Provably Secure PKI Schemes

Hemi Leibowitz
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, Israel

Amir Herzberg
Dept. of Computer Science and Engineering
University of Connecticut
Storrs, CT

Ewa Syta
Dept. of Computer Science
Trinity College
Hartford, CT**

***Abstract*—PKI schemes have significantly evolved since X.509, with more complex goals, e.g., *transparency*, to ensure security against corrupt issuers. However, due to the significant challenges involved and lack of suitable framework, the security properties of PKI schemes have not been rigorously defined or established. This is concerning as PKIs are the basis for security of many critical systems, and security concerns exist, even for well known and deployed PKI schemes, e.g., Certificate Transparency (CT).**

We present precise definitions allowing provably secure PKI schemes, with properties such as *accountability, transparency* and *non-equivocation*. We demonstrate usage of the PKI framework against X.509 version 2.

***Index Terms*—**

I. Introduction

Public Key Infrastructure (PKI) provides an essential foundation for applications which rely on public key cryptography, and it is crucial to achieve security in open networks and systems. Since its introduction in 1988, the deployment of PKI has been dominated by the X.509 standard [1], likely due to its integration with the TLS/SSL protocol [2], the most widespread protocol used to secure connections between servers and clients, most commonly web browsers. The resulting 'web-PKI' is necessary to provide confidentiality, integrity and authenticity of web services, and as such, is critical for the secure use of the web.

Unfortunately, the web-PKI deployment has inherent weaknesses. In particular, any CA is trusted to issue certificates for any domain [3], resulting in the weakest-link security model and making individual CAs prime targets for attacks. Over the years, we have seen many failures of this trusted-CA approach. For example, hackers stole the master keys of CAs [4], [5] and issued fake certificates for major websites. Furthermore, some CAs abused their powers by improperly delegating their certificate-issuing authority or even intentionally issuing unauthorized certificates [6]. Such PKI failures allow attackers to issue fake certificates, launch website spoofing and man-in-the-middle attacks, possibly leading to identity theft, surveillance, compromises of personal and confidential information, and other serious security breaches.

For many years, *accountability* was considered a sufficient deterrent. Namely, the fact that X.509 certificates are signed by a specific issuing CA, ensures that the issuing CA cannot deny having issued a certificate - or, more precisely, that its private key was used to sign the certificate. However, the many PKI failures brought the realization that *accountability is not sufficient*. Accountability is only effective if and when the fake certificate is found - which may not occur, especially if abused 'stealthily', and only if the misbehaving authority can be effectively punished.

These failures motivated efforts to develop and adopt *improved-security PKI schemes*, i.e., PKI schemes that ensure security against corrupt CAs. During the recent years, there have been extensive efforts toward this goal by researchers, developers and the IETF. These efforts focus on security properties such as *transparency*, *non-equivocation* and more. Proposals and designs include *Certificate Transparency (CT) [7], [8], Enhanced-CT [9], Sovereign Key [10], CONIKS [11], AKI [12], PoliCert [13], ARPKI [14], DTKI [15], CoSi [16], [17], IKP [18], CertCoin [19], PB-PKI [20], Catena [21], CertLedger [22]*, among others.

The goals of these designs are beyond those of X.509, and are significantly more complex than the X.509 goals. However, so far, these goals have not been rigorously defined and proven. Only few works present any analysis: [23], [24] analyze (only) the logging mechanism of CT, and ARPKI [14], [15] use automated symbolic analysis for system-specific properties. However, no work defines security goals (or proves such properties hold). In fact, even for the simple, 'classical' X.509 PKI, there is *no definition of security requirements* (and no proof). Arguably, this may not be as critical, since for X.509, both definitions and proof are quite straightforward (see within). However, this lack of definition (and proof) implies that works analyzing security of PKI-based protocols, e.g., IPsec/IKE [25], [26] and TLS [27], [28], mostly completely ignore the PKI and simply assume the use of correct public keys. Few works [29]–[31] study security of cryptographic protocols based on a *simplification* of X.509; however, the simplification *ignores revocation* and other advanced aspects of X.509, not to mention the properties of post-X.509 PKI schemes, such as transparency.

This is alarming, as most practical applications of cryptography involve certificates, and their security depends on the security of the PKI. The extensive efforts to prove security of cryptographic protocols may be moot when these protocols are deployed over a PKI scheme which was not proven secure. The concerns are even greater, considering that attacks against

** The work was partially completed during a visiting position at the Dept. of Computer Science and Engineering, University of Connecticut, Storrs, CT

PKI are not only a theoretical threat, but are a major concern in practice.

Indeed, defining and proving security for PKI schemes is a non-trivial challenge, especially for post-X.509 schemes, with more advanced and complex goals. PKI proposals vary greatly - even in terms of the types of parties involved, and in the communication and attack models. As a result, existing definitions and requirements are often informally defined and tailored to a specific design. The lack of *proper* definitions and proofs makes it challenging to build (provably) secure systems, which depend on PKI schemes, and to improve, compare and select PKI schemes. Evaluation of PKI schemes with new properties is especially challenging; for example, there are several schemes designed to achieve different *privacy* goals, but these cannot be properly compared - and, of course, are not proven secure. It is impossible to design and analyze schemes modularly, by provable reductions to simpler, already-analyzed schemes.

In this work, we present well-defined security specifications for PKI schemes, with reduction-based proofs of security. The definitions supports a wide range of PKI schemes, from X.509 to advanced, improved-security PKI schemes, independently of their specific designs. We focus on the classical security challenge of dealing with misbehaving parties, e.g., corrupt CAs, by *detecting* misbehavior and/or *preventing* damage due to misbehavior. Note that we do not address *trust-management* issues, such as the *decision to trust* a particular CA, typically based on cross-certification by already-trusted CAs ('basic constraints' in X.509), or restricting a CA to particular name-space ('naming constraints' in X.509). Other works address other important aspects of PKI schemes, mainly the *trust decision* - essentially, which CA should be trusted for a given certificate. A model of trust for PKI systems was proposed by Mauer [32], subsequently extended by [33], [34], and others [35]–[40].

To define the security specifications, we reviewed and analyzed existing PKI schemes and the properties they claim to provide. We present game-based definitions for the basic security specifications for PKI schemes designed for possibly faulty CAs. These include the following security specifications: *accountability*, Δ*-transparency*, *non-equivocation* (detection and prevention), *revocation accountability*, *non-revocation accountability* and Δ*-revocation status transparency*. We map these security specifications to existing PKIs in Table I.

Contributions. This work makes the following contributions:

1) PKI framework for provably-secure PKI schemes.
2) Formal definitions of PKI security specifications.
3) Specification comparison of popular PKI schemes.

Organization. Section II reviews the PKI landscape with respect to the security specifications we identify in PKI schemes and summarizes the related work. Section III briefly discusses the *Modular Specifications Security Framework (MoSS) [41]* which is used to formally prove the specifications achieved by PKI schemes under formally defined models and presents the PKI framework and its security specifications. Section IV defines the basic concepts of PKI schemes. Section V present the PKI framework, which includes the formal definition of PKI scheme's operations and security specifications. Section VI presents how the PKI framework can be applied in practice, and defines the X.509 version 2 protocol using the PKI framework and then formally analyzes the security of the protocol in Section VII. We conclude and discuss future work in Section VIII.

II. Security Specifications of PKI Schemes

The first step in developing the PKI framework was the identification of the security specifications of PKI schemes, where we focus on schemes designed for security against corrupt authorities (typically, corrupt CA). In this section, we first discuss, informally, these security specifications which we later formally define in §V-C as game-based definitions.

We put an extensive effort into a thorough comparison of the existing schemes with respect to the properties we identify. We present the results in Table I.

A. X.509 PKI Schemes and Related Specifications

The basic goal of a PKI scheme is to ensure *authenticity* of information in *public key certificates*. Certificates are issued and endorsed by *Certificate Authorities (CAs)*. A typical certificate links an *identifier* with some *public information*, typically, a public key. A certificate also typically includes a signature generated by the issuing CA, over the certificate's information; the signature serves as the CA's endorsement of the mapping between the identifier and the public information in the certificate. An *honest* CA issues a certificate only after it verifies that the entity requesting the certificate is entitled to receive it. This establishes the following basic goal of PKI schemes: *accountability*.

*1) Accountability Specification (*ACC*):* Accountability is the ability to *identify the CA that issued a given certificate*. Accountability provides a reactive defense against a corrupt CA; such CA can be ignored or otherwise punished. In most PKI schemes, including X.509, accountability is achieved by having the CA digitally sign certificates, i.e., a CA is accountable for any certificate signed using the CA's private key. CA accountability, in this sense, includes unauthorized use of the CA's private key, e.g., due to exposure or penetration, as well as intentionally issuing 'fake' certificate, where the public information does not correctly match the identifier. In particular, accountability serves as a motivation for honest CAs to take any available precaution to ensure that the certificates they issue are authentic, because if not, they will be held accountable. Note that we use the term accountability as a technical, well-defined specification, which does not necessarily have any specific legal or financial implications.

*2) Revocation Accountability Specification (*ReACC*):* Similarly to the accountability specification which focuses on issuing certificates, revocation accountability ensures accountability of *revoked* certificates. Namely, while certificates' validity period starts from their *issue date* and ends on their *expiration* *date* (both of which specified in the certificate), certificates can be invalidated before their expiration date, i.e., revoked. Hence, revocation accountability requires that each revoked certificate can be traced back to the revoking CA. This helps to ensures that clients will not have their certificates revoked without a legitimate reason (e.g., their request), unless the CA is malicious, or an attacker corrupts or tricks the CA - in which case, this can be exposed. A user can request to have its certificate revoked for a variety of reasons, including a loss or compromise of the private key corresponding to the public key endorsed in the certificate.

