the following description of our privacy property.

Initial The adversary constructs the DAA key pair sk_I and $\mathsf{pk}(sk_I)$, and publishes the public part $\mathsf{pk}(sk_I)$ along with any additional parameters. In addition, public keys of honest TPMs are published.

Algorithm	Input	Output
Setup	Security parameters.	A DAA key pair sk_I and $\mathsf{pk}(sk_I)$, and any implementation-specific parameters.
Join	Trusted platform's input: the system parameters (namely, the DAA public key and any implementation-specific parameters defined by the setup algorithm), the TPM's internal secret `DAASeed` (this value is defined during manufacture [TCG07]), a counter value `cnt` selected by the host, and the TPM's endorsement key. Issuer's input: the system parameters and the issuer's private key sk_I.	Trusted platform's output: a pair consisting of the attestation identity credential `cre` and secret `tsk`. Issuer's output: the public part of the TPM's endorsement key.
Sign	The system parameters, a verifier's basename `bsn`, a message m, an attestation identity credential `cre`, and a secret `tsk`.	A signature σ.
Verify	The system parameters, a verifier's basename `bsn`, a message m, a candidate signature σ for m, and a set of secret keys $\texttt{ROGUE}_{\texttt{tsk}}$.	1 (*accept*) or 0 (*reject*).
Link	The system parameters, and two candidate signatures σ and σ' for messages m and m'.	-1 (*invalid signature*) if the verify algorithm outputs 0 for signature σ (or σ') using the system parameters, basename $\perp$, the message m (respectively m'), and the empty set of secret keys. Otherwise, the algorithm returns 1 (*linked*) if the signatures can be linked and 0 (*not linked*) if the signatures cannot be linked.

Table 1: Summary of inputs and outputs for Direct Anonymous Attestation algorithms

Phase 1 The adversary makes the following requests to signers $\mathcal{A}$ and $\mathcal{B}$:

- Join. The signer executes the join algorithm with the adversary to create $\texttt{cre}$ and $\texttt{tsk}$. (The adversary, behaving as the issuer, will typically construct $\texttt{cre}$ but not learn $\texttt{tsk}$.)
- Sign. The adversary submits a basename $\texttt{bsn}$ and a message m. The signer runs the sign algorithm and returns the signature to the adversary.

We insist that sign requests to $\mathcal{A}$ (or $\mathcal{B}$) must be proceeded by at least one join request to $\mathcal{A}$ (respectively $\mathcal{B}$). Moreover, at the end of Phase 1, both signers are required to have run the join algorithm at least once.

Challenge The adversary submits a message m' and a basename $\texttt{bsn}'$ to the signers, with the restriction that the basename has not been previously used if $\texttt{bsn}' \neq \perp$. Each signer produces a signature on the message and returns the signature to the adversary.

Phase 2 The adversary continues to probe the signers with join and sign requests, but is explicitly forbidden to use the basename $\texttt{bsn}'$ used in the Challenge phase if $\texttt{bsn}' \neq \perp$.

Result The protocol satisfies privacy if the adversary cannot distinguish between the two signatures output during the challenge.

Intuitively, our description captures anonymity because the adversary cannot distinguish between the two signatures output during the challenge. This can be observed as follows. Suppose a protocol satisfies the above description of privacy but the identity of a signer can be revealed from a signature. It follows immediately that the adversary can test which challenge signature belongs to $\mathcal{A}$, therefore, allowing the signatures to be distinguished and, hence, deriving a contradiction. Moreover, our description also captures unlinkability. This can be observed as follows. Suppose a protocol satisfies our description of privacy, but a randomly sampled pair of signatures from the same signer can be linked with non-negligible probability, where the signatures were defined as unlinkable by the signer. Further suppose that the signatures are on messages m and m'. Let us consider an adversary that requests a signature σ_A from $\mathcal{A}$ during Phase 1 using the message m and basename $\texttt{bsn} = \perp$ (that is, signatures should be unlinkable). Furthermore, the adversary submits the message m' and basename $\texttt{bsn}' = \perp$ during the challenge, and the signers return signatures σ_1 and σ_2. The adversary tests if σ_A and σ_1 are linked or if σ_A and σ_2 are linked. Exactly one test will succeed with non-negligible probability, thereby allowing the adversary to distinguish between signatures σ_1 and σ_2. We have derived a contradiction and, therefore, a protocol satisfying our description of privacy provides unlinkability. Formally proving that our security definition (Definition 3) captures anonymity and unlinkability is beyond the scope of this article.

Comparison with Brickell, Chen & Li Our description of privacy clarifies some ambiguities in the cryptographic game proposed by Brickell, Chen & Li:

- The side condition that both signers must execute the join algorithm at least once during Phase 1 is only implicitly included in the cryptographic game by Brickell, Chen & Li with the requirement that *"*[the adversary] *chooses two signers' identities* [...]*"* [BCL09, §2.2.2]. We stress that their cryptographic game is unsatisfiable without this condition, in particular, privacy can never be achieved in a setting with one signer. Accordingly, we make the side condition explicit in our description.

- Sign queries with $\mathcal{A}$ or $\mathcal{B}$ are restricted to the basename $\perp$ in Phase 2 of the cryptographic game, more precisely, Brickell, Chen & Li state *"*[the adversary is] *not allowed to make Sign* [queries] *with* $\mathtt{bsn}$ *if* $\mathtt{bsn} \neq \perp$ [...]*"* [BCL09, §2.2.2]. However, Chen has confirmed that this was unintentional and we only forbid sign requests with $\mathcal{A}$ or $\mathcal{B}$ from using the challenge basename.

In addition, there are some high level distinctions between our description of the privacy property and the description proposed by Brickell, Chen & Li:

1. *No key verification.* In the original description, by Brickell, Chen & Li, a key constructed by the adversary in the Initial phase is verified, whereas, no verification of the key is performed in our description.

2. *Static corruption of honest TPMs.* In the original description the adversary can dynamically corrupt honest TPMs, whereas, all TPMs except two are assumed to be corrupt in our description.

3. *Indistinguishability definition.* In the original description either $\mathcal{A}$ or $\mathcal{B}$ signs the message during the Challenge and privacy is satisfied if the adversary has a negligible advantage over guessing the correct signer. By comparison, in our definition, we consider a run in which $\mathcal{A}$ signs during the challenge and a run in which $\mathcal{B}$ signs during the challenge, and privacy is satisfied if these runs are indistinguishable.

The first abstraction is trivially sound, but not complete (nonetheless, the level of abstraction in the symbolic model typically precludes attacks of this type). The second simplifying abstraction appears to be reasonable since TPMs can be simulated by the adversarial context. Indeed, this is a typical simplification in symbolic models, for example, definitions of ballot secrecy for electronic voting [KR05, DKR06, BHM08] and privacy for vehicular ad hoc networks [DDS10] also fix the set of honest participants. However, it is unknown if these simplifications are sound. The third abstraction is intuitively sound, since an adversary strategy that can detect whether a signature belongs to $\mathcal{A}$ or $\mathcal{B}$ can be transformed into a strategy that distinguishes the signatures of $\mathcal{A}$ and $\mathcal{B}$. More precisely, let $\mathcal{M}$ be an adversary that, given a signature σ, returns the identity

$id = \mathcal{M}(\sigma)$ of the signer. A strategy $\mathcal{M}'$ for distinguishing σ_1 and σ_2 simply tests whether $\mathcal{M}(\sigma_1) = \mathcal{M}(\sigma_2)$.

Finally, we will overcome the following shortcoming in the privacy game: the cryptographic game does not permit the adversary to interact with TPMs during the Initial phase. As a consequence, any DAA scheme satisfying the game-based definition may exhibit the following undesirable property, namely, if a malicious issuer interacts with TPMs before constructing a key, then no security assurances are offered. We avoid this limitation by modelling dishonest TPMs as part of the adversarial context and allow the adversary to receive the public keys of honest TPMs during the Initial phase. In future work, the cryptographic game could be revised to allow the adversary to probe the challenger during the Initial phase (the accountability game exhibits the same weakness and could be similarly revised).

4.1 Privacy as an equivalence

Informally, privacy asserts that an adversary cannot distinguish between signatures produced by two distinct signers. We formalise privacy as an observational equivalence property (Definition 3) using the *DAA game biprocess* $\mathsf{DAA\text{-}G}$ (Figure 4) parametrised by a pair of processes $\langle \mathsf{Join}, \mathsf{Sign} \rangle$, which model a DAA scheme's join and sign algorithms.

Definition 3 (Privacy). *Given a pair of processes* $\langle \mathsf{Join}, \mathsf{Sign} \rangle$*,* privacy *is satisfied if the DAA game biprocess* $\mathsf{DAA\text{-}G}$ *satisfies observational equivalence.*

Let us critique the suitability of Definition 3 by relating the operations performed in $\mathsf{DAA\text{-}G}$ to those performed in our description of privacy.

Modelling the Initial phase Biprocess $\mathsf{DAA\text{-}G}$ defines the TPM endorsement keys $\nu\, sk_A$ and $\nu\, sk_B$ (these keys are defined during manufacture [TCG07]), outputs the public part of these keys, and inputs w_{params}, where w_{params} captures inputting a public key $\mathsf{pk}(sk_I)$ and any additional parameters from the adversary, therefore, immediately corresponding to the *Initial* step of our description. (As observed by Rudolph [Rud07], and specified in our description and enforced by our biprocess $\mathsf{DAA\text{-}G}$, privacy can only be expected if both signers use the same system parameters, that is, the signers do not accept distinct system parameters from the issuer.)

