ADEM: An Authentic Digital EMblem

Felix Linker
flinker@inf.ethz.ch
ETH Zurich
Department of Computer Science
Zurich, Switzerland

David Basin
basin@inf.ethz.ch
ETH Zurich
Department of Computer Science
Zurich, Switzerland

ABSTRACT

In times of armed conflict, the emblems of the red cross, red crescent, and red crystal are used to mark *physical* infrastructure. This enables military units to identify assets as protected under international humanitarian law to avoid attacking them. In this paper, we tackle the novel security problem of how to extend such protection to *digital*, network-connected infrastructure through a digital emblem. A digital emblem has a unique combination of security requirements, namely, authentication, accountability, and a property that we call *covert inspection*. Covert inspection states that those wishing to authenticate assets as protected must be able to do so without revealing that they may attack unprotected entities.

In this paper, we (i) define the requirements of a digital emblem, emphasizing security requirements, (ii) present *ADEM*, a decentralized design that implements a digital emblem analogous to the physical emblems of the red cross, crescent, and crystal, and (iii) provide a comprehensive threat model and analysis that ADEM achieves strong security guarantees against an active network adversary.

In addition to our security analysis, ADEM was also evaluated in a series of domain expert meetings at the invitation of the International Committee of the Red Cross. We report on the feedback we received, which supports our thesis that ADEM is not just theoretically interesting but practically relevant to limit attacks on protected parties in cyberspace.

CCS CONCEPTS

• **Security and privacy** → *Security requirements*; *Formal security models*; *Logic and verification*; **Authentication**; *Security protocols*.

KEYWORDS

Accountability; Authentication; Formal Analysis; Security Requirements; Protocol Design

ACM Reference Format:
Felix Linker and David Basin. 2023. ADEM: An Authentic Digital EMblem. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3616578

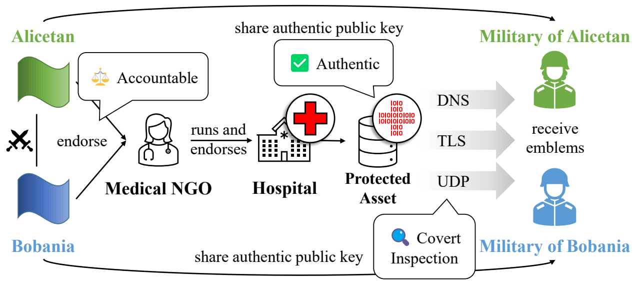

Figure 1: Overview of ADEM. The nation states Alicetan and Bobania endorse a protected party as legitimate, which in turn endorses a hospital as belonging to them. The hospital's protected asset distributes digital emblems using UDP, TLS, and DNS. The nation states' militaries can independently verify these emblems using their own nation states' public keys.

1 INTRODUCTION

In this paper, we tackle a novel security problem that arises in times of armed conflict. International humanitarian law (IHL) mandates that military units must not target medical facilities, such as hospitals. The emblems of the red cross, red crescent, and red crystal are used to mark *physical* infrastructure (e.g., by a red cross painted on a hospital's rooftop), thereby enabling military units to identify those assets as protected under IHL. The International Committee of the Red Cross (ICRC) recently [25] posed the question: How can one extend such markings to *digital* infrastructure such as servers and networks?

Extending protection to digital infrastructure raises unique security challenges.

(i) Digital emblems require authentication as assets must not be able to fake protection, for example, by transferring markings from medical to military infrastructures.
(ii) The marking of infrastructure must be designed so that one can hold misbehaving parties accountable who mark unprotected infrastructure. Protection under IHL stems from law, and laws must be enforceable to have an effect.
(iii) Those wishing to authenticate assets must be able to verify protective markings in a way that does not call attention to the fact that they are screening potential targets. We call this property *covert inspection*.
(iv) Protective markings must work in a decentralized way. What complies with IHL is subject to debate, and different parties (possibly at war with one another) must be able to express conflicting views and must not need to trust the same parties.

Standard authentication protocols are inadequate: they solve some, but not all, of these challenges. Such protocols typically require interaction between two participants and any interaction between a military unit and a potential target would reveal the unit's intention to attack that target, should it be unprotected. An unprotected target could use this knowledge to defend itself against an imminent attack, which in turn would deter military units from verifying whether their targets are protected in the first place. Moreover, typical designs providing authentication and accountability are usually centralized, e.g., in authenticated data structures such as Merkle hash trees (MHTs) [18]. Adding consensus protocols to maintain such data structures in a decentralized way does not help as they aim to establish *consensus*, which cannot be assumed in an international context.

To fill this gap, we present *Authentic Digital EMblem (ADEM)*, a design that solves all these challenges. Figure 1 provides an overview of ADEM. Digital emblems in ADEM are cryptographically signed messages and authenticate an asset as protected under IHL. Emblems are distributed actively by these assets using UDP, TLS, or DNS in a way that provides covert inspection. Emblems are accompanied by endorsements, certificate-like objects signed by independent authorities such as nation states. ADEM provides accountability through use of the Certificate Transparency (CT) log infrastructure, where parties commit to their public keys, and ADEM does not require any updates to the Internet's infrastructure.

In total, we present three contributions.

- The ICRC recently published a report that presents the idea of a digital emblem and proposes initial requirements [25]. These requirements, however, are mostly high-level and not actionable. For example, the ICRC requires that a digital emblem must be "obvious and easily visible." We take the report as a basis to precisely define the requirements for a digital emblem, emphasizing its security requirements.
- We present an *ADEM*, a design that achieves all these requirements and implements a digital emblem analogous to the physical emblems of the red cross, crescent, and crystal.
- We provide a comprehensive threat model and security analysis showing that ADEM achieves strong security guarantees against an active network adversary. This includes both the formal definition and proof of authentication and accountability using the protocol verifier Tamarin [16] and a security rationale for covert inspection. In doing so, we highlight that accountability is a subtle property, hard to root in intuition, and (perhaps not coincidentally) rarely considered in the literature.

In addition to our security analysis, ADEM was evaluated in a series of meetings with domain experts with various backgrounds, such as health care, the military, and cybersecurity, and including this paper's authors. The meetings were hosted by the ICRC in 2021, which invited the experts with the goal to provide feedback on the idea of a digital emblem in general and design proposals, such as ADEM, in particular. From these meetings, the ICRC concluded in their report to "continue research and consultation on a possible 'digital emblem', [which] will require further work on the technical development, validation and verification of possible solutions." The report lists ADEM as one of the solutions to pursue and notes that "[m]any experts considered the ADEM framework to be considered and elaborate for a 'digital emblem'" [25]. Based on this evaluation, we argue that ADEM not just works in theory but also fits the needs of protected parties and the ICRC in practice.

We proceed as follows. We start by defining the requirements of a digital emblem (Sec. 2). Afterwards, we present ADEM (Sec. 3) and analyze its security (Sec. 4). Finally, we present related work (Sec. 5) and draw conclusions (Sec. 6).

Artifacts

Section 3 details ADEM's design omitting some technical details. A full set of specifications for ADEM is hosted at https://adem-wg.github.io/adem-spec/. Note that the specification is still under active development and might change. The specification as referred to in this paper can be accessed as "Preview for branch v1.0." Our formal model and proofs, presented in Section 4.3, are available at https://github.com/adem-wg/adem-proofs/.

2 THE PROBLEM

We begin with the requirements for a digital emblem. For this, we introduce relevant legal and historical background, describe the problem domain, and finally provide a digital emblem's requirements, which follow from the ICRC's report on digital emblems [25]. However, whereas the report introduces requirements on a much higher, abstract level and without a detailed consideration of security, we present a comprehensive list of actionable, technical requirements and put security in focus.

2.1 Legal and Historical Background

The Geneva Conventions [7] and their Additional Protocols constitute the core of IHL and establish legal rules on the conduct of armed conflict. These rules codify the meaning and usage of protective emblems, namely, the red cross, crescent, and crystal, permitting protected parties (PPs) to mark their assets, such as vehicles, personnel, or buildings with these signs during armed conflicts. These signs inform other parties about an asset's affiliation with the International Red Cross and Red Crescent Movement (*indicative* use), or about an asset's protected status under IHL (*protective* use). Actors bound by IHL must respect an asset's protected status and not attack it, except in very limited circumstances.

Since 1949, the Geneva Conventions have been amended and extended. Additional Protocol I [8, 9], for example, contains additional regulations on how PPs could communicate their status using technical means, like radar transponders and radio signals. Recognizing that technology may progress rapidly, Additional Protocol I allows for the ICRC to convene state experts to review and suggest updates to the technical means by which the PPs may be identified. In order to initiate discussions among states, the ICRC proposed the idea of a digital emblem on an international stage [25] in collaboration with external experts, including the authors of this paper.

2.2 Problem Domain

We next introduce the different stakeholders and parties of a digital emblem. Figure 2 shows the five relevant kinds of actors. First and foremost are *protected parties (PPs)* who operate in areas of

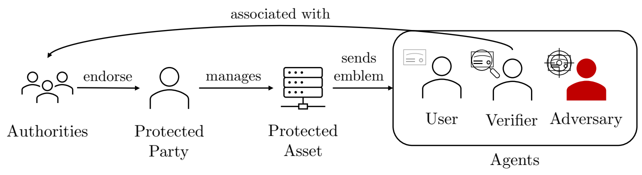

Figure 2: Parties involved in deployment and use of digital emblems.

conflict and conduct operations that enjoy protection under IHL utilizing *protected digital assets*, such as mobile devices (tablets, smartphones), servers (both virtual and dedicated), processes such as web servers, or a PP's intranet. Prior to commencing their operations, PPs must seek permission from competent *authorities*. When they are given permission, PPs can apply digital emblems to their protected assets, which *present* them to three types of *agents*.

