A Symbolic Analysis of ECC-based Direct Anonymous Attestation

Jorden Whitefield*, Liqun Chen*, Ralf Sasse†, Steve Schneider*, Helen Treharne* and Stephan Wesemeyer*
* Surrey Centre for Cyber Security, University of Surrey, United Kingdom
Email: {j.whitefield, liqun.chen, s.schneider, h.treharne, s.wesemeyer}@surrey.ac.uk
† Department of Computer Science, ETH Zurich, Switzerland
Email: ralf.sasse@inf.ethz.ch

***Abstract*—Direct Anonymous Attestation (DAA) is a cryptographic scheme that provides Trusted Platform Module (TPM)-backed anonymous credentials. We develop TAMARIN modelling of the ECC-based version of the protocol as it is standardised and provide the first mechanised analysis of this standard. Our analysis confirms that the scheme is secure when all TPMs are assumed honest, but reveals a break in the protocol's expected authentication and secrecy properties for all TPMs even if only one is compromised. We propose and formally verify a minimal fix to the standard. In addition to developing the first formal analysis of ECC-DAA, the paper contributes to the growing body of work demonstrating the use of formal tools in supporting standardisation processes for cryptographic protocols.**

***Index Terms*—Direct Anonymous Attestation, symbolic verification, TAMARIN PROVER, authentication, secrecy.**

I. INTRODUCTION

Devices such as laptops, smartphones and tablets, which connect to the Internet, are commonplace. Trusted computing is one approach that enhances the security on these devices by installing a "root of trust" (RoT), *e.g.*, through a Trusted Platform Module (TPM) [1], [2], ARM TrustZone Trusted Execution Environment [3], Intel Software Guard Extensions (SGX) [4], etc. These roots of trust are then used to attest that devices are in a "trustworthy" state, meaning that the devices behave as expected for a specific purpose. It is desirable that such device attestations be conducted in a privacy-preserving manner to protect users, and reduce the knowledge adversaries may learn. Two popular anonymous attestation schemes are Direct Anonymous Attestation (DAA) [5] for the TPM, and Enhanced Privacy ID (EPID) [6] for Intel's SGX. In this paper we focus on TPMs.

A TPM is a resource-constrained cryptographic co- processor which is embedded within a commodity device which we refer to as the host. The TPM supports *(i) isolation*: separate and protected from the host in the event of compromise; *(ii) protected execution*: ensures the operation is executed and not interfered with; and *(iii) secure storage*: storage which is only accessible by the TPM if the host is in a trustworthy state. The Trusted Computing Group (TCG) reports there are billions of TPMs installed in branded PCs, laptops and servers [7].

DAA is an anonymous digital signature scheme which provides authentication and privacy, to ensure the integrity of devices. There are two variants of a DAA scheme built from public-key cryptosystems, the RSA-DAA and the Elliptic Curve based ECC-DAA scheme respectively [8], [9], [10]. The ECC-DAA variant is more efficient for low-end resource constrained devices [10] which is appropriate for hardware RoTs.

ISO/IEC 20008-2:2013 [11] is implemented and deployed widely today within TPMs, more specifically, including three DAA related mechanisms: Mechanism 2 is an RSA-DAA scheme, which is implemented in the TPM 1.2 specification Mechanism 3 is the EPID scheme, which does not split the TPM and host operations, and which in the literature is called pre-DAA and Mechanism 4, the focus of this work, is an ECC-DAA scheme, which is implemented in early versions of the TPM 2.0 specification. The TPM API of ECC-DAA in the TPM 2.0 specification is designed to support two ECC-DAA schemes: Mechanism 4 and a modification of Mechanism 3 with the splitting operations. ISO/IEC 20008-2:2013 Mechanism 4 relies on the use of a "secure and authentic channel" but leaves such a mechanism out of scope. The standard refers to Chen et al. [10] to provide an appropriate mechanism. Throughout this paper we refer to the combination of ISO/IEC 20008-2:2013 Mechanism 4 and the recommended ways of implementing a secure and authentic channel as I-MECH4.

Brickell et al. [12] state that an ECC-DAA scheme must satisfy the notions of correctness, user-controlled anonymity and user-controlled traceability. Intuitively these correctness, security and privacy properties mean the following:

- *Correctness*: valid signatures are verifiable and linkable, when needed;
- *User-controlled anonymity*: the identity of a device cannot be revealed from the signature;
- *User-controlled traceability*: the host controls whether signatures can be linked.

The ISO/IEC standard directly cites Brickell et al. [12] as the reference for the scheme's security and privacy properties.

There have been substantial efforts from the academic community in the development of proofs analysing ECC-DAA, including simulation [5], game-based [13] and more recently within the UC-model [14]. The formal methods tool PROVERIF [15] has been used to analyse symbolic abstractions of DAA [16], [17], [18], [19]. These symbolic analysis efforts have not covered the host and TPM being split roles. All of these endeavours have helped to both find weaknesses in DAA and guide the design decisions of future releases of DAA [20].

A. Contributions

Our contribution in the paper is a complete, terminating and provable symbolic model for the suite of ECC-DAA operations with proofs of the security and privacy properties analysed using the TAMARIN PROVER (TAMARIN) [21]. We present a symbolic verification of the ISO/IEC 20008-2:2013 Mechanism 4, which is implemented in TPM 2.0. The analysis performed covers all operations of the ECC-DAA scheme: SETUP, JOIN, SIGN, VERIFY and LINK. Our main contributions in this work are as follows:

1) We develop a symbolic model of I-MECH4. We model the TPM and the host as two separate actors that communicate over a secure channel. Note that our model does not restrict the number of TPMS or hosts. The split of the hosts and TPMS is consistent with recent computational proofs. The roles of the hosts and TPMS are clearly separated and the communication between the roles is not under the control of an adversary, thus requiring the use of a secure channel. We provide a faithful abstraction of the operations of ECC-DAA which includes a representation of splitting an host and a TPM that is not covered by previous work.
2) We have defined a method for performing symbolic non-interactive zero-knowledge proofs for ECC-DAA within TAMARIN. We provide the first concrete example of zero-knowledge proofs in TAMARIN that did not require any modifications or additions to the tool. The formal model can be used in other protocol analysis containing zero-knowledge proofs.
3) This is the first symbolic model to prove all security and privacy properties of I-MECH4. Our analysis revealed a man-in-the-middle attack such that the compromise of a single TPM means that no other TPM can be authenticated reliably, and secrecy cannot be guaranteed. We proposed a solution for I-MECH4 based on the privacy CA protocol by Chen et al. [22], [23], and this provably fixes the attack found.
4) We present a clear mapping and encoding of I-MECH4 and its associated properties in a corresponding TAMARIN model. Our model is annotated against the standards document, which provides an easy way to validate that the formal model is a faithful representation of the standardised scheme. It also provides a foundational formal model for future symbolic ECC-DAA analysis within TAMARIN.

We provide all the artefacts needed to reproduce our results in [24].

B. Related Symbolic Analysis Work

Our work significantly goes beyond the state-of-the-art symbolic analysis of DAA schemes in the literature. Previous symbolic analysis work by Backes et al. [16] introduced a framework in the applied π calculus, for the reasoning and analysis of non-interactive zero-knowledge within the PROVERIF tool. Preliminary analysis of the RSA-DAA operations was used as a case study for their framework. The analysis revealed a weakness in the JOIN operation showing that if a TPM A was compromised and its endorsement key leaked then an adversary could perform a JOIN impersonating A. This attack was then fixed in the operation by including a TPMS identity in the zero-knowledge proof. Additional analysis of RSA-DAA anonymity in SIGN was performed, and showed that two DAA signatures were indistinguishable. Work by Backes et al. analysed the initial proposal by Brickell, Camenisch and Chen in [5] and pre-dates the standardisation of RSA-DAA in ISO/IEC 20008-2 2013 Mechanism 2. They do not consider a secure and authentic channel. In this paper we find a similar attack even though our model follows the recommended way of building a secure and authentic channel.

Smyth et al. in [17] found a vulnerability of the RSA-DAA scheme where user privacy could be violated in the presence of a corrupt ISSUER and VERIFIER which collude. They demonstrated that if a VERIFIER uses the same linking property (basename) as the ISSUER, then the identity of a PLATFORM can be revealed. An ISSUER could also be a VERIFIER and it is not unreasonable that this single entity would have the same basename in both operations. This is possible due to the way in which the basename is computed, *e.g.*, $hash(bsn)$ for both JOIN and SIGN. The privacy violation is fixed by making a minor alteration to the RSA-DAA scheme which introduces a 0 or 1 bit in the computation of basename, *e.g.*, $hash(0 \parallel bsn)$, for the operations JOIN and SIGN respectively thus preserving untraceability. Again this work pre-dates the standardisation of DAA in ISO/IEC 20008-2 2013, and the model was developed using the initial standard defined by the TCG in version 1.2 revision 85 [25] in 2005.

Additional research on the ECC-DAA scheme by Smyth et al. was presented in [18]. It is important to note that their model does not provide a full abstraction of the scheme, because the JOIN operation is omitted. Additionally, in the SIGN operation the adversary was forbidden from re-blinding signatures which limits the adversary ability. The authors analysed the user-controlled anonymity property of ECC-DAA using observational equivalence and this was shown to hold. However, no general conclusions regarding anonymity could be made due to the level of abstraction of the model. The authors state that the focus of their work was on the ISO/IEC 20008-2:2011 draft standard.

While a number of different symbolic modelling tools exist including TAMARIN and PROVERIF, we chose TAMARIN to model ECC-DAA due to the class of protocols it has successfully been able to analyse including TLS 1.3, eVoting, public-key infrastructure and future Intelligent Transportation Systems [26], [27], [28], [29].

C. Paper Organisation

The paper is organised as follows. In Section II we describe the ECC-DAA scheme, its suite of protocols and the security and privacy properties claimed by Brickell et al. in [12]. Section III describes an overview of the fundamentals of the TAMARIN PROVER and our TAMARIN model. In Section IV, we describe our threat model and formalise the security and privacy

guarantees. Section V describe our results and Section VI provides conclusions and future work.

II. ECC-BASED DIRECT ANONYMOUS ATTESTATION

In this section we provide a concise description of the ECC-DAA scheme's operations, as defined in ISO/IEC 20008-2:2013 [11], necessary for understanding our symbolic model.