Modelling Phases 1 & 2 The processes $\mathsf{Signer}\{^{b_A}/_{w_b}, ^{sk_A}/_{w_{\mathsf{ek}}}\}$ and $\mathsf{Signer}\{^{b_B}/_{w_b}, ^{sk_B}/_{w_{\mathsf{ek}}}\}$, which form part of biprocess $\mathsf{DAA\text{-}G}$, allow the adversary to initiate two signers and perform arbitrarily many join and sign requests, capturing *Phases 1 & 2* of our description (we over-approximate the capabilities of the signers and do not formally distinguish between Phases 1 & 2). Formally, we model a signer's ability to perform join and sign requests by permitting the Signer process to instantiate the Join and Sign processes, which are expected to behave like services, that is, they can be called by, and return results to,

Figure 4 Biprocess modelling privacy in DAA

Given a pair of processes $\langle \mathsf{Join}, \mathsf{Sign}\rangle$, the *DAA game biprocess* $\mathsf{DAA\text{-}G}$ is defined as

$$\begin{aligned}&\nu\, sk_A \,.\, \nu\, sk_B \,.\, \big(\overline{c}\langle \mathsf{pk}(sk_A)\rangle \mid \overline{c}\langle \mathsf{pk}(sk_B)\rangle \mid c(w_{\mathsf{params}}) \,.\\ &\qquad\qquad \nu\, b_A \,.\, \nu\, b_B \,.\, (\mathsf{Challenge} \mid \mathsf{Signer}\{^{b_A}/_{w_b}, {}^{sk_A}/_{w_{\mathsf{ek}}}\} \mid\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{Signer}\{^{b_B}/_{w_b}, {}^{sk_B}/_{w_{\mathsf{ek}}}\})\big)\end{aligned}$$

such that $b_A, b_B, sk_A, sk_B, w_b, w_{\mathsf{ek}} \notin (\mathrm{fn}(\mathsf{Sign}) \cup \mathrm{fv}(\mathsf{Sign}) \cup \mathrm{fn}(\mathsf{Join}) \cup \mathrm{fv}(\mathsf{Join}))$ and where

```
$\mathsf{Signer} = \nu\, \mathsf{a_j} \,.\, \nu\, \mathsf{a_j'} \,.\, \nu\, \mathsf{a_s} \,.\, \nu\, \mathsf{a_s'} \,.\, (\,(!\mathsf{Join}) \mid (!\mathsf{Sign}) \mid ($
  $\nu\, \mathtt{cnt} \,.\, \nu\, \mathtt{DAASeed} \,.$
  $!\overline{a_j}\langle \llparenthesis w_{\mathsf{params}}, \mathtt{DAASeed}, \mathtt{cnt}, w_{\mathsf{ek}} \rrparenthesis\rangle \,.\, a_j'(x) \,.$
  $\text{let } x_{\mathsf{cre}} = \pi_1(x) \text{ in let } x_{\mathsf{tsk}} = \pi_2(x) \text{ in } ($
    $!c(y) \,.\, \text{let } y_{\mathsf{bsn}} = \pi_1(y) \text{ in let } y_{\mathsf{msg}} = \pi_2(y) \text{ in}$
    $\text{if } y_{\mathsf{bsn}} = \bot \text{ then}$
      $\overline{a_s}\langle \llparenthesis w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathsf{tsk}} \rrparenthesis\rangle \,.$
      $a_s'(z) \,.\, \overline{c}\langle z\rangle$
     $\text{else}$
      $\overline{a_s}\langle \llparenthesis w_{\mathsf{params}}, \llparenthesis chl^+, y_{\mathsf{bsn}} \rrparenthesis, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathsf{tsk}} \rrparenthesis\rangle \,.$
      $a_s'(z) \,.\, \overline{c}\langle z\rangle$
  $) \mid ($
    $\overline{w_b}\langle \llparenthesis x_{\mathsf{cre}}, x_{\mathsf{tsk}} \rrparenthesis\rangle$
  $)$
$))$

$\mathsf{Challenge} = \nu\, \mathsf{a_s} \,.\, \nu\, \mathsf{a_s'} \,.\, (\,(\mathsf{Sign}) \mid ($
  $b_A(x) \,.\, \text{let } x_{\mathsf{cre}} = \pi_1(x) \text{ in let } x_{\mathsf{tsk}} = \pi_2(x) \text{ in}$
  $b_B(y) \,.\, \text{let } y_{\mathsf{cre}} = \pi_1(y) \text{ in let } y_{\mathsf{tsk}} = \pi_2(y) \text{ in}$
  $c(z) \,.\, \text{let } z_{\mathsf{bsn}} = \pi_1(z) \text{ in let } z_{\mathsf{msg}} = \pi_2(z) \text{ in}$
  $\text{if } z_{\mathsf{bsn}} = \bot \text{ then}$
    $\overline{a_s}\langle \llparenthesis w_{\mathsf{params}}, z_{\mathsf{bsn}}, z_{\mathsf{msg}}, \mathrm{diff}[x_{\mathsf{cre}}, y_{\mathsf{cre}}], \mathrm{diff}[x_{\mathsf{tsk}}, y_{\mathsf{tsk}}] \rrparenthesis\rangle \,.$
    $a_s'(z) \,.\, \overline{c}\langle z\rangle$
   $\text{else}$
    $\overline{a_s}\langle \llparenthesis w_{\mathsf{params}}, \llparenthesis chl^-, z_{\mathsf{bsn}} \rrparenthesis, z_{\mathsf{msg}},$
                          $\mathrm{diff}[x_{\mathsf{cre}}, y_{\mathsf{cre}}], \mathrm{diff}[x_{\mathsf{tsk}}, y_{\mathsf{tsk}}] \rrparenthesis\rangle \,.$
    $a_s'(z) \,.\, \overline{c}\langle z\rangle$
$))$
```

for some constants chl^+, chl^-.

the Signer process. The restricted channel names a_j and a'_j are introduced to ensure private communication between the Signer and Join processes; similarly, names a_s and a'_s ensure private communication between the Signer and Sign processes. The bound name $\mathtt{cnt}$ is a counter value selected by the host (in this article we consider a static counter value for simplicity, in particular, we do not model incrementing a counter) and the bound name $\mathtt{DAASeed}$ represents the TPM's internal secret (this value is defined during manufacture [TCG07]). The remainder of the Signer process models a signer's ability to execute arbitrarily many instances of the join and sign algorithms (as discussed below), and captures an aspect required by the Challenge phase.

- *Join process.* The Join process is assumed to act like a service and listens for input on channel a_j. It follows that the Signer process can invoke the service by message output $\overline{a_j}\langle(\!|w_{\mathsf{params}}, \mathtt{DAASeed}, \mathtt{cnt}, sk_M|\!)\rangle$, where $(\!|w_{\mathsf{params}}, \mathtt{DAASeed}, \mathtt{cnt}, sk_M|\!)$ models the join algorithm's parameters. The Join process is assumed to output results on channel a'_j, and this response can be received by the Signer process using message input $a'_j(x)$; the result is bound to the variable x, and is expected to consist of a pair $(\!|x_{\mathsf{cre}}, x_{\mathsf{tsk}}|\!)$ representing the attestation identity credential and TPM's secret.

Interaction between Sign and Signer is similar.

- *Sign process.* The Signer process first inputs a variable y which is expected to be a pair representing the verifier's basename y_{bsn} and a message y_{msg}. The invocation of the sign algorithm by the signer is modelled by the message output $\overline{a_s}\langle(\!|w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathsf{tsk}}|\!)\rangle$, where $(\!|w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathsf{tsk}}|\!)$ represents the algorithm's parameters. (We shall discuss the distinction between parameters $(\!|w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathsf{tsk}}|\!)$ and $(\!|w_{\mathsf{params}}, (\!|chl^+, y_{\mathsf{bsn}}|\!), y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathsf{tsk}}|\!)$ in Section 4.1.1.) The sign algorithm is expected to output a signature which can be sent to a verifier, in the Signer process this signature is received from the Sign process by message input $a'_s(z)$ and the variable z, representing the signature, is immediately output. The side condition that a sign request must be preceded by at least one join request is enforced by the sequential description of the Signer process.

In addition, processes $\mathsf{Signer}\{^{b_A}/_{w_b}, {}^{sk_A}/_{w_{\mathsf{ek}}}\}$ and $\mathsf{Signer}\{^{b_B}/_{w_b}, {}^{sk_B}/_{w_{\mathsf{ek}}}\}$ communicate their attestation identity credentials (denoted by x_{cre}) and TPM secrets (denoted by x_{tsk}) to biprocess $\mathsf{Challenge}$ using private channels b_A and b_B. Moreover, the side condition that signers $\mathcal{A}$ and $\mathcal{B}$ are both required to have run the join algorithm at least once during Phase 1 is enforced by the sequential inputs on channels b_A and b_B in biprocess $\mathsf{Challenge}$.

Modelling the Challenge phase Biprocess $\mathsf{Challenge}$ forms part of biprocess $\mathsf{DAA\text{-}G}$ and is designed to capture the behaviour of the signers in the Challenge phase of our description. Biprocess $\mathsf{Challenge}$ inputs the attestation identity credentials and TPM secrets produced by the signers. Biprocess $\mathsf{Challenge}$ also inputs a basename and a message from the environment. The inputs (namely,

x, y, and z) to biprocess $\mathsf{Challenge}$ are used to construct a signature and the biprocess uses $\mathrm{diff}[x_{\mathsf{cre}}, y_{\mathsf{cre}}]$ and $\mathrm{diff}[x_{\mathsf{tsk}}, y_{\mathsf{tsk}}]$ to ensure that the signature is produced by $\mathcal{A}$ in $\mathrm{fst}(\mathsf{DAA\text{-}G})$ and $\mathcal{B}$ in $\mathrm{snd}(\mathsf{DAA\text{-}G})$.