*3) Non-Revocation Accountability Specification (*NReACC*):* Since any certificate can be revoked at any time, it is imperative to verify that a valid certificate is still *not-revoked*, prior to using it. The non-revocation accountability specification ensures that an honest CA would never sign a revoked certificate as non-revoked; in fact, if a CA signs such statement, this provides a *proof of misbehavior*. In X.509, and even most post-X.509 PKI schemes, only the certificate's issuer can revoke it, making this specification easy to define and achieve. However, considering the history of CA failures, it may be useful to allow revocation of certificates by other authorities, but would be harder to define and achieve.

There are several revocation/non-revocation mechanisms which are used in practice, including *Certification Revocation List (CRL)* [42] and *Online Certificate Status Protocol (OCSP)* [43]. However, the actual guarantees that such mechanisms provide are more complex than intuitively implied. For example, some browsers often employ a soft-fail approach, which means that an adversary that can block CRL/OCSP responses could trick browsers into accepting revoked certificates. Therefore, the actual security guarantee of such mechanisms must be defined with respect to the actual used model.

B. Post-X.509 PKI schemes and specifications

Until now, we discussed accountability, revocation accountability and non-revocation accountability, which correspond to the basic PKI properties, provided already by X.509. We now discuss additional security goals, pursued by more recent PKI schemes, designed to improve security against corrupt CAs. These include Δ*-transparency* (ΔTRA), Δ*-revocation status transparency* (ΔReST), Δ*-equivocation detection* (ΔEQ-D) and *equivocation prevention* (EQ-P).

1) Δ-Transparency Specification (ΔTRA*):* Accountability, as described above, mainly serves as a *deterrent* against misbehavior, i.e., only offers retroactive security by punishing a CA 'caught' misbehaving, e.g., issuing a fraudulent certificate. For many years this reactive measure was viewed as a sufficient defense, under the assumption that CAs were highly respectable and trustworthy entities who would not risk, intentionally or otherwise, being implicated in issuing fraudulent certificates. However, repeated cases of fake-certificates, by compromised or dishonest CAs, have proven this assumption to be overly-optimistic. It turned out that punishing CAs is non-trivial: beyond negative publicity, any punishment was arbitrary, short-lived and overall ineffective [44]–[46].

Furthermore, 'punishment' could only be applied *after* the damage was committed and *discovered* - if it is discovered at all. An attacker or corrupt CA could reduce the risk of discovery, by minimizing the exposure of the fraudulent certificate. Except for efforts such as the Perspectives Project [47], or the EFF SSL Observatory [48] that aim to gather and inspect *all* SSL certificates used in practice, the burden of detecting and responding to fraudulent certificates is mostly on the clients that receive them; browsers typically cannot detect fraudulent certificates, much less to report them to a (non-existing) 'enforcement agency'.

This significant issue has motivated more recent PKI designs, e.g., CT, where valid certificates must be *transparently published*, i.e., publicly available, allowing third parties (e.g., trusted 'monitors') to inspect and detect any fraudulent certificates. Transparency requires a certificate to be recognized (signed) by a party which takes responsibility for making the certificate *available* to all monitoring-entities, within a specified time frame Δ. To clarify, Δ describes the maximum time it takes for the certificate to become publicly available, referred to as the Maximum Merge Delay (MMD) in CT, but Δ *also* includes the maximum time it takes for monitors to discover it once it is publicly available. By demanding transparency, a PKI system prevents a CA from 'silently' generating fraudulent yet validly-formed certificates, and exposing them only to selected victims during an attack, and furthermore, facilitates detection of fraudulent certificates, issued by a corrupt or compromised CA.

2) Δ-Revocation Status Transparency Specification (ΔReST*):* While $\Delta-$transparency guarantees that every valid certificate is available to all interested monitors within Δ time, $\Delta-$transparency does not guarantee protection against revoked certificates. That is to say, while $\Delta-$transparent certificates are available to monitors, there is no guarantee regarding what happens to those certificates later, and therefore, $\Delta-$transparency impose no guarantees regarding to certificates' revocation status. For example, if the private keys of a $\Delta-$transparent certificate were compromised and the subject of the certificate had the certificate revoked, there is no guarantee that others will learn of this revocation, and especially, there is no clear time frame describing *when* others will learn of this event.

This fundamental issue motivates $\Delta-$*revocation status transparency*. When an entity *attests* for a given certificate's revocation status, either revoked or non-revoked, is transparent (ΔReST), that entity commits to making this *status* available to all interested third parties, again, within a predetermined Δ time frame.

3) Detection and Prevention Specifications: Although the aforementioned transparency specifications require that certificates and their status are available to interested parties, it does not guarantee that a problematic certificate, either malicious or benign, will be *detected* nor it guarantees that such detection would in fact occur *before* certificate misuse occurs. In fact, even where detection is guaranteed to occur, this can only be *guaranteed* some time after the issuance of the fake certificate - although, this aspect is often overlooked. Finally, detection is only a passive measure, and some specifications could be instead active, i.e., strive for prevention instead of after the fact detection.

Fraudulent certificates which use the *same* identifier as the victim, may be abused for phishing, and for other attacks, e.g., stealing web cookies. For example, consider a scenario where two equivocating certificates are transparent, but they are logged over two different entities which are monitored by different monitors. In such scenario, Δ-transparency is not enough, as no single entity is guaranteed to be aware of both certificates.

In this work we focus on two examples of such specifications: Δ-equivocation detection and equivocation prevention. Future work can extended and support additional detection and prevention specifications.

4) Δ-Equivocation Detection Specification ($\Delta\mathsf{EQ\text{-}D}$): The Δ-equivocation detection specification ensures that any equivocating certificate is detected within Δ time of its issuance by at least one honest entity.

Often, a certificate is considered fraudulent when it uses a *misleading* identifier, such as a domain name which is identical or similar to that of a victim domain, e.g., g00gle.com, or with an identifier which users may expect to belong to a known domain, e.g., googleaccounts.com. Such misleading identifiers are often abused, e.g., for *phishing* attacks. A PKI which supports transparency, allows a domain to vigilantly watch for any certificate issued with identifiers which are identical, similar or otherwise misleading to be associated with its own domain names.

This design makes it possible to quickly detect misbehavior, such as issuing of a fraudulent certificate. Ideally, fraudulent certificates could likely be detected *before* they can be abused, or at least, before they can cause much harm.

5) Equivocation Prevention Specification ($\mathsf{EQ\text{-}P}$): A PKI which prevents equivocation, will prevent a corrupt CA from issuing a fake certificate for an already-certified identifier, e.g., domain name. This could *prevent*, rather than merely *detect*, man-in-the-middle and other attacks impersonating existing secure domains [49].

Note that transparency implies $\Delta\mathsf{EQ\text{-}D}$, but not $\mathsf{EQ\text{-}P}$. We still define equivocation detection as a separate specification, since it does *not imply* transparency, i.e., $\Delta\mathsf{EQ\text{-}D}$ is not equivalent to transparency. In fact, some PKI schemes, notably CONIKS [11], offer equivocation detection but *not transparency* - indeed, transparency would conflict with some of CONIKS privacy goals. Of course, providing non-equivocation but not transparency, may still allow issuing of misleading (but not identical) identifiers, e.g., misleading domain names which may be abused for phishing attacks.

We emphasize that specifications can be either a global requirement that applies to all the certificates in the system, or alternatively, to only a subset. To illustrate, consider some prevention specification such as equivocation prevention. One approach for a system would be to require all valid certificates to be unequivocal by default. Alternatively, another approach would be to allow certificates to choose whether they are unequivocal or not, yet they can be valid either way. One possible reasoning for that might be monetary, since unequivocability might require additional effort which involve additional costs.

C. PKI Schemes Comparison

The goal of the PKI framework is to allow analysis and provable-security, for existing and future PKI schemes. We designed the framework in a way that *embraces*, *complements* and *reflects* current PKI designs. To this end, we have methodically examined the existing PKI schemes by identifying and analyzing their properties. We present the results of our analysis in Table I, and summarize them below. The table includes twelve existing PKI systems, and, in addition, a "proof-of-concept" PKI we defined, and X.509v2, a minor extension of the CT specifications, which appears essential to ensure CT's security properties. We compared all schemes with respect to the specifications formally presented in §V; we also mention two additional properties, privacy and global name-spaces.

Notations in Table I. We use the $\mathsf{n/s}$ (not supported) symbol to indicate when a scheme does not seem to support a specification. Otherwise, we use one of the three following symbols, ●, ⊙, or ◐, to indicate the support of a specification by the scheme. The ● symbol indicates that a system comes with *rigorous*, reduction-based proof of the specification. We indicate with an appropriate comment when a scheme is supported by automated symbolic proof for a given property; note that such proofs are often of property specific to that scheme, not properties defined for arbitrary PKI schemes. The ◐ symbol indicates that although no formal proofs were provided, it seems *intuitively true* that the system achieves a specification; e.g., accountability in X.509 follows from the use of signature scheme to sign the certificate. The ⊙ symbol depicts the property is justified using an (informal) security argument; note that this may imply that additional assumptions or details may be needed to ensure security.

Following our discussion of the 'basic' PKI security properties in §II-A, we observe that most systems aim to achieve accountability, with the exception of CertCoin and PB-PKI. Both CertCoin and PB-PKI build on top of Namecoin [50], which is a decentralized namespace system rather than a centralized, CA-oriented system, where the CAs grant identifiers to clients. Instead, due to the fully decentralized nature, anyone can claim an identifier so long it is available; consequently, there is no accountability for assigning identifiers. Notice also that Catena is a witnessing (logging) scheme that allows to witness public-key directories using the Bitcoin blockchain. As a result, accountability of issuing certificates is handled by the directories themselves, which require unusual additional assumptions (which can be modelled using the framework).