Regular users of an asset do not pay attention to the emblem, for example, when they visit a website. *Verifiers* pay attention to the emblem and only wish to attack lawful (under IHL) targets. They are typically part of an armed force engaged in a conflict. We can assume that most verifiers (typically military units) will be associated with an authority (typically their nation state or an ally), and, hence, that such verifiers can obtain the authentic public keys of their affiliated authorities through secure, out-of-band channels. *Adversaries* are willing to violate IHL and search to abuse digital emblems. For example, they may seek to issue emblems to non-protected assets.

2.3 Requirements

The purpose of a digital emblem is to prevent attacks on protected assets by informing other parties about their status under IHL.[1] Therefore, a digital emblem can only prevent attacks by those who respect it. There are various reasons why one may respect IHL. Nation states or independent conflict parties may respect IHL because they are bound by it and, consequently, attacking protected assets can be a war crime. Verifiers not bound by IHL, e.g., cybercriminals, may also want to not target protected assets to avoid unwanted attention or because they respect the humanitarian cause. All verifiers, though, intend to attack assets not protected under IHL and, therefore, verifiers must be willing to pay attention to digital emblems for them to have an effect. This has important implications on our problem statement.

In particular, verifiers never want to reveal themselves as inspecting emblems. If verifiers were to reveal this, their targets could use that knowledge to protect themselves against an imminent attack, and verifiers would not inspect emblems. The ICRC lists two related requirements. First, it requires that "probing for a 'digital emblem' must not identify a cyber operator as a potential threat actor." Second, it requires that emblems must be "obvious and easily visible" [25]. We cover both notions as the positive security requirement *covert inspection* below. In particular, covert inspection implies that emblem presentation must be *active*, i.e., verifiers will be sent emblems and need not query them.

[1]Although IHL also permits the indicative use of an emblem, we subsume this case under the protective use of an emblem in the remainder of this paper.

The ICRC requires that a digital emblem must be "trustworthy" and that "cyber operators [must be] able to verify [an emblem's] validity" [25]. We understand "trustworthiness" as meaning that a digital emblem must be *authentic*, i.e., a digital emblem only marks assets that *truly enjoy IHL protection*. If it were not authentic, verifiers would risk that their lawful, i.e., unprotected, targets could avert attacks by displaying fraudulent emblems, and verifiers would again not inspect emblems.

Finally, the ICRC states that a digital emblem should fit into the existing framework of IHL. Compliance with IHL facilitates the diplomatic process of adopting a digital emblem, possibly integrating it into Additional Protocol I. However, should a digital emblem be codified in law, it is crucial that the illicit display of digital emblems can be prosecuted. Hence, a digital emblem must provide *accountability*, a property that was not considered in the ICRC's report.

To summarize, we find that a digital emblem must meet the following three *security requirements* (SR).

- **SR1** A digital emblem must be presented actively in a way that supports *covert inspection*: Agents who wish to verify whether an asset is protected under IHL must be indistinguishable from agents who interact with that asset for other purposes.
- **SR2** Emblems must be *verifiably authentic*. Agents must be able to correctly associate emblems to the issuing PP and the respectively marked assets.
- **SR3** A digital emblem must provide *accountability*. Independent parties must be able to identify parties that issued fraudulent digital emblems.

A digital emblem that integrates into existing IHL also has *functional requirements* (FR). First, compliance with existing IHL implies that there must be no fixed set of authorities that can regulate how a digital emblem is used. Second, and more broadly, this means that a digital emblem should work similarly to physical emblems. Just as a flag with a red cross can be affixed to and removed from supplies, personnel, buildings, or vehicles, a digital emblem must be applicable to the widest possible range of use cases and digital technologies, and also be easily removable.

Finally, agents must be able to verify the presence of digital emblems. With physical emblems, unprotected assets will simply not show the flag of the red cross. Likewise, in the digital domain, these assets will not present an invalid emblem, but rather no emblem. We summarize these considerations with the following *functional requirements* (FR).

- **FR1** A digital emblem must be *decentralized*. There can neither be fixed parties in charge of distributing emblems nor a fixed set of authorities deciding who is eligible to issue emblems.
- **FR2** A digital emblem must be *widely applicable* to a broad range of devices, infrastructures, and organizations.
- **FR3** A digital emblem must be *removable*.
- **FR4** A digital emblem's *presence must be verifiable*.

3 DESIGN

In this section, we start by giving an overview of ADEM and how its design meets the requirements just identified, then we proceed to technical details, and we close with an example of ADEM in practice.[2] Note that ADEM relies, in part, on the Web public key infrastructure (PKI), which associates domain names with TLS public keys, and the CT ecosystem, which monitors issued certificates. We briefly review both systems in Appendix A.

3.1 Overview

ADEM identifies protected assets via their network address, i.e., an address (IP address or domain name) and port combination, to be as *widely applicable* as possible (**FR2**). Clearly, an address/port combination can label network connected processes such as servers. Moreover, one can protect entire networks using address prefixes or protect devices by labelling every port of an address.

Digital emblems in ADEM are cryptographically signed and mark a set of assets as protected. Assets send emblems to anyone interacting with them. This distribution mechanism supports *covert inspection* (**SR1**), that emblems are *removable* (**FR3**), and that their *presence can be verified* (**FR4**). An emblem can be verified as *authentic* (**SR2**) by verifying its signature, but naturally this requires authentic public keys.

ADEM specifies three types of public/private key pairs: *asset keys* used to sign digital emblems, *intermediate keys* used to manage key hierarchies, and *root keys* identifying organizations (authorities and PPs). As a means to obtain authentic public keys, ADEM specifies *endorsements*, which are certificates that attest a given public key as belonging to a given organization, and that this organization is believed by the endorsement's signer to be eligible to issue digital emblems. Asset keys are endorsed by intermediate keys, and intermediate keys are endorsed by root keys. The root keys of one party can additionally endorse the root keys of another party.

Verifiers can freely choose which endorsing parties they trust, which makes ADEM *decentralized* (**FR1**). We expect that most verifiers (typically military units) will only accept emblems accompanied by an endorsement from an authority they trust, like their own nation state or an ally. As we pointed out in Section 2.2, we assume that such verifiers can obtain the authentic public keys of their affiliated authorities through secure channels. Such a verification practice establishes a closed loop of trust: Military units can see that their own nation state endorsed the protected asset in question.

To provide *accountability* (**SR3**), ADEM requires organizations to commit to their root keys. The non-repudiation of digital signatures alone is necessary but not sufficient to provide accountability: a key-holder cannot repudiate signatures but could still repudiate key ownership. ADEM implements this commitment mechanism by utilizing CT logs. Organizations must publish their root keys within certificates that bind them to a central, HTTPS-enabled website identifying the organization, e.g., `https://protected-party.org`, which we call their *organization identifier (OI)*.

This commitment mechanism comes with additional benefits: (a) Parties can monitor impersonation attempts by monitoring the CT logs. (b) Parties can revoke their root keys by revoking the binding certificate. (c) Independent verifiers such as cybercriminals, who may not be able to obtain authentic root keys out-of-band, can authenticate root public keys as belonging to a domain name.

Figure 3 shows a typical ADEM setup. Authorities issue endorsements to a PP, and the PP manages an internal public key hierarchy.

[2]See p. 2 for a reference to a full set of technical specifications of ADEM.

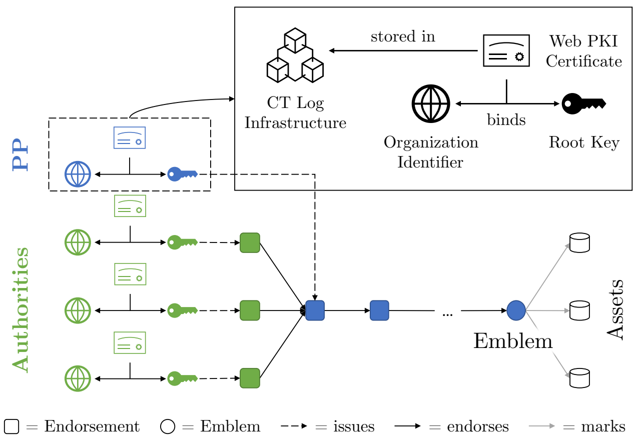

Figure 3: A typical ADEM setup. Multiple authorities (green) endorse a PP's root key. The PP (blue) maintains an internal hierarchy of public keys by signing multiple internal endorsements that ultimately endorse the emblem signing key as belonging to the PP. Parties commit to their root keys by submitting specifically encoded Web PKI certificates to CT logs, which are signed by standard certificate authorities.

Table 1: Fields of an emblem. `emb` maps to a JSON object with two keys that constrain the contexts in which the emblem may be used.

Field		Purpose
`ass`		List of protected assets (AIs)
`iss`		Emblem issuer (OI)
`iat`		Time of issuance
`nbf/exp`		Validity window
`emb`	`prp`	Purposes
	`dst`	Distribution method

The authorities and the PP are identified by a domain and a root key. Finally, the PP issues an emblem, which marks multiple assets as protected.

