Formal analysis of anonymity in ECC-based Direct Anonymous Attestation schemes*

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

[1] Toshiba Corporation, Kawasaki, Japan
[2] School of Computer Science, University of Birmingham, UK
[3] HP Laboratories, Bristol, UK

Abstract. A definition of user-controlled anonymity is introduced for Direct Anonymous Attestation schemes. The definition is expressed as an equivalence property suited to automated reasoning using ProVerif and the practicality of the definition is demonstrated by examining the ECC-based Direct Anonymous Attestation protocol by Brickell, Chen & Li. We show that this scheme is secure under the assumption that the adversary obtains no advantage from re-blinding a blind signature.

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. Cryptographic operations, by their nature, may reveal a platform's identity and as a consequence the TPM has been perceived as threat to privacy by some users. Brickell, Camenisch & Chen [1] have introduced the notion of Direct Anonymous Attestation (DAA) to overcome these privacy concerns. More precisely, DAA is a remote authentication mechanism for trusted platforms which provides user-controlled anonymity and traceability. The concept is based upon group signatures with stronger anonymity guarantees; in particular, the identity of a signer can never be revealed, but signatures may be linked with the signer's consent, 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 provided by an issuer. The issuer authenticates the host as a trusted platform and grants an *attestation identity credential* (occasionally abbreviated *credential*). The host can now produce signatures using the credential, thereby permitting a verifier to authenticate the host as a group member and therefore a trusted platform.

Brickell, Chen & Li [2, 3] and Chen [4, 5] characterise the following properties for Direct Anonymous Attestation schemes:

* Ben Smyth's work was partly done at Loria, CNRS & INRIA Nancy Grand Est, France and the School of Computer Science, University of Birmingham, UK.

- *User-controlled anonymity.*
  - *Privacy.* The identity of a signer cannot be revealed from a signature.
  - *Unlinkability.* Signatures cannot be linked without the signer's consent.
- *User-controlled traceability.*
  - *Unforgeability.* Signatures cannot be produced without a TPM.
  - *Basename linkability.* Signatures are linkable with the signer's consent.
- *Non-frameability.* An adversary cannot produce a signature associated with an honest TPM.
- *Correctness.* Valid signatures can be verified and, where applicable, linked.

The contrasting nature of anonymity and traceability properties aims to balance the privacy demands of users and the accountability needs of administrators.

Contribution. A definition of user-controlled anonymity is presented as an equivalence property which is suited to automated reasoning using ProVerif. Informally, the 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 examining user-controlled anonymity in the ECC-based DAA protocol [6, 3]. Support for the ECC-based scheme is mandated by the TPM.next specification [7], which is due to replace TPM version 1.2. Moreover, the scheme has been included in the ISO/IEC anonymous digital signature standard [8]. Unfortunately, we could not prove any results in the general case and we, therefore, focus on proving the security of the scheme in a model where the adversary is forbidden from re-blinding a blind signature.

Related work. In the computational model, Brickell, Camenisch & Chen [1] and Chen, Morrissey & Smart [9–11] introduce simulation-based models of security, and Brickell, Chen & Li [2, 3] propose a game-based security definition; the relationship between the simulation-based models and the game-based definition is unknown [11, pp158]. We consider a symbolic definition, based upon the game-based definition. By comparison, Backes, Maffei & Unruh [12] formalised an earlier notion of user-controlled anonymity (informally described in [1]) 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, the formalisation pre-dates the user-controlled anonymity definitions by Brickell, Chen & Li and Chen, Morrissey & Smart and considers a conceptually weaker adversary; for example, the following scenario is not considered: 1) signer $\mathcal{A}$ obtains a credential $\texttt{cre}_A$ and produces arbitrary many signatures; 2) signer $\mathcal{B}$ obtains a credential $\texttt{cre}_B$ and produces arbitrary many signatures; and 3) the adversary attempts to distinguish between two fresh signatures produced by the signers using credentials $\texttt{cre}_A$ and $\texttt{cre}_B$. Finally, our definition is intuitively simpler, which should aid analysis and, in particular, be better suited to automated reasoning.

2 Preliminaries: Calculus of ProVerif

We adopt a dialect [13, 14] of the applied pi calculus [15, 16] which is suited to automated reasoning using Blanchet's ProVerif [17].

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. Substitutions $\{^M/_x\}$ replace 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 an equational theory E, that is, a finite set of equations of the form $M = N$. We define $=_E$ as the smallest equivalence relation on terms that contains E, and is closed under application of constructors, substitution of terms for variables, and bijective renaming of names. 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; moreover, the variables of M' are bound in $M'_1, \ldots, M'_l$, and variables are subject to renaming. The term $g(M_1, \ldots, M_l)$ is defined iff 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 $M'\sigma$.

The grammar for terms and processes is presented 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. The syntax does not include the conditional if $M = N$ then P else Q, but this 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 $\mathsf{eq}(x, x) \to x$. We always include this destructor in Σ. The rest of the syntax is standard (see Blanchet [13, 14] 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 ($\to_\Sigma$) in association with the auxiliary rules for term evaluation ($\Downarrow_\Sigma$) and structural equivalence ($\equiv$), the structural equivalence rules are standard and we omit them for brevity (see Blanchet [13, 14] for details). Both $\equiv$ and $\to_\Sigma$ are defined only on closed processes. We write $\to^*_\Sigma$ for the reflexive and transitive closure of $\to_\Sigma$, and $\to^*_\Sigma{\equiv}$ for its union with $\equiv$; we occasionally abbreviate $\to_\Sigma$ as $\to$ and $\Downarrow_\Sigma$ as $\Downarrow$.

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

Fig. 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

Fig. 2 Semantics for terms and processes

$M \Downarrow M$

eval $h(D_1, \ldots, D_n) \Downarrow N\sigma$
if $h(N_1, \ldots, N_n) \to N \in \mathrm{def}_\Sigma(h)$,
and σ is such that for all i, $D_i \Downarrow M_i$ and $\Sigma \vdash M_i = N_i\sigma$

$\overline{N}\langle M\rangle.Q \mid N'(x).P \;\to\; Q \mid P\{^M/_x\}$ (Red I/O)
if $\Sigma \vdash N = N'$

let $x = D$ in P else $Q \to P\{^M/_x\}$ (Red Fun 1)
if $D \Downarrow M$

let $x = D$ in P else $Q \to Q$ (Red Fun 2)
if there is no M such that $D \Downarrow M$

$!P \;\to\; P \mid !P$ (Red Repl)

$P \;\to\; Q \;\Rightarrow\; P \mid R \;\to\; Q \mid R$ (Red Par)

$P \;\to\; Q \;\Rightarrow\; \nu\, a.P \;\to\; \nu\, a.Q$ (Red Res)

$P' \equiv P,\ P \;\to\; Q,\ Q \equiv Q' \;\Rightarrow\; P' \;\to\; Q'$ (Red $\equiv$)

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 the definition of contexts to permit the use of diff, and sometimes refer to contexts without diff as plain contexts.