Direct Anonymous Attestation [5] is an authentication mechanism that enables the provision of privacy-preserving and accountable authentication services. ECC-DAA is based on group signatures that give strong anonymity guarantees, unlike traditional digital signature mechanisms [30], [31], [32] that are used to provide entity authentication, non-repudiation and data integrity.

Traditional digital signature mechanisms enable the holder(s) of a private key to generate a digital signature for a message. The related verification key (public key) is then used to verify the validity of a signed message. In contrast, an anonymous digital signature mechanism is a special class of a digital signature where no (authorised or unauthorised) entity can discover the identity of the user who signed the message. As with traditional digital signature mechanisms, anonymous digital signature mechanisms are based on asymmetric cryptography. The difference between traditional digital signature mechanisms and anonymous digital signatures is that to verify an anonymous signature a user makes use of a group public key (group signature) or multiple public keys (ring signature), neither of which is bound to an individual user.

The ECC-DAA scheme is an anonymous digital signature mechanism, and the motivation for applying ECC-DAA is the ability to split the signer role between a secure device (TPM), and a commodity computing device (host). In ISO/IEC 20008-2:2013 mechanism 4 a TPM is referred to as a principal signer and a host is referred to as an assistant signer. Throughout the paper a principal signer is referred to as PSIGNER and an assistant signer as ASIGNER. Essentially, a PSIGNER can sign any arbitrary message collaboratively with an ASIGNER which is the commodity device. This split utilises the high level of security offered by a PSIGNER, in conjunction with the computational ability and storage capacity offered by an ASIGNER (see [10] for a practical example).

An ECC-DAA scheme considers a set of entities: ISSUERS, ASIGNERS, PSIGNERS, and VERIFIERS; the ASIGNER and PSIGNER together form a trusted PLATFORM. The ISSUER is a trusted third-party responsible for attesting and authorising PLATFORMS to join the network of PLATFORMS. A VERIFIER is any other system entity or trusted third-party that can verify a PLATFORM's credentials in a privacy-preserving manner using ECC-DAA operations; without the need of knowing a PLATFORM's identity. The ECC-DAA scheme is a two phase process with five operations. Phase one consists of SETUP and JOIN while phase two uses SIGN, VERIFY and LINK. The interactions between entities involved in the JOIN, SIGN and VERIFY operations are shown in Figure 1. The DAA notation followed in this paper are as presented in the ISO/IEC 20008-2:2013 [11] document. Briefly, $[x]P$ is a multiplication operator that takes a positive integer x and a point P on an elliptic curve. The '$+$' operator represents addition, '$-$' operator is subtraction, and '$\|$' operator is the concatenation of two data items.

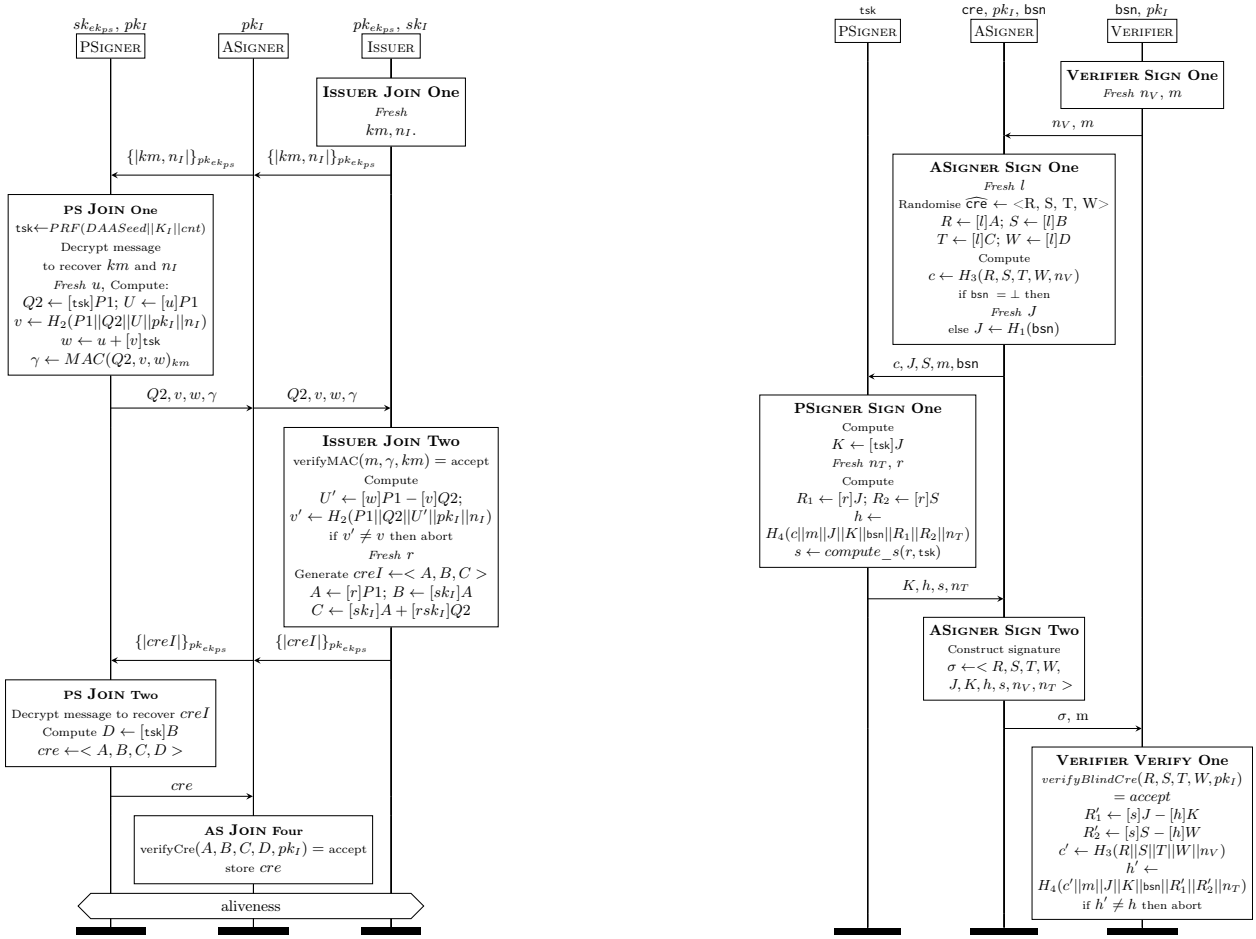
We next describe each of the ECC-DAA operations in turn.

SETUP: The SETUP operation initialises the system with the security parameters (K_i, $P1$), for each of the operations and long-term parameters of the ISSUER, and these parameters are published to all entities. Prior to this operation, we assume during the manufacture time of a PSIGNER, an endorsement key-pair ($sk_{ek_{ps}}$ / $pk_{ek_{ps}}$) is embedded by the manufacturer in read-only memory (ROM) and ISSUERS have access to public endorsement keys. Furthermore, a unique internal secret value `DAASeed` is set, and a monotonic counter (`cnt`) is implemented on the PSIGNER, and an external counter is available to the ASIGNER. The ISSUER also generates its ECC-DAA key-pair (sk_I / pk_I), and publishes its public key.

JOIN: This operation of the ECC-DAA scheme is run between a PLATFORM (the ASIGNER and PSIGNER) and an ISSUER. The JOIN operation executes as shown in Figure 1a and upon successful completion attests a PLATFORM as being a genuine member of the group, and the PLATFORM receives a credential (`cre`) from the ISSUER for use in future communications with VERIFIERS. The `cre` attests that the PSIGNER is valid, and the PSIGNER computes the D element of `cre` containing the ECC-DAA secret key `tsk`. The communication is conducted over a public channel between the ISSUER and an ASIGNER. The TCG recommends that encryption is applied to this communication using the TPM endorsement keys [33].

SIGN / VERIFY: The SIGN operation is run between a given PSIGNER and its associated ASIGNER when a VERIFIER sends a message to be signed to the ASIGNER. The VERIFIER then performs the subsequent VERIFY operation. Other VERIFIERS in the group can also verify a signed message. Figure 1b describes the various steps of the SIGN and VERIFY operations, and the interaction between entities.

An ASIGNER initiates the SIGN operation when it receives a message, n_V, from a VERIFIER. The ASIGNER SIGN step of the operation constructs one portion of the ECC-DAA signature which includes randomising the ASIGNER's credentials `cre` yielding R, S, T and W. The VERIFIER basename `bsn`, which is either a fixed string value associated with the VERIFIER or not specified (denoted by the special symbol $\perp$), determines whether J is a randomly selected group element or fixed as $H_1(\texttt{bsn})$. The latter case is used when signatures are required to be linkable. The ASIGNER sends its part of the ECC-DAA signature to the PSIGNER which then uses these values to construct a proof of knowledge of its secret key `tsk` which it returns to the ASIGNER. The ASIGNER completes the SIGN operation, by incorporating the various computed values into the signature σ, which the VERIFIER can now use to verify that the message has been signed by a PLATFORM that is a member of the group. These proofs convince a VERIFIER that a message is signed by a ECC-DAA key that was certified by the ISSUER, without knowledge of the PSIGNER's ECC-DAA key or `cre` (VERIFY). Of course, the VERIFIER has to trust that the ISSUER only issues `cres` to valid PSIGNERS.



(a) JOIN operation

(b) SIGN/VERIFY operation

Fig. 1: Symbolic representation of ECC-DAA message flow diagrams

LINK: The LINK operation may be used by a VERIFIER to check if two or more signatures, σ, are linked. Linkability is controlled in ECC-DAA by the value of the basename bsn either being set or unset. If $\mathsf{bsn} = \perp$ then an ASIGNER will select a fresh group element, J, uniformly at random, else it computes $J = H_1(\mathsf{bsn})$. The ASIGNER then sends J to its PSIGNER, which then computes $K = [\mathsf{tsk}]J$. Thus, if J is always a hash of bsn then a PLATFORM's messages will be linkable because J and K remain constant in all ECC-DAA SIGN responses. This assumes that each of the signatures came from the same PLATFORM signed with its ECC-DAA secret key tsk.

III. MODELLING THE PROTOCOL IN TAMARIN

A. *The* TAMARIN PROVER

TAMARIN [21], [34] is a state-of-the-art protocol verification tool for *symbolic modelling*. It supports unbounded verification, mutable global state, and flexible user- defined equational theories. Protocols are modelled using multiset rewriting rules and properties are specified in a first-order logic fragment. The tool offers automatic verification succeeding in many cases, as well as an interactive verification mode with manual proof tree traversal. The tool provides both proofs, and disproofs by counter-example, but may not terminate due to the undecidability of the underlying problem.