In the rest of this section, we explain the syntax and semantics of emblems and endorsements and how organizations publish their root keys. We also explain how protected assets actively distribute digital emblems. Finally, we explain how verifiers obtain and verify digital emblems.

3.2 Emblems

We call both emblems and endorsements *tokens*. Emblems resemble the statement: *"Asset A belongs to PP P and is protected under IHL."* Emblems encode this statement within JSON Web Signatures (JWSs) [10], a popular standard for signed attribute-value stores, using the attributes listed in Table 1.[3]

[3]In this section, we omit some technical details. We only include critical attributes, omitting those with purely technical purposes, such as version numbers. For a complete reference, see p. 2.

```
asset-identifier = address [ ":" port ]
address = [ "*." ] domain-name
        | "[" IPv6 "]"

org-identifier = "https://" domain-name
```

Figure 4: Syntax of AIs, identifying network-connected processes, and OIs, identifying organizations. `port` is an integer. Domain names and IP addresses are encoded as usual.

Table 2: Endorsement fields. `log` maps to an array of JSON objects that specify in which logs a root key-binding certificate was included. `emb` maps to a JSON object whose attributes constrain emblems signed under this endorsement. The constraints apply to the respectively named emblem attributes.

Field		Purpose
`iss`		Endorsement issuer (OI)
`sub`		Endorsed party (OI)
`key`		Endorsed public key
`nbf/exp`		Validity window
`log`	`ver`	Version of CT log
	`id`	ID of CT log
	`hash`	Certificate leaf hash
`emb`	`prp`	Purpose constraints
	`dst`	Distribution constraint
	`ass`	Asset constraints (allowlist)
	`wnd`	Max emblem lifetime

The attribute `iss` encodes an organization (*who* issues a token) and `ass` encodes assets (*what* is protected). Organizations are identified by *organization identifiers (OIs)*, and assets by *asset identifiers (AIs)*, which both closely resemble URIs [1].

Figure 4 depicts the syntax of OIs and AIs. OIs are encoded as domain names prefixed by `https://`. AIs are an address (IP address or domain name) and port combination and point to network-connected processes. For example, `https://example.com` is an OI and `example.com:8080` or `[::FFFF:93.184.216.34]:22` are AIs. As AIs permit IP address prefixes and wildcards in domain names, one AI defines a set of protected assets, more concretely, protected processes. A process reachable under the IP address *IP* and port *P* is included in the set of an AI *AI* if both (i) either the domain name encoded within *AI* resolves to *IP* or to a prefix of *IP*, or if the IP address encoded within *AI* is *IP* or a prefix of *IP*, and (ii) if *AI*'s port is equal to *P*, or *AI* does not include a port. An emblem can include multiple AIs and marks all assets that these AIs point to as protected under IHL.

3.3 Endorsements

Endorsements are also encoded as JWSs and resemble the statement: *"Party P_1 attests that public key K belongs to party P_2 and that P_2 may signal protection under IHL."* They provide a means to obtain authentic public keys for emblem verification. Additionally, endorsements may come with constraints. For example, a nation state may endorse a PP to mark its website `humanitarian.org` as protected, but nothing else. Constraints allow PPs and authorities to mitigate the consequences of private key compromises and to prevent abuse within PPs. Technically, anyone can endorse anyone else, even oneself, e.g., to manage key hierarchies within one's own organization. In practice, we expect that most party-to-party endorsements will be signed by nation states or supranational alliances like the Arab League or European Union.

Endorsements include the attributes listed in Table 2. An endorsement's constraints can specify upper bounds on an emblem's lifetime, over which channels emblems may be distributed, and what kind of assets may be labelled (listing permitted AIs). `log` lists CT logs that provide the party's root key commitment for the purpose of accountability, as we will explain in Section 4.3.3.

3.4 Root Keys

Naturally, chains of endorsements (as specified in the previous section) will terminate in some key that is not further endorsed. We call these keys *root keys*. Recall that most verifiers will be able to authenticate the root keys of at least some authorities out-of-band (see Sec. 2.2). Such verifiers can authenticate a claim of protection by verifying that it was (transitively) endorsed by a public key they trust. In designing ADEM to provide accountability, however, we do not rely on out-of-band authenticated public keys. ADEM should (and will) provide accountability also in cases where public keys cannot be authenticated by requiring parties to bind their root keys to their OI, which implements a root key commitment mechanism.

Our commitment mechanism enables verifiers to associate emblems and endorsements to domain name holders and to hold these domain name holders accountable for misbehavior. Further, this commitment mechanism enables verifiers to authenticate root keys of authorities they are not affiliated with, it allows parties to revoke root keys, and it enables organizations to detect impersonation attempts, i.e., attempts to maliciously associate public keys with their OI.

Our commitment mechanism works as follows. Organizations commit to a root key using a mechanism inspired by the concept of "self-authenticating traditional (SAT) domains" [28]. First, they calculate their root key's hash $h(pk)$ and register the subdomain "$h(pk)$`.adem-configuration.`OI", for example:

`z247co3xrah...danumgcfx7a.adem-configuration.pp.org`.

Second, they request a TLS certificate [2] that is valid for both their OI and this new subdomain. Finally, they submit this certificate to the CT logs. A root key is specified as correctly associated to an OI if there is a certificate that (i) is not revoked, (ii) properly logged in the CT infrastructure, and (iii) binds the key to the OI as described above. Note that proper CT log inclusion requires that the certificate's signature is valid, and that there is a validation path to a root certificate accepted by the log in question.

ADEM does not specify how verifiers check a certificate's revocation status. We recommend, though, to use offline verification mechanisms that are employed and maintained by modern browsers such as Mozilla Firefox [21] or Google Chrome [4]. Using these mechanisms, verifiers would regularly download a set of revoked certificates that has been assembled by a third-party (Mozilla or Google, in the examples above). Using offline certificate revocation checks, verifiers only reveal that they use the mechanism at the time they update it. We will cover revocation checks in more detail in our security analysis (Sec. 4.4).

3.5 Distribution

ADEM specifies three interfaces for token distribution. Either, tokens are actively pushed to anyone interacting with a protected asset via TLS [24] or UDP [23], or they are stored in DNS records [19, 20]. In each case, ADEM distributes emblems with the endorsements necessary to cryptographically verify an emblem as genuine.

Extending ADEM to support additional interfaces is straightforward, and there is no limit to the number of interfaces ADEM could support. However, the more interfaces ADEM supports, the greater the burden that ADEM puts on verifiers, who would need to implement detection on every interface to ensure that they are not attacking a PP.

TLS. We envision TLS to be the most popular interface to transmit tokens. Many application-level protocols, such as HTTPS, run over TLS. Moreover, using TLS provides attractive security properties as tokens cannot be dropped by an on-path attacker.

Tokens distributed over TLS are appended as a custom extension to the `NewSessionTicket` message. The `NewSessionTicket` message is intended to establish pre-shared keys for session resumption, but is dropped by clients that do not support its extensions. Thus, our TLS distribution mechanism is backwards-compatible with existing TLS clients.

UDP. To support the labeling of arbitrary network traffic, we also use UDP to distribute tokens. UDP does not guarantee reliability, so an adversary may drop UDP traffic that includes tokens. However, an emblem sent over UDP still provides authenticity and can prevent attacks when received. Whenever a protected asset receives a packet from a new network address, it sends tokens to a specified port at that address.

DNS. Finally, protected assets can be labelled using DNS TXT records, which enable associating arbitrary information with a domain name [26]. Naturally, this is only possible if the respective asset has an associated domain name. A benefit of labeling an asset via DNS is that access to the asset is not required for distribution or verification and no additional software must be deployed.

Note that DNS distribution requires explicit queries from verifiers, suggesting a possible conflict with the *covert inspection* requirement (**SR1**). We will cover this tension in our security analysis later (Sec. 4.4).

3.6 Determining Protection Status

The distribution mechanisms just presented specify how a verifier could receive digital emblems given a domain name or given an asset with a port that runs TLS. Still, verifiers require a procedure to decide whether an *arbitrary* digital asset is protected under IHL. In particular, they must be able to determine this for digital assets *not* protected under IHL, while not alerting those assets about a planned attack.

To help verifiers decide whether an asset enjoys protection, assets must distribute emblems the following way. If an asset distributes emblems via UDP, the asset must monitor all incoming requests, e.g., via firewall rules. Whenever an asset receives a request from a new IP and port combination, it must send an emblem via UDP to that address. Assets need not parse incoming packets, however, and may respond in any way they like, e.g., with a TCP RST (reset), Internet control message protocol (ICMP) error messages, or not at all. If an asset distributes emblems via TLS, it must do so with every TLS connection. Such assets must also run TLS servers on ports 443 (HTTPS) and 853 (DNS over TLS). These servers can be minimal and solely distribute emblems.

To decide whether a given asset is protected, verifiers should run a procedure that depends on the address they wish to verify. Given a domain name, they should check the domain's TXT records for emblems and resolve the domain name to IP addresses to check those for digital emblems too.

Given an IP address, verifiers can send an arbitrary packet to that address and wait for an emblem in response via UDP. If they do not receive an emblem, they should also try to establish a TLS connection to port 443 or 853 and wait for an emblem to be sent as part of a `NewSessionTicket`. Should a verifier learn an asset's domain name only while checking for an emblem, e.g., during the TLS handshake, they should additionally check these domain names for emblems stored in DNS TXT records.