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

Fig. 3 Generalised semantics for biprocesses

$\overline{N}\langle M\rangle.Q \mid N'(x).P \rightarrow Q \mid P\{^{M}/_{x}\}$ (Red I/O)
if $\Sigma \vdash \mathrm{fst}(N) = \mathrm{fst}(N')$ and $\Sigma \vdash \mathrm{snd}(N) = \mathrm{snd}(N')$

$\mathrm{let}\ x = D\ \mathrm{in}\ P\ \mathrm{else}\ Q \rightarrow P\{^{\mathrm{diff}[M_1, M_2]}/_{x}\}$ (Red Fun 1)
if $\mathrm{fst}(D) \Downarrow M_1$ and $\mathrm{snd}(D) \Downarrow M_2$

$\mathrm{let}\ x = D\ \mathrm{in}\ P\ \mathrm{else}\ Q \rightarrow Q$ (Red Fun 2)
if there is no M_1 such that $\mathrm{fst}(D) \Downarrow M_1$ and
there is no M_2 such that $\mathrm{snd}(D) \Downarrow M_2$

Assumptions and notation. In this paper, 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 $\mathsf{fst}(\mathsf{pair}(x, y)) \rightarrow x$ and $\mathsf{snd}(\mathsf{pair}(x, y)) \rightarrow y$. 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 $\pi_i(M)$.

2.2 Observational equivalence

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 $\mathrm{fn}(C) \cap \mathrm{fn}(M) = \emptyset$ and $\Sigma \vdash M = M'$. The definition of observational equivalence [13, 14] 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 define observational equivalence as a property of biprocesses.

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

Blanchet, Abadi & Fournet [13, 14] have shown that a biprocess P satisfies observational equivalence when reductions in $\mathrm{fst}(P)$ or $\mathrm{snd}(P)$ imply reductions in P; this proof technique is formalised using the notion of *uniformity*.

Definition 3 (Uniform). *A biprocess* P *is* uniform *if for all processes* Q_1 *such that* $\mathrm{fst}(P) \rightarrow Q_1$, *then* $P \rightarrow Q$ *for some biprocess* Q, *where* $\mathrm{fst}(Q) \equiv Q_1$, *and symmetrically for* $\mathrm{snd}(P) \rightarrow Q_2$.

Definition 4 (Strong uniformity). *A closed biprocess* P satisfies strong uniformity *if for all plain evaluation contexts* C *and biprocesses* Q *such that* $C[P] \rightarrow^* \equiv Q$, *then* Q *is uniform.*

Theorem 1 (Strong uniformity implies equivalence). *Given a closed biprocess* P, *if* P *satisfies strong uniformity, then* P *satisfies observational equivalence.*

3 Formalising DAA protocols

A Direct Anonymous Attestation scheme allows remote authentication of trusted platforms, and comprises of five algorithms, each of which will now be discussed.

Setup. The setup algorithm is primarily used by the issuer to construct a public 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. On successful completion of the join algorithm, the issuer grants the trusted platform with an attestation identity credential `cre` based upon a secret `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 `cre` and secret `tsk`, which asserts group membership and therefore trusted platform status. In addition to `cre` and `tsk`, the algorithm takes as input a message m and a basename `bsn`. The basename is used to control linkability between signatures: if $\texttt{bsn} = \perp$, then signatures should be unlinkable; otherwise, signatures produced by the same signer and based upon the same basename can be linked.

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.

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

3.1 Applied pi process specification

This paper considers user-controlled anonymity, which is dependent on a trusted platform's behaviour, that is, the join and sign algorithms. Formally, these algorithms are modelled by a pair of processes $\langle \mathsf{Join}, \mathsf{Sign} \rangle$.

The signer (or trusted platform) is able to execute arbitrarily many instances of the join and sign algorithms to become a member of a group and, subsequently, produce signatures as a group member. This behaviour is captured by the Signer process modelled below. The join and sign algorithms are modelled by the processes Join and Sign, which are expected to behave like services, that is, they can be called by, and return results to, the Signer process. The communication between the Signer and Join/Sign processes is achieved using private communication over channels a_j, a'_j, a_s, and a'_s. In essence, the private channel communication models the internal bus used by computer systems for communication between the host and TPM.

The process Signer instantiates arbitrarily many instances of the Join and Sign processes. 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.

$$
\begin{aligned}
\mathsf{Signer} = \nu\, a_j.\nu\, a'_j.\nu\, a_s.\nu\, a'_s\ .\ (\ (!\mathsf{Join}) \mid (!\mathsf{Sign}) \mid (\nu\, \mathtt{cnt}.\nu\, \mathtt{DAASeed}.\nu\, sk_M.\overline{c}\langle \mathsf{pk}(sk_M)\rangle.\\
\quad !c(w_{\mathsf{params}}).\overline{a_j}\langle(\!|w_{\mathsf{params}}, \mathtt{DAASeed}, \mathtt{cnt}, sk_M|\!)\rangle.a'_j(x).\\
\quad \text{let } x_{\mathsf{cre}} = \pi_1(x) \text{ in let } x_{\mathtt{tsk}} = \pi_2(x) \text{ in } (\\
\qquad !c(y).\text{let } y_{\mathsf{bsn}} = \pi_1(y) \text{ in let } y_{\mathsf{msg}} = \pi_2(y) \text{ in}\\
\qquad \overline{a_s}\langle(\!|w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathtt{tsk}}|\!)\rangle.a'_s(z).\overline{c}\langle z\rangle\\
\quad)\\
))
\end{aligned}
$$

The bound name `cnt` is a counter value selected by the host. The bound name `DAASeed` represents the TPM's internal secret and sk_M represents the TPM's endorsement key (these values are defined during manufacture [18]). The public part of the endorsement key is published by the Signer process. The remainder of the Signer process models a signer's ability to execute arbitrarily many instances of the join and sign algorithms. The Signer process must first input system parameters w_{params}, provided by the issuer. 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 $\overline{a_j}\langle(\!|w_{\mathsf{params}}, \mathtt{DAASeed}, \mathtt{cnt}, w_{\mathsf{ek}}|\!)\rangle$, where $(\!|w_{\mathsf{params}}, \mathtt{DAASeed}, \mathtt{cnt}, w_{\mathsf{ek}}|\!)$ 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 $(\!|x_{\mathsf{cre}}, x_{\mathtt{tsk}}|\!)$ representing the attestation identity credential and TPM's secret. The interaction between the Sign and Signer processes is similar. 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 $\overline{a_s}\langle(\!|w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathtt{tsk}}|\!)\rangle$, where $(\!|w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathtt{tsk}}|\!)$ represents the algorithm's parameters. 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.