Dolev-Yao adversary. We use the standard adversary in the symbolic model: the Dolev-Yao (DY) adversary [35]. The DY adversary is a very strong network adversary, with full control over the network: it can intercept, block, replay, and send any message on the network. Additionally, the adversary learns all the content in all messages it sees, unless they are cryptographically protected. Furthermore, if the DY adversary knows (or has derived) the appropriate keys, it can encrypt and decrypt messages. The cryptography used is assumed perfect, *i.e.*, the hash functions used are cryptographically secure and the adversary cannot encrypt or decrypt messages without knowledge of the right key. The precise capabilities of the adversary and the assumptions on the cryptographic primitives used are encoded in the equational theory specified.

Terms and equations. In a symbolic model, all messages are described as terms, for example $aenc(m, pk(k))$ represents the asymmetric encryption of some plaintext m under a public key $pk(k)$, rather than dealing with bitstrings and probabilities, as is done in the computational model. Then, one defines a signature Σ as a number of operators, *e.g.*, $aenc$, each equipped

with an arity, *i.e.*, the number of arguments it accepts (Note that Σ is not a cryptographic signature). Terms are constructed by applying operators to constants, variables, and other operators recursively.

The cryptographic properties of the used primitives are then specified as equations. The example of asymmetric encryption introduced above would also contain a decryption operator $adec$, and an equation $adec(aenc(m, pk(k)), k) = m$ that allows extracting the plaintext message m using the private key k associated with the used public key $pk(k)$. Note that the equations specified *completely* characterise all possible derivations, as the perfect cryptography assumption is used, meaning there is no other way to break the used primitives. A set of equations, together with the underlying signature, is called an equational theory. TAMARIN allows convergent (*i.e.*, both confluent and terminating) equational theories [36] that additionally satisfy the Finite Variant Property [37].

Facts, states, rules, and labelled multiset rewriting.

Distinguished terms, called *facts*, consisting of a top-level fact symbol of fixed arity and with standard terms as arguments, build the *state*. Specifically, a state is a finite multiset of facts, and it represents the current state of a protocol's execution, including all participants local states, the adversary knowledge, and messages currently on the network. The distinguished $Fr(x)$ fact represents fresh values and the semantics of all other facts is given by the specified *rules*. The rules model the possible actions of protocol participants as well as the adversary actions.

Rules are given as triples, written `[l]--[a]->[r]` with `l`, `a`, `r` finite sequences of facts, representing the *premises*, *actions*, and *conclusions* respectively. As a general DY is considered, a modelling convention is that messages are sent to the network using a special `Out` fact, received from the network by `In` and the adversary knowledge is represented by `K` facts. Note that the adversary can apply all the equations given in the equational theory specified and modify messages as it wants, assuming it has the necessary cryptographic keys available.

The set of rules specifying the protocol and adversary then yields a labelled transition system, with the initial state being the empty multiset. TAMARIN changes its state multiset by finding an applicable rule, *i.e.* one whose premises match existing facts within the current state multiset to obtain a new state multiset where the facts used in the premise are replaced with those from the rule's conclusion. The actions associated with each rule instance in the execution yield the *trace*. Each rule instance and all associated actions are timestamped with the timepoint of their occurrence with ordered timepoints. Properties are then specified on top of the action trace using first-order logic. First, we give an example protocol, and then will consider properties in more detail.

Example 1. Let us consider a simple protocol that sends a message encrypted under a public key. The first rule creates a public/private key pair, which can be used arbitrarily often. The second rule describes the sender, which picks a fresh value to send, and looks up the intended recipient's public key and outputs that value encrypted under this key. The third rule shows the recipient receiving this message, looking up its own private key k and accepting the message only if it was encrypted under the related public key $pk(k)$. Otherwise the rule cannot be triggered.

```
Create:  [Fr(~k)]--[]->[!Ltk($A,~k),!Pk($A,pk(~k))]
Send:    [!Pk($R,pubk), Fr(~m)] --[Sent(~m)]-> [Out(aenc(~m,pubk))]
Receive: [!Ltk($R,k), In(aenc(m,pk(k)))] --[Received(m)]-> []
```

Security property specification. Trace properties such as secrecy and agreement are expressed as first-order logic formulae. These formulae introduce variables to reason about the ordering of actions traces ([38] provides more detail). A formula ϕ may hold on trace tr and we lift the semantics to a set of traces Tr. We say a formula holds for all traces when it is satisfied by any trace in the set (which we use to prove security properties), and we say that there exists a trace satisfying the formula ("exists- trace" semantics) when there is at least one trace on which the formula holds. We use this semantics in general to show that some protocol is executable, or a specific state can be reached.

Example 2. Extending Example 1 we define an executable property which states that the final rule may be executed:

```
lemma executable_example: exists-trace
"Ex z #i. Received(z) @#i"
```

and this formula is satisfied in the example by executing the three rules once each, in order, with the message being forwarded from the `Send` rule's `Out` fact to the `Receive` rule's `In` fact by the network adversary.

An additional feature we make use of is that of *restrictions*. Restrictions are useful to limit the set of traces one wants to consider in protocol analysis. We employ a number of restrictions on our model of ECC-DAA to restrict the branching behaviour.

In addition to trace properties, TAMARIN supports equivalence properties as well. These are necessary for privacy properties such as unlinkability. For equivalence properties it is required that two instances of the protocol are indistinguishable for the adversary. The instances are defined by use of *diff*-terms (taking two arguments), and in essence yield two versions of the same protocol, with the differences limited to the terms under the *diff*-operator. TAMARIN checks observational equivalence on this (see [39] for details) by comparing the two resulting systems and ensuring that the adversary cannot distinguish the two for any protocol execution and adversary behaviour.

B. The model

Using TAMARIN we implemented a symbolic model of ECC-DAA that captures the behaviour and split roles of I-MECH4. Our model captures these behaviours in the presence of a DY adversary. From Figure 1b it is clear that the functionality of the protocol is different depending on whether the basename (`bsn`) is fresh or fixed. This distinction is best captured using different variants of our model to aid traceability and will only be important during the analysis. It would have been possible to produce one model to reflect the functionality of both J

being fixed and fresh but this would have meant duplication of rules and introduction new state facts to control the firing of the rules. This would have made the model less readable. By producing two variants of the model it means that the rules within each variant are more easily matched to the functionality of the ECC-DAA scheme and provides a traceable mapping to the standards document.

Our model is more comprehensive than [18] as it is finer-grained and closer to the standards document. In addition, we also, for the first time, capture in a symbolic setting, all the authentication and privacy properties of ECC-DAA in a single model. The full TAMARIN model is available [24] and contains 23 rules and comprises of over 1300 lines of TAMARIN code.

We begin by describing the equational theory for our ECC-DAA models. We construct a signature, Σ, to capture the cryptographic operators, where $aenc/2$ is a binary operator and more generally f/n introduces an n-ary operator called f.

$$\Sigma = \{aenc/2,\ adec/2,\ pk/1,\ MAC/2,\ verifyMAC/3,\ accept/0,\ H2/5,\ multp/2,\ plus/2,\ minus/2,\ U/2,\ calcU/1,\ verifyCre/5,\ verifyBlindCre/5,\ H1/1,\ H3/5,\ H4/8,\ PRF/3,\ compute_s/2,\ calcR1/1,\ calcR2/1,\ checkAnon/5,\ deanon/0\}$$

The $aenc$, $adec$ and pk operators come from the `asymmetric-encryption` built-in. The multiplication operator is represented as '`multp`', addition by '`plus`', and subtraction by '`minus`'. The properties of the other operators in Σ are defined as equations and we describe each in turn.

Equation 1. **Message Authentication Codes (MAC):** To model MACs used in the JOIN operation for providing authentication and integrity during the challenge-response between a PSIGNER and ISSUER we define the following equation:

$$verifyMAC(m, MAC(m,k), k) = accept$$

Given a message m, the MAC of m signed under key k, and the key k, we can model that a MAC has been signed and constructed correctly with knowledge of k. Successful application of this equation will reduce to the $accept$ constant.

Equation 2. **DAA Credential Verification:** ECC-DAA has two different credential verification stages, one which verifies that a credential, `cre`, received by the ISSUER and signed by a PSIGNER was correctly constructed in the JOIN operation. The other allows other VERIFIERS to verify a randomised credential $\widehat{\texttt{cre}}$ in the SIGN operation.

One of the final steps of the JOIN operation is to verify that the credentials, (A, B, C), are received from the ISSUER and that the element D constructed by the PSIGNER, that is dependent on B also originates from the same ISSUER. To achieve this we express the equation $verifyCre$ to take A, B, C, D and the ISSUER's pk_I as inputs. If the ISSUER's secret key embedded within A, B, C and D corresponds to the same ISSUER pk_I, then the `cre` is valid.

$$verifyCre(A, B, C, D, pk(sk_I)) = accept$$

The equation in the model fully defines A, B, C and D to be the appropriate terms, for example A is `multp(creRandom,P1)`.

During the VERIFY operation VERIFIERS are required to validate the randomised credential, $\widehat{\texttt{cre}}$=$< R, S, T, W >$ that is constructed from A, B, C and D by multiplying i by a randomly chosen factor l. The randomising of the credential takes place during the SIGN operation by the ASIGNER, where each element of the `cre` is randomised by l; for example, `multp(l,multp(creRandom,P1))`, etc. The validation is expressed as the equation $verifyBlindCre$, and as in the previous equation ensures that the secret key of the ISSUER and public key of the ISSUER match. Again R, S, T and W in the equation are fully expanded in the model.

$$verifyBlindCre(R, S, T, W, pk(sk_I)) = accept$$

The following two equations capture our mathematical abstractions of the ECC-DAA non-interactive proof of knowledge within our ECC-DAA model.

Equation 3. **Calculation of U':**

The zero-knowledge proof of knowledge (ZKPK) in the ISSUER JOIN Two step of the JOIN operation (Figure 1a) is that the two values of the hash H_2, *i.e.*, v and v', are computed equally by the PSIGNER and ISSUER respectively. The structures of the hashes are identical, differing on only the U term. Therefore, to demonstrate ZKPK in the symbolic setting we are required to show that U' is equal to U during the construction of the ISSUER's v'. This is defined using the following equation.

$$calcU(minus(multp(w, P1), multp(v, Q2))) = U(u, P1)$$

The functionality of the equation represents the following reduction:

$$\begin{aligned} U' &= [w]P1 - [v]Q2 \\ &= [u + v_{\texttt{tsk}}]P1 - [v][\texttt{tsk}]P1 \\ &= [u]P1 + [v_{\texttt{tsk}}]P1 - [v_{\texttt{tsk}}]P1 \\ &= [u]P1 \\ &= U \end{aligned}$$

In the TAMARIN model the equation is fully expanded to define w and v explicitly and what we show here is the structure of the equation.