Interestingly, many systems directly focus on more advanced properties, such as transparency and non-equivocation, and treat more 'basic' properties, such as accountability and

System [reference]	Safety requirements						Additional req.	
	ACC	ΔTRA	ΔEQ-D	EQ-P	ReACC/ NReACC	ΔReST	Privacy^{1}	Global namespace
Provably-secure X.509v2 ^{2} (this work)	●	n/s	n/s	n/s	●	n/s	n/s	✓
X.509v3 ^{2}	◐	n/s	n/s	n/s	◐	n/s	n/s	✓
Catena [21]	⊙^{7}	⊙	⊙	⊙	⊙^{7}	⊙	n/s	✓
CertCoin [19]	n/s	⊙	⊙	⊙	n/s	⊙	n/s	✓
PB-PKI [20]	n/s	⊙	⊙	⊙	n/s	⊙	⊙	✓
CoSi [17]	◐	⊙	⊙	⊙	n/s	n/s	n/s	✓
Enhanced-CT [9] DTKI [15] ^{3}	◐	⊙	⊙	n/s	◐	⊙	n/s	✓
AKI [12]	◐	⊙	⊙	n/s	◐	⊙	⊙	✓
CONIKS [11]	◐	n/s	⊙	n/s	◐	⊙	⊙	✗
ARPKI [14] ^{4}	◐	⊙	⊙	⊙	◐	⊙	n/s	✓
CertLedger [22]	◐	⊙	⊙	⊙	⊙	⊙	n/s	✓
Certificate Transparency (CT) [7]	◐	⊙^{5}	⊙^{5}	n/s	◐	n/s^{6}	n/s	✓
PoC-PKI	●	●	●	●	●	●	n/s	✓

TABLE I
Comparison of PKI schemes with respect to PKI framework. Symbols: ● - reduction-based proofs, ◐ - intuitively true, ⊙ - security arguments (a proof may require assumptions), n/s - not supported. ^{1}Different privacy definitions, goals. ^{2}X.509 with PKIX, and CRL or OCSP (OCSP ensures NReACC). ^{3}DTKI has symbolic proofs of some aspects. ^{4}ARPKI has symbolic proofs of some aspects. ^{5}Proofs of logging properties in [23], [24]. ^{6}CT is extended to include revocation transparency in [8].

revocation, as intrinsic to PKI, often without even stating them. This phenomenon is especially apparent in case of revocation; many systems (e.g., CertCoin, Catena, PB-PKI, CoSi) do not directly address revocation at all, and do not discuss how revocation should be handled, by whom and under which conditions. Other PKI schemes use the X.509 notion of a certificate, and implicitly rely on the X.509 revocation mechanisms (CRLs and OCSP). This approach is somewhat understandable due to the pervasiveness of X.509, but also establishes the X.509 revocation mechanisms as the status quo of revocation, despite known weaknesses.

In Table I, we label accountability, revocation accountability and non-revocation accountability as 'intuitively true' for all systems, except for CertCoin, Catena, PB-PKI, and CoSi. These properties are typically achieved using a secure signing scheme, and therefore a formal proof seems straightforward and not essential. Note that CertCoin, PB-PKI and CONIKS allow clients to revoke their own certificates, but revocation can also be done by an adversary that compromised the client's secret keys, or alternatively, the client may be unable to perform revocation if the secret keys are lost.

Transparency, on the other hand, is supported by all post-X.509 PKI schemes, except CONIKS. The fact that transparency is so pervasively provided is likely in response to one of the main weaknesses of X.509 widely abused in practice, i.e., the lack of a mechanism to effectively propagate all issued certificates among CAs and clients. CONIKS, on the other hand, offers a limited notion of transparency of the identity / value map, which hides the actual identifiers and their corresponding values, as a trade-off between security and privacy. The clients can only query for individual identifiers. Furthermore, even that must be within a specific namespace, as CONIKS does not support global namespaces, where multiple CAs are authorized to issue for the same namespace. The use of separate namespaces, while problematic for the web PKI, works well for many applications such as chat rooms or messaging boards, that require secure key distribution but are under control of a single entity.

As Table I indicates, most previously-published PKI schemes have only informal security arguments for transparency. The exception are CT, DTKI, and ARPKI, which have different types of automated proofs for scheme-specific properties. Namely, the properties and their proofs are not relevant to PKIs per se. Rather, they focus on details of the design of the particular scheme. Specifically, Dowling et al. [23] formalized security properties and provided reduction-based proofs for logging aspects of CT, that cover two classes of security goals involving malicious loggers and malicious monitors. Chase and Meiklejohn [24], on the other hand, focus on formalizing transparency through "transparency overlays", a generic construction they use to rigorously prove transparency in CT and Bitcoin. While their approach is elegant and can be used in other systems as a primitive that achieves transparency, it focuses on the "CT-style transparency" and does not consider other PKI properties such as revocation or non-equivocation.

Some of the systems, such as DTKI and ARPKI, verify their core security properties using automated symbolic proofs via the Tamarin prover [51]. Symbolic proofs provide an important added value for the security of proposed systems. Unfortunately, symbolic proofs often use abstractions; for example, in DTKI and ARPKI, a Merkle tree is modeled as a list. Such abstractions present an obstacle towards 'air-tight' security proofs. This strengthens the importance of a formal framework which on the one hand does not rely on specific implementations, yet, on the other, can be easily used by any implementation. We leave it to future work to explore ways to use symbolic proofs to add automatic verification capabilities to the framework described in this paper.

The post-X.509 safety specifications - transparency and non-equivocation - are significantly more complex to understand, define and to achieve, compared to the X.509 properties of accountability, revocation accountability and non-revocation accountability. Hence, we did not consider any of these post-X.509 properties to be 'intuitively true' - we believe they all require a proper definition and proof, as we provide in this paper; we spent considerable effort in properly defining these specifications in a precise and complete manner, and made every effort to keep things simple - but we admit that these definitions still require considerable effort to fully understand.

We separated between properties which are not-supported, and properties which are claimed to be supported using some security arguments. Note also that most systems do not discuss revocation status transparency at all, even though in certain cases, e.g., CoSi, it seems relatively easy to achieve it. CT originally did not have a built-in support for revocation status transparency, and it was only later formalized as Revocation Transparency [8].

III. Preliminaries

In this section, we discuss the Modular Specifications Security (MoSS) [41] framework, which we use to define and analyze PKI schemes. In Section III-A, we briefly describe the MoSS framework, specifically, the concepts of an *execution process*, *models* and *specifications*. Later, in Sections VI-VII we formally analyze the security of X.509 version 2 using the MoSS framework. This analysis also relies on the security of a the signature scheme used. For the traditional definitions of a signature scheme and its security, i.e., indistinguishability, see Appendix A.

A. MoSS Concepts: Execution Process, Model and Specifications.

The MoSS framework establishes three components: an *execution process*, *model* and *specifications*. We now briefly explain each component, focusing on the 'classical' asymptotic-security definitions[1]. Note, however, that we omit some of the advanced tools provided by the framework which are not relevant to this work. For the full description of the framework, see [41].

1) Execution Process: The first component is an adversary-driven execution process, which defines the process of executing a protocol $\mathcal{P}$ under adversary $\mathcal{A}$, giving the adversary extensive control over the operation of the environment, including communication, local-clock values, inputs from the application, and faults. The execution process is a precise, algorithmic process that outputs a *transcript* containing the events in a run of the protocol with a given adversary. All events are serializable on a 'real-time' axis, and defined iteratively, as a sequence of invocations of specific entity.

More precisely, **Exec** is a randomized algorithm, which receives three inputs: two efficient (PPT) algorithms, $\mathcal{A}$ for the adversary and $\mathcal{P}$ for the PKI scheme, and a *security parameter*, which in our case is simply a unary string 1^κ. The execution process outputs a *transcript* $T \leftarrow \mathbf{Exec}_{\mathcal{A},\mathcal{P}}(1^\kappa)$, which provides details on the events in the execution, specifically:

$T.out_{\mathcal{A}}$	Adversary's output.
$T.e$	Number of events in the execution.
$T.\mathsf{N}$	Set of entities (determined by adversary).
$T.\mathsf{F}$	Faulty (adversary-controlled) entities.
$T.ent[\hat{e}]$	Entity invoked in event $\hat{e} \leq T.e$.
$T.opr[\hat{e}]$	Operation invoked in event $\hat{e} \leq T.e$.
$T.inp[\hat{e}]$	Input to event $\hat{e} \leq T.e$.
$T.clk[\hat{e}]$	Clock value of entity $T.ent[\hat{e}]$, at event $\hat{e} \leq T.e$.
$T.\tau[\hat{e}]$	Global real-time at event $\hat{e} \leq T.e$.
$T.out[\hat{e}]$	Output of entity $T.ent[\hat{e}]$, at event $\hat{e} \leq T.e$.

Appendix C contains the pseudo-code of the execution process, from [41].

2) Models: The second component of the framework is the definition of *models*. A model predicate $\mathcal{M}$ classifies the execution of a protocol $\mathcal{P}$ under adversary $\mathcal{A}$ as 'valid' ($\top$) or 'invalid' ($\bot$), effectively enforcing one or multiple restrictions on the adversarial control of the execution, including initialization assumptions, limitations on number and type of faults, maximal delay and/or maximal clock drift. Furthermore, a model predicate may enforce restrictions on the use of resources by the adversary, such as numbers of queries or running time.