4 Security definition: User-controlled anonymity

Informally, the notion of user-controlled anonymity 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, from another one in which $\mathcal{B}$ signs a message. Based upon the game-based definition by Brickell, Chen & Li [2, 3] we present the following security definition.

Initial: The adversary constructs the key pair sk_I and $\mathsf{pk}(sk_I)$, and publishes the public part $\mathsf{pk}(sk_I)$. The adversary also publishes any additional parameters.

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

– Join. The signer executes the join algorithm to create `cre` and `tsk`. The adversary, as the issuer, learns `cre` but typically not `tsk`.
– Sign. The adversary submits a basename `bsn` and a message m. The signer runs the sign algorithm and returns the signature to the adversary.

At the end of Phase 1, both signers are required to have run the join algorithm at least once.

Phase 2 (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 3: The adversary continues to probe the signers with join and sign requests, but is explicitly forbidden to use the basename used in Phase 2 if $\texttt{bsn} \neq \perp$.

Result: The protocol satisfies user-controlled anonymity if the adversary cannot distinguish between the two signatures output during the challenge.

Our definition intuitively captures privacy because the adversary cannot distinguish between the two signatures output during the challenge. We can witness that this is a sufficent condition for privacy as follows: suppose a protocol satisfies user-controlled anonymity but the identity of a signer can be revealed from a signature, it follows immediately that the adversary can test which signature output during the challenge belongs to $\mathcal{A}$, allowing the signatures to be distinguished and therefore deriving a contradiction. Moreover, our definition also captures unlinkability. This can be witnessed as follows. Suppose a protocol satisfies user-controlled anonymity but signatures can be linked without the signer's consent. It follows from our security definition that no adversary can distinguish between two signatures output during the challenge. Let us now consider an adversary that requests a signature σ_A from $\mathcal{A}$ during Phase 1 using basename $\texttt{bsn} = \perp$ (that is, the signer does not consent to linkability) and an arbitrary message m. The adversary submits an arbitrary message m' and basename $\texttt{bsn}' = \perp$ during the challenge, and the signers return signatures σ_1 and σ_2. Since signatures can be linked without the signer's consent, the adversary is able to test if σ_A and σ_1 are linked, or whether σ_A and σ_2 are linked; exactly one test will succeed allowing the adversary to distinguish between signatures σ_1 and σ_2. We have derived a contradiction and therefore a protocol satisfying our definition of user-controlled anonymity provides unlinkability.

Formally, our definition of user-controlled anonymity can be modelled as an observational equivalence property (Definition 5) using the *DAA game biprocess* DAA-G presented in Figure 4. The Challenge process, which forms part of the process DAA-G, is designed to capture the behaviour of the signers in Phase 2. This is achieved by outputting the attestation identity credential x_{cre} and TPM's secret $x_{\mathtt{tsk}}$, produced by the signers in Phase 1, on the private channels

Fig. 4 Biprocess modelling user-controlled anonymity in DAA

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

$$\nu\, b_A.\nu\, b_B.c(w_{\mathsf{params}})\ .\ (\mathsf{Challenge} \mid \mathsf{Signer}^{+}\{^{b_A}/_{w_b}\} \mid \mathsf{Signer}^{+}\{^{b_B}/_{w_b}\})$$

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

$$\begin{aligned}
\mathsf{Signer}^{+} \quad &= \nu\, a_j.\nu\, a'_j.\nu\, a_s.\nu\, a'_s.(\ (!\mathsf{Join}) \mid (!\mathsf{Sign}) \mid (\nu\, \mathtt{cnt}.\nu\, \mathtt{DAASeed}.\nu\, sk_M.\overline{c}\langle \mathsf{pk}(sk_M)\rangle \\
&\quad\ !\overline{a_j}\langle (\!| w_{\mathsf{params}}, \mathtt{DAASeed}, \mathtt{cnt}, sk_M |\!)\rangle.a'_j(x). \\
&\quad\ \text{let } x_{\mathsf{cre}} = \pi_1(x) \text{ in let } x_{\mathtt{tsk}} = \pi_2(x) \text{ in } (\\
&\quad\quad !c(y).\text{let } y_{\mathsf{bsn}} = \pi_1(y) \text{ in let } y_{\mathsf{msg}} = \pi_2(y) \text{ in} \\
&\quad\quad \text{if } y_{\mathsf{bsn}} = \perp \text{ then} \\
&\quad\quad\quad \overline{a_s}\langle (\!| w_{\mathsf{params}}, y_{\mathsf{bsn}}, y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathtt{tsk}} |\!)\rangle.a'_s(z).\overline{c}\langle z\rangle \\
&\quad\quad \text{ else} \\
&\quad\quad\quad \overline{a_s}\langle (\!| w_{\mathsf{params}}, (\!| chl^{+}, y_{\mathsf{bsn}} |\!), y_{\mathsf{msg}}, x_{\mathsf{cre}}, x_{\mathtt{tsk}} |\!)\rangle.a'_s(z).\overline{c}\langle z\rangle \\
&\quad\ ) \mid (\\
&\quad\quad \overline{w_b}\langle (\!| x_{\mathsf{cre}}, x_{\mathtt{tsk}} |\!)\rangle \\
&\quad\ ) \\
&\ \ )) \\
\mathsf{Challenge} &= \nu\, a_s.\nu\, a'_s\ .\ (\ (\mathsf{Sign}) \mid (\\
&\quad\ b_A(x).\text{let } x_{\mathsf{cre}} = \pi_1(x) \text{ in let } x_{\mathtt{tsk}} = \pi_2(x) \text{ in} \\
&\quad\ b_B(y).\text{let } y_{\mathsf{cre}} = \pi_1(y) \text{ in let } y_{\mathsf{tsk}} = \pi_2(y) \text{ in} \\
&\quad\ c(z).\text{let } z_{\mathsf{bsn}} = \pi_1(z) \text{ in let } z_{\mathsf{msg}} = \pi_2(z) \text{ in} \\
&\quad\ \text{if } z_{\mathsf{bsn}} = \perp \text{ then} \\
&\quad\quad \overline{a_s}\langle (\!| w_{\mathsf{params}}, z_{\mathsf{bsn}}, z_{\mathsf{msg}}, \mathrm{diff}[x_{\mathsf{cre}}, y_{\mathsf{cre}}], \mathrm{diff}[x_{\mathtt{tsk}}, y_{\mathsf{tsk}}] |\!)\rangle.a'_s(z).\overline{c}\langle z\rangle \\
&\quad\ \text{ else} \\
&\quad\quad \overline{a_s}\langle (\!| w_{\mathsf{params}}, (\!| chl^{-}, z_{\mathsf{bsn}} |\!), z_{\mathsf{msg}}, \mathrm{diff}[x_{\mathsf{cre}}, y_{\mathsf{cre}}], \mathrm{diff}[x_{\mathtt{tsk}}, y_{\mathsf{tsk}}] |\!)\rangle.a'_s(z).\overline{c}\langle z\rangle \\
&\ \ ))
\end{aligned}$$

for some constants chl^{+}, chl^{-}.