Equation 4. **DAA Signature Verification:** The ZKPK in the ECC-DAA VERIFY operation requires that the two hash values, *i.e.*, h and h', are the same, and h is defined in PSIGNER SIGN One in Figure 1b. Their H_4 structures are identical apart from the R_1 and R_2 terms in h and R_1' and R_2' terms in h'. Therefore, we provide two equations to state that R_1' reduces to R_1 and similarly for R_2':

$$calcR1(minus(multp(s, J), multp(h, K))) = multp(r, J)$$

and R_2' is equal to R_2:

$$calcR2(minus(multp(s, S), multp(h, W))) = multp(r, S)$$

Similar to the above $calcU$ equation the details of each term within $calcR1$ and $calcR2$ are fully expanded in the TAMARIN model.

Equation 5. **DAA De-anonymisation:** In the event a PSIGNER DAA key, tsk, is compromised and known by the adversary, it is possible to identify messages produced by a specific PSIGNER. We captured this in the $checkAnon$ equation which takes as its input the following terms from a given ECC-DAA signature:

$$checkAnon(S, W, J, K, tsk) = deanon$$

If the tsk matches in these formulae then the signature can be linked to the PSIGNER whose tsk was revealed. The functionality of the equation represents the following computation:

$$\begin{aligned} S &= [l]B \\ W &= [l]D = [l][\mathsf{tsk}]B = [\mathsf{tsk}][l]B = [\mathsf{tsk}]S \\ K &= [\mathsf{tsk}]J \end{aligned}$$

Hence, an adversary with knowledge of a PLATFORMS DAA key and DAA signatures can check whether W and K can be computed by multiplying S and J by tsk respectively to reveal whether the PLATFORM produced the signature.

Channels. Chen et al. in [10] note that the communication between an ASIGNER and a PSIGNER is done in a secure manner. In our TAMARIN model we define the communication between these two entities over a *Secure Channel* to provide an appropriate abstraction. Secure channels have the property of being both confidential and authentic. This means that an adversary can neither modify nor learn messages that are sent over the channel. They have previously been used in TAMARIN in [40] and we follow their modelling ideas.

Secure channel communication uses two rules, `ChanOut_S` and `ChanIn_S`, to create an extra layer of abstraction based on linear facts to explicitly model secure channels. This prevents communication being broadcast via the adversary, over the standard `In` and `Out` channels.

```
rule ChanOut_S:
[ Out_S( $A, $B, x ), !Paired( $A, $B ) ]
--[ ChanOut_S( $A, $B, x ) ]->
[ Sec( $A, $B, x ) ]
```

The fact `Out_S($A,$B,x)` models that the PSIGNER or ASIGNER (A, B or vice versa) sends a message `x` on the secure channel. The persistent fact `!Paired($A,$B)` is a predicate on the channel storing state information about the one to one association between a ASIGNER and PSIGNER. This ensures that only the designated PSIGNER can communicate with its corresponding ASIGNER. The conclusion of the rule is a linear fact containing the message x, that the adversary cannot see or forge.

```
rule ChanIn_S:
[ Sec( $A, $B, x ) ]
--[ ChanIn_S( $A, $B, x ) ]->
[ In_S( $A, $B, x ) ]
```

The linear fact `Sec(...)` ensures that the secure channel is replay protected, i.e., when the message x is consumed by one of the paired entities, x is not stored to be replayed later as a consequence of the shared `In_S($A,$B,x)` fact which is not known to the adversary. The secure channel is justified as being replay protected as this channel is only ever used on the PLATFORM between the ASIGNER and PSIGNER.

Recall that in a JOIN operation the communication between an ISSUER and a PSIGNER needs to be encrypted under the public endorsement key. Note that — unlike the secure channel between PSIGNER and ASIGNER— this communication can be observed by the adversary, as shown in Figure 2.

C. Model Abstractions and Restrictions

To simplify the number of cases in the proof, we consider the VERIFIER to be an abstract role. This means that a VERIFIER is not a PLATFORM in our model whereas in reality it could be another PLATFORM or some other device, *e.g.*, embedded device. An abstraction of this role is possible as a PSIGNER is not required to verify a ECC-DAA signature. We have abstracted the ISSUER's public key (X, Y) and private key (x, y) to pk_I and sk_I respectively.

The most important abstraction is how we modelled "*a secure and authentic channel between the principal signer and the group membership issuer*", since the definition of such a channel was outside the scope of the standards document. As stated earlier ISO/IEC 20008-2:2013 refers to Chen et al. [10] who propose the use of a MAC and the TCG propose the use of a public endorsement key [33] to create a secure and authentic channel within mechanism 4. The model of using a MAC is already captured by Equation 1 and in the next section we describe our abstraction of the TCG mechanism.

We also employ a number of restrictions in our model of ECC-DAA (we refer to them as *A1 - A6* to avoid confusion with R_1 and R_2 in the model description):

A1 - Single Issuer: We consider the ISSUER to be a distinct role in the protocol. This choice has been made to simplify the proof, and it is important to note that the ISSUER can still be a corrupt entity.

A2 - Unique Pairing: We constrain a PSIGNER to belong to a single unique ASIGNER, and a ASIGNER to have exactly one PSIGNER. This models an ideal system and is representative of the real world.

A3 - Single Platform Initialisation: Once a PLATFORM has performed its SETUP to generate its unique values and endorsement key-pair, it is not allowed to do this again. This restriction captures the manufacture process where secrets are installed to a PSIGNER at manufacture time.

A4 - Equality checks: We use the TAMARIN equality restriction so that all instances of an equality action in a trace ensure that both arguments within an action are equal. We use them for modelling the verification of MACs, `cre`, $\widehat{\mathsf{cre}}$ and ECC-DAA signatures.

A5 - Inequality checks: Such checks are used to ensure that two arguments of a check are not equal in a trace. This is used specifically during the PLATFORM initialisation rule to specify the ASIGNER and PSIGNER identities are different, which is also the case in practice.

D. Modelling the ECC-DAA operations

In Section II we noted that there are five ECC-DAA operations and each of them match to one or more TAMARIN rules. This section summarises the mapping from operations to TAMARIN

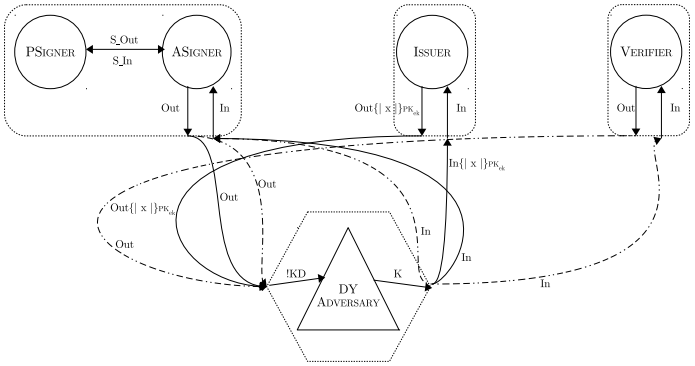


Fig. 2: Network with all communication routed through the adversary

rules and provides some illustrative examples. The SETUP operation corresponds to two rules, one to capture the setup of the ISSUER and one to setup a PLATFORM. The LINK operation maps to one rule. The JOIN operation maps to eight rules, representing ISSUER JOIN One etc. from Figure 1a and three of the rules represent the forwarding of messages from the ISSUER to the PSIGNER via the ASIGNER over the secure channel. Similarly the SIGN operation maps to four sign rules. The VERIFY operation maps to two rules in order to capture the behaviour of a VERIFIER verifying the message it sent and verifying a message sent by a different VERIFIER.

The rule for setting up a PLATFORM is defined as follows:

```
rule PLATFORM_SETUP:
  let pk_ek = pk( ~sk_ek ) in
    [ Fr( ~sk_ek ) ]
  --[ PlatformInit(), PlatformStart( $AS, $PS ),
      Create( $PS ), Neq($AS, $PS),
      UniqueExecJoin( 'PS_SETUP' ),
      Unique_Pairing( $PS ),
      Unique_Pairing( $AS ) ]->
    [ !F_PSEk($PS,~sk_ek),!F_PSPkEk($PS,pk_ek),
      !F_Paired($PS,$AS),!F_Paired($AS,$PS),
      St_PlatformInit( $AS, $PS ),
      Out( pk_ek ) ]
```

The PLATFORM_SETUP rule initialises the PLATFORM, which is a combination of an ASIGNER and PSIGNER.

In the premise three fresh terms are generated to compute the tsk and the PSIGNER endorsement key-pair. The ISSUER parameters Ki are also required in the generation of tsk. There are a number of action labels, for example Unique_Pairing that captures the restriction *A2*. The conclusion of the rule stores the generated terms and introduces a linear fact to control moving to the next step of the ECC-DAA scheme. Note that the persistent fact !F_PSPkEk stores a PSIGNER's public endorsement key. The fact that this key is installed in the PSIGNER at manufacturing time and shared with the ISSUER is modelled using the Out(pk_ek) fact. Note that this also makes the key available to the adversary in our model. The public endorsement key, $pk_{ek_{ps}}$ is only ever used in the JOIN operation by the ISSUER so that the ISSUER can authenticate to a PSIGNER. This is shown in the following fragment of the PS_JOIN_ONE rule:

```
rule PS_JOIN_ONE:
let
  tsk      = PRF( ~DAASeed, Ki, ~cnt )
  pk_ek    = pk( sk_ek )
  msg      = aenc( < 'ISSUER_REQ', km, ni >, pk_ek )
  gamma    = MAC( < 'gamma', P1, Q2, v, w >, km )
  ...
in
  [ In_S( $AS, $PS, msg ), Fr(~DAASeed), Fr(~cnt),
    !F_IssuerKi( $I, Ki ), ...
  ]
  --[ ... HonestPS( tsk ) ... ]->
  [ Out_S( $PS, $AS, <'PS_RESP_OUT', ..., gamma> )
    !F_PSDaaSeed( $PS, ~DAASeed ),
    !F_PSCnt($PS,~cnt),!F_PSTsk($PS,tsk) ]
```

The tsk is generated using a pseudo-random function, PRF, which was introduced as part of Σ. It provides a fine-grained traceable abstraction to the construction of tsk, rather than a fresh term. While these example rules do not allow the adversary to learn the secret endorsement key, $sk_{ek_{ps}}$, or the secret DAA key tsk, we have also defined two additional TAMARIN rules that allow the adversary to learn the secret keys. This models a possible threat of corrupted PLATFORMS that can be identified during the security analysis. Recall that that I-MECH4 requires a secure and authentic channel for communication between PLATFORMS and ISSUERS and within the model a MAC is constructed to ensure the integrity of $Q2$, v and w in the JOIN operation. The verification of the MAC occurs in another rule in the JOIN operation, and this is where Equation 1 is called.