The execution process allows the adversary to fully control the execution. Assumptions and restrictions on the execution, such as communication, synchronization and adversary model, are defined by efficient *model predicates*. The model predicate $\mathcal{M}$ is applied to the execution transcript T; we say that execution transcript T *satisfies* model $\mathcal{M}$, if $\mathcal{M}(T) = \top$. Intuitively, a PPT adversary $\mathcal{A}$ *satisfies* model $\mathcal{M}$, which we denote by $\mathcal{A} \models \mathcal{M}$, if for every protocol $\mathcal{P}$, executions of $\mathcal{P}$ with $\mathcal{A}$ satisfy $\mathcal{M}$ with overwhelming probability. In this work, we assume a simple model, that enforces bounded-drift clocks and that $T.\mathsf{F}$ would contain every entity that the adversary 'corrupted' by reading its state. This model is formally defined in Appendix C.

Definition 1. *(Model-satisfying adversary.) Let a model $\mathcal{M}$ be an algorithm with binary output, i.e., a predicate. We say that adversary $\mathcal{A}$ **(asymptotically) satisfies model** $\mathcal{M}$, denoted as $\mathcal{A} \models \mathcal{M}$, if for every PKI scheme $\mathcal{P}$ and security parameter 1^κ, the model predicate $\mathcal{M}$, applied to a random resulting execution-transcript $T \leftarrow \mathbf{Exec}_{\mathcal{A},\mathcal{P}}(1^\kappa)$, is satisfied with overwhelming probability, i.e.:*

$$\mathcal{A} \models \mathcal{M} \overset{def}{=} (\forall\, \mathcal{P}, 1^\kappa)\, \epsilon^{\mathcal{M}}_{\mathcal{A},\mathcal{P}}(1^\kappa) \in Negl(1^\kappa) \qquad (1)$$

*Where, for any predicate π, e.g., $\pi \equiv \mathcal{M}$, we define the π-*advantage of adversary $\mathcal{A}$ against protocol $\mathcal{P}$ *as:*

$$\epsilon^{\pi}_{\mathcal{A},\mathcal{P}}(1^\kappa) \overset{def}{=} \Pr\left[\begin{array}{l} \pi(T) = \bot \ : \\ T \leftarrow \mathbf{Exec}_{\mathcal{A},\mathcal{P}}(1^\kappa) \end{array}\right] \qquad (2)$$

3) Specifications: The third component of the framework is the definition of *specifications*, which define what is provided

[1]It is not difficult to extend our results to concrete security, as modeled in [41].

by a protocol or scheme, given some model.

Similarly, *specifications* are also defined as efficient predicates applied to the transcript T applied to execution transcripts. We define that a protocol $\mathcal{P}$ (e.g., in our case, a PKI scheme), *satisfies* a specification as follows.

Definition 2. *(Specification-satisfying protocol.) We say that protocol $\mathcal{P}$* ***(asymptotically) satisfies specification*** *ξ under model $\mathcal{M}$, denoted as $\mathcal{P} \models^{\mathcal{M}} \xi$, if for every PPT* adversary $\mathcal{A}$ that satisfies $\mathcal{M}$ *and security parameter 1^{κ}, the specification ξ, applied to a random resulting execution-transcript $T \leftarrow$ $\mathbf{Exec}_{\mathcal{A},\mathcal{P}}(1^{\kappa})$, is satisfied with overwhelming probability, i.e.:*

$$\mathcal{P} \models^{\mathcal{M}} \xi \overset{def}{=} (\forall\, \mathcal{A}, 1^{\kappa}\ |\mathcal{A} \models \mathcal{M})\, \epsilon^{\xi}_{\mathcal{A},\mathcal{P}}(1^{\kappa}) \in Negl(1^{\kappa}) \quad (3)$$

Where $\epsilon^{\xi}_{\mathcal{A},\mathcal{P}}(1^{\kappa})$ is defined as in Equation 2.

IV. PKI Concepts

In this section, we formally define the building blocks of a PKI. In §IV-A, we define who are the entities involved and what is a certificate. Then, in Section IV-B, we explain what it means for certificates to have attributes.

A. Entities and Certificates

PKI schemes are protocols for a set N of *authorities*, such as *certificate authorities (CAs)*. A *certificate authority (CA)*, also referred to as an *issuer*, is an authority that issues *certificates* to subjects, where a certificate is a verifiable association of the subject's identifier with some public information; in a *public key certificate*, the public information includes the subject's public key. PKI schemes that wish to satisfy non-trivial specification, can introduce more authorities into N. For example, in a specific PKI scheme such as CT, the set of authorities N contains additional types of authorities, such as loggers, monitors etc., which are imperative for advance specifications, e.g., transparency.

PKI schemes have two types of users (clients): *subjects*, which typically use a CA to obtain and manage their public key certificate, and *relying parties*, which use the certificates of the subjects, in order to determine if they want to communicate with the subject (using the certified public key). Note, however, that we do not explicitly include the clients as part of the set of entities N.

In Definition 3, we define the structure of a typical certificate.

Definition 3 (Certificate)**.** *A* certificate *is a tuple:*

$$\psi = (serial, subject, pub, from, to, issuer, \tau, \gamma, \sigma)$$

where:

- *$\psi.serial$: CA-assigned serial number.*
- *$\psi.subject$: to whom the certificate is issued.*
- *$\psi.pub$: public information associated with $\psi.subject$.*
- *$\psi.from$: when the validity period starts.*
- *$\psi.to$: when the validity period ends.*
- *$\psi.issuer$: the CA that issued the certificate.*
- *$\psi.\tau$: issue time.*
- *$\psi.\gamma$: certificate's extensions (may contain any additional data that the issuer is certifying along with the public information and subject fields).*
- *$\psi.\sigma$: the CA's signature over the certificate's fields.*

B. Attribute Attestations

Each certificate can be accompanied with different *attributes attestations*. For example, a certificate ψ can be attested as non-revoked using an additional *attestation information*, which contains the additional information needed to verify the claim that ψ was indeed not revoked. Note that the notion of attestation is closely-related yet significantly different from notion a certification extension.

In practice, the structure of an attestation can be of any form; that said, we now define and later use a specific format, which we found useful in defining different PKI specifications.

Definition 4 (Attribute attestation)**.** *An* attestation of attribute *is a tuple:*

$$\rho = (attr, serial, \iota, tbs, \sigma, \tau)$$

where:

- *$\rho.attr$ is the attribute attested, e.g.,* NREV.
- *$\rho.serial$ is the serial number of the corresponding certificate.*
- *$\rho.\iota$ is the identifier of the attesting entity.*
- *$\rho.tbs$ is the data To Be Signed (tbs) by the attestation.*
- *$\rho.\sigma$ is the proof of the $\rho.tbs$ data.*
- *$\rho.\tau$ is the time of attestation.*

V. The PKI Framework

We now define the PKI framework. In Section V-A, we informally describe the operations which a PKI scheme is required to support, and then formally define what is a PKI scheme using these operations in Section V-B. Then, we formally define the security specifications used in PKI scheme, which we informally introduced in Section II. In Section V-C we define the basic PKI security specifications and in Section V-D we define the advanced security specifications, used in post-X.509 PKI schemes.

A. Informal Overview of PKI Functionalities

We now identify the basic functionalities required to be supported by a PKI scheme, and present an informal description of the PKI scheme definition, which will be formally defined next in §V-B. We begin with the most basic PKI operation, certificate issuance.

1) Certificate issuance: Certificates are *issued* by an entity $\iota \in \mathsf{N}$, as a result of a request generated by the certificate's subject. The subject provides the certificate's details, i.e., the subject identifier and public information (e.g., public key). Then, the issuing entity uses a dedicated operation, which we denote as the Issue operation, to generate the certificate. Namely, this Issue operation bundles the certificate details together according to the relevant format, along with a proof that the issuer has issued that exact certificate. As a result, the issuer's proof (signature) attests that the issuer has sufficiently verified the eligibility of the requesting client to request certificate issuance for the relevant subject. That said, the actual verification process is not part of the properties we model.

2) Certificate revocation: Revocation is a major aspect of PKI schemes, addressed already in X.509. The revoking entity uses a dedicated operation, which we denote as the Revoke operation. Namely, this Revoke operation will mark the certificate as revoked, and optionally, can output relevant related attestations, e.g., revocation attestation (REV). Notice, however, that such output attestations are non-mandatory; indeed, in practice, when CAs are requested to revoke a certificate, they often output an *indication* whether the certificate was revoked or not, but the actual *proof of revocation*, i.e., attestation, is acquired through other mechanisms, e.g., CRLs.

3) Attribute attestation: Once a certificate is issued, authorities in N can *attest* for attributes regarding this certificate. In fact, certificates can be attested to have any kind of attribute, however, such attestations needs to be: (1) well defined, and (2) acceptable by relying parties. A valid non-expired certificate can (theoretically) be attested to be revoked or not revoked by any authority; nevertheless, usually relying parties would only accept such attestations from relevant authorities. For example, accepting revoked and not revoked attestation only from the authority that issued the certificate. We denote a dedicated operation Attest to produce attestations. In this work, we identify two basic types attestations: revocation and non-revocation, along with more advanced attributes: Δ-transparency, Δ-revocation status transparency, Δ-equivocation detection and equivocation prevention. We discuss these attributes in depth in §V-C-§V-D.