b_A and b_B in Signer^{+}, and inputting these values in the Challenge process. The Challenge process proceeds by producing a signature in the standard manner, but uses $\mathrm{diff}[x_{\mathsf{cre}}, y_{\mathsf{cre}}]$ and $\mathrm{diff}[x_{\mathtt{tsk}}, y_{\mathsf{tsk}}]$ to ensure that the signature is produced by $\mathcal{A}$ in fst(DAA-G) and $\mathcal{B}$ in snd(DAA-G). Finally, the necessity for a distinct basename in Phase 2 (when $\mathtt{bsn} \neq \perp$) is enforced by prefixing the basename used by Challenge with chl^{-} and, similarly, prefixing the basenames used by Signer^{+} with chl^{+}. Our definition of user-controlled anonymity (Definition 5) follows naturally.

Definition 5 (User-controlled anonymity). *Given a pair of processes* $\langle \mathsf{Join}, \mathsf{Sign} \rangle$, user-controlled anonymity *is satisfied if the DAA game biprocess* DAA-G *satisfies observational equivalence.*

Comparison with the game-based definition. In the game-based definition by Brickell, Chen & Li [2, 3] either $\mathcal{A}$ or $\mathcal{B}$ signs the message during Phase 2 and user-controlled anonymity is satisfied if the adversary has a negligible advantage over guessing the correct signer. By comparison, in our definition, both $\mathcal{A}$ and $\mathcal{B}$ sign the message during Phase 2 and user-controlled anonymity is satisfied if

these signatures are indistinguishable. Intuitively, any adversary strategy that exists where either $\mathcal{A}$ or $\mathcal{B}$ signs can be exploited in the setting where both $\mathcal{A}$ and $\mathcal{B}$ sign, and we therefore believe that our definition is at least as strong. This result can be stated as follows. Let *Game* be the game-based definition presented by Brickell, Chen & Li and let *Game*$'$ be the variant presented here. If there exists an adversary $\mathcal{M}$ that is able to win *Game*, then there exists an adversary $\mathcal{M}'$ that can win *Game*$'$. It follows immediately that any attack against a protocol can be discovered in *Game*$'$, that is, *Game*$'$ is as strong as *Game*. We now sketch our proof. Suppose σ_1 and σ_2 are signatures produced during the challenge of *Game*$'$. The adversary $\mathcal{M}'$ can submit σ_1 to $\mathcal{M}$ and the adversary $\mathcal{M}$ can reveal the signer's identity id_1; similarly, the adversary $\mathcal{M}'$ can exploit $\mathcal{M}$ to reveal the signer's identity id_2 from σ_2. The adversary $\mathcal{M}'$ tests $id_1 \stackrel{?}{=} \mathcal{A}$ and $id_2 \stackrel{?}{=} \mathcal{A}$, exactly one of these tests will succeed allowing $\mathcal{M}'$ to distinguish the two signatures output during the challenge. For a complete proof it would be necessary to either: 1) define a symbolic version of the game by Brickell, Chen & Li, or 2) formalise our definition in the computational model. The former appears problematic and hence explains why we base our definition on Brickell, Chen & Li rather than stricty follow it. The complete proof remains as future work. Computational soundness results would also be an interesting future direction.

5 Case study: ECC-based DAA

The ECC-based DAA protocol was introduced by Brickell, Chen & Li [6, 3] to overcome efficiency issues with the RSA-based DAA protocol.

5.1 Primitives and building blocks

We first recall the details of Camenisch-Lysyanskaya (CL) signatures [19], which form the foundations of the ECC-based DAA protocol.

Randomised signature scheme. A CL signature is denoted $\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$, where x_{sk} is the secret key, x_{nonce} is a nonce, and x_{msg} is a message. The random component $\mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{nonce}})$ can be derived from a signature $\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$. Verification is standard given a signature, message, and public key, that is, $\mathsf{checkclsign}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{msg}}, \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \rightarrow \mathsf{accept}$. The scheme allows randomisation of signatures, and we denote the randomisation of the signature $\sigma = \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$ as $\mathsf{clrand}(y_{\mathsf{nonce}}, \sigma)$ for some random nonce y_{nonce} such that $\mathsf{clrand}(y_{\mathsf{nonce}}, \sigma) \rightarrow \mathsf{clsign}(x_{\mathsf{sk}}, \mathsf{mul}(x_{\mathsf{nonce}}, y_{\mathsf{nonce}}), x_{\mathsf{msg}})$.

Signature scheme for committed values. Given the public part of a signing key $\mathsf{pk}(x_{\mathsf{sk}})$ and a message x_{csk}, the corresponding commitment is $U = \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{csk}})$ and the associated signature is $\mathsf{clsigncommit}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, U) \rightarrow \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{csk}})$, where x_{nonce} is a nonce. To maintain security of the signature scheme, knowledge of x_{csk} must be demonstrated.

Tractability of DDH problem. The DDH problem is tractable for cyclic groups with symmetric pairing, that is, given a pairing $e : G_1 \times G_1 \to G_2$ and integers $g, g^a, g^b, g^c \in G_1$, the distribution $\{(g, g^a, g^b, g^{ab})\}$ is distinguishable from $\{(g, g^a, g^b, g^c)\}$, where G_1 and G_2 are groups of prime order. This is due to the *bilinear* property of the pairing function: for all integers $g, h \in G_1$ and integers $a, b \in \mathbb{Z}$ we have $e(g^a, h^b) = e(g, h)^{ab}$. It follows immediately that $\{(g, g^a, g^b, g^{ab})\}$ is distinguishable from $\{(g, g^a, g^b, g^c)\}$ because $e(g, g^c) = e(g^a, g^b)$ iff $c = ab$. Moreover, we have $\{(g, h, g^a, h^a)\}$ is distinguishable from $\{(g, h, g^a, h^b)\}$ because $e(g^a, h) = e(g, h^b)$ iff $a = b$. Finally, $e(g, a^{rx+rmxy}) = e(g, a)^{rx+rmxy} = e(g^x, a^r) \cdot e(g^m, a^{rxy})$ and hence, by reference to the cryptographic description of CL-signatures, the commitment $M = g^m$ can be linked to the randomised signature $(a', b', c') = (a^r, a^{ry}, a^{rx+rmxy})$ when the secret key x is known. We explicitly include the following properties in our symbolic model.