One example of a SIGN rule is as follows:

```
rule PS_SIGN_ONE:
  let
    PSSign   = < 'PSSign', c, ~J, S, nv, bsn >
    tsk      = PRF( DAASeed, Ki, cnt )
    K        = multp( tsk, ~J )
    R1       = multp( ~randS1, ~J )
    R2       = multp( ~randS1, S )
    h        = H5( c, nv, ~J, K, bsn, R1, R2, ~nt )
    s        = compute_s( ~randS1, tsk )
    PSResp   = < 'PSSignResp', K, h, s, ~nt >
  in
    [ In_S( $AS, $PS, PSSign ), Fr( ~nt ),
      Fr( ~randS1 ), !F_PSTsk( $PS, tsk ) ]
  --[ PSSignOne( ), DAASign( tsk, ~J, nv ),
      UniqueExecSign( 'PS_SIGN_ONE' ) ]->
    [ Out_S( $PS, $AS, PSResp ) ]
```

Goal	Lemma	Model A	Model B
G1	functional_correctness_group_verification	✓	✓
G2	functional_correctness	✓	✓
G3	functional_correctness_dishonest_send	✓	✓
G4	aliveness	✓	✓
G5	weak_agreement_any_reveal	✓	✓
G6	weak_agreement	×	×
G7	ni_agreement_any_reveal	✓	✓
G8	ni_agreement	×	×
G9	i_agreement	×	×
G10	secrecy_cre	×	×
G11	can_be_deanonymised	✓	✓
G12	user_controlled_independent_link_tokens	✓	n/a
G13	user_controlled_linkability	n/a	✓
Goal	**Observational Equivalence**	**Model C**	
G14	unlinkability	✓	

Fig. 3: Summary of Results

This rule represents the PSIGNER SIGN One step in Figure 1b where the input to the rule over the secure channel In_S are the terms produced by the ASIGNER from ASIGNER SIGN One in Figure 1b. The premise also generates two fresh terms nt and randS1, which correspond to n_T and r respectively. The ECC-DAA key, tsk, is known by the PSIGNER and is used in the generation of K which, together with the term J, is used for controlled traceability of signed messages to a PLATFORM. The first action in the PS_SIGN_ONE rule simply denotes what rule is being fired, and the others support the expression of security and privacy properties. The conclusion securely outputs PSResp to the ASIGNER which includes K to control traceability, the hash h capturing the proof of knowledge for the ECC-DAA SIGN operation, and s which is needed to recompute h by the VERIFIER and a nonce n_T.

IV. THREAT MODEL AND PROPERTIES

All the properties we establish for our ECC-DAA model are identified in Figure 3. We use ✓ to indicate that the property holds, and × is used when the property does not hold. Recall in Section III-B two variants of the model were introduced and they are referred to as Model A and Model B in Figure 3 respectively. Model A represents the basename being unset whereas Model B represents the basename as a constant. In the analysis of unlinkability in Section IV-C we introduce and justify Model C.

A. Threat Model

As stated in Section III-A the model considers a Dolev-Yao adversary. Notably, the DY can compromise a PSIGNER to gain knowledge of the secret endorsement key, and the secret DAA key. Additionally, the adversary can corrupt the ISSUER and learn its secret key.

B. Security Properties

In this section we focus on how we encode the *correctness*, *authentication*, and *secrecy* properties, as given by goals $G1$ - $G10$. We provide the TAMARIN code for lemmas $G1$ and $G6$ as examples in Appendix B. The definition of correctness from [12] simply refers to a correct execution of the scheme. Normally within a symbolic setting correctness is a notion that applies over all traces. Hence correctness here in a symbolic setting is also in the context of a single run of the protocol (and uses exists-trace) but we refer to it as *functional correctness* to avoid confusion. Our lemmas not only encode correctness from [12] we also explore correctness of a group signature to build confidence in our model.

In the model we additionally define authentication lemmas to determine what level of authentication the scheme satisfies. This enables us to establish the authentication in the scheme in the usual way using formal analysis.

$G1$ *Functional Correctness (group verification)*: ISO/IEC 20008-2:2013 mechanism 4 [11] defines that group signature verification is required for ECC-DAA which means there exist two VERIFIERS, one that sends a message to be signed by a PLATFORM, and another VERIFIER that verifies the message. This property is captured by our lemma functional_correctness_group_verification which states that there exists a send from one VERIFIER that is then signed by a signer PLATFORM, and verified by a different VERIFIER than the one which sent the message to be signed.

$G2$ *Functional Correctness*: We encode the ECC-DAA property as one lemma. The lemma states that if both the PLATFORM and VERIFIER are honest, the signatures and their links will be accepted by a VERIFIER. This means that the following must have occurred: A) SETUP has occurred and the ISSUER has generated its secret key and published its parameters. B) JOIN has successfully executed under the ISSUER secret key and parameters, therefore producing a ECC-DAA credential including the PSIGNERS generated ECC-DAA key (tsk). C) SIGN has produced a ECC-DAA signature σ_0 on the message m_0 with tsk and a randomised credential $\widehat{\text{cre}}$. Given the steps A, B and C have successfully executed then a ECC-DAA VERIFY on σ_0 has executed to accept the signature.

$G3$ *Functional Correctness of SIGN in presence of an adversary*: The lemma is used to prove the ECC-DAA SIGN operation, in the presence of an adversary that sends a message on the network. The intuition for this lemma is that there exists a VERIFIER that receives and verifies a signed ECC-DAA message which may have been sent by the adversary or the adversary has corrupted a PLATFORM so that she could forge a message. Given the capabilities of the adversary she would be able to generate and inject a message on the network and request that a signer executes the SIGN protocol and signs the message. This would result in a signed message that was crafted by the adversary, which all other group members would be able to verify. This property is captured by our lemma functional_correctness_dishonest_send.

$G4 - G9$ *Authentication in JOIN*: In [41], Lowe identifies a hierarchy of authentication specifications. In this paper we explore which form of authentication as defined by Lowe our ECC-DAA scheme satisfies: *aliveness*, *weak agreement*, *non-injective agreement* and *injective agreement*.

Aliveness is a form of authentication which guarantees that when an initiator A completes a run of the protocol, apparently interacting with another agent B, then B has run the protocol,

```
lemma secrecy_cre:
"All A B x #i. Secret(A,B,x) @ i
==>
not(Ex #k. K(x) @ k)
|(Ex C #r. IssuerKeyReveal(C) @ r & Honest(C) @ i)
|(Ex C #r. RevealEK(C) @ r & Honest(C) @ i)
|(Ex C #r. RevealTsk(C) @ r & Honest(C) @ i)"
```

(a) secrecy_cre ($G10$)

```
lemma can_be_deanonymised: exists-trace
"Ex AS PS sigma tsk #i #j #k #l.
( PlatformStart(AS, PS) @ i
& RevealPSTsk(PS, tsk) @ j
& ASSendFullSignature(AS, PS, sigma)@ k
& DeAnonymised(PS, tsk, sigma) @ l )"
```

(b) can_be_deanonymised ($G11$)

```
lemma user_controlled_independent_link_tokens:
"All k kP j jP #i .
    CompareLinkTokens(k,kP,j,jP) @ i
    & not(j = jP)
      ==> not(k = kP)"
```

(c) user_controlled_independent_link_tokens ($G12$)

```
lemma user_controlled_linkability:
"All k kP j jP #i .
  (All #i #j x . UniqueExecJoin(x) @ i
                & UniqueExecJoin(x) @ j
                  ==> #i = #j)
  & CompareLinkTokens(k,kP,j,jP) @ i
  & j = jP
    ==> k = kP"
```

(d) user_controlled_linkability ($G13$)

Fig. 4: Secrecy and User-controlled Linkability lemmas

but not necessarily with A. In the ECC-DAA scheme the initiator A is the ISSUER and an agent B is a PLATFORM.

Weak agreement is a slightly stronger form of authentication that guarantees when an initiator A completes a run of the protocol apparently with another agent B, then B has also been running the protocol apparently with A. This gives a stronger claim about the ISSUER running with a PLATFORM.

Lowe's non-injective agreement is a stronger authentication property than weak agreement; it adds a further condition to ensure that the two agents, A and B, agree on the roles they are taking and agree on the data items used in their message exchange. In the ECC-DAA scheme non-injective agreement would guarantee that the ISSUER and a PLATFORM both agree upon the completion of a JOIN operation. This means that in the JOIN operation, the contents of the received messages correspond to the sent messages for the specific JOIN session.

Lowe in [41] also defines *injective agreement* as an authentication property. Injective agreement adds a further constraint on top of non-injective agreement, which is that there is a unique matching partner run for each completed run of an agent. The idea of injective agreement is to prevent relay attacks.

We capture these properties in lemmas as $G4$, $G5$, $G6$, $G7$, $G8$ and $G9$ respectively. Note that weak agreement and non-injective agreement are captured two lemmas respectively. The first variant of the lemmas enables us to prove authentication if any key reveal has happened and if keys have been revealed then the lemma hold vacuously. The second variant of the lemmas guarantees that both the PSIGNER and ISSUER are honest when a PSIGNER has authenticated to an ISSUER (i.e. they have completed the JOIN operation) but all other entities may have leaked their keys by that point.

$G10$ *Secrecy in* JOIN: The lemma is used to prove secrecy of a credential `cre` in the ECC-DAA JOIN operation. The intuition for this lemma is that the ISSUER and a PLATFORM have established a shared secret, since the `cre` is sent to a PLATFORM encrypted under its public endorsement key. We model this by the adversary not knowing the `cre`, unless any of the involved parties keys have been revealed. This means that the ISSUER's key has been revealed, or the endorsement key reveal of the used PLATFORM, or the `tsk` key reveal of the used PLATFORM. This property is captured by our lemma `secrecy_cre`.