3.7 Emblem Verification

After probing an asset as just described, if a verifier has received emblems and endorsements, they will next verify them. The verification procedure takes as input: a digital emblem, a set of endorsements associated with the emblem (possibly empty), and a set of trusted public keys (possibly empty). For example, if the verifier were a military unit, they might use their nation state's root public key as a trusted public key. The verification procedure returns a set of *security levels.*

First, the procedure verifies every received token's signature and checks that it is valid with respect to its lifetime. The procedure discards any token that fails this check. The procedure also checks that the emblem is valid with regard to every endorsement that passed previous checks, e.g., that the emblem's designated protected assets comply with every respective endorsement's constraints, etc.

Second, the procedure determines and returns all the emblem's security levels, which indicate whether an emblem was endorsed and by whom. The security levels are as follows.

Invalid: If the emblem was discarded as invalid during preprocessing or none of the other security levels apply.
Unsigned: The emblem does not bear a signature.
Signed: The emblem does not designate an issuer.
Organizational: All tokens designate the same issuer, all endorsements transitively endorse the emblem's verification key, and the topmost endorsement's public key is correctly configured as the PP's root key (Sec. 3.4).
Endorsed: Same as *organizational*, but additionally, there are endorsements with a different issuer than that of the emblem. All such endorsements' verification keys are correctly configured as the respective issuer's root key and endorse the emblem issuer's root key.

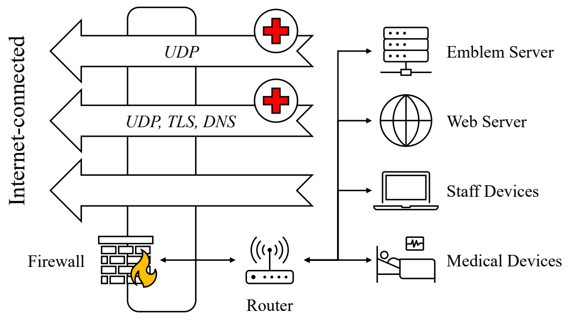

Figure 5: Imaginary Hospital Network

Signed/Organizational/Endorsed-Trusted: Same as the security levels *signed*/*organizational*/*endorsed*, but a trusted public key was used in the verification of the respective level.

The different security levels come with different security guarantees. Security levels other than endorsed were included upon request of the ICRC and increase ADEM's flexibility, e.g., enabling easy deployment in cases of emergency. PPs could announce their root or their asset public keys through other channels, allowing verifiers to authentically associate emblems to those parties even when they are only signed or organizational. In practice, though, we expect that most verifiers will only accept emblems with the security level *endorsed-trusted*.

If a verifier only wants to check if an emblem was endorsed by at least one trusted public key, they could skip the verification of correctly configured root keys, etc. Verification would then require no network calls beyond the initial probing. This leaves them vulnerable to certain kinds of attacks, e.g., they would not notice that a PP revoked its root key, but this highlights how simple verification can be. Even the full verification procedure for an endorsed emblem only requires additional network calls to check that the certificates binding a root key to an OI are correctly submitted to the CT logs. One must also check that these certificates have not been revoked. However, as we recommend using offline verification for these checks, their network-overhead is constant.

3.8 Example

We close this section with a brief example that showcases how ADEM might be deployed in practice. Consider an imaginary hospital network as depicted in Figure 5. The network is connected to the Internet via a router that also hosts a firewall, restricting outside access to certain assets. The network comprises four types of assets: network-connected medical devices, staff devices such as laptops and tablets, a web server advertising the hospital's services, and an emblem server that distributes emblems.

The medical devices do not have the technical means to distribute emblems themselves. The emblem server protectively marks these devices by regularly broadcasting a digital emblem via UDP to the local network. Such emblems serve as a defense in-depth against attackers who penetrated the hospital's network. Such an attacker may not have had a chance to inspect emblems from the outside, e.g., when they deployed malware via a malicious email attachment.

The staff's devices can monitor external traffic themselves, using firewall rules to log the traffic. Whenever they observe a packet from a new network address, they forward the address to the emblem server, which sends an emblem to that address via UDP.

Finally, the web server also marks itself as protected through all the interfaces available (TLS, UDP, and DNS). It requests emblems for itself at the emblem server each time its last emblem expired, but it need not request a new emblem for every recipient, as emblems are not recipient constrained. The server then sends those emblems to clients attempting to connect.

Now let us also imagine that the hospital was operating in a region of conflict between two countries: Alicetan and Bobania. Utilizing endorsement constraints, both countries could endorse the PP in question to issue emblems within the IP prefix that is managed by the router. Whenever the countries' militaries would receive an emblem, they could see that their own nation state endorsed the issuing PP, giving them good reason to trust that the emblem is legitimate. But also independent verifiers could take the fact that two rivaling nation states endorsed the PP as an argument to trust emblems issued by the PP. The chance that two warring countries would collude to set up a fake "PP" is negligible.

This example highlights that ADEM is flexible in that it applies to a wide variety of protected assets and network setups, and that it enables a separation of concerns: not all protected assets need to be equipped with key material, only the emblem server requires that. Moreover, ADEM's design does not require the emblem server to authenticate emblem requests.[4] The emblem server could issue emblems for the IP-range of the hospital's network, and thus they would only apply to actually protected assets.

4 SECURITY ANALYSIS

ADEM is designed to provide three security properties: authentication, accountability, and covert inspection (see Sec. 2.3). Authentication and accountability go hand in hand. Authentication expresses: "Any marked asset is protected, unless…," and accountability adds: "Whenever a marked asset is not protected, we can blame…"

We formally define and verify authentication and accountability, and model both properties and ADEM in the protocol verifier Tamarin [16].[5] In this section, we define our threat model (Sec. 4.1), introduce Tamarin (Sec. 4.2), and present the formalization of ADEM, authentication, and accountability in Tamarin, as well as our findings (Sec. 4.3). Finally, we provide a separate security argument for covert inspection (Sec. 4.4).

4.1 Threat Model

We consider an active network adversary who can inject, intercept, read, reorder, and replay any message and can corrupt any party's keys. We only constrain our adversary by the following assumptions:

(1) PPs only issue endorsements for keys under their control, and their root keys are uncompromised.

[4]For the purposes of spam prevention, it would be desirable, though, that the server authenticates requests or throttles emblem generation.
[5]Our formal model and proofs are available for download at https://github.com/adem-wg/adem-proofs.

(2) Internal PP endorsements are constrained in such a way that they only permit the issuance of emblems for protected infrastructure.
(3) PPs only use root keys that are endorsed by at least one authority.
(4) Not *all* authorities who endorsed a PP collude or have their signing keys compromised. Some may endorse illegitimate parties as PPs, but cannot convince all other authorities to endorse such parties.
(5) Organizations monitor CT logs for fraudulent public keys associated to their OI.
(6) Verifiers have access to a trusted offline revocation mechanism.

While Assumption 1 expresses that we do not consider the compromise of PP root private keys, PPs can recover from root key compromise through revocation in practice. Assumption 2 practically implies that the compromise of PP non-root private keys has no effect as the respective endorsements' constraints prevent any misuse.

Assumptions 3 and 4 follow the idea that we laid out in Section 2.2: we can assume that most verifiers will be affiliated to one or more authorities and will only accept emblems endorsed by those authorities, i.e., will only accept emblems with the security level *endorsed*. Assumption 3 expresses this fact, and Assumption 4 expresses that the authority of a verifier's choosing is indeed to be trusted (all but the trusted authorities may be compromised). Assumption 5 concerns the Web PKI domain: we assume that parties utilize the CT logs to monitor their own domains.

Finally, we note with Assumption 6 that we do not consider revocation mechanisms in our formal model explicitly. As ADEM does not specify the precise means verifiers use to perform revocation checks, a formal model of all the possible ways is out of scope in this work. Nevertheless, we will still consider the implications of revocation for authentication and accountability (Sec. 4.3.4) and covert inspection (Sec. 4.4).

To put our security analysis into context, recall that in Section 2.2, we introduced adversaries as those agents who do not respect IHL and seek to abuse digital emblems. Note that adversaries who only disregard IHL and are willing to attack unprotected entities cannot be defended against using a digital emblem. As we noted in Section 2.3: An emblem can only prevent attacks by those who respect it. The goal of our security analysis is to establish that digital emblems cannot be abused, e.g., to mark unprotected infrastructure.

4.2 The Tamarin Protocol Verifier

We formally modeled the authentication and accountability properties and verified that ADEM satisfies them using the protocol verifier Tamarin [16]. Whenever Tamarin attempts to prove a property, it considers infinitely many participants and parallel protocol sessions, and unbounded message exchanges between participants. Tamarin verifies trace properties using backwards constraint solving, attempting to find a counterexample. Tamarin terminates either upon the successful construction of a counterexample (the property is false) or after it considered all possible valid traces of the protocol (the property is true).

```
rule Encrypt:
  [In(aenc('req', pk(sk))), !Ltk(A, sk)]
  -->
  [Out(aenc('ack', pk(sk)))]
```

Figure 6: Tamarin Input Language Example

Tamarin models consist of *rules* and an *equational theory* that constitute a multiset rewriting system (MRS). The rules typically encode the steps protocol participants can take, and the equational theory encodes message semantics, capturing the adversary's reasoning. Rules take the form `lhs --labels-> rhs` and modify the global protocol state. The global protocol state is a multiset of *facts*. Facts model partial state, e.g., which key belongs to which participant, and are either persistent or not. A rule's `lhs` encodes the required partial state for the rule to apply, and the `rhs` encodes an update to the state. When a rule is applied, all non-persistent facts in `lhs` are removed from the global state, and all facts in `rhs` are added to the state. `labels` is a set of labels, which constitute the modelled protocol's trace.