- $\mathsf{clcommit}(x_{\mathsf{base}}, x_{\mathsf{msg}})$ is related to $\mathsf{clcommit}(x'_{\mathsf{base}}, x_{\mathsf{msg}})$, where x_{base} and x'_{base} are known.
- $\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$ is related to $\mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{msg}})$, where x_{sk} is known.

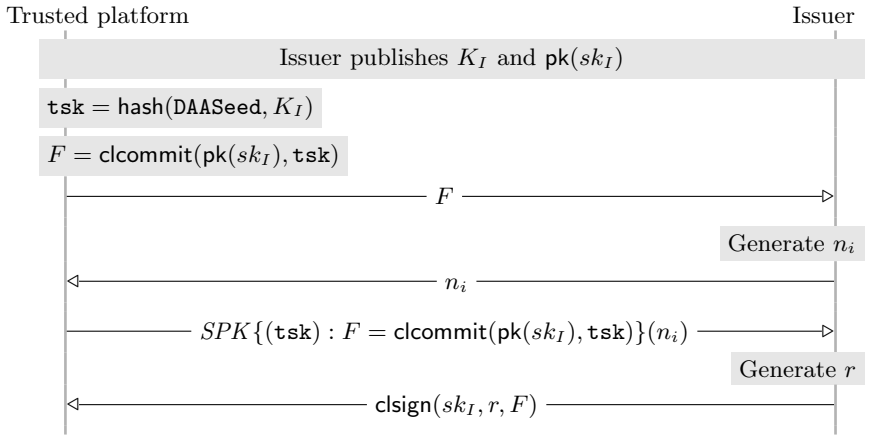
We shall also use a commitment function commit in which the DDH problem is intractable, that is, $\mathsf{commit}(x_{\mathsf{base}}, x_{\mathsf{msg}})$ and $\mathsf{commit}(x'_{\mathsf{base}}, x_{\mathsf{msg}})$ are indistinguishable when $x_{\mathsf{base}} \neq x'_{\mathsf{base}}$.

Proving knowledge of a signature. The signature scheme for committed values can be used to build an anonymous credential system. Given a signature $\sigma = \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{csk}})$, random nonce y_{nonce} and blinding factor y_{blind}, the anonymous credential $\hat{\sigma} = \mathsf{clblind}(y_{\mathsf{blind}}, \mathsf{clrand}(y_{\mathsf{nonce}}, \sigma))$. (We remark that the random component $\mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{nonce}})$ can be recovered from both the signature σ and the blind signature $\mathsf{clblind}(y_{\mathsf{blind}}, \sigma)$, hence the two signatures can be linked; it follows that blinding is insufficient to derive an anonymous credential.) A zero-knowledge proof of knowledge can then be used to demonstrate that the anonymous credential $\hat{\sigma}$ is indeed a blinded signature on message x_{csk} using blinding factor x_{blind}. We will adopt the notation introduced by Camenisch & Stadler [20] to describe signatures of knowledge. For instance, $SPK\{(\alpha) : F = \mathsf{clcommit}(\mathsf{pk}(sk_I), \alpha)\}(m)$ denotes a *"Signature Proof of Knowledge of α such that $F = \mathsf{commit}(\mathsf{pk}(sk_I), \alpha)$ holds, where m is the message being signed."* In the example, the Greek letters in parentheses are used for values about which knowledge is being proved and these values are kept secret by the prover.

5.2 Protocol description

For the purpose of studying user-controlled anonymity, it is sufficient to consider the join and sign algorithms. The join algorithm (Figure 5) is defined below, given the algorithm's input: system parameter K_I and $\mathsf{pk}(sk_I)$ (that is, the issuer's long- and short-term public keys), the TPM's secret `DAASeed`, a counter value `cnt`, and the TPM's endorsement key pair sk_M and $\mathsf{pk}(sk_M)$. We remark that

Fig. 5 ECC-based DAA join algorithm



the issuer's basename $\mathtt{bsn}_I$ is not provided as input (unlike the RSA-based DAA protocol) in the standard mode of operation.

1. The TPM computes the secret $\mathtt{tsk} = \mathsf{hash}(\mathtt{DAASeed}, K_I)$, derives the commitment $F = \mathsf{clcommit}(\mathsf{pk}(sk_I), \mathtt{tsk})$ and sends F to the issuer.
2. The issuer generates a nonce n_i and sends it to the trusted platform.
3. The trusted platform generates a signature proof of knowledge $SPK\{(\mathtt{tsk}) : F = \mathsf{clcommit}(\mathsf{pk}(sk_I), \mathtt{tsk})\}(n_i)$ that the message F is correctly formed. The host sends the proof to the issuer.
4. The issuer verifies the proof and generates a credential $\mathtt{cre} = \mathsf{clsign}(sk_I, r, F)$. The signature $\mathtt{cre}$ is sent to the trusted platform and the platform verifies the signature.

At the end of the algorithm, the credential $\mathtt{cre}$ can be public (in particular, it is known by the host and the issuer), but only the TPM knows the corresponding secret $\mathtt{tsk}$. The credential $\mathtt{cre}$ and secret $\mathtt{tsk}$ can be provided as input to the sign algorithm, along with a basename $\mathtt{bsn}$ and message m. The sign algorithm proceeds as follows.

5. If $\mathtt{bsn} = \perp$, then the host generates a nonce ζ; otherwise, the host computes $\zeta = \mathsf{hash}(\mathtt{1}, \mathtt{bsn})$. The host provides the TPM with ζ. The TPM computes the commitment $N_V = \mathsf{commit}(\zeta, \mathtt{tsk})$. The host generates a random nonce r' and blinding factor $\hat{r}$, which are used to compute the anonymous credential $\widehat{\mathtt{cre}} = \mathsf{clblind}(\hat{r}, \mathsf{clrand}(r', \mathtt{cre}))$. The trusted platform then produces a signature proof of knowledge that $\widehat{\mathtt{cre}}$ is a valid blinded credential on message $\mathtt{tsk}$ using blinding factor $\hat{r}$, 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.