C. Privacy Properties

The purpose of this section is to capture the user-controlled anonymity, and user-controlled tracability properties of ECC-DAA by Brickell et al. in [12].

$G11$ and $G14$ *User-controlled Anonymity*: User-controlled anonymity in [12] requires two properties to hold. Firstly *anonymity*, it is hard to recover the identity of the signer from its signature unless its secret key is known. Secondly *user- controlled unlinkability*, ensures that an adversary cannot tell if the signatures were produced by one or two PSIGNERS.

Within TAMARIN unlinkability is established via observational equivalence ($G14$). The intuition behind this is that the adversary is given two signatures from one signer, or one each from two signers. However, the adversary is unable to distinguish between these two instances. To encode this in TAMARIN requires augmenting our model to generate two signatures from one PLATFORM, or two signatures from two different PLATFORMS. It is possible to express this using TAMARIN rules in the existing models but the resultant state space was too large for exploration. Therefore, we developed a third model, Model C, to support this analysis which simplified the model of the JOIN operation but did not reduce the adversary's capability as a result of the simplification. With this revised model observational equivalence holds applying *diff* on two signatures, and hence unlinkability is established.

As stated above anonymity should not be broken unless the key is leaked. Therefore, the unlinkability above already covers the case that anonymity cannot be broken when the key is not revealed. Model C does not permit a `tsk` key to be revealed and since two signatures cannot be distinguished to be from the same or different PSIGNERS then this undesirable behaviour is not possible. In models A and B which do allow for a `tsk` reveal,

we formulate an additional lemma, `can_be_deanonymised` ($G11$). It states that, if the `tsk` is known to the adversary then the identity of the PLATFORM is revealed and therefore anonymity is broken. Thus $G14$ and $G11$ together address user-controlled anonymity.

$G12-13$ *User-controlled Traceability*: The user-controlled traceability in [12] requires two properties to hold; the first property is *unforgeability* and relies on the perfect cryptography assumption; while the second one is *user-controlled linkability*.

In our ECC-DAA model we do not need to capture the first property as a lemma because the symbolic method relies on cryptography being perfect. The second property means that if two ECC-DAA signatures are computed with the same `bsn`, *i.e.*, the J values are equal, then signatures are linkable as coming from the same PLATFORM. Alternatively, if the ECC-DAA signatures are computed with two different `bsn`s and hence the J values are different then the signatures are unlinkable. The LINK operation in the ECC-DAA scheme is used to ascertain whether signatures are linkable or not.

We encode two lemmas to determine whether two signatures are linkable. In Model A we define the lemma `user_controlled_independent_link_tokens` ($G12$). The intuition for this lemma is that the corresponding link tokens of two ECC-DAA signatures are different since the J's are unique. The action `CompareLinkTokens` appears in a trace only after two signatures have been constructed.

In Model B we expect to establish linkability. We define a lemma `user_controlled_linkability` ($G13$) to capture linkability. It is expressed as the contrapositive to the way linkability is expressed in [12]. The lemma states that if two ECC-DAA signatures are computed by the same PLATFORM with the fixed J then the LINK operation would yield linkability and hence the K's would be equal.

In $G13$ we only consider traces for a single PLATFORM. If the traces included two PLATFORMs and each created a signature with a fixed J we would not be able to establish linkability since the K values within the signatures would be different. Note that anonymity within this variant of the model does not hold since linkability is present.

Our model allows us to create two signatures explicitly within a trace and therefore, linkability can be expressed as a trace property as discussed. This means we do not have to encode user-controlled traceability via observational equivalence.

V. ANALYSIS AND RESULTS

In this section we review our analysis of our ECC-DAA model and present both a weakness in the SIGN operation and an attack on the JOIN operation.

Clarifying parts of the ECC-DAA operations which were not stated in I-MECH4, *e.g.*, discovering the source of the basename used in the SIGN operation, was a significant task even before the TAMARIN modelling analysis. Identifying the appropriate modelling abstractions and restrictions for the analysis was necessary because TAMARIN would not have been able to support proving the model otherwise. For example, to encode our `calcR1` equation requires 34 operators that comprises 49 terms on the LHS of the equation with the abstraction of the `s` term. Similarly for `calcR2` with the abstraction the equation requires 40 operators and 54 terms, whereas without the abstraction it requires 67 operators and 94 terms.

Determining how to express the lemmas to capture the security and privacy properties was also an iterative process in order to ensure that they clearly mapped to those in [12].

We analysed our models on the following machine specification: Intel i7-7600U (2 cores) @ 2.80GHz and 16GB RAM using TAMARIN version 1.5.0 [42]. The model itself is reasonably efficient on memory and consumes 3GB of RAM in the course of a proof, however a great deal of the processor resources is required to perform the proof. For the models presented in this paper, the proofs and disproofs for Model A (which focuses on user-controlled untraceability) takes 3 minutes, and Model B (which focuses on user-controlled traceability) takes 10 minutes to verify. Figure 3 summarises the results of all of the properties of our ECC-DAA models.

The results affirm that our formal model of the ECC-DAA scheme meets all the functional correctness properties in [12] ($G1$ to $G3$). It also demonstrates that standard authentication properties are met when all the participants are honest and there are no key reveals and that the highest level of authentication that can be achieved is non-injective agreement ($G4$, $G5$ and $G7$ hold). When keys are revealed the highest level of authentication that can be achieved is only *aliveness* ($G4$) of a PSIGNER whenever the ISSUER completes the run of a JOIN operation, apparently with a PSIGNER that has previously been running the JOIN operation.

The privacy analysis also demonstrates that it is not possible to link two ECC-DAA signatures when the J's are fresh ($G12$). Conversely, the analysis shows that it is possible to link two ECC-DAA signatures when the J are fixed ($G13$). $G12$ vacuously holds in Model B and $G13$ vacuously holds in Model A. $G11$ also holds in both Models.

Notably, Models A and B are auto-provable using TAMARIN's default heuristics, and this was invaluable when changes to the model occurred as it allowed all the ECC-DAA operations to be re-proved quickly.

The analysis of $G14$ established unlinkability but proving this using Model C required a guided proof.

Our analysis of functional correctness of the SIGN operation in the presence of an adversary indicates that I-MECH4 may not be protected from honest sender starvation and more interestingly resource exhaustion. Honest sender starvation is expected in a DY setting where the adversary can block or modify a message sent by a VERIFIER, therefore the VERIFIER would never receive a response to its message. Resource exhaustion is where the signer expends effort to continually sign messages since an adversary can submit its own message m' and n'_V to a signer and it will produce a valid ECC-DAA signature. The production of such a signature is a costly operation.

Moreover, our model revealed that I-MECH4 is vulnerable to a man-in-the-middle attack and an attack on secrecy, when the security of any PSIGNER secret endorsement key

is compromised. This is the first symbolic analysis to highlight these attacks for an ECC-DAA scheme. Therefore the lemmas `weak_agreement`, `ni_agreement`, and `secrecy_cre` do not hold when one or more secret endorsement key is compromised and revealed to the adversary.

Note also that injective agreement does not hold since the encrypted message that communicates the generated credential to a PLATFORM can be replayed by the adversary. This is because there is no freshness in the message communication from the ISSUER in the JOIN operation. Hence, this violates the requirement of a unique running session in order to establish injective agreement.

A. Man-in-the-middle and Secrecy Attack and Fix

Our TAMARIN analysis indicates that the JOIN operation (Figure 1a) cannot guarantee that the ISSUER authenticates any PLATFORM if a single PSIGNER is corrupted, but that the corrupted PLATFORM is still regarded by the ISSUER as being honest. Additionally, in the JOIN operation our analysis indicates that secrecy of any PLATFORM's `cre` cannot be guaranteed, if a single PSIGNER is corrupted. These attacks were found by a man-in-the-middle attack when performing authentication analysis in the context of an adversary revealing a PSIGNER secret endorsement key ($G6$ and $G8$), and when performing secrecy analysis in the context of another PSIGNER's secret endorsement key reveal ($G10$). Therefore, the security of I-MECH4 relies heavily on the integrity of *all* PSIGNERS.

Both attacks require an ISSUER and two PLATFORMS, of which one PLATFORM is corrupted by the adversary. The attack on authentication shows that the ISSUER believes it has authenticated with PLATFORM_A, whereas it actually authenticated with PLATFORM_B. The attack is detailed in Figure 5. Additionally, the attack on secrecy shows that the ISSUER believes it has established a shared secret, `cre`, with PLATFORM_A, whereas the secret is shared with PLATFORM_B and it is known by the adversary. Our work affirms that these attacks are possible in the standardised ECC-DAA scheme even when considering the recommended ways of establishing a secure and authentic channel (I-MECH4).

The following details the steps of the attack, shown in Figure 5, on authentication for an ISSUER and two PLATFORMS one of which is corrupt:

1) For some PSIGNER the secret endorsement key ($sk_{ek_{ps}A}$) is compromised, modelled by the DY revealing this key. We refer to this entity as PLATFORM_A.
2) There has been a honest PLATFORM and ISSUER SETUP, we refer to these as PLATFORM_B and ISSUER respectively. Note that the ISSUER is unaware that the unrelated $sk_{ek_{ps}A}$ has been leaked.
3) The ISSUER sends out a JOIN request, JOIN_A, containing n_{IA} and MAC key km_A encrypted under the public endorsement key $pk_{ek_{ps}A}$ for PLATFORM_A.
4) The DY intercepts the JOIN_A request and with knowledge of $sk_{ek_{ps}A}$ decrypts the message and gains knowledge of the nonce n_{IA} and MAC key km_A. The DY then encrypts n_{IA} and km_A under PLATFORM_B's public endorsement key $pk_{ek_{ps}B}$.
5) PLATFORM_B receives JOIN_A and forwards JOIN_A to its PSIGNER. The PSIGNER follows the command to produce $Q2_B$, v_B, w_B and γ_B, and returns it to PLATFORM_B.
6) This is the key step to the attack: the ISSUER receives and validates the response for JOIN_A that was performed by PLATFORM_B. The ISSUER then continues to create the `cre` elements $< A_A, B_A, C_A >$ and encrypts it under $pk_{ek_{ps}A}$ and sends the message, $\{| < A_A, B_A, C_A > |\}_{pk_{ek_{ps}A}}$, out on the network.
7) The DY intercepts the `cre` message encrypted under $pk_{ek_{ps}A}$ and decrypts it with knowledge of $sk_{ek_{ps}A}$ to retrieve $< A_A, B_A, C_A >$. The DY then re-encrypts this under $pk_{ek_{ps}B}$ and forwards the message to ASIGNER_B.
8) PLATFORM_B receives $\{| < A_A, B_A, C_A > |\}_{pk_{ek_{ps}B}}$ and forwards to its PSIGNER. The PSIGNER then creates its part of the `cre` D_B and returns $< A_A, B_A, C_A, D_B >$ to the ASIGNER. The ASIGNER then verifies this as a valid credential.
9) The ISSUER believes it was running the JOIN operation session with PLATFORM_A under $< A_A, B_A, C_A >$, when actually PLATFORM_B was committed to the JOIN operation session with its self constructed D_B.