Tamarin analyzes protocols in a *symbolic model*, where protocol messages are modelled as terms. For example, `aenc(m, pk(sk))` encodes the asymmetric encryption of a message `m` using the public key `pk(sk)`, which has the corresponding private key `sk`. Note that, while `aenc` represents a function, it is never evaluated, i.e., the term `aenc(m, pk(sk))` itself represents the asymmetric encryption. Terms are given a semantics through an equational theory. For example, the equation `adec(aenc(m, pk(sk)), sk) = m` expresses that, given the right private key, one can decrypt an asymmetrically encrypted message. Using such functions, the adversary can reason about the messages it receives. The adversary in Tamarin is an active network adversary who can inject, intercept, reroute, and replay any message.

As an example, consider the rule depicted in Figure 6. A protocol participant identified by `A` receives a request, asymmetrically encrypted with `A`'s public key, and looks up its private key and sends an encrypted response.

4.3 Formal Analysis of Authentication & Accountability

We explain next how we formally modelled ADEM, authentication, and accountability using Tamarin. In total, our formal model comprises 327 lines of model specification and 12 lemmas, and it required the implementation of a proof heuristic in Python and auxiliary lemmas for the proofs to terminate. With these proof heuristics and using a laptop with 16GB of RAM and an Intel i7-1065G7 CPU (4 cores @ 1.30 GHz), Tamarin can find a proof for all lemmas except one in around 23s. The remaining lemma must be proven semi-automatically, and the repository hosting our proof file provides documentation how to do so (see p. 2).

4.3.1 Formal Model of ADEM. Tamarin is well suited for ADEM's verification because it supports non-deterministic models, e.g., allowing us to represent endorsement verification as unbounded, non-deterministic loops. We need not specify how many endorsements a PP gathers, but rather let Tamarin exhaustively inspect all

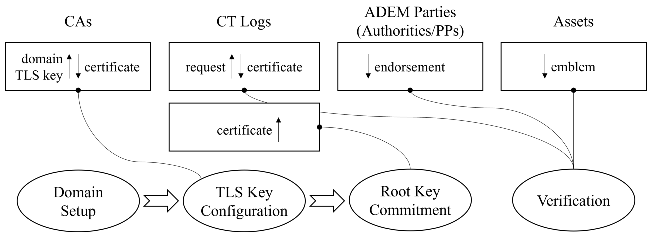

Figure 7: Components of our formal model. We model four kinds of parties with five interfaces (e.g., certificate authorities (CAs) have an interface for certificate signing). The model supports two main protocol flows: PP endorsement and emblem distribution. Lines with dots at the end denote that the (partial) protocol flow uses the connected interface.

possible executions of our model and hence all possible combinations of endorsements.

Figure 7 provides an overview of our Tamarin model. It includes certificate authorities (CAs), CT logs, authorities, and PPs as protocol participants. These parties control keys associated to different protocols: TLS/HTTPS, certificate signing, and emblem/endorsement signing. Modelling the CT specification, which requires that logs must be monitored such that they preserve the append-only property, we represent CT entries using persistent facts, i.e., certificates can either be in a log or not, and once they are included, they cannot be removed. This means that our model abstracts from an ordering on log entries. The adversary can, though, include arbitrary entries in compromised logs.

Our model is intended to run in two phases. Note that we do not enforce that these phases are executed in order, i.e., our proofs also cover executions where the phases are executed interleaved. In the first phase, we non-deterministically assign domain and subdomain names to parties, TLS keys and certificates to these domains, and let ADEM parties commit to root keys by requesting the corresponding Web PKI certificates. In the second phase, a verifier receives an emblem and a non-deterministically determined number of endorsements and verifies them accordingly.

We modelled the adversary as able to compromise any party and their keys. Moreover, all certificates, emblems, endorsements, and the like were modelled as being distributed over an insecure network.

4.3.2 Authentication. Within the model just described, we proved that ADEM provides the following authentication property:

Theorem. If an endorsed emblem is successfully verified then the emblem claims protection for an actually protected asset that belongs to the organization endorsed.

ADEM provides this property using chains of endorsements that root in a set of (not all compromised) authorities. Interestingly, and perhaps surprisingly, authentication does not rely on the security of a PP's OI. We only utilize a PP's OI to commit to root keys, not to authenticate them. In practice, this means that ADEM can handle a partial failure of the Web PKI: Even if the adversary managed to obtain a malicious Web PKI certificate for the PP's OI, they could only succeed in forging unauthentic claims of protection if they additionally compromised some authorities' OIs for which the verifier has no other means of obtaining authentic public keys. Recall, however, that it will likely be trivial for, e.g., military units to obtain at least one authority's authentic public key, namely their own nation state's public key (Sec. 2.2). Moreover, the fewer external endorsements an emblem is accompanied by, the less believable it is. For example, similarly to the CT log infrastructure, we foresee that emblems will be required to be accompanied by some minimal number of external endorsements before they are accepted.

```
 1 All oi asset ak rk.
 2     ( Emblem(oi, asset, ak)
 3     & RootEnd(oi, rk))
 4 ==> ( (Ex pp. OI(pp, oi)
 5         & IsAsset(pp, asset, ak))
 6     | (Ex p. OI(p, oi)
 7         & CompromisedParty(p))
 8     | (Ex otherA.
 9         CompromisedKey(otherA, ak))
10     | (not Ex oiA rkA endK.
11         AuthEnd(rkA, oiA, oi, endK))
12     | (All oiA rkA endK.
13           AuthEnd(oiA, rkA, oi, endK)
14       ==> ( (Ex p. OI(p, oiA)
15               & CompromisedParty(p))
16           | (not Ex p.
17               IsRootPK(p, oiA, rkA)))))
```

Figure 8: Tamarin formalization of the authentication property. `Emblem` represents that a verifier received and verified an emblem. `RootEnd` and `AuthEnd` represent the same for PP and authority endorsements respectively. `IsAsset` expresses that `asset` is a protected asset and `IsRootPK` associates an authentic public key to an OI. Lines 6f.,8f., and 10f. represent Assumption 1, 2, and 3 respectively.

The formalization of ADEM's authentication property is given in Figure 8 (we simplify the presentation of Tamarin lemmas for clarity; see Appendix B for additional details.). The main implication's left-hand side encodes that a verifier received and successfully verified an emblem with an accompanying root endorsement from a (proclaimed) PP. The right-hand side encodes that either the emblem marks a protected asset of that PP (line 4), or one of our threat model's assumptions was violated (see Fig. 8 for details), or that for all authorities, either their root key was compromised (lines 14f.) or an unauthentic key was used to verify their endorsements (lines 16f.).

Note that an attack where the verifier uses unauthentic root keys for all uncompromised authorities is permitted by the authentication property (line 16f.) but not ruled out by our threat model. As we will see next, however, this case is covered by accountability. The use of unauthentic root keys requires malicious root key commitments, which can be detected and which allow us to hold misbehaving parties accountable.

4.3.3 Accountability. Formalizing accountability is subtle. In contrast to authentication, there is no standard definition, and it is

```
All p log ca d pkF skCA.
    ( Dispute(p, log, ca, d, pkF)
    & CASk(ca, skCA)
    & LogInclusion(log, cert))
==> (Ex. CompromisedParty(ca))
```

Figure 9: Accountability constraint for misbehaving CAs. *cert* is a signed certificate from CA ca, binding d to pkF.

unclear from the outset what constitutes a good accountability property. We follow Küsters et al. [12], who defined accountability as a set of *accountability constraints*, which are conditions under which one can blame specific parties for misbehavior. Our accountability property comprises three constraints.

Theorem. The following conditions specify when we can identify parties as misbehaving:

(1) If a domain owner monitors a fraudulent certificate issued for their domain, then the signing CA misbehaved.
(2) If an authority endorsed a fraudulent public key, the authority misbehaved.
(3) If a PP endorsed a fraudulent public key, the PP misbehaved.

The first constraint is straightforward as CT was designed to provide it. The last two constraints allow us to blame misbehaving PPs or authorities respectively, and these constraints go beyond what CT provides for the Web PKI. They are conceptually similar: in both cases, one can blame key holders (PP or authority), when they used their authentic root key to endorse a malicious key (either as a root or asset key).

Our accountability property, which comprises these constraints, has two attractive features. First, it provides what Küsters et al. [12] call *individual accountability*: for each constraint there is at least one party that we can blame. Second, our accountability property is *complete* in that whenever an unauthentic emblem is observed, one of the accountability constraints applies (requiring Assumption 5, i.e., that parties monitor their domains in CT logs).

Consequently, our accountability property admits a decision procedure for verifiers to decide whether there is someone to blame. Whenever a verifier decides to use a root key, they can know that a) as they verified that the CT logs include the root key-binding certificate, the organization in question could have disputed the root key, and b) as they verified that the root key is still active, the organization in question did not dispute the root key. Thus, the verifier can consider this root key as authentic for the given OI, and should this root key be used to issue fraudulent endorsements, the issuing party could be held accountable.