We remark that our attack against the RSA-based DAA scheme [21] (see also [22, §4.4.5]) cannot be launched in the ECC setting, because it is not possible to select a public key $\mathsf{pk}(sk_I)$ and basename $\mathtt{bsn}$ such that the commitments $F = \mathsf{clcommit}(\mathsf{pk}(sk_I), \mathtt{tsk})$ and $N_V = \mathsf{commit}(\mathsf{hash}(1, \mathtt{bsn}), \mathtt{tsk})$ can be linked; this is due to the constraints placed upon the public key, the pre-image resistance of hash functions, and properties of the respective commitment functions.

5.3 Signature and equational theory

We construct a signature Σ to capture the cryptographic primitives used by the scheme and define rewrite rules to capture the relationship between these primitives. Let $\Sigma = \{\mathsf{accept}, \bot, 1, \mathsf{F}_{\mathsf{join}}, \mathsf{F}_{\mathsf{sign}}, \mathsf{clgetnonce}, \mathsf{hash}, \mathsf{pk}, \mathsf{clblind}, \mathsf{clcommit}, \mathsf{commit}, \mathsf{clrand}, \mathsf{mul}, \mathsf{checkclsign}, \mathsf{checkspk}, \mathsf{clsign}, \mathsf{clsigncommit}, \mathsf{linksigcomm}, \mathsf{spk}, \mathsf{clbsign}, \mathsf{linkcomm}\}$. Functions accept, $\bot$, 1, $\mathsf{F}_{\mathsf{join}}$, $\mathsf{F}_{\mathsf{sign}}$ are constants; $\mathsf{clgetnonce}$, hash, pk are unary functions; $\mathsf{clblind}$, $\mathsf{clcommit}$, commit, clrand, mul are binary functions; $\mathsf{checkclsign}$, $\mathsf{checkspk}$, clsign, $\mathsf{clsigncommit}$, $\mathsf{linksigcomm}$, spk are ternary functions; and $\mathsf{clbsign}$, $\mathsf{linkcomm}$ are functions of arity four. We occasionally write $\mathsf{hash}(x_{\mathsf{plain},1}, \ldots, x_{\mathsf{plain,n}})$ to denote $\mathsf{hash}(\langle\!| x_{\mathsf{plain},1}, \ldots, x_{\mathsf{plain,n}} |\!\rangle)$. The rewrite rules associated with the destructors in Σ are defined below.

$$\mathsf{clsigncommit}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{msg}})) \to \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$$

$$\mathsf{clrand}(y_{\mathsf{nonce}}, \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \to \mathsf{clsign}(x_{\mathsf{sk}}, \mathsf{mul}(x_{\mathsf{nonce}}, y_{\mathsf{nonce}}), x_{\mathsf{msg}})$$

$$\mathsf{clrand}(y_{\mathsf{nonce}}, \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \to \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, \mathsf{mul}(x_{\mathsf{nonce}}, y_{\mathsf{nonce}}), x_{\mathsf{msg}})$$

$$\mathsf{clblind}(y_{\mathsf{blind}}, \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \to \mathsf{clbsign}(y_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$$

$$\mathsf{clblind}(y_{\mathsf{blind}}, \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \to \mathsf{clbsign}(\mathsf{mul}(x_{\mathsf{blind}}, y_{\mathsf{blind}}), x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$$

$$\mathsf{clgetnonce}(\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \to \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{nonce}})$$

$$\mathsf{clgetnonce}(\mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \to \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{nonce}})$$

$$\mathsf{checkclsign}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{msg}}, \mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \to \mathsf{accept}$$

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 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.

$$\mathsf{checkspk}(\mathsf{F}_{\mathsf{join}}, V, \mathsf{spk}(\mathsf{F}_{\mathsf{join}}, x_{\mathtt{tsk}}, V)) \to \mathsf{accept} \quad \text{where } V = \langle\!| y_{\mathsf{pk}}, \mathsf{clcommit}(y_{\mathsf{pk}}, x_{\mathtt{tsk}}), y_{\mathsf{msg}} |\!\rangle$$

$$\mathsf{checkspk}(\mathsf{F}_{\mathsf{sign}}, V, \mathsf{spk}(\mathsf{F}_{\mathsf{sign}}, \langle\!| x_{\mathtt{tsk}}, x_{\mathsf{blind}} |\!\rangle, V)) \to \mathsf{accept} \quad \text{where } V = \langle\!| y_\zeta, \mathsf{pk}(z_{\mathsf{sk}}), \mathsf{commit}(y_\zeta, x_{\mathtt{tsk}}), \mathsf{clbsign}(x_{\mathsf{blind}}, z_{\mathsf{sk}}, z_{\mathsf{nonce}}, x_{\mathtt{tsk}}), y_{\mathsf{msg}} |\!\rangle$$