Note, however, that the Attest operation might not be able to *immediately* output the requested attestation. The reason is that in some PKI systems, advanced attributes might involve interaction with other authorities, e.g., to avoid equivocation (when required). Therefore, to support such non-immediate response mechanism, the Attest operation can output a *pending* attestation, i.e., a *commitment* to produce the requested attestation (or failure indication), within Λ time, where Λ is typically known from the given attribute or request. Whenever the Λ time period ends, an execution of the Attest algorithm with the pending attestation guaranteed to output the final response, which is either the expected *non-pending* attribute attestation, or $\bot$, i.e., the request was declined.

4) Attestation validity: While Issue, Revoke and Attest are *stateful* operations, checking whether a specific certificate is valid with respect to a specific attribute attestation must be a *stateless* operation, since *anyone* can perform this operation, including clients. As clients do not have any certificate-related state, thus, they must be able to validate certificates without any extra information beyond the actual data to be validated.

Thus, we denote the *stateless* $\mathsf{WasValid}$ function which checks whether a given attribute attestation ρ attested for a certificate ψ *was* valid when ρ was attested with respect to a public key pk. Note that pk can be either an entity's public key, or alternatively, a group public key, e.g., threshold signature verification key.

5) Audit and misbehavior: In order to formalise some of the PKI-related specifications, it is required to be able to query entities with respect to their local state. For example, consider the Δ-transparency specification, which requires that a Δ-transparent certificate will be available to all interested entities within Δ time. Such specification needs a way to query the local state of those interested entities, to make sure that Δ-transparent certificate were indeed available to them. Furthermore, entities might produce a proof-of-misbehavior, i.e., verifiable accusation against other entities' misdeeds.

Hence, we denote the Audit function, which can be invoked in a couple of ways. When Audit is invoked with a subject $subject$ and an attribute $attr \in \mathsf{AttrSet}$, it either returns a valid set of certificates Ψ issued for $subject$ along with their attribute attestations set P, according to the local state, or $\bot$ otherwise (no such certificates). When Audit is invoked with a certificate ψ and an attribute attestation ρ, it outputs $\top$ if the current state indicates that ψ is valid with respect to ρ. In contrast, if the current state indicates that $\rho.attr$ should *not* have been attested to ψ, then Audit outputs an Indicator of Accusation (IA) or a proof of misbehavior ζ. Otherwise, e.g., if the current state does not provide the necessary information, then the algorithm returns $\bot$. To verify if an output proof of misbehavior is valid, we denote the PoM operation.

6) Monitoring: As discussed earlier, non-trivial specifications, e.g., equivocation, require that certificates will be monitored; otherwise, they cannot be guaranteed. Therefore, we denote the $\mathsf{Monitor}$ operation which is used to assign entities to monitor other entities. Namely, an entity invoking the $\mathsf{Monitor}$ operation with an input $\iota \in \mathsf{N}$ starts to monitor ι. As a result, the monitoring entity receives periodic updates from the monitored entity with any changes made from the last update.

7) Additional operations: Finally, a PKI scheme also requires a few basic functions that will help bind the entire scheme together. Namely, a PKI scheme requires an initialization operation, denoted as Init, to allow entities to perform any required operation at the beginning, e.g., generating cryptographic keys locally, exchanging initial information with other entities etc. A second required operation regards handling time-based events, e.g., the ability to perform operations periodically. The assigned operation, denoted as Wakeup, is expected to be invoked whenever a predefined event has occurred. To illustrate, consider CT loggers which are required to periodically update and sign the log; handling such periodic event is done by the Wakeup operation. Lastly, since each entity receives incoming requests, either from other entities or from clients, these requests are handled via a dedicated function, denoted as the $\mathsf{Incoming}$ algorithm.

B. Formal Definition

We now define the basic set of operations which are required from a PKI scheme. However, we emphasize that not all PKI schemes requires to implement all of them, as we demonstrate for X.509 in section VI. Furthermore, some schemes might require additional or optional inputs. Therefore, our specifications should be interpreted as holding for any values of these additional inputs.

For simplicity of exposition, we use a simplified notations and explicitly write only the input parameters of each operation, although the implementation will, obviously, also refer to the clock and the state. (Note that the WasValid operation is *stateless*, hence, its implementation cannot refer to the clock or state.)

Definition 5 (PKI scheme)**.** *A PKI scheme $\mathcal{P}$ is a PPT algorithm, that supports (at least) the following set of operations:*

$$\mathcal{P} = (\mathsf{Init}, \mathsf{Issue}, \mathsf{Revoke}, \mathsf{Attest}, \mathsf{Audit}, \mathsf{WasValid}, \mathsf{Wakeup}, \mathsf{Incoming}, \mathsf{PoM}, \mathsf{Monitor})$$

where:

- $\mathsf{Init}(x) \rightarrow y$*: The algorithm takes as input information x, performs the relevant initialization operations and outputs y.*
- $\mathsf{Issue}(subject, pub, from, to) \rightarrow (\psi, P)/\bot$*: The algorithm takes as input a subjects identifier $subject$, public information pub, start of validity date $from$ and expiration date to, and outputs either a matching certificate $\psi \in \Psi$ along with a set of attribute attestations P, or failure indicator $\bot$.*
- $\mathsf{Revoke}(\psi) \rightarrow P/\bot$*: The algorithm takes as input a certificate $\psi \in \Psi$, and outputs either a set of attribute attestations P, or failure indicator $\bot$.*
- $\mathsf{Attest}(\psi, attr) \rightarrow \rho/\bot$*: The algorithm takes as input a certificate $\psi \in \Psi$ and an attribute attestation $attr \in$, and outputs either an attribute attestation ρ, or failure indicator $\bot$.*
- $\mathsf{Audit}(subject, attr) \rightarrow (\Psi, P)/\bot$ *or* $\mathsf{Audit}(\psi, \rho) \rightarrow \top/IA/\zeta/\bot$*: The algorithm subject identifier $subject$ and attribute $attr \in$ AttrSet, and outputs either a valid set of certificates Ψ along with a set of attribute attestations P, or failure indicator $\bot$. Alternatively, the algorithm takes as input a certificate $\psi \in \Psi$ and attribute attestation ρ, and outputs either $\top$ or $\bot$.*
- $\mathsf{WasValid}(\psi, pk, \rho) \rightarrow \top/\bot$*: This* stateless *algorithm takes as input a certificate $\psi \in \Psi$, a public key* pk*, and an attribute attestation ρ, and outputs either $\top$ or $\bot$.*
- $\mathsf{Wakeup}(data)$*: The algorithm takes as input a wake-up event information $data$ and perform the time-related operation.*
- $\mathsf{Incoming}(x)$*: The algorithm takes as input information x and process it accordingly.*
- $\mathsf{PoM}(pk, \sigma) \rightarrow \top/\bot$*: The algorithm takes as input a public key pk and proof σ, and outputs either $\top$ or $\bot$.*
- $\mathsf{Monitor}(\iota)$*: The algorithm takes as input an entity identifier ι and starts to periodically monitor ι.*

C. Basic PKI Security Specifications

In algorithms 1-7, we define the specification predicates for PKI schemes, corresponding to the security requirements informally introduced in Section II. As per Definition 2, PKI scheme $\mathcal{P}$ fails to satisfy specification ξ, if there is significant (non-negligible) probability that an execution of some PPT adversary $\mathcal{A}$ will result in execution transcript T s.t. $\xi(T) = \bot$, i.e., that $\mathcal{A}$ will 'win'. In this subsection we present the 'basic' specifications: accountability, revocation and non-revocation, which are relevant even to 'classical' PKI schemes such as X.509 [52] and PKIX [53]. The more advanced specifications are in the following subsection.

1) The public-key association convention 'PubKey' and role-based certificates: The goal of PKI is to establish public keys, however, PKI also utilizes known public keys associated with specific identifiers. For example, in TLS, these are referred to as 'anchor' public keys, and associated with well-known entities - the 'root CAs'; and threshold/proactive public-key systems, use keys which are associated with a *role* rather than with any single entity, for example, the public key used to verify periodical re-certifications by the set of entities in the proactive authenticated communication scheme of [54]. Some PKI schemes need such a 'role' which is not a specific entity, e.g., to identify public keys trusted to attest for non-equivocation. We adopt a simple convention to identify such associations of a public key pk with an identifier $role$ (which may be a 'named role' or simply an entity $role \in \mathsf{N}$). The convention is that a non-faulty party output the 'reserved tuple' ('PubKey', $role$, pk).

2) Functions, e.g. IsValidAttr*:* We define few functions, for operations used by multiple predicates; the definition of each function appears with the *first* predicate using this function. For example, the IsValidAttr function checks that the given attestation of a given certificate is valid with for one of the attributes in a given set[2] ATTR of attributes, and signed properly with the specified public key (given as one of the inputs). Furthermore, IsValidAttr validates that this public key, is indeed a public key associated with either an entity or an attribute, denoted as $role$, which is also input to IsValidAttr. To validate this, IsValidAttr uses the public-key association convention just introduced, i.e., it checks that a *non-faulty entity* $\iota \in T.\mathsf{N} - T.\mathsf{F}$, has output the 'reserved tuple' ('PubKey', $role$, pk). In most calls, $role$ identifies one of the entities, $role \in \mathsf{N}$, in fact usually $role = \iota$; but sometimes $role$ is not an entity, e.g., we use it to refer to the public key trusted to attest for non-equivocation.

3) Accountability specification: The accountability predicate (Algorithm 1) returns $\bot$ if accountability fails, namely, if there exists a valid, accountable certificate ψ, whose issuer is honest $\psi.issuer \in T.\mathsf{N} - T.\mathsf{F}$, yet during the execution, the issuer $\psi.issuer$ was *not* instructed to issue a certificate with the subject, public key and validity period in ψ, i.e., there was no *'Issue'* operation in $\psi.issuer$ with these inputs.