We proved the three accountability constraints and a lemma showing that our property is complete using Tamarin. Proving ADEM to provide accountability was significantly more challenging than proving it to provide authentication, as it required to identify precise and concise conditions of the constraints; their formalization is given in Figures 9-11.

The lemma for completeness is depicted in Figure 12. This lemma states that whenever a verifier uses a root key for verification, a valid certificate binding this root key to the OI must have been included in the CT logs. The proof of this lemma is straightforward

```
All p pA oi rkT rkF oiA rkA.
    ( IsRootPK(p, oi, rkT)
    & IsRootPK(pA, oiA, rkA)
    & AuthEnd(oiA, rkA, oi, rkF)
    & not (rkT = rkF))
==> (Ex. CompromisedParty(pA))
```

Figure 10: Accountability constraint for misbehaving authorities

```
All p oi rk e1 ek i.
    ( IsRootPK(p, oi, rk)
    & RootEnd(oi, rk)
    & Emblem(oi, e1, ek)
    & (not Ex e2. IsAsset(p, e2, ek)))
==> (Ex. CompromisedParty(p))
```

Figure 11: Accountability constraint for misbehaving PPs

```
All oi rk.
    UsedRootKey(oi, rk)
  ==> (Ex ca caSk log c1 c2 k.
    CASk(ca, caSk)
  & c1 = <'cert', ca, oi, k>
  & c2 = <'cert', ca, <oi, sha256(rk)>, k>
  & InLog(log, <c1, sign(c1, caSk)>)
  & InLog(log, <c2, sign(c2, caSk)>))
```

Figure 12: Completeness property of accountability constraints. UsedRootKey expresses that the verifier used the given root key during emblem verification. c1 and c2 model the body of the root key binding certificate. We modelled certificates that are valid for multiple domains using two certificates that are valid for the same key (here k).

as it follows directly from the specification (Sec. 3.4), but it is vital nonetheless. The completeness lemma together with the reasoning from the previous paragraph shows that one of the accountability constraints always applies.

4.3.4 Discussion. To support the automation of our formal proofs, our formal model employed abstractions, e.g., regarding the IP and the Border Gateway Protocol (BGP), DNS, and certificate issuance and revocation. In this section, we discuss these abstractions and how they affect our findings, in particular how our results relate to the real-world.

We assumed that parties can obtain authentic CA and CT log keys. And, for simplicity, we did not model intermediate CA certificates. This should not affect our findings, as intermediate certificates would change our model only in so far that whenever we speak about one CA now, we would instead need to speak about a set of CAs. In practice, it would require just a bit more work to determine which CA precisely was to be blamed in case of misconduct.

In Section 4.3, we described that we modelled CT logs assuming that they are properly monitored and thus cannot equivocate. When a CT log equivocates, it misbehaves by providing different clients different views of the log. [30] recently showcased how utilizing anonymous routing, such as the Tor network, can help to avoid the need for monitoring: If an equivocating CT log cannot see who queries them, they can only guess when providing different clients with different views, defeating the purpose of equivocation. Thus, we suggest that verifiers contact CT logs using anonymous routing protocols should they wish to defend themselves against equivocating CT logs.

Additionally, we did not model CT precertificates, a mechanism that includes certificates in logs prior to their issuance. Precertificates must also be signed by the issuing CA and should thus be covered by our model of standard certificate inclusion. We also did not model internal endorsements of PPs for their own infrastructure and endorsement constraints as, under Assumption 2, internal endorsements must always be constrained in such a way that the endorsed key can only be used for legitimate purposes. Thus, the adversary cannot violate the authentication property by compromising intermediate or asset keys.

Finally, one additional threat vector to ADEM lies in the IP/BGP protocols and DNS. Namely, as emblems identify protected assets via their domain names or IP addresses, an adversary could gain illegitimate protection by hijacking either the respective DNS records and have it point to their IP address, or by BGP hijacking an IP address to have their assets appear as if they were in control of that IP address. However, such attacks would have to be conducted at a massive scale as, by the covert inspection property, adversaries cannot know who wants to attack them. And if they knew who wanted to attack them, it would make little sense to launch a BGP hijack merely to thwart an attack: the adversary could instead deploy more targeted, and cheaper countermeasures, such as dropping all traffic from the adversary.

Certificate Issuance and Revocation. We modelled CAs as flawlessly authenticating certificate requests. We do model fraudulent certificates as we allow CA key compromise, but our model captures two distinct ways how an adversary might get hold of a fraudulent certificate in the same way. An adversary can either have access to a CA's private key, or they can trick the CA. For example, if they managed to compromise a party's DNS entries, they might be able to obtain a fraudulent certificate without the CA acting maliciously. As a consequence, the property that the CA is compromised subsumes both cases just mentioned in our model. In practice, though, this has only limited impact as an adversary would still need to compromise *all* non-malicious authorities' OIs to associate a fraudulent key with the OIs or significantly reduce an emblem's believability by dropping the endorsements of uncompromised authorities.

As for revocation, we do not consider the publication of fraudulent certificate revocations. Such revocations constitute a denial of service attack and ADEM does not provide availability guarantees. Beyond availability, we can abstract possible misbehavior into two categories. Either the CA in question does not publish a certificate revocation, or the offline revocation mechanism does not provide the respective revocation. In both cases, the party in question might misbehave intentionally or due to secret key compromise.

In the case of a misbehaving CA, the PP can easily detect this misbehavior and take actions accordingly (they will see that the revocation was not published). To mitigate the cases of misbehaving offline revocation mechanisms, verifiers can consult multiple such mechanisms. In either case, we recommend short-lived root key endorsements to mitigate any kind of key compromise. The short-lived endorsements can be realized easily as endorsements are neither confidential, nor does resigning require new requests. Authorities could provide newly signed endorsements regularly and publicly after they initially approved a request.

4.4 Covert Inspection

Covert inspection is independent of authentication and accountability. Moreover, adversaries trying to violate this property have completely different motivations: The adversary seeks to distinguish processes that pay attention to an emblem from those that do not, given a process's network transcript and no matter whether it interacts with a protected or an unprotected asset. In this section, we first explain why a formal analysis for covert inspection is infeasible, and after provide a security argument.

One could formalize the idea of covert inspection in a game-based setting where the adversary is given two processes (one checking for the emblem and the other not) and tasked with identifying the one that checks for the emblem. A digital emblem scheme would then provide covert inspection if there cannot be an adversary that recognizes the emblem-checking process with a chance non-negligibly higher than the chance to guess correctly.

Unfortunately, such a definition would be too strong. On the one hand, there are certain classes of programs that can be trivially recognized as non-emblem checking, for example, any process sending no network packets, or programs not using TLS to interact with an asset only signalling protection via TLS. One would need to rule out certain classes of "obviously non-emblem checking" processes; however, it is unclear how one could define such classes without rendering the security definition trivial. On the other hand, the adversary could gain a non-negligible advantage over random guessing by checking if one of the processes sent "non-standard traffic" to the respective asset. Web-servers, for example, see different "standard traffic" than database servers. One would need to represent these differences in typical traffic in the security game, but this would require a classification of digital assets, and a probability distribution on "typical interacting processes" for each of these classes. This has, to the best of our knowledge, not been previously considered in the literature.

Both of these problems put a formal analysis out of reach. Instead, we analyze what a verifier must do to check an emblem and how these steps might reveal them. The goal of the verifier is to generate traffic that "does not stand out" from standard traffic to the respective asset so that they can hide in a sufficiently large anonymity set of clients performing standard queries. A verifier will take the following steps to decide whether a given asset is protected (see Sec. 3.6 and 3.7):

- Check DNS TXT, A, or AAAA records of a (possibly adversarially provided) domain name. TXT records may contain tokens, and A and AAAA records map domain names to IPv4 and IPv6 addresses.

- Check (possibly adversarially chosen) CT logs for the inclusion of root key-binding certificates.
- Send an arbitrary packet to the asset.
- Attempt to perform a TLS handshake with one of the ports 443 (HTTPS) or 853 (DNS over TLS).

The first two steps are necessary to check for emblems in DNS entries and to verify root key setups. To identify a client as emblem-checking through these two steps requires monitoring the client's traffic. This could be done either by an on-path adversary, or because the adversary controlled the DNS servers or CT logs in question. However, in both cases, verifiers are unlikely to be detectable or can take easy steps that reduce the probability of being detected.

Regarding an on-path adversary, traffic to DNS servers and CT logs is usually encrypted, i.e., one could unlikely determine the true contents of queries. Additionally, the verifier could use a virtual private network (VPN) to mask the service they query. Regarding an adversary who controls DNS servers or CT logs, both types of servers usually receive so many queries that an adversary monitoring all these would have a high false-positive rate in identifying verifying clients. And this constitutes exactly what the verifier needs: a large anonymity set. Additionally, the verifier can choose the DNS servers they want to query, reducing the likelihood that the respective server colludes with the adversary.

The same reasoning applies to the packets that the verifier must send to the asset directly. For UDP distribution, we neither require the verifier to send specific packets nor define the port they must contact, verifiers a) will likely query these ports in the intelligence phase of their planned attack anyway, and b) can hide in other traffic that these ports are likely to receive. These two points apply to TLS distribution as well. Verifiers only need to attempt to *connect* to two ports that are well-known to often run TLS.