Fig. 6 Applied pi process specification for the ECC-based DAA protocol

$$
\begin{aligned}
\mathsf{Join}_{\mathsf{ECC}} \;\hat{=}\; & a_j(w_{\mathsf{params}}, w_{\mathsf{DAASeed}}, w_{\mathsf{cnt}}, w_{\mathsf{ek}}) \,. \\
& \text{let } w_{\mathsf{K}} = \pi_1(w_{\mathsf{params}}) \text{ in let } w_{\mathsf{pk}} = \pi_2(w_{\mathsf{params}}) \text{ in} \\
& \text{let } \mathtt{tsk} = \mathsf{hash}(w_{\mathsf{DAASeed}}, w_{\mathsf{K}}) \text{ in} \\
& \text{let } F = \mathsf{clcommit}(w_{\mathsf{pk}}, \mathtt{tsk}) \text{ in} \\
& \overline{c}\langle F \rangle \,. \\
& c(x) \,. \\
& \overline{c}\langle \mathsf{spk}(\mathsf{F}_{\mathsf{join}}, \mathtt{tsk}, (\!| w_{\mathsf{K}}, F, x |\!)) \rangle \,. \\
& c(y_{\mathsf{cre}}) \,. \\
& \text{if } \mathsf{checkclsign}(w_{\mathsf{pk}}, \mathtt{tsk}, y_{\mathsf{cre}}) = \mathsf{accept} \text{ then} \\
& \overline{a'_j}\langle (\!| y_{\mathsf{cre}}, \mathtt{tsk} |\!) \rangle
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{Sign}_{\mathsf{ECC}} \;\hat{=}\; & a_s(w_{\mathsf{params}}, w_{\mathsf{bsn}}, w_{\mathsf{msg}}, w_{\mathsf{cre}}, w_{\mathsf{tsk}}) \text{ . let } w_{\mathsf{pk}} = \pi_2(w_{\mathsf{params}}) \text{ in} \\
& \nu\, r' \,.\, \nu\, \hat{r} \,.\, \nu\, n_t \,.\, c(x) \,. \\
& \text{if } w_{\mathsf{bsn}} = \perp \text{ then} \\
& \quad \nu\, \zeta \,. \\
& \quad \text{let } \widehat{\mathtt{cre}} = \mathsf{clblind}(\hat{r}, \mathsf{clrand}(r', w_{\mathsf{cre}})) \text{ in} \\
& \quad \text{let } N_V = \mathsf{commit}(\zeta, w_{\mathsf{tsk}}) \text{ in} \\
& \quad \text{let } spk = \mathsf{spk}(\mathsf{F}_{\mathsf{sign}}, (\!| w_{\mathsf{tsk}}, \hat{r} |\!), (\!| \zeta, w_{\mathsf{pk}}, N_V, \widehat{\mathtt{cre}}, (\!| n_t, x, w_{\mathsf{msg}} |\!) |\!)) \text{ in} \\
& \quad \overline{a'_s}\langle (\!| \zeta, N_V, \widehat{\mathtt{cre}}, n_t, spk |\!) \rangle \\
& \text{else} \\
& \quad \text{let } \zeta = \mathsf{hash}(\mathtt{1}, w_{\mathsf{bsn}}) \text{ in} \\
& \quad \text{let } \widehat{\mathtt{cre}} = \mathsf{clblind}(\hat{r}, \mathsf{clrand}(r', w_{\mathsf{cre}})) \text{ in} \\
& \quad \text{let } N_V = \mathsf{commit}(\zeta, w_{\mathsf{tsk}}) \text{ in} \\
& \quad \text{let } spk = \mathsf{spk}(\mathsf{F}_{\mathsf{sign}}, (\!| w_{\mathsf{tsk}}, \hat{r} |\!), (\!| \zeta, w_{\mathsf{pk}}, N_V, \widehat{\mathtt{cre}}, (\!| n_t, x, w_{\mathsf{msg}} |\!) |\!)) \text{ in} \\
& \quad \overline{a'_s}\langle (\!| \zeta, N_V, \widehat{\mathtt{cre}}, n_t, spk |\!) \rangle
\end{aligned}
$$

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. Finally, the tractability of the DDH problem for cyclic groups with symmetric pairing requires the following equations.

$$\mathsf{linkcomm}(x_{\mathsf{base}}, x'_{\mathsf{base}}, \mathsf{clcommit}(x_{\mathsf{base}}, z_{\mathsf{msg}}), \mathsf{clcommit}(x'_{\mathsf{base}}, z_{\mathsf{msg}})) \to \mathsf{accept}$$

$$\mathsf{linksigcomm}(x_{\mathsf{sk}}, \mathsf{clcommit}(\mathsf{pk}(x_{\mathsf{sk}}), z_{\mathsf{msg}}), \mathsf{clsign}(x_{\mathsf{sk}}, z_{\mathsf{nonce}}, z_{\mathsf{msg}})) \to \mathsf{accept}$$

5.4 Model in applied pi

The ECC-based join and sign algorithms are modelled by the pair of processes $\langle \mathsf{Join}_{\mathsf{ECC}}, \mathsf{Sign}_{\mathsf{ECC}} \rangle$ presented in Figure 6, in which we abbreviate $c(x)$.let $x_1 = \pi_1(x)$ in $\ldots$ let $x_n = \pi_n(x)$ in P as $c(x_1, \ldots, x_n).P$.

The join process $\mathsf{Join}_{\mathsf{ECC}}$ is instantiated by inputting the join algorithm's parameters: the ECC-based DAA system parameters w_{params}, the TPM's internal secret w_{DAASeed}, the counter 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 pair containing the issuer's long-term public key K_I and short-term public key $\mathsf{pk}(sk_I)$.

The process constructs the terms $\mathtt{tsk}$ and F in accordance with the protocol's description (Section 5.2), and outputs F to the issuer. A nonce x is then input and a signature proof of knowledge is produced. Finally, the process inputs a signature y_{cre} on the secret $\mathtt{tsk}$ and concludes by outputting the attestation identity credential y_{cre} and TPM's secret $\mathtt{tsk}$ on the private channel a'_j, that is, the $\mathsf{Join}_{\mathsf{ECC}}$ process returns the values y_{cre} and $\mathtt{tsk}$ to the Signer^+ process.

The sign process $\mathsf{Sign}_{\mathsf{ECC}}$ is instantiated by inputting the sign algorithm's parameters: the ECC-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}. The process inputs a nonce x from the verifier. The if-then-else branch models the signer's ability to produce 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 $\mathsf{Sign}_{\mathsf{ECC}}$ process returns the signature to the Signer^+ process.

5.5 Analysis: User-controlled anonymity

The DAA game biprocess derived from $\langle \mathsf{Join}_{\mathsf{ECC}}, \mathsf{Sign}_{\mathsf{ECC}} \rangle$ can be automatically analysed using ProVerif; unfortunately, however, ProVerif cannot prove equivalence in the general case. Accordingly, rather than prove that the specification $\langle \mathsf{Join}_{\mathsf{ECC}}, \mathsf{Sign}_{\mathsf{ECC}} \rangle$ satisfies user-controlled anonymity, we attempt to prove security in a setting where the adversary is forbidden from re-blinding a blind signature, this is an under-approximation and introduces the assumption that the adversary obtains no advantage from re-blinding a blind signature. This under-approximation can be acheived by removing the rewrite rule $\mathsf{clblind}(y_{\mathsf{blind}}, \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \rightarrow \mathsf{clbsign}(\mathsf{mul}(x_{\mathsf{blind}}, y_{\mathsf{blind}}), x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})$.