Modelling the Result phase The *Result* phase of our description is captured using observational equivalence.

4.1.1 Limitations

We assume that the processes Join and Sign are initiated by input on channels a_j and a_s and, similarly, output results on channels a'_j and a'_s. Intuitively, it follows that some processes not satisfying these conditions will satisfy our definition of privacy, in fact, the pair of processes $\langle 0, 0\rangle$ will satisfy our definition. We tolerate this limitation here, and future work could consider a complete definition of the DAA properties, including: correctness, linkability, non-frameability, and unforgeability. The correctness property will exclude degenerate pairs of processes such as $\langle 0, 0\rangle$. Similar considerations are made in the literature, for example, degenerate processes can satisfy the definition of ballot secrecy for electronic voting by Delaune, Kremer & Ryan [DKR09, DKR10] and the definition of privacy for vehicular ad-hoc networks by Dahl, Delaune & Steel [DDS10].

The necessity for a distinct basename bsn' in the Challenge phase (when $bsn' \neq \bot$) is enforced by prefixing the basename z_{bsn} used by biprocess $\mathsf{Challenge}$ with chl^- and, similarly, prefixing the basenames y_{bsn} used by the Signer process with chl^+. Capturing distinct basenames in this manner introduces an abstraction.

5 Case study: RSA-based DAA

The first concrete Direct Anonymous Attestation scheme was introduced by Brickell, Camenisch & Chen [BCC04] and is based on RSA. The TPM specification version 1.2 [TCG07], which has been defined as an ISO/IEC international standard [Int09], mandates support for the RSA-based scheme, and the scheme has also been included in the ISO/IEC anonymous digital signature standard [Int11]. In this section, we analyse privacy in the RSA-based protocol using our definition.

5.1 Primitives and building blocks

Camenisch-Lysyanskaya (CL) signatures [CL03, Lys02] form the foundations of RSA-based DAA. Unlike most signature schemes, CL signatures are particularly suited to DAA since the scheme supports signing committed values and proving knowledge of a signature is efficient.

Signature scheme A CL signature is denoted $\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{prime}}, x_{\mathsf{rand}}, x_{\mathsf{msg}})$, where x_{sk} is the secret key, x_{prime} is a random prime, x_{rand} is a nonce, and

x_{msg} is a message. The prime and nonce components can be derived from a signature. Verification is standard given a signature, message and public key, namely, $\mathsf{checkclsign}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{msg}}, \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{prime}}, x_{\mathsf{rand}}, x_{\mathsf{msg}})) = \mathsf{accept}$.

Signature scheme for committed values Given the public part of a signing key $\mathsf{pk}(x_{\mathsf{sk}})$, a message x_{csk}, and commitment factor x_{cf}, the corresponding commitment is $U = \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{cf}}, x_{\mathsf{csk}})$ and the associated signature is $\mathsf{clsign}(x_{\mathsf{sk}}, y_{\mathsf{prime}}, y_{\mathsf{rand}}, U)$, where y_{prime} is a randomly chosen prime and y_{rand} is a nonce. This signature can be opened to recover $\sigma = \mathsf{clopen}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{cf}}, \mathsf{clsign}(x_{\mathsf{sk}}, y_{\mathsf{prime}}, y_{\mathsf{rand}}, U)) = \mathsf{clsign}(x_{\mathsf{sk}}, y_{\mathsf{prime}}, y_{\mathsf{rand}} \circ x_{\mathsf{cf}}, x_{\mathsf{csk}})$ – that is, the signature on x_{csk} – where $\circ$ is commutative and associative. (A proof should also be provided to demonstrate that σ does not contain a covert channel – such details will be omitted from the model presented here – see Brickell, Camenisch & Chen [BCC04] or Smyth [Smy11, pp159] for further details.)

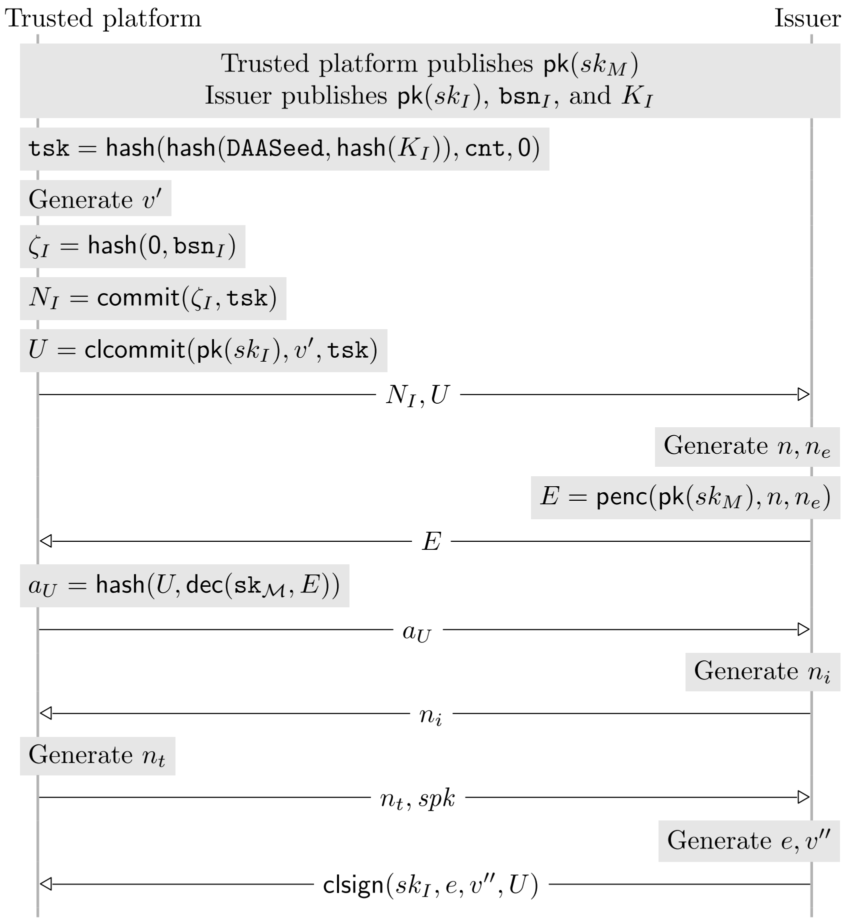
We also consider commitments $\mathsf{commit}(x_{\mathsf{pub}}, x_{\mathsf{csk}})$, where x_{pub} is a public parameter and x_{csk} is a message. Our ternary commitment function differs from our binary commitment function by taking a commitment factor as an additional parameter, this corresponds to the distinction made by the underlying mathematical primitives and permits signatures on committed values to be constructed.

We will adopt the notation introduced by Camenisch & Stadler [CS97] to describe primitives which prove knowledge of, and relations among, discrete logarithms. For instance, given values K, N, T and U, the expression $SPK\{(\alpha, \beta) : N = \mathsf{commit}(T, \alpha) \wedge U = \mathsf{clcommit}(\mathsf{pk}(K), \alpha, \beta)\}(m)$ denotes a signature proof of knowledge of α, β such that $N = \mathsf{commit}(T, \alpha)$ and $U = \mathsf{clcommit}(\mathsf{pk}(K), \alpha, \beta)$, where m is the message being signed. In the example, the Greek letters are used for values about which knowledge is being proved and these values are kept secret by the prover. All other values, that is, those from the Latin alphabet, are known to the verifier. The RSA-based DAA scheme uses signature proofs of knowledge to demonstrate possession of attestation identity credentials.

Proving knowledge of a signature The signature scheme for committed values can be used to build an anonymous credential system. Given a signature $\sigma = \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{prime}}, x_{\mathsf{rand}}, x_{\mathsf{csk}})$ and commitment factor x_{cf}, an anonymous credential $\hat{\sigma} = \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{cf}}, \sigma)$. The signature proof of knowledge $SPK\{(x_{\mathsf{csk}}, x_{\mathsf{cf}}) : \mathsf{checkclsign}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{csk}}, \mathsf{clopen}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{cf}}, \hat{\sigma})) = \mathsf{accept}\}(m)$ can then be used to demonstrate that the anonymous credential $\hat{\sigma}$ is indeed a commitment to a signature on the message x_{csk} using commitment factor x_{cf}.

The application of our primitives to construct the RSA-based DAA protocol will be considered in the next section.

Figure 5 RSA-based DAA join algorithm


Trusted platform
Issuer
Trusted platform publishes $\mathsf{pk}(sk_M)$
Issuer publishes $\mathsf{pk}(sk_I)$, $\mathtt{bsn}_I$, and K_I
$\mathtt{tsk} = \mathsf{hash}(\mathsf{hash}(\mathtt{DAASeed}, \mathsf{hash}(K_I)), \mathtt{cnt}, 0)$
Generate v'
$\zeta_I = \mathsf{hash}(0, \mathtt{bsn}_I)$
$N_I = \mathsf{commit}(\zeta_I, \mathtt{tsk})$
$U = \mathsf{clcommit}(\mathsf{pk}(sk_I), v', \mathtt{tsk})$
N_I, U
Generate n, n_e
$E = \mathsf{penc}(\mathsf{pk}(sk_M), n, n_e)$
E
$a_U = \mathsf{hash}(U, \mathsf{dec}(\mathtt{sk}_{\mathcal{M}}, E))$
a_U
Generate n_i
n_i
Generate n_t
n_t, spk
Generate e, v''
$\mathsf{clsign}(sk_I, e, v'', U)$

where spk is shorthand for $SPK\{(\mathtt{tsk}, v') : N_I = \mathsf{commit}(\zeta_I, \mathtt{tsk}) \wedge U = \mathsf{clcommit}(\mathsf{pk}(sk_I), v', \mathtt{tsk})\}(n_i, n_t)$

5.2 Protocol description

For the purpose of studying privacy, it is sufficient to consider the join and sign algorithms. The join algorithm (Figure 5) is defined below, given the algorithm's input: system parameters $\mathsf{pk}(sk_I)$, $\mathtt{bsn}_I$, and K_I (that is, the DAA public key, basename, and the long-term key); the TPM's secret $\mathtt{DAASeed}$; a counter value $\mathtt{cnt}$; and the TPM's endorsement key $\mathsf{pk}(sk_M)$.