[2]Usually we check for only a specific attribute, but in some predicates, we only care if the attestation is for one of multiple attributes.

Algorithm 1 Accountability predicate $\mathsf{ACC}(T)$

```
1: $(\psi, \rho, pk) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho$, and public key $pk$
2: $test \leftarrow$ IsValidAtt($T, \{\mathsf{ACC}\}, \psi, \rho, pk,$
                $\psi.issuer, \psi.issuer$)    ▷ $\psi.issuer$ is honest and $\psi$ was attested by $\psi.issuer$ to be accountable on time $\rho.\tau$
3:         and $\nexists \bar{e}$ s.t. $T.ent[\bar{e}] = \psi.issuer$
               and $T.opr[\bar{e}] =$ 'Issue'
               and $T.inp[\bar{e}] = (\psi.subject, \psi.pub,$
                    $\psi.from, \psi.to)$    ▷ Yet, $\psi.issuer$ was not asked to issue $\psi$
4: if $test$ then return $\bot$ else return $\top$

5: procedure IsValidAtt($T, ATTR, \psi, \rho, pk, \iota, role$)
6:     return $\iota \in T.\mathsf{N} - T.\mathsf{F}$
         and $\rho.attr \in ATTR$
         and $\mathcal{P}.\mathsf{WasValid}(\psi, pk, \rho)$
         and $\exists \hat{e}$ s.t. $T.ent[\hat{e}] = \iota$
             and $T.out[\hat{e}] =$ ('PubKey', $role, pk$)    ▷ Certificate $\psi$ has a valid attestation $\rho$ with an attribute from set[2] ATTR, verified by the public key $pk$ of entity $role$. The mapping of $pk$ to $role$ was outputted by a honest entity $\iota$ (so should be correct).
7: end procedure
```

Algorithm 2 Revocation accountability predicate $\mathsf{ReACC}(T)$

```
1: $(\psi, \rho, pk) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho$, and public key $pk$
2: $test \leftarrow$ IsValidAtt($T, \{\mathsf{REV}\}, \psi, \rho, pk,$
                $\psi.issuer, \psi.issuer$)    ▷ $\psi.issuer$ is honest and $\psi$ was attested by $\psi.issuer$ as revoked on time $\rho.\tau$ (see Alg. 1)
3:         and $\nexists$ $\bar{e}$ s.t. $T.ent[\bar{e}] = \psi.issuer$
               and $T.opr[\bar{e}] =$ 'Revoke'
               and $T.inp[\bar{e}] = \psi$
               and $T.\tau[\bar{e}] \leq \rho.\tau$    ▷ Yet, $\psi.issuer$ was not asked to revoke $\psi$ before time $\rho.\tau$
4: if $test$ then return $\bot$ else return $\top$
```

Algorithm 3 Non-revocation accountability predicate $\mathsf{NReACC}(T)$

```
1: $(\psi, \rho, pk) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho$, and public key $pk$
2:     $test \leftarrow$ IsValidAtt($T, \{\mathsf{NREV}\}, \psi, \rho, pk,$
                $\psi.issuer, \psi.issuer$)    ▷ $\psi.issuer$ is honest and $\psi$ was attested by $\psi.issuer$ as non-revoked on time $\rho.\tau$ (see Alg. 1)
3:         and $\exists \bar{e}$ s.t. $T.ent[\bar{e}] = \psi.issuer$
               and $T.opr[\bar{e}] =$ 'Revoke'
               and $T.inp[\bar{e}] = \psi$
               and $T.\tau[\bar{e}] \leq \rho.\tau$    ▷ Yet, $\psi.issuer$ was asked to revoke $\psi$ before time $\rho.\tau$
4: if $test$ then return $\bot$ else return $\top$
```

Algorithm 4 Equivocation-prevention predicate $\mathsf{EQ\text{-}P}(T)$

```
1: if EquivocationOccured($T$, $\mathsf{EQ\text{-}P}$) then return $\bot$ else return $\top$

2: procedure EquivocationOccured($T, attr$)
3:     $\left( \psi, \psi', \rho_e, \rho'_e, pk_e, pk'_e, \iota, \iota' \right) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificates $\psi$,$\psi'$, attribute attestations $\rho_e, \rho'_e$, public keys $pk_e, pk'_e$ and two (honest) entities $\iota, \iota'$
4:     return IsValidAtt($T, \{attr\}, \psi, \rho_e, pk_e,$
                $\iota, attr$)    ▷ $\psi$ was attested by $\rho_e.\iota$ to be unequivocal on time $\rho_e.\tau$ (see Alg. 1)
5:         and IsValidAtt($T, \{attr\}, \psi', \rho'_e, pk'_e,$
                $\iota', attr$)    ▷ $\psi'$ was attested by $\rho'_e.\iota$ to be unequivocal on time $\rho'_e.\tau$ (see Alg. 1)
6:         and $\psi.subject = \psi'.subject$
           and $\psi.from < \psi'.from < \psi.to$
           and $\psi.pub \neq \psi'.pub$    ▷ $\psi$ and $\psi'$ have the same subject identifier and overlapping validity periods, yet, they have different public information, i.e., $\psi$ and $\psi'$ are equivocating
7:         and $\forall \iota \in T.\mathsf{N} - T.\mathsf{F}$
               $\exists \dot{e}$ s.t. $T.opr[\dot{e}] =$ 'Audit'
                 and $T.ent[\dot{e}] = \iota$
                 and $T.inp[\dot{e}] = \psi.subject$
                 and $T.\tau[\dot{e}] > \rho'_e.\tau$
                 and $\nexists \rho \in T.out[\dot{e}]$ s.t. $\rho.attr =$ 'REV'    ▷ None of the honest entities think that $\psi$ was revoked before the equivocating certificate $\psi'$ was attested as unequivocal
8: end procedure
```

Algorithm 5 $\Delta-$Equivocation detection predicate $\Delta\mathsf{EQ\text{-}D}(T)$

```
1: $\begin{pmatrix} \psi, \psi', pk_s, pk_e, pk'_e, \\ pk_t, \rho_e, \rho'_e, \rho_t \end{pmatrix} \leftarrow T.out_{\mathcal{A}}$    ▷ Certificates $\psi$,$\psi'$, entity $\iota$, public keys $pk_s, pk_e, pk_e, pk_t$, and attribute attestations $\rho_e, \rho'_e, \rho_t, \rho'_t$
2: $test \leftarrow$ EquivocationOccured($T$, $\Delta\mathsf{EQ\text{-}D}$)    ▷ See Alg. 4
3:         and $\forall \iota \in T.\mathsf{N} - T.\mathsf{F}$
             $\exists \dot{e}$ s.t. $T.opr[\dot{e}] =$ 'Audit'
               and $T.ent[\dot{e}] = \iota$
               and $T.inp[\dot{e}] = (\psi', \Delta\mathsf{EQ\text{-}D})$
               and $T.out[\dot{e}] = \top$
               and $T.\tau[\dot{e}] > \Delta + max(\rho_e.\tau, \rho'_e.\tau)$    ▷ No one detected the equivocation during the $\Delta$ time frame
4: if $test$ then return $\bot$ else return $\top$
```

* Note that not all the parameters that are outputted by the adversary in line 1, are used in the algorithm, due to the call for EquivocationOccured in line 3. However, we explicitly wrote all these arguments here, since otherwise, it is harder to understand the context of line 4, for example, the fact that ψ' in line 4 is the second equivocating certificate.

4) Revocation and non-revocation accountability specifications: The *revocation accountability* predicate (Algorithm 2) checks for a valid revocation attestation ρ for a certificate ψ s.t. $\psi.issuer \in T.\mathsf{N} - T.\mathsf{F}$ ($\psi.issuer$ is honest), yet, until the time specified in the attestation (as time of revocation), $\psi.issuer$ was *not* instructed to revoke ψ. In this case, the attacker 'wins' (predicate returns $\bot$).

Similarly, the *non-revocation accountability* predicate checks for a valid non-revocation attestation ρ for a certificate ψ s.t. $\psi.issuer$ is honest, although, before the revocation time as specified in the attestation, $\psi.issuer$ *was* instructed to revoke ψ. Again, if this holds, then the attacker 'wins' (predicate returns $\bot$).

5) Extensions: Our specification does not address *extensions*, an important part of X.509 from version 3. Some extensions are very important and relevant to PKI security, e.g., the basic, naming and length constraints. It is easy to add the extensions as content to the certificates and validate that certificates only contain the extensions specified in 'issue'; the challenge is to reflect the security implications of extensions, mainly, their extensive, non-trivial use for cross-certification. This important issue is beyond our scope.

6) Role-based issuers and subjects: We usually think of the *subject* and *issuer* fields in certificates as *identifiers*, e.g., *X.509 distinguished name* or *domain name*. However, these fields may also contain a 'label' identifying the issuer or subject by their *role* or *property*, or as belonging to a *group*. This type of more general attribution of issuers and/or subjects may have different advantages, such as allowing role-based access control and preserving anonymity; see [55], [56]. In particular, there may be value in systems where (some) certificates are issued by a *group of cooperating entities* - for example, for non-equivocation. This may require extensions to the predicates.

7) Non-issuer revocations: Current predicates only allow revocation and non-revocation attestations by the issuer of the certificate. This is the common approach also in deployed PKI schemes, but there may be value in extending this in the future. Specifically, certain systems might allow entities *other* than the issuers to revoke certificates and attest for their status. Such change, however, must be carefully designed, to avoid undesirable side effects; for instance, if multiple entities may revoke a certificate, then they must be coordinated, to avoid one entity issuing non-revoked attestation to a certificate revoked by another entity.