In addition, we make two over-approximations of attacker knowledge. First, we replace the rewrite rules associated with $\mathsf{clgetnonce}$ with the following rules.

$$\mathsf{clgetnonce}(\mathsf{clsign}(x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \rightarrow x_{\mathsf{nonce}}$$

$$\mathsf{clgetnonce}(\mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \rightarrow x_{\mathsf{nonce}}$$

Secondly, we modify the rewrite rule $\mathsf{clrand}(y_{\mathsf{nonce}}, \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \rightarrow \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, \mathsf{mul}(x_{\mathsf{nonce}}, y_{\mathsf{nonce}}), x_{\mathsf{msg}})$ to use pairing rather than the function mul, that is, we use the following rule.

$$\mathsf{clrand}(y_{\mathsf{nonce}}, \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, x_{\mathsf{nonce}}, x_{\mathsf{msg}})) \rightarrow \mathsf{clbsign}(x_{\mathsf{blind}}, x_{\mathsf{sk}}, (\!|x_{\mathsf{nonce}}, y_{\mathsf{nonce}}|\!), x_{\mathsf{msg}})$$

Although, in general, the revised rewrite rules for $\mathsf{clgetnonce}$ are not equivalent to the original rules, they intuitively overapproximate attacker knowledge in our setting. The specification $\langle \mathsf{Join}_{\mathsf{ECC}}, \mathsf{Sign}_{\mathsf{ECC}} \rangle$ does not restrict any signing keys and thus public keys can be freely constructed by application

of the function symbol pk. If the environment has knowledge of a signature $\mathsf{clsign}(K, R, M)$, then the environment can recover the nonce R and construct the commitment $\mathsf{clcommit}(\mathsf{pk}(K), R)$, that is, we have an over-approximation (the reasoning is similar for blind signatures). Moreover, given the modified rewrite rules for $\mathsf{clgetnonce}$, the revised rewrite rule for clrand is also an over-approximation; in particular, witness that the environment can recover $(\!|R, R'|\!)$ from an arbitrary blind signature $\mathsf{clbsign}(R'', K, (\!|R, R'|\!), M)$ and trivially construct $\mathsf{clcommit}(\mathsf{pk}(K), \mathsf{mul}(R, R'))$.

Over-approximations are the norm in ProVerif (it introduces them anyway); if ProVerif concludes that a protocol is correct even with over-approximations of attacker knowledge, then one may conclude it is correct. Under-approximations are more problematic, however: they introduce an assumption. Thus, our proof of correctness is valid only on the assumption introduced by our under-approximation, namely, that the attacker obtains no advantage from re-blinding a blind signature.

In our revised model, ProVerif is able to automatically verify *user-controlled anonymity*. The ProVerif scripts associated with our analysis are available online: `http://www.bensmyth.com/publications/11-anonymity-in-DAA/`.

6 Further work and conclusion

Direct Anonymous Attestation is a relatively new concept and its properties merit further study. In particular, user-controlled traceability, non-frameability, and correctness 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, correctness, non-frameability, user-controlled anonymity and user-controlled traceability) 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 user-controlled anonymity 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 traceability 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.

Unfortunately, we are unable to derive any conclusions about user-controlled anonymity in the ECC-based DAA scheme in the general setting; however, using an approximation of our rewrite rules, we show the scheme is secure under the assumption that the adversary obtains no advantage from re-blinding a blind signature. In addition, two over-approximations were required. Future work could develop theory for the sound abstraction between sets of rewrite rules, thereby providing more confidence in our over-approximations; indeed, Backes, Maffei & Unruh [12] make some progress in this direction.

Conclusion. This paper presents a definition of user-controlled anonymity for Direct Anonymous Attestation protocols. The definition is expressed as an equivalence property suitable for automated reasoning. The practicality of the approach is demonstrated by examining the ECC-based Direct Anonymous Attestation protocol. The ECC-based scheme is particularly significant because support is mandated by the TPM.next specification which is due to replace TPM version 1.2 and, moreover, the protocol has been included in the ISO/IEC anonymous digital signature standard. Our analysis demonstrates the absence of attacks in a model where we restrict the adversary from re-blinding a blind signature.

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 the anonymous reviewers who provided constructive criticism. This research was conducted as part of the EPSRC projects *UbiVal* (EP/D076625/2) and *Verifying Interoperability Requirements in Pervasive Systems* (EP/F033540/1), and as part of the ProSecure project which is funded by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865.

References

1. Brickell, E., Camenisch, J., Chen, L.: Direct Anonymous Attestation. In: CCS'04: 11th ACM Conference on Computer and Communications Security, ACM Press (2004) 132–145
2. Brickell, E., Chen, L., Li, J.: Simplified Security Notions of Direct Anonymous Attestation and a Concrete Scheme from Pairings. Cryptology ePrint Archive, Report 2008/104 (2008)
3. Brickell, E., Chen, L., Li, J.: Simplified security notions of Direct Anonymous Attestation and a concrete scheme from pairings. International Journal of Information Security **8**(5) (2009) 315–330
4. Chen, L.: A DAA Scheme Requiring Less TPM Resources. Cryptology ePrint Archive, Report 2010/008 (2010)
5. Chen, L.: A DAA Scheme Requiring Less TPM Resources. In: INSCRYPT'09: 5th International Conference on Information Security and Cryptology. Volume 6151 of LNCS., Springer (2011) 350–365
6. Brickell, E., Chen, L., Li, J.: 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. (2008) 166–178
7. Trusted Computing Group: Draft TPM 2.0 Specification, Revision oo79. (2011)
8. 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)
9. Chen, L., Morrissey, P., Smart, N.P.: DAA: Fixing the pairing based protocols. Unpublished draft (2010)

10. Chen, L., Morrissey, P., Smart, N.P.: DAA: Fixing the pairing based protocols. Cryptology ePrint Archive, Report 2009/198 (2009)
11. Chen, L., Morrissey, P., Smart, N.P.: On Proofs of Security for DAA Schemes. In: ProvSec'08: 2nd International Conference on Provable Security. Volume 5324 of LNCS., Springer (2008) 156–175
12. Backes, M., Maffei, M., Unruh, D.: 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, IEEE Computer Society (2008) 202–215
13. Blanchet, B.: Automatic Proof of Strong Secrecy for Security Protocols. In: S&P'04: 25th IEEE Symposium on Security and Privacy, IEEE Computer Society (2004) 86–100
14. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. Journal of Logic and Algebraic Programming **75**(1) (February–March 2008) 3–51
15. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: POPL'01: 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press (2001) 104–115
16. Ryan, M.D., Smyth, B.: Applied pi calculus. In Cortier, V., Kremer, S., eds.: Formal Models and Techniques for Analyzing Security Protocols. IOS Press (2011)
17. Blanchet, B., Smyth, B.: ProVerif: Automatic Cryptographic Protocol Verifier User Manual & Tutorial. `http://www.proverif.ens.fr/` (2011)
18. Trusted Computing Group: TPM Specification version 1.2. (2007)
19. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: CRYPTO'04: 24th International Cryptology Conference. Volume 3152 of LNCS., Springer (2004) 56–72
20. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups. In: CRYPTO'97: 17th International Cryptology Conference. Volume 1294 of LNCS., Springer (1997) 410–424
21. Smyth, B., Ryan, M.D., Chen, L.: 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., Springer (2007) 218–231
22. Smyth, B.: Formal verification of cryptographic protocols with automated reasoning. PhD thesis, School of Computer Science, University of Birmingham (2011)