1. The host computes $\zeta_I = \mathsf{hash}(0, \mathtt{bsn}_I)$ and sends ζ_I to the TPM. The TPM computes secret $\mathtt{tsk} = \mathsf{hash}(\mathsf{hash}(\mathtt{DAASeed}, \mathsf{hash}(K_I)), \mathtt{cnt}, 0)$ and derives the commitment $N_I = \mathsf{commit}(\zeta_I, \mathtt{tsk})$. The TPM also gen-

erates a blinding factor v', which is used to compute the commitment $U = \mathsf{clcommit}(\mathsf{pk}(sk_I), v', \mathtt{tsk})$. The trusted platform sends U and N_I to the issuer.

2. The issuer generates a nonce n_e, encrypts the nonce with the TPM's endorsement key $\mathsf{pk}(sk_M)$, and sends the encrypted nonce to the TPM. The TPM decrypts the ciphertext to recover n_e, computes $a_U = \mathsf{hash}(U, n_e)$ and sends a_U to the issuer, therefore authenticating as a trusted platform. (Note that the RSA-based DAA protocol does not rely on the authentication technique recommended by the Trusted Computing Group.)

3. The trusted platform generates a signature proof of knowledge that the messages U and N_I are correctly formed and sends it to the issuer. The proof includes a nonce n_i generated by the issuer and a nonce n_t generated by the TPM, thereby ensuring that both the issuer and the TPM contribute to the proof's randomness.

4. The issuer verifies the proof and evaluates a policy to decide if a new credential should be granted (the policy dictates how many distinct credentials may be issued to a particular trusted platform). To proceed, the issuer generates a signature $\mathsf{clsign}(sk_I, e, v'', U)$ and sends it to the trusted platform.

5. The trusted platform verifies the signature and opens it to reveal the credential $\mathtt{cre} = \mathsf{clsign}(sk_I, e, v' \circ v'', \mathtt{tsk})$, that is, the TPM's secret $\mathtt{tsk}$ signed by the issuer.

The join algorithm outputs $\mathtt{cre}$ and $\mathtt{tsk}$, which can be provided as input to the sign algorithm, along with the system parameters, a basename $\mathtt{bsn}$, and message m. The sign algorithm proceeds as follows.

6. If $\mathtt{bsn} = \bot$, then the host generates a nonce ζ, otherwise, the host computes $\zeta = \mathsf{hash}(0, bsn)$. The host provides the TPM with ζ. The TPM generates a nonce w, and computes the commitment $N_V = \mathsf{commit}(\zeta, \mathtt{tsk})$ and anonymous credential $\widehat{\mathtt{cre}} = \mathsf{clcommit}(\mathsf{pk}(sk_I), w, \mathtt{cre})$. The trusted platform then produces a signature proof of knowledge that $\widehat{\mathtt{cre}}$ is a commitment to a valid credential and that N_V is correctly formed.

The sign algorithm outputs the signature proof of knowledge which is sent to the verifier. Intuitively, if a verifier is presented with such a proof, then the verifier is convinced that it is communicating with a trusted platform.

5.3 Signature and equational theory

Before modelling the RSA-based DAA scheme as a process, we construct a suitable signature Σ (defined below) to capture the cryptographic primitives used and define an equational theory to capture the relationships between these primitives.

$$\begin{aligned}\Sigma = \{&\mathsf{accept}, \bot, \mathsf{0}, \mathsf{1}, \mathsf{F}_{\mathsf{join}}, \mathsf{F}_{\mathsf{sign}}, \mathsf{clgetnonce}, \mathsf{clgetprime}, \mathsf{hash}, \mathsf{pk}, \mathsf{commit},\\ &\circ, \mathsf{dec}, \mathsf{checkclsign}, \mathsf{checkspk}, \mathsf{clcommit}, \mathsf{clopen}, \mathsf{penc}, \mathsf{spk}, \mathsf{clsign}\}\end{aligned}$$

Functions accept, $\bot$, $\mathsf{0}$, $\mathsf{1}$, $\mathsf{F}_{\mathsf{join}}$, $\mathsf{F}_{\mathsf{sign}}$ are constant symbols; $\mathsf{clgetnonce}$, $\mathsf{clgetprime}$, hash, pk are unary functions; commit, $\circ$, dec are binary functions; $\mathsf{checkclsign}$, $\mathsf{checkspk}$, $\mathsf{clcommit}$, clopen, penc, spk are ternary functions; and clsign is a function of arity four. We occasionally write $\mathsf{hash}(x_{\mathsf{plain,1}}, \ldots, x_{\mathsf{plain,n}})$ to denote $\mathsf{hash}((\!|x_{\mathsf{plain,1}}, \ldots, x_{\mathsf{plain,n}}|\!))$. The equations associated with these functions are defined below:

$$\mathsf{dec}(x_{\mathsf{sk}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})) = x_{\mathsf{plain}}$$

$$\mathsf{clgetprime}(\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{prime}}, x_{\mathsf{rand}}, x_{\mathsf{msg}})) = x_{\mathsf{prime}}$$

$$\mathsf{clgetnonce}(\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{prime}}, x_{\mathsf{rand}}, x_{\mathsf{msg}})) = x_{\mathsf{rand}}$$

$$\mathsf{checkclsign}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{msg}}, \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{prime}}, x_{\mathsf{rand}}, x_{\mathsf{msg}})) = \mathsf{accept}$$

$$\mathsf{clopen}(x, x_{\mathsf{rand}}, \mathsf{clcommit}(x, x_{\mathsf{rand}}, x_{\mathsf{plain}})) = x_{\mathsf{plain}}$$

$$\begin{aligned}&\mathsf{clopen}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, \mathsf{clsign}(x_{\mathsf{sk}}, y_{\mathsf{prime}}, y_{\mathsf{rand}}, \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{msg}})))\\ &\qquad = \mathsf{clsign}(x_{\mathsf{sk}}, y_{\mathsf{prime}}, y_{\mathsf{rand}} \circ x_{\mathsf{rand}}, x_{\mathsf{msg}})\end{aligned}$$

A signature proof of knowledge is encoded in the form $\mathsf{spk}(F, U, V)$, where F is a constant declaring the particular proof in use, U denotes the witness (or private component) of a signature proof of knowledge, and V defines the public parameters and message being signed. The function $\mathsf{checkspk}$ is used to verify a signature and we define the following equations.

$$\begin{aligned}&\mathsf{checkspk}(\mathsf{F}_{\mathsf{join}}, V, \mathsf{spk}(\mathsf{F}_{\mathsf{join}}, (x_{\mathsf{tsk}}, x_{\mathsf{cf}}), V)) = \mathsf{accept}\\ &\quad \text{where } V = (x_{\zeta_{\mathsf{I}}}, x_{\mathsf{pk}}, \mathsf{commit}(x_{\zeta_{\mathsf{I}}}, x_{\mathsf{tsk}}), \mathsf{clcommit}(x_{\mathsf{pk}}, x_{\mathsf{cf}}, x_{\mathsf{tsk}}), x_{\mathsf{msg}})\end{aligned}$$

$$\begin{aligned}&\mathsf{checkspk}(\mathsf{F}_{\mathsf{sign}}, V, \mathsf{spk}(\mathsf{F}_{\mathsf{sign}}, (x_{\mathsf{tsk}}, x_{\mathsf{cf}}), V)) = \mathsf{accept}\\ &\quad \text{where } V = (x_{\zeta}, \mathsf{pk}(x_{\mathsf{sk}}), \mathsf{commit}(x_{\zeta}, x_{\mathsf{tsk}}),\\ &\quad \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{cf}}, \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{prime}}, x_{\mathsf{rand}}, x_{\mathsf{tsk}})), x_{\mathsf{msg}})\end{aligned}$$

The first equation is used to verify the signature proof of knowledge produced by the trusted platform during the join algorithm and the second is used by a trusted platform during the sign algorithm to assert group membership.