8) Issuer public key: The predicates rely on the (stateless) function $\mathcal{P}.\mathsf{WasValid}$ to check whether the certificate outputted by the adversary has a claimed attribute. $\mathcal{P}.\mathsf{WasValid}$ needs to be provided with the issuer's public key, to verify thtat the certificate has a specific attribute endorsed with respect to that key. This public key, however, comes from the adversary controlled output $T.out_{\mathcal{A}}$ and cannot be trusted on its own. Hence, we rely on the following convention: each entity ι outputs its own public key in the form $(\text{'PubKey'}, \iota, pk)$. Each time the adversary outputs an attribute attestation, it also outputs the public key of the attesting entity, and the ISVALIDATTR function checks that this public key was indeed outputted by a honest entity (in the $(\text{'PubKey'}, \iota, pk)$ form).

D. Post-X.509-PKI Security specifications

We now define more advanced security specifications, relevant to 'post-X.509' PKI systems, such as Certificate Transparency (CT) [7]. We start the discussion with specifications related to equivocation prevention and detection, and then transparency. While this order of describing the 'strongest' specification, and hence usually most complex to achieve, might be counter intuitive, it results in a more comprehensible presentation.

1) Equivocation prevention ($\mathsf{EQ\text{-}P}$*):* The equivocation prevention predicate (Algorithm 4) is designed to ensure that if equivocation prevention holds, then at no point in time will there exist two unequivocal certificates with different public information issued for the same identifier with overlapping validity periods. This property is relatively easy to achieve if we disregard possible scenarios where an unequivocal certificate ψ is revoked and an unequivocal replacement certificate ψ' needs to be issued, since, in theory ψ' would conflict with ψ due to the same identifier, different public information and overlapping validity.

An overly simplistic solution would be to simply prohibit unequivocal certificates replacements prior to their stated expiration date. We adopt a different approach (described in Algorithm 4), however, that allows replacements and require that if a replacement is issued, then the original certificate must be revoked first. The equivocation prevention predicate checks if an adversary is able to present two conflicting, unequivocal certificates and that none of the honest entities considered the first certificate as revoked prior to the replacement being issued. This approach ensures that the second certificate was not issued in good faith by an honest entity who was led to believe that the first certificate is revoked, i.e., based on a fraudulent revocation attestation issued by an adversary controlled entity.

2) Δ-Equivocation detection ($\Delta\mathsf{EQ\text{-}D}$*):* The Δ-equivocation detection predicate (Algorithm 5) is designed to ensure that if the specification holds, then each occurrence of equivocating certificates will be detected within Δ time by (at least) one honest authority. Equivocation detection provides a 'weaker' guarantee than prevention, since equivocating certificates can be issued, unlike in case of prevention, and only after some time are discovered. However, in terms of complexity of their specifications, the Δ-equivocation detection specification has more requirements than equivocation prevention. Namely, to break equivocation prevention, the adversary needs to show that it is able to produce two equivocating certificates; however, for the adversary to break equivocation detection, it must, *in addition* to the existence of such certificates, show that no honest entity detected the equivocation within Δ. Both specifications rely on a procedure to detect equivocation (EQUIVOCATIONOCCURED, described in Algorithm 4) but the specification of equivocation detection includes additional check that ensures that no honest entity detected equivocation within the Δ time frame.

3) Δ-Transparency (ΔTRA*):* The Δ-transparency predicate (Algorithm 6) is designed to ensure that if Δ-transparency holds, then a Δ-transparent certificate is available to "interested" parties, i.e., *monitors*, within Δ time of its transparency attestation being issued. In order for the adversary to break the transparency guarantee, the adversary must show that for some valid Δ-transparent certificate ψ, which was attested by an entity $\rho.\iota$ on time $\rho.\tau$ as Δ-transparent, there is a honest entity ι_M which monitors $\rho.\iota$, yet ι_M is not aware of ψ although it should, i.e., ι_M is not aware of ψ *after* $\rho.\tau + \Delta$, and ι_M did not accuse $\rho.\iota$ of some misbehavior or has a proof of any misbehavior.

In a typical scenario ι_M monitors $\rho.\iota$, i.e., it receives periodic updates from $\rho.\iota$ about the set of certificates $\rho.\iota$ maintains and has committed to make transparent. If $\rho.\iota$ is honest, then ι_M will receive the certificates and the transparency guarantee will hold. If, on the other hand, $\rho.\iota$ is compromised, i.e., in any way controlled by the adversary, then $\rho.\iota$ can choose one of two strategies: to completely cease to communicate (in which case, ι_M will formally accuse $\rho.\iota$) or to continue to communicate but to withhold a specific certificate ψ from ι_M (in which case ι_M will eventually have a proof that $\rho.\iota$ withheld ψ once it obtains ρ which commits $\rho.\iota$ to make ψ transparent at a specific time). The predicate checks for the proof using the *'Audit'* operation. Consequently, the adversary must be able to withhold ρ from ι_M without ι_M being able to detect it (through an accusation or a proof of misbehavior produced as a result of an *'Audit'* operation on ι_M).

4) Δ-Revocation status transparency (ΔReST*):* While Δ-transparency focuses on the transparency of certificates, the Δ-revocation status transparency specification applies to the certificates' revocation status. A ΔReST attestation ρ_s specifies a certificate status as either revoked or not revoked as endorsed by a specific entity $\rho_s.\iota$. The Δ-revocation status transparency predicate (Algorithm 7) ensures that if the specification holds, then whenever an entity $\rho_t.\iota$ produces an transparency attestation ρ_t of ρ_s, then $\rho_t.\iota$ commits to make ρ_s available to monitors within the Δ time frame. Therefore, similarly to the Δ-transparency predicate, the Δ-revocation status transparency predicate checks if the adversary is able to produce a transparency attestation ρ_t of ρ_s such that there exists a honest monitor ι_M that monitors $\rho_t.\iota$, yet ι_M is unaware of ρ_s after Δ and ι_M did not accuse $\rho_t.\iota$ during Δ as uncooperative, nor it (eventually) has a proof of $\rho_t.\iota$'s misbehavior.

VI. Provably-secure X.509 PKI Scheme

In this section, we formalize the design of the X.509 version 2 scheme (Algorithms 8-13). The resulting construction follows the common practices of the currently deployed X.509-based systems, except for allowing certain extensions for simplicity and clarity of exposition, e.g., naming constraints for cross-certification. The construction supports two revocation mechanisms commonly used in practice, CRLs and OCSP.

A. System Entities and their Local State

In X.509, the set of authorities N consists of *certificate authorities* (CAs) only who perform the same tasks. Each CAs is responsible for issuing and revoking certificates as well as providing information about the certificates it revoked. Each CA maintains a local state s which contains the following information:

- $s.serial$: a counter for the issued certificates; initialized to 0.
- $s.\iota$: the global identifier of the entity.
- $s.sec$: the system's security parameter; initialized within the execution process.
- $s.sk$: the secret signing key.
- $s.pk$: the public verification key.
- $s.certs$: the list of all issued certificates by the specific entity; initialized as an empty list.
- $s.CRL$: the list of all revoked certificates; initialized as an empty list.
- $s.SignedCRL$: proof over the latest CRL.
- $s.\tau$: the value of the current local clock of the specific entity.

B. PKI Algorithms

We now present the implementations of the PKI algorithms specified in Definition 5 relevant to X.509 in Algorithms 8-13. We do not present the Incoming, Monitor, PoM and Audit algorithms as they are not required given the X.509 design and goals. Specifically, there is no interaction between the CAs and consequently, no need for Incoming. X.509 does not provide advanced PKI security features, such as transparency and equivocation, and therefore does not need Monitor, PoM and Audit algorithms.

1) The Init *Algorithm:* The initialization algorithm Init (Algorithm 8) enables each entity to generate a private/public signing key pair; the secret key is stored locally and the algorithm outputs the public key following the convention presented in §V-C. The security parameter $s.sec$ used to generate the keys is defined as a part of the execution process.

2) The Issue *Algorithm:* The certificate issuing algorithm Issue (Algorithm 9) is used by a CA to issue an *accountable* certificate ψ. An honest CA invokes the algorithm only if the requesting client is eligible for the $subject$ to be included in ψ; the specific process of verifying such eligibility varies from CA to CA and is beyond the scope of this paper. The algorithm uses the signing algorithm $\mathcal{S}$.Sign to produce a signature over the specific certificate fields (its serial number, subject, public information, and validity period). The resulting signature serves as the accountability proof, and therefore, the accountability attestation outputted along with the certificate contains the certificate and proof. The algorithm stores the certificate locally and then outputs the certificate along with the accountability attestation.