Finally, we recommend that verifiers also perform offline revocation checks and thereby regularly update these mechanisms. However, offline revocation mechanisms by their very nature cannot reveal which certificates the verifier is interested in. Beyond that, mainstream offline mechanisms are maintained by, e.g., Google for Chromium-based browsers [4] and Mozilla Firefox [21], which gives verifiers a large anonymity set of clients to hide in when updating their mechanisms.

5 RELATED WORK

In this section, we compare with related work from three areas. We discuss well-known authentication systems and highlight their differences to ADEM, how other authors tackled the formal verification of accountability properties, and how covert inspection relates to anonymity.

5.1 Authentication

Authenticated Data Structures. Instead of endorsing PPs, authorities could maintain lists of protected assets using authenticated data structures, for instance, using MHTs such as in the CT log ecosystem [14], or as proposed by CONIKS [17]. With such data structures, however, it is hard to maintain the covert inspection property. Assets would likely need to inform clients connecting to them about the logs that they are included in, e.g., by providing clients with a URL. However, an adversary could easily direct clients attempting to verify an asset's protection to a honey pot instead of logs. The honey pot would not even need to be such a log: the mere connection to a URL would inform the adversary about an imminent attack. At the same time, emblems could not be removed from protected assets, only invalidated.

To avert the honey pot problem, one could instead aim for a global directory of protected assets or public keys that are eligible to claim protection, maintained by consensus protocols, e.g., establishing Byzantine fault tolerance [13]. This would still not solve the issue of removing emblems from assets. Even more critically, consensus protocols require (by definition) *consensus*, which cannot be assumed in the context of a digital emblem: the parties who must reach a consensus may be at war which each other. To avoid the need for consensus, one could instead include different parties' claims about who enjoys protection, but then one would require an additional mechanism to authenticate these claims, rendering the addition of such logs pointless.

ADEM also relies on authenticated data structures, namely the CT log infrastructure. However, we do not utilize these logs for authentication but rather for accountability.

Certificate-based Authentication. In certificate-based approaches, most notably the Web PKI [5], authorities attest that certain public keys belong to certain identities. It is clear that adapting the Web PKI to our purposes would introduce many practical hurdles. X.509 certificates [2] have different semantics than emblems and endorsements. Additionally, existing CAs can hardly authenticate PPs or authorities as such, and likewise, PPs and authorities can hardly become CAs in the Web PKI.

However, ADEM shares some similarities with the Web PKI or parts thereof. Internal endorsements resemble proxy certificates in the X.509 ecosystem [29] (a standard for Web PKI key delegation), endorsement constraints resemble X.509 name constraints [2], and altogether, endorsements resemble standard certificates.

In contrast to the Web PKI, we can hold key-holders, i.e., parties using ADEM, accountable and not just CAs and CT logs. Moreover, ADEM's design centers around the idea that verifiers can choose the “root CA” (authority) they want to trust, rather than relying on a small set of “root CAs” that then endorse other authorities.

ADEM's trust model does not implement a Web of Trust [3] either, although ADEM supports certificate chains of arbitrary length and with arbitrarily many “root CAs.” Authorities endorse PPs directly, and, as there are no other party-to-party endorsements, “certificate” (endorsement) chains between different issuers always have length one. Verifiers must choose the authorities they directly trust and do *not* “transitively trust” them.

5.2 Formal Verification of Accountability

In comparison to properties like secrecy or authentication, a standard definition of accountability does not exist, and accountability is harder to root in intuition. We already noted in Section 4.3.3 that our definition was inspired by Küsters et al. [12]. In this section, we will explain the key differences.

Küsters et al. [12] consider protocols that include a judge who can blame individual protocol participants. They define accountability properties as a set of accountability constraints of the form $C_i = \psi_i \implies V_i^1 \mid \cdots \mid V_i^2$, where ψ_i is a condition under which the constraint applies, and V_i^j are verdicts blaming parties as misbehaving. A protocol provides an accountability property when a) its judge is *fair* (only blames misbehaving parties) and b) the property is *complete* (whenever ψ_i applies, the judges blames parties that match some verdict V_i^j).

Their definitions do not apply in our setting because they do not consider an active network adversary but rather an adversary who cannot intercept communication between honest participants and the judge. Given an active network adversary as in this paper, a judge could be prevented from stating a verdict. The adversary could simply drop all messages to the judge. Generally speaking, it is unlikely that a violation of a protocol's security property, usually encoded within ψ_i, implies that the judge made a verdict V_i^j.

One way to resolve this problem is to strengthen the accountability constraints' conditions ψ_i such that they imply an active judge. Strengthening them too much, though, risks trivializing them: They might take the form "If the judge blames A, then they blame A." We circumvented this issue by modelling the judge within the accountability constraints themselves. Namely, our accountability constraints take the form: "If ψ_i, then a verdict V applies," where V does not blame but identifies corrupted parties. Consequently, the fairness of our "judge" is implicit.

Our approach, however, raises the question: How can we know that a judge as modelled in the accountability property can exist? Given an arbitrary condition ψ_i, it is possible that no *real* judge can decide (as a protocol participant and from their viewpoint) whether ψ_i is the case. Küsters et al. need not consider this problem as their judge is a protocol participant and thus only relies on incoming messages from other participants. We addressed this question by modelling the constraints of our accountability properties such that they only referred to facts that could be checked by a real judge, e.g., whether a certificate was included in a CT log, or whether a verifier received an endorsement of a specific form.

5.3 Covert Inspection, Anonymity, and Undetectability

In this section, we compare covert inspection to the definitions of anonymity and undetectability, two properties that both seem related to covert inspection. However, we will argue that neither matches covert inspection.

Anonymity itself is often defined as that a subject of an action is not identifiable [22]. This definition of anonymity has, e.g., been applied in the analysis of systems like the Tor network [6], but it does not suit our needs: It may already suffice for an adversary to know that *someone* verifies an emblem rather than to know *who*. Depending on the adversary, though, tools that provide anonymity can still help with covert inspection, e.g., onion routing could help against an on-path adversary trying to identify verifiers.

[22] additionally introduces a definition of *undetectability*: the "[u]ndetectability of an item [...] means that the attacker cannot sufficiently distinguish whether it exists or not." Clearly, this definition is very similar to covert inspection, but it still does not quite match. ADEM requires verifiers to make *some* queries, e.g., the resolution of a DNS TXT record. An adversary trying to misuse ADEM as a honeypot might notice the queries themselves. However, the adversary could not use them to identify verifiers reliably (see Sec. 4.4) because queries as part of ADEM are "too standard" and would be performed by many non-verifiers as well.

Undetectability has often been studied in the context of undetectable signals like power or radio signals [11, 15]. The IP network does not support undetectability in the same sense: a packet is sent or not and cannot "not be picked up by my receiver." Undetectability was also studied in the analysis of steganography [11, 15], which again has similarities with covert inspection. In steganography, two parties try to exchange information over an insecure network such that an adversary monitoring all traffic cannot detect the presence of that communication. In contrast, in covert inspection, verifiers a) do not send hidden information, and b) possibly directly communicate with the adversary.

6 CONCLUSION

We have presented ADEM, a design that implements a digital emblem, enabling the marking of digital assets as protected under IHL analogously to the physical emblems of the red cross, red crescent, and red crystal. ADEM is a decentralized design that provides authentication, accountability, and covert inspection. Moreover, ADEM can be deployed by PPs autonomously and does not require updating the Internet's infrastructure. We evaluated ADEM through a formal and informal security analysis, and through a series of meetings with domain experts that were conducted in 2021 at the invitation of the ICRC. Both evaluations show that ADEM is secure and that it should fit the needs of PPs and the ICRC in practice.

We see three promising directions for future work. First, we are developing prototypes for ADEM in collaboration with the ICRC. Our prototypes include clients for emblem and endorsement generation, emblem distribution, and emblem verification. Second, we would like to support our security rationale for covert inspection with an empirical analysis. This involves gathering data or using existing data of real-world traffic to estimate how likely it would be in different scenarios that verifiers can hide their emblem-probing queries in benign traffic. Finally, we have focussed on the network distribution of digital emblems in this paper. However, there are attacks that do not identify their targets through network addresses, such as malware in malicious e-mail attachments or malicious JavaScript. We aim to investigate how a digital emblem could be presented to malware that is already present on a given host.

ACKNOWLEDGMENTS

This work was funded by the Werner Siemens-Stiftung (WSS). We thank the WSS for their generous support of this project.

We also thank Dennis Jackson for his critical work in the initial development and consultation phase of ADEM and Tilman Rodenhäuser for his proof-reading regarding the legal aspects of this paper.