5.4 Applied pi processes

The RSA-based join and sign algorithms are modelled by the pair of processes $\langle \mathsf{Join}_{\mathsf{RSA}}, \mathsf{Sign}_{\mathsf{RSA}} \rangle$ presented in Figure 6, where $c(x_1, \ldots, x_n).P$ denotes $c(x).\text{let } x_1 = \pi_1(x) \text{ in } \ldots \text{ let } x_n = \pi_n(x) \text{ in } P$. The join process $\mathsf{Join}_{\mathsf{RSA}}$ is instantiated by inputting the join algorithm's parameters: the RSA-based DAA system parameters w_{params}, the TPM's internal secret w_{DAASeed}, the counter

Figure 6 Applied pi processes for RSA-based DAA

```
$\mathsf{Join}_{\mathsf{RSA}} \ \hat{=}$
  $a_j(w_{\mathsf{params}}, w_{\mathsf{DAASeed}}, w_{\mathsf{cnt}}, w_{\mathsf{ek}})\ .\ \nu\, v'\ .$
  let $w_{\mathsf{pk}} = \pi_1(w_{\mathsf{params}})$ in
  let $w_{\mathsf{bsn_I}} = \pi_2(w_{\mathsf{params}})$ in
  let $w_{\mathsf{K}} = \pi_3(w_{\mathsf{params}})$ in
  let $\zeta_I = \mathsf{hash}(0, w_{\mathsf{bsn_I}})$ in
  let $\mathtt{tsk} = \mathsf{hash}(\mathsf{hash}(w_{\mathsf{DAASeed}}, \mathsf{hash}(w_{\mathsf{K}})), w_{\mathsf{cnt}}, 0)$ in
  let $N_I = \mathsf{commit}(\zeta_I, \mathtt{tsk})$ in
  let $U = \mathsf{clcommit}(w_{\mathsf{pk}}, v', \mathtt{tsk})$ in
  $\overline{c}\langle (\!| N_I, U |\!) \rangle\ .\ c(x)\ .\ \overline{c}\langle \mathsf{hash}(U, \mathsf{dec}(w_{\mathsf{ek}}, x)) \rangle\ .\ c(y)\ .\ \nu\, n_t\ .$
  $\overline{c}\langle (\!| n_t, \mathsf{spk}(\mathsf{F}_{\mathsf{join}}, (\!| \mathtt{tsk}, v' |\!), (\!| \zeta_I, w_{\mathsf{pk}}, N_I, U, (\!| n_t, y |\!) |\!)) |\!) \rangle\ .$
  $c(z)$ . let $\mathtt{cre} = \mathsf{clopen}(w_{\mathsf{pk}}, v', z)$ in
  if $\mathsf{checkclsign}(w_{\mathsf{pk}}, \mathtt{tsk}, \mathtt{cre}) = \mathsf{accept}$ then
  $\overline{a'_j}\langle (\!| \mathtt{cre}, \mathtt{tsk} |\!) \rangle$

$\mathsf{Sign}_{\mathsf{RSA}} \ \hat{=}$
  $a_s(w_{\mathsf{params}}, w_{\mathsf{bsn}}, w_{\mathsf{msg}}, w_{\mathsf{cre}}, w_{\mathsf{tsk}})\ .$
  let $w_{\mathsf{pk}} = \pi_1(w_{\mathsf{params}})$ in
  $c(x)\ .\ \nu\, n_t\ .\ \nu\, w\ .$
  if $w_{\mathsf{bsn}} = \perp$ then
    $\nu\, \zeta\ .$
    let $\widehat{\mathtt{cre}} = \mathsf{clcommit}(w_{\mathsf{pk}}, w, w_{\mathsf{cre}})$ in
    let $N_V = \mathsf{commit}(\zeta, w_{\mathsf{tsk}})$ in
    let $spk = \mathsf{spk}(\mathsf{F}_{\mathsf{sign}}, (\!| w_{\mathsf{tsk}}, w |\!), (\!| \zeta, w_{\mathsf{pk}}, N_V, \widehat{\mathtt{cre}}, (\!| n_t, x, w_{\mathsf{msg}} |\!) |\!))$ in
    $\overline{a'_s}\langle (\!| \zeta, w_{\mathsf{pk}}, N_V, \widehat{\mathtt{cre}}, n_t, spk |\!) \rangle$
   else
    let $\zeta = \mathsf{hash}(0, w_{\mathsf{bsn}})$ in
    let $\widehat{\mathtt{cre}} = \mathsf{clcommit}(w_{\mathsf{pk}}, w, w_{\mathsf{cre}})$ in
    let $N_V = \mathsf{commit}(\zeta, w_{\mathsf{tsk}})$ in
    let $spk = \mathsf{spk}(\mathsf{F}_{\mathsf{sign}}, (\!| w_{\mathsf{tsk}}, w |\!), (\!| \zeta, w_{\mathsf{pk}}, N_V, \widehat{\mathtt{cre}}, (\!| n_t, x, w_{\mathsf{msg}} |\!) |\!))$ in
    $\overline{a'_s}\langle (\!| \zeta, w_{\mathsf{pk}}, N_V, \widehat{\mathtt{cre}}, n_t, spk |\!) \rangle$
```

value w_{cnt} chosen by the host, and the TPM's endorsement key w_{ek}. The system parameters w_{params} are expected to be a triple containing the DAA public key w_{pk}, basename $w_{\mathsf{bsn_I}}$, and long-term key K_I. The process constructs the terms N_I and U in accordance with the protocol's description (Section 5.2) and outputs the values to the issuer. The process then receives a ciphertext x, which it decrypts, and outputs the hash of the plaintext paired with U. A nonce y is then input and a signature proof of knowledge is produced, using y as a source of randomness in the proof. Finally, the process inputs a signature z on the commitment U and concludes by outputting the attestation identity credential $\mathtt{cre}$ and TPM's secret $\mathtt{tsk}$ on the private channel a'_j, that is, the $\mathsf{Join}_{\mathsf{RSA}}$ process returns the values $\mathtt{cre}$ and $\mathtt{tsk}$ to the Signer process. The sign process $\mathsf{Sign}_{\mathsf{RSA}}$

is instantiated by inputting the sign algorithm's parameters: the RSA-based DAA system parameters w_{params}, the verifier's basename w_{bsn}, the message w_{msg} to be signed, the attestation identity credential w_{cre}, and the TPM's secret w_{tsk}. (For simplicity, we do not capture the possibility of signing a message produced by the TPM using private data.) The process recovers the DAA public key w_{pk} from the system parameters, and inputs a nonce x from the verifier. The if-then-else branch models the signer's ability to produce either linkable or unlinkable signatures, based upon the parameter w_{bsn}, in particular, the if-branch produces an unlinkable signature, whereas the else-branch produces a linkable signature. The process concludes by outputting a signature on the private channel a'_s, that is, the $\mathsf{Sign}_{\mathsf{RSA}}$ process returns the signature to the Signer process.

5.5 Analysis: Violating privacy

The DAA game biprocess $\mathsf{DAA\text{-}G}_{\mathsf{RSA}}$ derived from $\langle \mathsf{Join}_{\mathsf{RSA}}, \mathsf{Sign}_{\mathsf{RSA}} \rangle$ does not satisfy privacy. Informally, this can be observed by consideration of the following adversaries.

Passive adversary A passive adversary can violate privacy under the following assumptions: first, the identity of a trusted platform can be observed during the join algorithm[2]; secondly, there exists a basename which is shared between an issuer and a verifier; and, thirdly, a signer is willing to use the same basename with an issuer and verifier. By our second assumption, there exists an issuer's basename $\mathtt{bsn}_I$ and a verifier's basename $\mathtt{bsn}$ such that $\mathtt{bsn}_I = \mathtt{bsn}$. The attack proceeds as follows. Let us suppose that the trusted platform executes the join protocol with the issuer and subsequently runs the sign protocol with the verifier. Since the signer is willing to use the same basename with an issuer and verifier, it follows that $\zeta_I = \zeta$ and $N_I = N_V$. The commitments N_I and N_V are unique for a particular signer and the adversary knows the identity of the trusted platform that produced N_I during the join algorithm, it follows that the signer's identity can be revealed.

Corrupt issuers and verifiers Corrupt issuers and verifiers can violate privacy under the assumption that a signer is willing to use the same basename with an issuer and verifier. This is a special case of our passive attack: an issuer and verifier conspire to use the same basename (that is, $\mathtt{bsn}_I = \mathtt{bsn}$) and since the issuer knows the identity of the trusted platform that produced N_I, the identity of the signer can be revealed.

[2] The RSA-based DAA protocol [BCC04] does not specify how the issuer learns a trusted platform's public endorsement key during an execution of the join algorithm. However, it seems reasonable to assume that the public key would be sent as plaintext. By contrast, Cesena *et al.* [CLR+10, Ces10] define an extension of RSA-based DAA which uses TLS to hide the affiliation between groups and trusted platforms, this variant would thwart a passive adversary, but not corrupt issuers and verifiers.

The linchpin of these attacks is the willingness of a signer to use the same basename with an issuer and verifier. This can be justified as follows. Firstly, this mode of operation is not explicitly forbidden by the protocol definition [BCC04]. Secondly, this behaviour is expected when the issuer and verifier are the same entity, as demonstrated by Camenisch *et al.* [CL01, CH02] in the *idemix* system, for example. Finally, the signer has insufficient resources to detect the use of the same basename with an issuer and verifier.

Formally, Theorem 1 demonstrates that $\mathsf{DAA\text{-}G}_{\mathsf{RSA}}$ does not satisfy privacy. This result is witnessed using a context $C[\_]$ such that $\mathrm{fst}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}]) \to^* Q$ and Q can output on channel b, but there is no reduction $\mathrm{snd}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}]) \to^* Q'$ such that Q' can output on b, where both reductions are of the same length. Intuitively, the context behaves as follows. First, the context outputs system parameters $\llparenthesis \mathsf{pk}(sk_I), \mathtt{bsn}, K_I \rrparenthesis$. Secondly, the context executes the join algorithm with both signers and binds $\llparenthesis \mathsf{commit}(\mathsf{hash}(0, \mathtt{bsn}), \mathtt{tsk}), \mathsf{clcommit}(\mathsf{pk}(sk_I), v', \mathtt{tsk}) \rrparenthesis$ to x, where $\mathtt{tsk} = \mathsf{hash}(\mathsf{hash}(\mathtt{DAASeed}, \mathsf{hash}(K_I)), \mathtt{cnt}, 0)$ and $\mathtt{cnt}$, $\mathtt{DAASeed}$ and v' are restricted names. Thirdly, the context issues a challenge using the basename $\mathtt{bsn}$ and message msg, and binds T to y, where $\pi_3(T) = \mathsf{commit}(\mathsf{hash}(0, \mathtt{bsn}), \mathrm{diff}[\mathtt{tsk}, \mathtt{tsk}'])$, $\mathtt{tsk}' = \mathsf{hash}(\mathsf{hash}(\mathtt{DAASeed}', \mathsf{hash}(K_I)), \mathtt{cnt}', 0)$, and $\mathtt{cnt}'$ and $\mathtt{DAASeed}'$ are restricted names. Finally, the context compares $\pi_1(x)$ and $\pi_3(y)$ to derive a distinction between $\mathrm{fst}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}])$ and $\mathrm{snd}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}])$.