The attack we identified is similar to one described by Backes et al. in [16] on the pre-standardised RSA-DAA scheme. They include the identity of a joining TPM in the zero-knowledge proof as a fix to the attack. Chen et al. in [23] proposed an alternative anonymous authentication to DAA which was also susceptible to a comparable attack for a compromised TPM, referred to as the Chosen Compromised TPM attack. For their privacy-preserving Certificate Authority protocol they similarly suggested that this type of attack could be removed by including the public endorsement key of a TPM in the JOIN operation.

We also propose to include the PSIGNER's identity using its public endorsement key in the proof of knowledge v as a fix to both attacks. This solution does not require any change in the overall functionality of I-MECH4. Therefore $v \leftarrow H_2(P_1||Q_2||U||pk_I||n_I)$ in PS JOIN One in Figure 1a is amended to $v \leftarrow H_2(pk_{ek_{ps}}||P_1||Q_2||U||pk_I||n_I)$ and v' in ISSUER JOIN Two is similarly amended.

We extended our model to capture this fix by modifying the equation $calcU$ to encapsulate the public endorsement key within v. Additionally, we amended the two rules `PS_JOIN_ONE` and `ISSUER_JOIN_TWO` to represent v and v' respectively.

With this fix the v_B in step 5 of the attack would additionally contain $pk_{ek_{ps}B}$ for PLATFORM_B. Therefore, step 6 of the attack would not be possible since the ISSUER would produce a v' based on its knowledge of $pk_{ek_{ps}A}$. Consequently v_B and v'_A computed by the ISSUER would not be equal and the JOIN operation would abort. The TAMARIN theory for this fix can be found in [24] and verifies the effectiveness of the change.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we present the development of a fine-grained symbolic model of the ISO/IEC 20008-2:2013 Mechanism 4

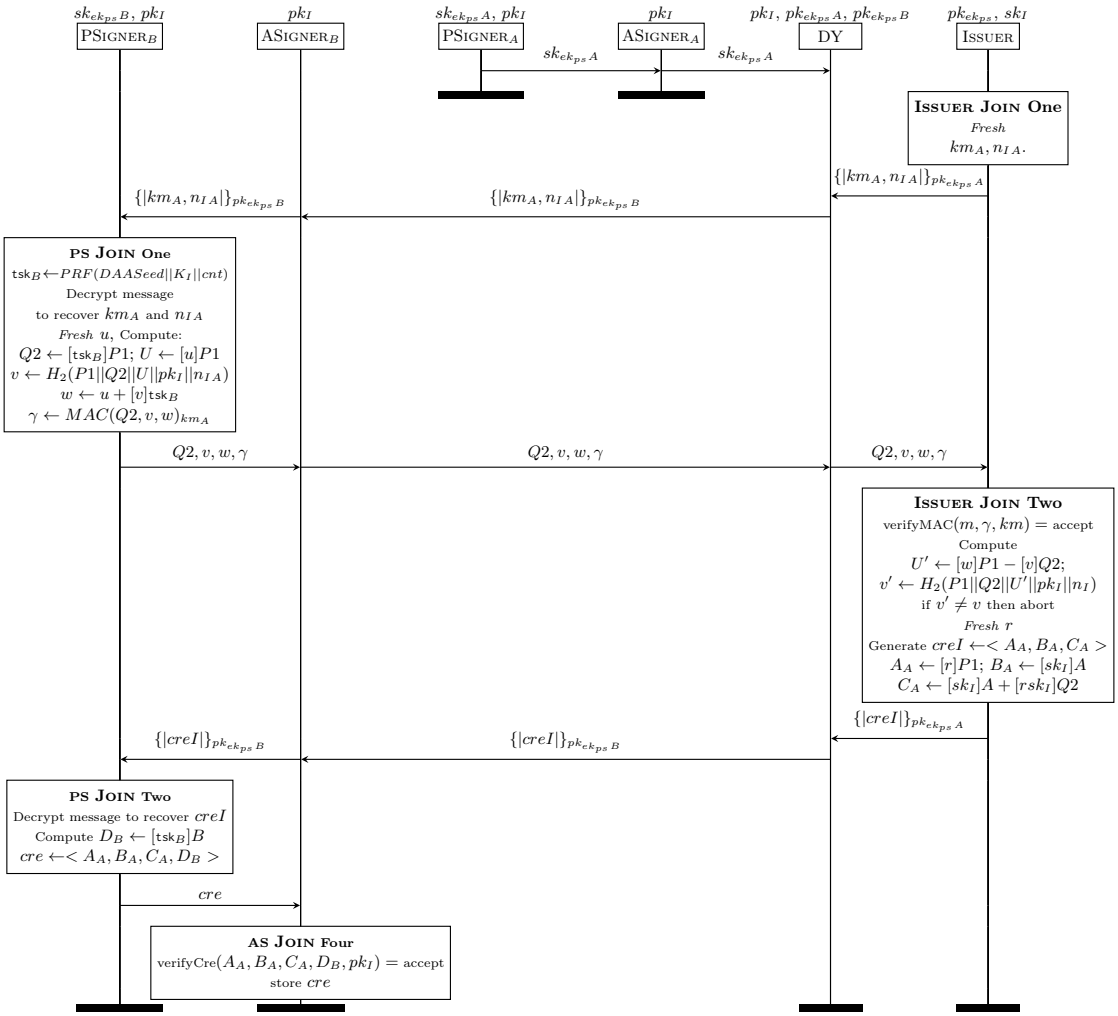


Fig. 5: Man-in-the-middle attack in the JOIN operation

standard together with the proposed implementations mechanisms for a secure and authentic channel [10] [33]. The paper also makes a contribution to how complex zero-knowledge proofs can be captured for symbolic reasoning. This involved defining an appropriate equational theory to represent the mathematical properties of the underlying cryptographic concepts in the protocol. Our approach could similarly be used in other schemes. The model contains lemmas to capture all the correctness, security and privacy properties required by the ISO/IEC 20008-2:2013 Mechanism 4. Even though our TAMARIN model employed the secure and authentic channel as recommended in the standard, we identified an attack using TAMARIN. The attack reveals a fundamental issue with the scheme, *i.e.*, if a single PSIGNER is compromised then no PSIGNER can be authenticated reliably. The attack is similar to that reported by Backes et al. for the RSA-DAA scheme and therefore highlights that the weakness is still present in ECC-DAA. Backes et al. proposed a fix for the RSA-DAA scheme and verified associated security properties. Our fix follows a similar embedding of the public endorsement key of a TPM in the proof-of-knowledge which provides the basis for establishing all security and privacy properties.

Our model can be used for future ECC-DAA formal analysis. Through the modelling process we gained valuable insights. The main lesson learned is the approach we applied for representing the mathematical equations involved in protocol using TAMARIN's equational theory. While TAMARIN's equational theory does not perform the mathematical operations involved in the protocol, it does provide a level of abstraction which maps the mathematical formulae directly to the corresponding TAMARIN syntax. This allows for our TAMARIN model to be closer to the implementation detail than previously verified models. We believe this is a useful modelling style that we and others will be able to apply in future modelling of complex protocols.

The TCG, which is an industrial standards body and the developer of TPM specifications, is continuously working on improvement of the TPM technology. Since ISO/IEC 20008-2 was published in 2013, several modifications on the ECC-DAA TPM API in the TPM 2.0 specification have been made and adopted by ISO/IEC as another international standard, ISO/IEC 11889:2015. These modifications are due to attacks found

against early versions of the ECC-DAA TPM API on TPM 2.0 [43], [44]. Acar et al. in [43] demonstrated that the API for the TPM allow an adversary to use a TPM as a static Diffie-Hellman (DH) oracle. Brown and Gallant in [45] found that although solving a static DH problem is still computationally infeasible but is simpler to solve than the computational DH problem. The modifications to the ECC-DAA scheme are as follows:

- Instead of receiving the elliptic curve point $J \leftarrow H_1(bsn)$, the TPM receives two values s_2, y_2, where $J = (x_2, y_2)$ and $x_2 = H(s_2)$ for a hash function H. The computation of this hash function can avoid that the TPM is used as a static DH oracle.
- The operation of receiving the elliptic curve point B and computing the point D by the TPM is removed for the same reason of avoiding to make the TPM as a static DH oracle. The replacement is that the Issuer computes the D value and provides a Schnorr signature, σ, to prove that the computation is correct. As a result, the DAA credential is (A, B, C, D, σ) instead of (A, B, C).

With these modifications, the ECC-DAA implementation in the current version of the TPM 2.0 specification is not compatible with ISO/IEC 20008-2:2013 mechanism 4. Extending our models to include these modifications will then provide a formal analysis of the ISO/IEC 11889:2015 for ECC-DAA implementations.

REFERENCES

[1] Trusted Computing Group, "Trusted Platform Module (TPM) | Trusted Computing Group (TPM)," https://trustedcomputinggroup.org/work-groups/trusted-platform-module/ [Online; accessed 24-October-2018].

[2] "Information technology — TPM Library — Part 1: Architecture," International Organization for Standardization, Geneva, CH, Standard, 2015.

[3] ARM, "TrustZone - ARM," https://www.arm.com/products/silicon-ip-security [Online; accessed 24-October-2018].

[4] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution," in *Hardware and Architectural Support for Security and Privacy HASP*, 2013.

[5] E. F. Brickell, J. Camenisch, and L. Chen, "Direct Anonymous Attestation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004*, 2004, pp. 132–145.

[6] E. Brickell and J. Li, "Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities," *IEEE Trans. Dependable Sec. Comput.*, vol. 9, no. 3, pp. 345–360, 2012. [Online]. Available: https://doi.org/10.1109/TDSC.2011.63

[7] Trusted Computing Group, "Trusted Computing," https://trustedcomputinggroup.org/trusted-computing/ [Online; accessed 24-October-2018].