REFERENCES

[1] Tim Berners-Lee, Roy T. Fielding, and Larry M. Masinter. 2005. *Uniform Resource Identifier (URI): Generic Syntax.* Request for Comments RFC 3986. Internet Engineering Task Force. https://doi.org/10.17487/RFC3986

[2] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. 2008. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* Request for Comments RFC 5280. Internet Engineering Task Force. https://doi.org/10.17487/RFC5280

[3] Germano Caronni. 2000. Walking the Web of Trust. In *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE Computer Society, Gaithersburg, MD, USA, 153–158. https://doi.org/10.1109/ENABL.2000.883720

[4] Chromium. [n.d.]. CRLSets. https://chromium.googlesource.com/playground/chromium-org-site/+/refs/heads/main/Home/chromium-security/crlsets.md

[5] Laurent Chuat, AbdelRahman Abdou, Ralf Sasse, Christoph Sprenger, David Basin, and Adrian Perrig. 2020. SoK: Delegation and Revocation, the Missing Links in the Web's Chain of Trust. In *2020 IEEE European Symposium on Security and Privacy*. IEEE Computer Society, Genoa, Italy, 624–638. https://doi.org/10.1109/EuroSP48549.2020.00046

[6] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. 2012. Probabilistic Analysis of Onion Routing in a Black-Box Model. *ACM Transactions on Information and System Security* 15, 3 (Nov. 2012), 14:1–14:28. https://doi.org/10.1145/2382448.2382452

[7] International Committee of the Red Cross. 1949. The Geneva Conventions of August 12, 1949. https://www.icrc.org/en/publication/0173-geneva-conventions-august-12-1949

[8] International Committee of the Red Cross. 1977. Annex I (to Protocol Additional I to the Geneva Conventions of 1949) : Regulations Concerning Identification, 6 June 1977. https://ihl-databases.icrc.org/en/ihl-treaties/api-annex-i-1977

[9] International Committee of the Red Cross. 1977. Protocol Additional to the Geneva Conventions of 12 August 1949, and Relating to the Protection of Victims of International Armed Conflicts (Protocol I), 8 June 1977. https://ihl-databases.icrc.org/en/ihl-treaties/api-1977

[10] Michael Jones, John Bradley, and Nat Sakimura. 2015. *JSON Web Signature (JWS)*. Request for Comments RFC 7515. Internet Engineering Task Force. https://doi.org/10.17487/RFC7515

[11] Georgios Kalogridis, Costas Efthymiou, Stojan Z. Denic, Tim A. Lewis, and Rafael Cepeda. 2010. Privacy for Smart Meters: Towards Undetectable Appliance Load Signatures. In *2010 First IEEE International Conference on Smart Grid Communications*. IEEE Computer Society, Gaithersburg, MD, USA, 232–237. https://doi.org/10.1109/SMARTGRID.2010.5622047

[12] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. 2010. Accountability: Definition and Relationship to Verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*. Association for Computing Machinery, New York, NY, USA, 526–535. https://doi.org/10.1145/1866307.1866366

[13] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems* 4, 3 (July 1982), 382–401. https://doi.org/10.1145/357172.357176

[14] Ben Laurie, Adam Langley, Emilia Kasper, Eran Messeri, and Rob Stradling. 2021. *Certificate Transparency Version 2.0*. Request for Comments RFC 9162. Internet Engineering Task Force. https://doi.org/10.17487/RFC9162

[15] Seonwoo Lee, Robert J. Baxley, Mary Ann Weitnauer, and Brett Walkenhorst. 2015. Achieving Undetectable Communication. *IEEE Journal of Selected Topics in Signal Processing* 9, 7 (Oct. 2015), 1195–1205. https://doi.org/10.1109/JSTSP.2015.2421477

[16] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification (Lecture Notes in Computer Science)*, Natasha Sharygina and Helmut Veith (Eds.). Springer, Berlin, Heidelberg, 696–701. https://doi.org/10.1007/978-3-642-39799-8_48

[17] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. 2015. CONIKS: Bringing Key Transparency to End Users. In *24th USENIX Security Symposium*. USENIX Association, Washington, D.C., USA, 383–398. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/melara

[18] Ralph C. Merkle. 1988. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology — CRYPTO '87 (Lecture Notes in Computer Science)*, Carl Pomerance (Ed.). Springer, Berlin, Heidelberg, 369–378. https://doi.org/10.1007/3-540-48184-2_32

[19] P. Mockapetris. 1987. *Domain Names - Concepts and Facilities*. Request for Comments RFC 1034. Internet Engineering Task Force. https://doi.org/10.17487/RFC1034

[20] P. Mockapetris. 1987. *Domain Names - Implementation and Specification*. Request for Comments RFC 1035. Internet Engineering Task Force. https://doi.org/10.17487/RFC1035

[21] MozillaWiki. [n.d.]. CA/Revocation Checking in Firefox. https://wiki.mozilla.org/CA/Revocation_Checking_in_Firefox

[22] Andreas Pfitzmann and Marit Hansen. 2010. A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[23] J. Postel. 1980. *User Datagram Protocol*. Request for Comments RFC 768. Internet Engineering Task Force. https://doi.org/10.17487/RFC0768

[24] Eric Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3*. Request for Comments RFC 8446. Internet Engineering Task Force. https://doi.org/10.17487/RFC8446

[25] Tilman Rodenhäuser, Mauro Vignati, Larry Maybee, and Hollie Johnston. 2022. Digitalizing the Red Cross, Red Crescent and Red Crystal Emblems. https://www.icrc.org/en/document/icrc-digital-emblems-report

[26] Rich Rosenbaum. 1993. *Using the Domain Name System To Store Arbitrary String Attributes*. Request for Comments RFC 1464. Internet Engineering Task Force. https://doi.org/10.17487/RFC1464

[27] Emily Stark, Joe DeBlasio, and Devon O'Brien. 2021. Certificate Transparency in Google Chrome: Past, Present, and Future. *IEEE Security & Privacy* 19, 6 (Nov. 2021), 112–118. https://doi.org/10.1109/MSEC.2021.3103461

[28] Paul Syverson and Matthew Traudt. 2019. Self-Authenticating Traditional Domain Names. In *2019 IEEE Cybersecurity Development*. IEEE Computer Society, Tysons Corner, VA, USA, 147–160. https://doi.org/10.1109/SecDev.2019.00030

[29] Von Welch, Mary Thompson, Douglas E. Engert, Steven Tuecke, and Laura Pearlman. 2004. *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*. Request for Comments RFC 3820. Internet Engineering Task Force. https://doi.org/10.17487/RFC3820

[30] Tarun Kumar Yadav, Devashish Gosain, Amir Herzberg, Daniel Zappala, and Kent Seamons. 2022. Automatic Detection of Fake Key Attacks in Secure Messaging. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, New York, NY, USA, 3019–3032. https://doi.org/10.1145/3548606.3560588

A THE WEB PKI ECOSYSTEM

We briefly introduce the Web PKI and CT ecosystem, on which ADEM in parts relies. See [5, 27] for a more detailed overview.

In the Web PKI, *certificate authorities (CAs)* sign certificates that associate public keys with domain names such that clients, e.g., browsers, can establish encrypted and authentic channels to, e.g., websites. Browsers and operating systems usually come preinstalled with a set of *root CAs* alongside their public keys. Root CAs typically sign intermediate certificates for other CAs, which can again issue intermediate certificates to other CAs or sign certificates for domain names.

A client accepts a Web PKI certificate if there is a chain of certificates originating from one of their root CAs. Usually, there are no restrictions on which CAs can issue certificates for which domain names and, hence, the compromise of a CA can have devastating effects. To tackle this issue, *Certificate Transparency (CT)* [14] was introduced. The idea of CT is that issued certificates must be stored in append-only, authenticated logs (more precisely, Merkle hash trees (MHTs)), such that these logs can be monitored for fraudulent certificates.

Certificates are submitted to logs by the issuing CA or the respective certificate holder. Whenever a client submits a certificate to a log, they are returned a *signed certificate timestamp (SCT)*, which is a promise of inclusion in the log within (at most) 24 hours. Nowadays, most browsers only accept Web PKI certificates that are accompanied by at least one signed certificate timestamp (SCT).

SCTs themselves, however, do not make the Web PKI ecosystem more secure. Additionally, logs must be monitored to (i) actually include certificates for which they issued an SCT and (ii) preserve the append-only property of their logs. To monitor the former, at least some clients must regularly check that SCTs they witness actually refer to certificates included in logs. To enable the monitoring of the append-only property, logs regularly issue *signed tree heads (STHs)*, which (as the name suggest) are signatures of the MHT's root hash value, alongside a consistency proof with the log's prior STH. Auditors must monitor such issued STHs and verify the consistency between them.

B LEMMA DETAILS

We simplified the presentation of the lemmas in Section 4 for clarity. Namely, we dropped loop identifiers, time variables, and slightly renamed labels for brevity.

Loop Identifiers. We represented that a verifier successfully verified an emblem or endorsement using the labels `Emblem`, `RootEnd`, and `AuthEnd`. As Tamarin permits arbitrarily many parallel protocol sessions, we require a means to identify and connect different verification sessions within a lemma. We do this using a loop identifier `id`, which is the first argument of all these labels. In the presentation of our lemmas, we dropped this loop identifier as all labels referring to emblem or endorsement verification anyways referred to the same loop identifier, i.e., `id` was always quantified within the outermost quantifier.

Time Variables. In Tamarin lemmas, one must not only specify *what* has happened using labels (`Emblem` represents that an emblem was verified), but also *when* something happened using time variables. In our model, however, we do not rely on the order of events, and hence, all time variables were omitted in the presentation of lemmas. More precisely, time variables are always quantified by the innermost quantifier. For example, the lemma

```
All x. P(x) ==> Ex. Q(x)
```

should be understood as

```
All x #t1. P(x) @ #t1 ==> Ex #t2. Q(x) @ #t2
```

where the pattern `@ #t` encodes that, e.g., `P(x)` occurred at time `#t1`.