Theorem 1. *The pair of processes* $\langle \mathsf{Join}_{\mathsf{RSA}}, \mathsf{Sign}_{\mathsf{RSA}} \rangle$ *do not satisfy privacy.*

Proof. Let $\mathsf{DAA\text{-}G}_{\mathsf{RSA}}$ be the DAA game biprocess derived from $\langle \mathsf{Join}_{\mathsf{RSA}}, \mathsf{Sign}_{\mathsf{RSA}} \rangle$ and consider the evaluation context $C[\_]$ defined below:

$$\begin{array}{l}
\_ \mid \overline{c}\langle \llparenthesis \mathsf{pk}(sk_I), \mathtt{bsn}, K_I \rrparenthesis \rangle \ . \\
c(w) \ . \ c(x) \ . \ \overline{c}\langle \mathsf{penc}(w, n, n_e) \rangle \ . \ c(z_a) \ . \ \text{if } z_a = \mathsf{hash}(n_e, \pi_2(x)) \text{ then} \\
\overline{c}\langle n_i \rangle \ . \ c(z_s) \ . \ \overline{c}\langle \mathsf{clsign}(sk_I, e, v'', \pi_2(x)) \rangle \ . \\
c(w') \ . \ c(x') \ . \ \overline{c}\langle \mathsf{penc}(w', n, n_e) \rangle \ . \ c(z'_a) \ . \ \text{if } z'_a = \mathsf{hash}(n_e, \pi_2(x')) \text{ then} \\
\overline{c}\langle n_i \rangle \ . \ c(z'_s) \ . \ \overline{c}\langle \mathsf{clsign}(sk_I, e, v'', \pi_2(x')) \rangle \ . \\
\overline{c}\langle \llparenthesis \mathtt{bsn}, \mathsf{msg} \rrparenthesis \rangle \ . \ \overline{c}\langle n_v \rangle \ . \ c(y) \ . \ \text{if } \pi_1(x) = \pi_3(y) \text{ then } \overline{b}\langle \mathsf{fail} \rangle \text{ else } 0
\end{array}$$

We have the following reductions:

$$\begin{aligned}
\mathrm{fst}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}]) &\to^* C'[\text{if } M = M \text{ then } \overline{b}\langle \mathsf{fail} \rangle \text{ else } 0] \\
\mathrm{snd}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}]) &\to^* C'[\text{if } M = N \text{ then } \overline{b}\langle \mathsf{fail} \rangle \text{ else } 0],
\end{aligned}$$

where $M = \mathsf{commit}(\mathsf{hash}(0, \mathtt{bsn}), \mathsf{hash}(\mathsf{hash}(\mathtt{DAASeed}, \mathsf{hash}(K_I)), \mathtt{cnt}, 0))$ and $N = \mathsf{commit}(\mathsf{hash}(0, \mathtt{bsn}), \mathsf{hash}(\mathsf{hash}(\mathtt{DAASeed}', \mathsf{hash}(K_I)), \mathtt{cnt}', 0))$. It follows that $\mathrm{fst}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}]) \not\sim \mathrm{snd}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}])$ because $\mathrm{fst}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}])$ can output on channel b but $\mathrm{snd}(C[\mathsf{DAA\text{-}G}_{\mathsf{RSA}}])$ cannot and, therefore, $\mathsf{DAA\text{-}G}_{\mathsf{RSA}}$ does not satisfy privacy. □

The context $C[\_]$ was discovered by manual reasoning and, subsequently, we found that ProVerif finds a similar context. (Analysis takes 39 seconds using ProVerif 1.86pl3 on CentOS 5.7 with 2.40GHz Intel Xeon and 4GB memory. Our

ProVerif scripts are available online.) An attack in the computational model follows immediately from our result in the symbolic setting (Theorem 1), see Smyth [Smy11, Appendix B] for details.

5.6 Solution: Fixing RSA-based DAA

The protocol can be fixed by refining the definition of ζ, namely, let $\langle \mathsf{Join}_{\mathsf{RSA}'}, \mathsf{Sign}_{\mathsf{RSA}'} \rangle$ refine $\langle \mathsf{Join}_{\mathsf{RSA}}, \mathsf{Sign}_{\mathsf{RSA}} \rangle$ by redefining ζ as $\zeta = \mathsf{hash}(1, \mathtt{bsn})$. The attacks presented are no longer possible, regardless of whether $\mathtt{bsn}_I = \mathtt{bsn}$. Furthermore, the revised pair of processes $\langle \mathsf{Join}_{\mathsf{RSA}'}, \mathsf{Sign}_{\mathsf{RSA}'} \rangle$ satisfy privacy; this can be automatically verified using ProVerif (verification takes 26 seconds using ProVerif 1.86pl3 on CentOS 5.7 with 2.40GHz Intel Xeon and 4GB memory; our ProVerif scripts are available online).

5.7 Limitations

Our representation of cryptographic primitives and relations between primitives (Section 5.3) is an abstraction. For instance, we assume cryptography is perfect (i.e., the attacker is only able to perform cryptographic operations when in possession of the required keys) and do not capture the low-level mathematical details of the cryptography. Moreover, on the basis that both the host and TPM must be honest for privacy, we do not distinguish between operations performed by the host and TPM. The difficulty of capturing cryptography in the symbolic model is well known [RS98, AR00, AR02, War03, War05] and is particularly difficult in this instance due to the complexity of DAA. Nevertheless, our analysis (Section 5.5) discovers an attack against the RSA-based DAA protocol, thereby demonstrating the practicalitiy of symbolic analysis. However, we acknowledge that our security results for the fixed RSA-based DAA protocol (Section 5.6) do not imply the absence of real-world attacks, due to the gulf between our symbolic representation of cryptography and the concrete implementation.

6 Balancing privacy and accountability

Balancing the privacy demands of users and the accountability needs of issuers and verifiers is a fundamental objective of Direct Anonymous Attestation schemes, in particular, DAA schemes permit signatures to be linked, without revealing the identity of the signer. The degrees of linkability are identified below, with reference to an application domain in which an honest issuer offers membership to a single group of signers and several verifiers offer multiple services.

- *Single-service linkability.* A verifier offering a single service is able to link transactions made by a given signer.
- *Cross-service linkability.* A verifier offering multiple services is able to link transactions made by a given signer over multiple services, when the

services share the same basename.

- *Cross-verifier linkability.* Multiple verifiers offering services are able to link transactions made by a given signer across all the verifiers, when the services share the same basename.

In this section, we reflect upon the notions of linkability for Direct Anonymous Attestation schemes and extend the degree of privacy available in such schemes.

6.1 Linkability between an issuer's groups

Let us identify an issuer by its long-term key K_I, and recall that the game-based security definition by Brickell, Chen & Li [BCL08b, BCL09] assumes that an issuer controls a single group of signers, where the group of signers is identified by a public key $\mathsf{pk}(sk_I)$. In this section, we generalise to the situation in which an issuer may issue credentials to several groups of signers, where each group of signers is associated with a different key $\mathsf{pk}(sk_I)$. In this situation, one can ask the following question:

- Can a verifier link two signatures constructed using distinct DAA public keys $\mathsf{pk}(sk_I)$ and $\mathsf{pk}(sk'_I)$, each belonging to the same issuer? We call this *linkability between an issuer's groups.*

The RSA-based scheme permits linkability between an issuer's groups, when the signatures share the same basename. This can be observed as follows: given the issuer's long-term key K_I and the basename $\mathtt{bsn}$ such that $\mathtt{bsn} \neq \bot$, the TPM's secret $\mathtt{tsk} = \mathsf{hash}(\mathsf{hash}(\mathtt{DAASeed}, \mathsf{hash}(K_I)), \mathtt{cnt}, \mathtt{0})$ and signatures produced using $\mathtt{tsk}$ will include $N_V = \mathsf{commit}(\zeta, \mathtt{tsk})$, where $\zeta = \mathsf{hash}(\mathtt{1}, \mathtt{bsn})$. (In the computational setting, linkability between an issuer's groups assumes that the groups' public keys share the same modulus Γ and order ρ, see [BCC04, §4.3] for definitions of Γ and ρ.) We can modify the RSA-based scheme to prevent linkability between an issuer's groups by defining $\zeta = \mathsf{hash}(\mathtt{1}, \mathtt{bsn}, \mathsf{pk}(sk_I))$, rather than $\zeta = \mathsf{hash}(\mathtt{1}, \mathtt{bsn})$. Intuitively, linkability between an issuer's groups strengthens accountability and weakens privacy, hence, the original RSA-based scheme provides stronger accountability, whereas our modification provides stronger privacy.

6.2 Practical guidelines for basenames

Basenames are particularly sensitive for DAA because they enable linkability and the ability to uniquely identify a set of services for which a basename can be used is a prerequisite of user-controlled linkability. However, no methodology for basename construction has been defined and this may lead to a security vulnerability. For example, a signer may inadvertently permit signatures to be linked, simply by using the same basename for multiple signatures; we argue that this scenario is likely because signers have insufficient resources to maintain a history of all basenames. We overcome this problem with the presentation of guidelines for the construction of basenames. First, basenames should be constructed from service-specific data such as the following:

- Service information, for example, issuer's public key, verifier's public key, service URL, and terms and conditions of service.
- Basename validity date, for example, start and expiry dates.
- DAA signing mode, for example, Attestation Identity Key (AIK) signing, Platform Configuration Register (PCR) signing, and external input signing.