[8] E. Brickell, L. Chen, and J. Li, "A new direct anonymous attestation scheme from bilinear maps," *Trusted Computing-Challenges and Applications*, 2008.

[9] L. Chen, "A DAA scheme requiring less TPM resources." in *Inscrypt*, 2009.

[10] L. Chen, D. Page, and N. P. Smart, "On the design and implementation of an efficient DAA scheme," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2010, pp. 223–237.

[11] "Information technology - Security techniques - Anonymous digital Signatures Part 2: Mechanisms using a group public key," International Organization for Standardization, Geneva, CH, Standard, 2013.

[12] E. Brickell, L. Chen, and J. Li, "Simplified security notions of direct anonymous attestation and a concrete scheme from pairings," *Int. J. Inf. Sec.*, 2009.

[13] L. Chen, "A DAA Scheme Requiring Less TPM Resources," in *Information Security and Cryptology - 5th International Conference, Inscrypt 2009, Beijing, China, December 12-15, 2009. Revised Selected Papers*, 2009, pp. 350–365.

[14] J. Camenisch, M. Drijvers, and A. Lehmann, "Universally Composable Direct Anonymous Attestation," in *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II*, 2016, pp. 234–264.

[15] B. Blanchet, "Automatic Verification of Security Protocols in the Symbolic Model: the Verifier ProVerif," in *Foundations of Security Analysis and Design VII, FOSAD Tutorial Lectures*, ser. Lecture Notes on Computer Science, A. Aldini, J. Lopez, and F. Martinelli, Eds. Springer Verlag, 2014, vol. 8604, pp. 54–87.

[16] M. Backes, M. Maffei, and D. Unruh, "Zero-Knowledge in the Applied Pi-calculus and Automated Verification of the Direct Anonymous Attestation Protocol," in *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, 2008, pp. 202–215.

[17] B. Smyth, M. Ryan, and L. Chen, "Direct Anonymous Attestation (DAA): Ensuring Privacy with Corrupt Administrators," in *Security and Privacy in Ad-hoc and Sensor Networks, 4th European Workshop, ESAS 2007, Cambridge, UK, July 2-3, 2007, Proceedings*, 2007, pp. 218–231.

[18] ——, "Formal Analysis of Anonymity in ECC-Based Direct Anonymous Attestation Schemes," in *Formal Aspects of Security and Trust - 8th International Workshop, FAST 2011, Leuven, Belgium, September 12-14, 2011. Revised Selected Papers*, 2011, pp. 245–262.

[19] B. Smyth, M. D. Ryan, and L. Chen, "Formal analysis of privacy in Direct Anonymous Attestation schemes," *Sci. Comput. Program.*, vol. 111, pp. 300–317, 2015. [Online]. Available: https://doi.org/10.1016/j.scico.2015.04.004

[20] J. Camenisch, L. Chen, M. Drijvers, A. Lehmann, D. Novick, and R. Urian, "One TPM to Bind Them All: Fixing TPM 2.0 for Provably Secure Anonymous Attestation," in *2017 IEEE Symposium on Security and Privacy, SP*.

[21] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," in *Computer Aided Verification, CAV 2013*.

[22] L. Chen and B. Warinschi, "Security of the TCG Privacy-CA Solution," in *IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing, EUC 2010, Hong Kong, China, 11-13 December 2010*, 2010, pp. 609–616. [Online]. Available: https://doi.org/10.1109/EUC.2010.98

[23] L. Chen, M. Lee, and B. Warinschi, "Security of the Enhanced TCG Privacy-CA Solution," in *Trustworthy Global Computing - 6th International Symposium, TGC 2011, Aachen, Germany, June 9-10, 2011. Revised Selected Papers*, 2011, pp. 121–141. [Online]. Available: https://doi.org/10.1007/978-3-642-30065-3_8

[24] "ECC-DAA Tamarin Source," https://github.com/tamarin-prover/tamarin-prover/tree/develop/examples/eurosp19-eccDAA [Online; accessed 15-March-2019].

[25] "Specification Version 1.2. Revision 85," Trusted Computing Group, Standard, 2016.

[26] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A Comprehensive Symbolic Analysis of TLS 1.3," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, 2017, pp. 1773–1788.

[27] A. Bruni, E. Drewsen, and C. Schürmann, "Towards a Mechanized Proof of Selene Receipt-Freeness and Vote-Privacy," in *Electronic Voting - Second International Joint Conference, E-Vote-ID 2017, Bregenz, Austria, October 24-27, 2017, Proceedings*, 2017, pp. 110–126.

[28] D. A. Basin, C. J. F. Cremers, T. H. Kim, A. Perrig, R. Sasse, and P. Szalachowski, "ARPKI: Attack Resilient Public-Key Infrastructure," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 2014, pp. 382–393.

[29] J. Whitefield, L. Chen, F. Kargl, A. Paverd, S. Schneider, H. Treharne, and S. Wesemeyer, "Formal Analysis of V2X Revocation Protocols," in *Security and Trust Management - 13th International Workshop, STM 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, 2017, pp. 147–163.

[30] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A Practical and Provably Secure Coalition-Resistant Group Signature Scheme," in *Advances in Cryptology - CRYPTO 2000, 20th Annual International*

Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings, 2000, pp. 255–270.

[31] M. Bellare, J. A. Garay, and T. Rabin, "Fast Batch Verification for Modular Exponentiation and Digital Signatures," in *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, 1998, pp. 236–250.

[32] J. Camenisch and M. Stadler, "Efficient Group Signature Schemes for Large Groups (Extended Abstract)," in *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, 1997, pp. 410–424.

[33] "Part 1: Architecture Family "2.0"," Trusted Computing Group, Standard, 2016.

[34] B. Schmidt, S. Meier, C. J. F. Cremers, and D. A. Basin, "Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties," in *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, 2012, pp. 78–94. [Online]. Available: https://doi.org/10.1109/CSF.2012.25

[35] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Information Theory*, 1983.

[36] J. Dreier, C. Duménil, S. Kremer, and R. Sasse, "Beyond Subterm-Convergent Equational Theories in Automated Verification of Stateful Protocols," in *Maffei M., Ryan M. (eds) Principles of Security and Trust. POST 2017. Lecture Notes in Computer Science, vol 10204.*, 2017, pp. 117–140.

[37] H. Comon-Lundh and S. Delaune, "The Finite Variant Property: How to Get Rid of Some Algebraic Properties," in *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings*, 2005, pp. 294–307. [Online]. Available: https://doi.org/10.1007/978-3-540-32033-3_22

[38] "Tamarin prover website," https://tamarin-prover.github.io/ [Online; accessed 24-October-2018].

[39] D. Basin, J. Dreier, and R. Sasse, "Automated Symbolic Proofs of Observational Equivalence," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 1144–1155. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813662

[40] D. A. Basin, S. Radomirovic, and M. Schläpfer, "A Complete Characterization of Secure Human-Server Communication," in *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, 2015, pp. 199–213. [Online]. Available: http://dx.doi.org/10.1109/CSF.2015.21

[41] G. Lowe, "A Hierarchy of Authentication Specification," in *10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA*, 1997, pp. 31–44.

[42] "Github: Tamarin Prover Version 1.5.0 (develop)," https://github.com/tamarin-prover/tamarin-prover/commit/44d5ecbc2097ee99a22a01876e445047f2a31c54 [Online; accessed 01-November-2018].

[43] T. Acar, L. Nguyen, and G. Zaverucha, "A TPM Diffie-Hellman Oracle," Cryptology ePrint Archive, Report 2013/667, 2013, https://eprint.iacr.org/2013/667.

[44] L. Xi, K. Yang, Z. Zhang, and D. Feng, "DAA-Related APIs in TPM 2.0 Revisited," in *Trust and Trustworthy Computing*, T. Holz and S. Ioannidis, Eds. Cham: Springer International Publishing, 2014, pp. 1–18.

[45] D. R. L. Brown and R. P. Gallant, "The Static Diffie-Hellman Problem," Cryptology ePrint Archive, Report 2004/306, 2004, https://eprint.iacr.org/2004/306.

APPENDIX A

The equational theory detailed in Section III-B demonstrates how the reductions are modelled in TAMARIN. Here we demonstrate how the equation for the calculation of U' is encoded within TAMARIN:

```
1  // U' = [w]P1 - [v]Q2
2  calcU( minus(
3    multp(
4   plus( u, multp(
5       H2( P1, multp( P1, PRF( DAASeed,
6            Ki, cnt ) ),
7        U( u, P1 ), pk( isk ),
8        ni ),
9        PRF( DAASeed, Ki, cnt ) ) ),
10  P1 ),
11   multp(
12    H2( P1, multp( P1, PRF( DAASeed, Ki, cnt ) ),
13   U( u, P1 ), pk( isk ), ni ),
14  multp( P1, PRF( DAASeed, Ki, cnt ) ) )
15 ) ) = U( u, P1 ) // U = [u]P1
16
```

Lines 3 to 10 represents the symbolic form $[w]P_1$ and embedded within that from lines 4 to 8 is the representation of $u+[v]tsk$. The encoding for the equations `calcR1` and `calcR2` can be seen in our models [24] and follows a similar pattern to the equation above.

APPENDIX B

The lemmas detailed in Section IV-B addressed functional correctness and authentication. All the lemmas are available in our TAMARIN scripts [24]. An example of a TAMARIN functional correctness lemma is given as follows:

```
(G1) lemma functional_correctness_group_verification
    : exists-trace
"Ex V V1 nv #i #j .
Send( V, nv ) @ i & Confirm( V1, nv ) @ j
& not( V = V1 )"
```

The `weak_agreement` lemma is an example of an authentication lemma:

```
(G6) lemma weak_agreement:
"All a b n #i .
Commit( a, b, n ) @ i
==> ( Ex n2 #j . Running( b, a, n2 ) @ j )
| ( Ex C #r. IssuerKeyReveal( C ) @ r
& Honest( C ) @ i )
| ( Ex C #r . RevealEK( C ) @ r
& Honest( C ) @ i )
| ( Ex C #r . RevealTsk( C ) @ r
& Honest( C ) @ i )"
```

The lemma states that whenever a commit action `Commit(a,b,n)` occurs at time `i`, then either this is a conclusion of a valid protocol run or an agent claimed to be honest at time `i` has been compromised at time `r`. This is the key lemma whose failure indicates the attack identified in Section V-A.