Secondly, given a basename constructed in this manner, a signer can evaluate whether the basename is suitable for use with a particular service. It follows immediately that our construction helps *inform* a signer that signatures can be linked.

7 Further work and conclusion

Direct Anonymous Attestation is a relatively new concept and its properties merit further study, in particular, correctness, linkability, non-frameability and unforgeability have received limited attention. Extending this work to include a complete definition of DAA properties would be an interesting direction for the future. Moreover, establishing a unified definition which includes all properties (that is, anonymity, correctness, linkability, non-frameability, unforgeability, and unlinkability) would be of interest to reduce the verification workload. As a starting point, this could be achieved by developing the formalisation of join and sign algorithms, modelled by $\langle \mathsf{Join}, \mathsf{Sign} \rangle$, to distinguish between operations performed by the host and those performed by the TPM. This distinction is not necessary for our definition of privacy because this property can only be achieved if both the host and TPM are trusted. By contrast, a corrupt host – even in collaboration with a corrupt TPM (where the TPM is known to be rogue) – should not be able to violate accountability properties and, therefore, an alternative model of $\langle \mathsf{Join}, \mathsf{Sign} \rangle$ would be required such that the actions performed by the host and TPM are distinguished.

For privacy it is necessary to ensure a distinct basename is used during the Challenge. Since the applied pi calculus does not record state, this is achieved by an abstraction. Accordingly, we believe the definition is necessary, but may not be sufficient. This limitation could be overcome by introducing a stateful variant of the applied pi calculus, indeed, Arapinis, Ritter & Ryan [ARR11] make some progress in this direction. A further limitation of our privacy definition is the restriction to settings with one issuer, indeed, this corresponds to the cryptographic game. Extending the definition to multiple issuers remains as future work.

Conclusion This article presents a definition of privacy for Direct Anonymous Attestation protocols. The definition is expressed as an equivalence property which is suited to automated reasoning and the practicality of the approach is demonstrated by evaluating the RSA-based Direct Anonymous Attestation protocol. The RSA-based scheme is particularly significant because support is mandated by the TPM specification version 1.2, which has been implemented and deployed in over 500 million computers (although the number of TPMs in active use is estimated to be significantly smaller). The analysis discovers a vulnerability which can be exploited by a passive adversary and, under weaker assumptions, by corrupt issuers and verifiers. A security fix is identified and the revised protocol is shown to satisfy our definition of privacy. The fix only affects the host's part of the protocol and, therefore, no hardware changes to the TPM are required. Furthermore, the fix has influenced the design of subsequent DAA schemes, for example, [BCL08a, BCL09].

Acknowledgements

We are particularly grateful to Tom Chothia and Andy Gordon for their careful reading of an earlier version of this work and their constructive feedback. We are also grateful to David Bernhard for his discussion with regards Section 6 and the anonymous reviewers who provided constructive criticism. This research was funded by the ERC projects *CRYSP* (259639) and *ProSecure* (258865). Smyth's work was partly conducted at: the School of Computer Science, University of Birmingham, UK; Loria, CNRS & INRIA Nancy Grand Est, France; and INRIA Paris-Rocquencourt, France

A A brief review of DAA schemes

The first concrete Direct Anonymous Attestation scheme was introduced by Brickell, Camenisch & Chen [BCC04] and is based upon RSA. However, RSA-based cryptography requires larger keys than equivalent ECC-based schemes. Moreover, the RSA-based DAA protocol is reliant on the strong RSA and decisional Diffie-Hellman assumptions, and some users are uncomfortable with the strong RSA assumption. This motivated the work of Brickell, Chen & Li [BCL08a, BCL09] who provide the first ECC-based DAA protocol using symmetric pairing. This scheme is more efficient and, therefore, better suited to devices with limited resources, such as the TPM. Furthermore, the ECC-based protocol is reliant on the LRSW [LRSW00] and decisional Bilinear Diffie-Hellman assumptions, which some users may prefer. Chen, Morrissey & Smart [CMS08a, CMS08b] extended the scheme based upon symmetric pairing to an asymmetric setting to improve efficiency. However, Li discovered a vulnerability in the asymmetric scheme which violates basename linkability and Chen & Li propose a fix [CL10a]; a further attack has been identified by Chen, Morrissey & Smart [CMS09] which, in theory, violates unforgeability. In addition, Chen, Morrissey & Smart [CMS11] have found theoretical accountability attacks against the symmetric pairing based scheme [BCL08a, BCL09] and the original

RSA-based scheme [BCC04]. The Chen, Morrissey & Smart [CMS09, CMS11] attacks allow a malicious host to extract the TPM's secret `tsk`, if the protocol is implemented in hardware without *stage control mechanisms*; the host can then forge signatures. However, since the TPM provides stage control protection, there is no practical threat in the current setting; but, these attacks are of practical interest because they identify settings in which DAA protocols cannot be deployed (for example, in other trusted computing settings which do not use the TPM). We remark that the analysis of unforgeability in the RSA-based scheme by Backes, Maffei & Unruh [BMU08] could not identify the Chen, Morrissey & Smart attack because they consider a setting where the host and TPM are both honest. Chen, Morrissey & Smart [CMS09, CMS11] also propose a new asymmetric scheme and Chen, Page & Smart [CPS10] propose an optimisation, moreover, Chen [Che10b] provides a further optimisation to the Chen, Page & Smart scheme. Brickell, Chen & Li [BCL11] have shown that an adversary can forge signatures in the variant by Chen and propose a fix. We are aware of six further ECC-based DAA protocols: Chen & Feng [CF08], Brickell & Li [BL09a, BL09b], Chen [Che10a, Che11], Brickell & Li [BL10], Bernhard *et al.* [BFG$^+$11], and Bernhard, Fuchsbauer & Ghadafi [BFG12].

References

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *POPL'01: 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 104–115. ACM Press, 2001.

[And03] Ross J. Anderson. Cryptography and Competition Policy - Issues with 'Trusted Computing'. In *PODC'03: 22nd ACM Symposium on Principles of Distributed Computing*, pages 3–10. ACM Press, 2003.

[And04] Ross J. Anderson. Cryptography and Competition Policy - Issues with 'Trusted Computing'. In L. Jean Camp and Stephen Lewis, editors, *Economics of Information Security*, volume 12 of *Advances in Information Security*, pages 35–52. Springer, 2004.

[AR00] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). In *IFIP TCS'00: 1st International Conference on Theoretical Computer Science*, volume 1872 of *LNCS*, pages 3–22. Springer, 2000.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[ARR11] Myrto Arapinis, Eike Ritter, and Mark Ryan. StatVerif: Verification of Stateful Processes. In *CSF'11: 24th IEEE Computer Secu-*

rity Foundations Symposium, pages 33–47. IEEE Computer Society, 2011.

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, February–March 2008.

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct Anonymous Attestation. In *CCS'04: 11th ACM Conference on Computer and Communications Security*, pages 132–145. ACM Press, 2004.

[BCC05] Ernie Brickell, Jan Camenisch, and Liqun Chen. The DAA scheme in context. In Chris Mitchell, editor, *Trusted Computing*, volume 6 of *Professional Applications of Computing Series*, pages 143–174. The Institute of Engineering and Technology, 2005.

[BCL08a] Ernie Brickell, Liqun Chen, and Jiangtao Li. A New Direct Anonymous Attestation Scheme from Bilinear Maps. In *Trust'08: 1st International Conference on Trusted Computing and Trust in Information Technologies*, volume 4968 of *LNCS*, pages 166–178, 2008.

[BCL08b] Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified Security Notions of Direct Anonymous Attestation and a Concrete Scheme from Pairings. Cryptology ePrint Archive, Report 2008/104, 2008.

[BCL09] Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of Direct Anonymous Attestation and a concrete scheme from pairings. *International Journal of Information Security*, 8(5):315–330, 2009.

[BCL11] Ernie Brickell, Liqun Chen, and Jiangtao Li. A (Corrected) DAA Scheme Using Batch Proof and Verification. In *INTRUST'11: 3rd International Conference on Trusted Systems*, volume 7222 of *LNCS*, pages 304–337. Springer, 2011.

[BFG$^+$11] D. Bernhard, G. Fuchsbauer, E. Ghadafi, N.P. Smart, and B. Warinschi. Anonymous attestation with user-controlled linkability. Cryptology ePrint Archive, Report 2011/658, 2011.

[BFG12] D. Bernhard, G. Fuchsbauer, and E. Ghadafi. Efficient Signatures of Knowledge and DAA in the Standard Model. Cryptology ePrint Archive, Report 2012/475, 2012.

[BHM08] Michael Backes, Cătălin Hriţcu, and Matteo Maffei. Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In *CSF'08: 21st IEEE Computer Security Foundations Symposium*, pages 195–209. IEEE Computer Society, 2008.

[BL09a] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: A Remote Anonymous Attestation Scheme for Hardware Devices. *Intel Technology Journal Security*, 13(2):96–111, June 2009.

[BL09b] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID from Bilinear Pairing. Cryptology ePrint Archive, Report 2009/095, 2009.

[BL10] Ernie Brickell and Jiangtao Li. A Pairing-Based DAA Scheme Further Reducing TPM Resources. In *TRUST'10: 3rd International Conference on Trust and Trustworthy Computing*, volume 6101 of *LNCS*, pages 181–195. Springer, 2010.

[Bla04] Bruno Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *S&P'04: 25th IEEE Symposium on Security and Privacy*, pages 86–100. IEEE Computer Society, 2004.

[BMU08] Michael Backes, Matteo Maffei, and Dominique Unruh. Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol. In *S&P'08: 29th IEEE Symposium on Security and Privacy*, pages 202–215. IEEE Computer Society, 2008.

[BS11] Bruno Blanchet and Ben Smyth. ProVerif 1.86pl3: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial. `http://www.proverif.ens.fr/`, 2011.

[Ces10] Emanuele Cesena. *Trace Zero Varieties in Pairing-based Cryptography.* PhD thesis, Department of Mathematics, Universitá degli Studi Roma Tre, 2010.

[CF08] Xiaofeng Chen and Dengguo Feng. Direct Anonymous Attestation for Next Generation TPM. *Journal of Computers*, 3(12):43–50, December 2008.

[CH02] Jan Camenisch and Els Van Herreweghen. Design and Implementation of the *idemix* Anonymous Credential System. In *CCS'02: 9th ACM Conference on Computer and Communications Security*, pages 21–30. ACM Press, 2002.

[Che10a] Liqun Chen. A DAA Scheme Requiring Less TPM Resources. Cryptology ePrint Archive, Report 2010/008, 2010.

[Che10b] Liqun Chen. A DAA Scheme Using Batch Proof and Verification. In *TRUST'10: 3rd International Conference on Trust and Trustworthy Computing*, volume 6101 of *LNCS*, pages 166–180. Springer, 2010.

[Che11] Liqun Chen. A DAA Scheme Requiring Less TPM Resources. In *INSCRYPT'09: 5th International Conference on Information Security and Cryptology*, volume 6151 of *LNCS*, pages 350–365. Springer, 2011.

[CL01] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT'01: 20th International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[CL03] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *SCN'02: 3rd Conference on Security in Communication Networks*, volume 2576 of *LNCS*, pages 268–289. Springer, 2003.

[CL10a] Liqun Chen and Jiangtao Li. A note on the Chen-Morrissey-Smart DAA scheme. *Information Processing Letters*, 110(12-13):485–488, 2010.

[CL10b] Liqun Chen and Jiangtao Li. Revocation of Direct Anonymous Attestation. In *INTRUST'10: 2nd International Conference on Trusted Systems*, volume 6802 of *LNCS*, pages 128–147. Springer, 2010.

[CLR$^+$10] Emanuele Cesena, Hans Löhr, Gianluca Ramunno, Ahmad-Reza Sadeghi, and Davide Vernizzi. Anonymous Authentication with TLS and DAA. In *TRUST'10: 3rd International Conference on Trust and Trustworthy Computing*, volume 6101 of *LNCS*, pages 47–62. Springer, 2010.

[CMS08a] Liqun Chen, Paul Morrissey, and Nigel P. Smart. On Proofs of Security for DAA Schemes. In *ProvSec'08: 2nd International Conference on Provable Security*, volume 5324 of *LNCS*, pages 156–175. Springer, 2008.

[CMS08b] Liqun Chen, Paul Morrissey, and Nigel P. Smart. Pairings in Trusted Computing. In *Pairing'08: 2nd International Conference on Pairing-Based Cryptography*, volume 5209 of *LNCS*, pages 1–17. Springer, 2008.

[CMS09] Liqun Chen, Paul Morrissey, and Nigel P. Smart. DAA: Fixing the pairing based protocols. Cryptology ePrint Archive, Report 2009/198, 2009. *This paper was withdrawn on 4 Dec 2011 because the simulation-based security definition is unsatisfiable, nevertheless, the unforgeability attacks (which we discuss in A) are still valid.*

[CMS11] Liqun Chen, Paul Morrissey, and Nigel P. Smart. DAA: Fixing the pairing based protocols. Unpublished draft, 2011.

[CPS10] Liqun Chen, Dan Page, and Nigel P. Smart. On the Design and Implementation of an Efficient DAA Scheme. In *CARDIS'10: 8th Internation Conference on Smart Card Research and Advanced Application*, volume 6035 of *LNCS*, pages 223–237. Springer, 2010.

[CS97] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups. In *CRYPTO'97: 17th International Cryptology Conference*, volume 1294 of *LNCS*, pages 410–424. Springer, 1997.

[DDS10] Morten Dahl, Stéphanie Delaune, and Graham Steel. Formal Analysis of Privacy for Vehicular Mix-Zones. In *ESORICS'10: 15th European Symposium on Research in Computer Security*, volume 6345 of *LNCS*, pages 55–70. Springer, 2010.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *CSFW'06: 19th Computer Security Foundations Workshop*, pages 28–42. IEEE Computer Society, 2006.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, July 2009.

[DKR10] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster. In David Chaum, Markus Jakobsson, Ronald L. Rivest, and Peter Y. A. Ryan, editors, *Towards Trustworthy Elections: New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 289–309. Springer, 2010.

[DRS07] Stéphanie Delaune, Mark D. Ryan, and Ben Smyth. Automatic verification of privacy properties in the applied pi-calculus. Technical Report CSR-07-12, School of Computer Science, University of Birmingham, 2007.

[DRS08] Stéphanie Delaune, Mark D. Ryan, and Ben Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, volume 263 of *International Federation for Information Processing (IFIP)*, pages 263–278. Springer, 2008.

[Int09] International Organization for Standardization. *ISO/IEC 11889: Information technology – Trusted Platform Module*, 2009.

[Int11] International Organization for Standardization. *ISO/IEC WD 20008-2 (Working Draft) Information technology – Security techniques – Anonymous digital signature – Part 2: Mechanisms using a group public key*, 2011.

[KM06] Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". II. In *INDOCRYPT'06: 7th International Conference on Cryptology*, volume 4329 of *LNCS*, pages 148–175. Springer, 2006.

[KM07] Neal Koblitz and Alfred Menezes. Another Look at “Provable Security”. *Journal of Cryptology*, 20(1):3–37, 2007.

[KR05] Steve Kremer and Mark D. Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In *ESOP'05: 14th European Symposium on Programming*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

[LRSW00] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym Systems. In *SAC'99: 6th International Workshop on Selected Areas in Cryptography*, volume 1758 of *LNCS*, pages 184–199. Springer, 2000.

[Lys02] Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2002.

[Mar08] Andrew Martin. The ten-page introduction to Trusted Computing. Technical Report CS-RR-08-11, University of Oxford, 2008.

[Mea03] Catherine Meadows. Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends. *Selected Areas in Communications*, 21(1):44–54, 2003.

[Moo88] Judy H. Moore. Protocol Failures in Cryptosystems. In *Proceedings of the IEEE*, volume 76, pages 594–602. IEEE Computer Society, 1988.

[RS98] Peter Y. A. Ryan and Steve A. Schneider. An Attack on a Recursive Authentication Protocol. A Cautionary Tale. *Information Processing Letters*, 65(1):7–10, 1998.

[RS11] Mark D. Ryan and Ben Smyth. Applied pi calculus. In Véronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, chapter 6. IOS Press, 2011.

[Rud07] Carsten Rudolph. Covert Identity Information in Direct Anonymous Attestation (DAA). In *SEC'07: 22nd International Information Security Conference*, volume 232 of *International Federation for Information Processing (IFIP)*, pages 443–448. Springer, 2007.

[Smy11] Ben Smyth. *Formal verification of cryptographic protocols with automated reasoning.* PhD thesis, School of Computer Science, University of Birmingham, 2011.

[SRC07] Ben Smyth, Mark D. Ryan, and Liqun Chen. Direct Anonymous Attestation (DAA): Ensuring privacy with corrupt administrators. In *ESAS'07: 4th European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, volume 4572 of *LNCS*, pages 218–231. Springer, 2007.

[SRC11] Ben Smyth, Mark D. Ryan, and Liqun Chen. Formal analysis of anonymity in ECC-based Direct Anonymous Attestation schemes. In *FAST'11: 8th International Workshop on Formal Aspects of Security and Trust*, volume 7140 of *LNCS*, pages 245–262. Springer, 2011.

[Sta02] Richard Stallman. Can You Trust Your Computer? Free Software Foundation, 2002. `http://www.gnu.org/philosophy/can-you-trust.html`.

[Sta10] Richard Stallman. Can You Trust Your Computer? In *Free Software, Free Society*, chapter 32. GNU Press, 2nd edition, 2010.

[Tar10] Christopher Tarnovsky. Deconstructing a 'Secure' Processor. In *Black Hat DC 2010*, 2010. `https://media.blackhat.com/bh-dc-10/video/Tarnovsky_Chris/BlackHat-DC-2010-Tarnovsky-DeconstructProcessor-video.m4v`.

[TCG07] Trusted Computing Group. *TPM Specification version 1.2*, 2007.

[Tru11] Trusted Computing Group. Do You Know? A Few Notes on Trusted Computing Out in the World. `http://www.trustedcomputinggroup.org/community/2011/03/do_you_know_a_few_notes_on_trusted_computing_out_in_the_world`, March 2011.

[Tru12] Trusted Computing Group. The Case for Turning on Trusted Platform Modules. `http://www.trustedcomputinggroup.org/resources/the_case_for_turning_on_trusted_platform_modules`, November 2012.

[War03] Bogdan Warinschi. A Computational Analysis of the Needham-Schröeder-(Lowe) Protocol. In *CSFW'03: 16th IEEE Computer Security Foundations Workshop*, pages 248–262. IEEE Computer Society, 2003.

[War05] Bogdan Warinschi. A computational analysis of the Needham-Schroeder-(Lowe) protocol. *Journal of Computer Security*, 13(3):565–591, 2005.