3) The Revoke *Algorithm:* The certificate revocation Revoke algorithm (Algorithm 10) is used by an *eligible* CA to revoke a certificate ψ. In X.509, an eligible CA is the one who initially issued the specific certificate. CAs revoke

Algorithm 6 $\Delta-$Transparency predicate $\Delta\mathsf{TRA}(T)$

```
1: $(\psi, pk, \rho, \iota_M) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho$, public key $pk$, and entity $\iota_M$
2: $test \leftarrow$ IsValidAtt$(T, \{\Delta\mathsf{TRA}\}, \psi, \rho, pk, \rho.\iota, \rho.\iota)$    ▷ $\psi$ was attested by $\rho.\iota$ to be $\Delta-$transparent on time $\rho.\tau$ (see Alg. 1)
3:          and $\iota_M \in T.\mathsf{N} - T.\mathsf{F}$    ▷ $\iota_M$ is honest and $\iota_M$ monitors $\rho.\iota$ since (at least) $\Delta$ before the transparency-attestation.
            and $\exists\tilde{e}$ s.t. $T.ent[\tilde{e}] = \iota_M$
                and $T.opr[\tilde{e}] =$ ‘Monitor’
                and $T.inp[\tilde{e}] = \rho.\iota$
                and $T.\tau[\tilde{e}] \leq \rho.\tau - \Delta$
4:          and WasNotAccused$(T, \psi, pk, \rho, \iota_M)$    ▷ Monitor $\iota_M$ did not accuse entity $\rho.\iota$ as uncooperative or misbehaving
5:          and $\exists\bar{e}$ s.t. $T.ent[\bar{e}] = \iota_M$    ▷ $\iota_M$ is not aware of $\psi$ even though the $\Delta$ time period passed
                and $T.opr[\bar{e}] =$ ‘Audit’
                and $T.inp[\bar{e}] = \psi.id$
                and $\psi \notin T.out[\bar{e}]$
                and $T.\tau[\bar{e}] \geq \rho.\tau + \Delta$
6: if $test$ then return $\perp$ else return $\top$

7: procedure WasNotAccused$(T, \psi, pk, \rho, \iota)$
8:      return $\nexists\ddot{e}$ s.t. $T.ent[\ddot{e}] = \iota$    ▷ $\iota$ did not ‘indicate/accuse’ $\rho.\iota$, at least not until $\Delta$ time units after attestation
                and $T.out[\ddot{e}] = (\text{IA}, \rho.\iota)$
                and $T.\tau[\ddot{e}] \leq \rho.\tau + \Delta$
9:          and $\exists\check{e}$ s.t. $T.ent[\check{e}] = \iota$    ▷ $\iota$ failed to produce ‘proof’ of $\rho.\iota$’s misbehavior, upon ‘Audit’ with input $\psi$ and $\rho$
                and $T.opr[\check{e}] =$ ‘Audit’
                and $T.inp[\check{e}] = (\psi, \rho)$
                and $\mathcal{P}.\mathsf{PoM}(pk, T.out[\check{e}]) = \perp$
                and $T.\tau[\check{e}] > \rho.\tau + \Delta$
10: end procedure
```

Algorithm 7 $\Delta-$Revocation Status Transparency predicate $\Delta\mathsf{ReST}(T)$

```
1: $(\psi, \rho_t, pk_s, pk_t, \iota_M) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho_t$, public keys $pk_s, pk_t$ and entity $\iota_M$
2: $\rho_s \leftarrow \rho_t.data$    ▷ Extract attestation $\rho_s$ from attestation $\rho_t$
3: $test \leftarrow$ IsValidAtt$(T, \{\mathsf{REV}, \mathsf{NREV}\}, \psi, pk_s, \rho_s, \psi.issuer, \psi.issuer)$    ▷ $\psi$ was attested by $\psi.issuer$ as revoked/non-revoked on time $\rho_s.\tau$ (see Alg. 1)
4:          and IsValidAtt$(T, \{\Delta\mathsf{ReST}\}, \psi, pk_t, \rho_t, \rho_t.\iota, \rho_t.\iota)$    ▷ $\psi$’s revocation status $\rho_s$ was attested by $\rho_t.\iota$ to be $\Delta-$revocation status transparent on time $\rho_t.\tau$ (see Alg. 1)
5:          and $\iota_M \in T.\mathsf{N} - T.\mathsf{F}$    ▷ $\iota_M$ is honest and $\iota_M$ monitors $\rho_t.\iota$ since (at least) $\Delta$ before the transparency-attestation
            and $\exists\tilde{e}$ s.t. $T.ent[\tilde{e}] = \iota_M$
                and $T.opr[\tilde{e}] =$ ‘Monitor’
                and $T.inp[\tilde{e}] = \rho_t.\iota$
                and $T.\tau[\tilde{e}] \leq \rho_t.\tau - \Delta$
6:          and WasNotAccused$(T, \psi, pk_t, \rho_t, \iota_M)$    ▷ Monitor $\iota_M$ did not accuse entity $\rho_t.\iota$ as uncooperative or misbehaving
7:          and $\exists\bar{e}$ s.t. $T.ent[\bar{e}] = \iota_M$    ▷ However, there is an event $\bar{e}$ in $T$ proving that $\rho_s$ did not become transparent after $\Delta$, i.e., there exists an honest monitor $\iota_M$ that is not aware of $\rho_s$ although it should. Namely, either $\rho_s$ attests $\psi$ as non-revoked on $\rho_s.\tau$ but all non-revocation attestations known to $\iota_M$ are older, or $\rho_s$ attests $\psi$ was revoked on $\rho_s.\tau$ but $\iota_M$ has a non-revocation attestation for $\psi$- attested **after** $\rho_t.\tau$
                and $T.opr[\bar{e}] =$ ‘Audit’
                and $T.inp[\bar{e}] = \psi.id$
                and $T.\tau[\bar{e}] \geq \rho_t.\tau + \Delta$
                and (
                    ($\rho_s.attr = \mathsf{NREV}$
                     and $\nexists\rho' \in T.out[\bar{e}]$ s.t.
                     $\rho'.attr = \mathsf{NREV}$ and $\rho'.\tau \geq \rho_s.\tau$)
                  or
                    ($\rho_s.attr = \mathsf{REV}$
                     and $\nexists\rho' \in T.out[\bar{e}]$ s.t.
                     $\rho'.attr = \mathsf{REV}$)
                )
8: if $test$ then return $\perp$ else return $\top$
```

certificates which are not expired or already revoked at the time of the revocation request. Similarly to the process of issuing certificates, an honest CA revokes a certificate only following a legitimate revocation request from the certificate subject.

In X.509, CAs revoke a certificate by locally adding it to their CRLs; however, CAs do not immediately produce and output a *proof* of revocation. Such a proof is only produced later on: either by issuing a periodic CRL update or in response to an OCSP request.

4) The Wakeup *Algorithm:* The time-based operations Wakeup algorithm (Algorithm 11) is used in X.509 for only one operation. Namely, whenever it is time to re-publish the latest CRL, the algorithm is executed. The algorithm takes the latest CRL, signs it and store it locally, so it can be retrieved via the Attest algorithm (Algorithm 12). According to the rfc, the CRL should be republished periodically, even if no new certificates were revoked. Specifically, each publication contains the next update time, and the next publication should occur before the latest one ‘expires’. In practice, many CAs publish updated CRLs frequently, a lot before the ‘next update’ statement; some even republish the CRL after every certificate revocation. Therefore, the Wakeup algorithm outputs the next wake-up request time, specifying when the next CRL update will occur. CAs who wish to sign the CRL after every revocation can do so by outputing a matching wake-up request in the Revoke algorithm (Algorithm 10).

5) The Attest *Algorithm:* In X.509, the certificate attestation algorithm Attest (Algorithm 12) supports two types of attestations: REV, indicating that a certificate is revoked and NREV, indicating that a certificate is *not* revoked. As in the case of revocation, certificate attestation requests are handled by the issuing CAs. Before issuing a specific attestation, the algorithm verifies the request to ensure that only revoked certificates receive the REV attestation and only non-revoked certificates receive the NREV attestation. The algorithm does

Algorithm 8 X.509: initialization algorithm

```
1: procedure Init ()
2:     $(s.sk, s.pk) \leftarrow \mathcal{S}.\mathsf{Gen}(s.sec)$        ▷ Generate signature keys
3:     return ('PubKey', $s.\iota, s.pk$)        ▷ Output public key, see §V-C
4: end procedure
```

Algorithm 9 X.509: certificate issuance algorithm

```
1: procedure Issue($subject, pub, from, to$)
2:     $s.serial \leftarrow s.serial + 1$        ▷ Increase counter
3:     $\psi \leftarrow (s.serial, subject, pub, from,$
           $to, s.\iota, s.\tau)$        ▷ Initialize certificate fields
4:     $\psi.\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(s.sk, \psi)$        ▷ Sign certificate
5:     $s.certs \mathrel{+}= \psi$        ▷ Store certificate locally
6:     $\rho \leftarrow (\mathsf{ACC}, \psi.serial, s.\iota, \psi, \psi.\sigma, \psi.\tau)$        ▷ Accountability attestation
7:     return $(\psi, \{\rho\})$
8: end procedure
```

Algorithm 10 X.509: certificate revocation algorithm

```
1: procedure Revoke($\psi$)
2:     if $\psi \notin s.certs$ or $\psi.to < s.\tau$ then
3:         return $\perp$        ▷ $\psi$ was issued by another CA or is expired
4:     end if
5:     if $\psi.serial \notin s.CRL$ then        ▷ If $\psi$ was not already revoked
6:         $s.CRL \mathrel{+}= \psi.serial$        ▷ Add to local CRL
7:     end if
8:     return $\{\}$
9: end procedure
```

Algorithm 11 X.509: time-based operations

```
1: procedure Wakeup (DATA)
2:     $tbs \leftarrow \{$'CRL'$, s.\iota, s.\tau, s.CRL\}$        ▷ Local CRL data to be signed
3:     $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(s.sk, tbs)$        ▷ Sign CRL data
4:     $s.SignedCRL \leftarrow (tbs, \sigma)$        ▷ Store proof locally
5:     return ('Wake-up request', $x$)        ▷ Set next wake-up for CRL update
6: end procedure
```

not check the expiration date of the certificate as it only issues an attestation to the *existing* state of the certificate and it does not alter it.

An attestation can be produced for both revocation approaches, CRLs and OCSP, and the optional input α is used to indicate the specific method. The output of the algorithm is always in the same format and only the value of $attr$ varies. The procedure of verifying the attestation will vary based on the specific attribute endorsed, however.

6) The WasValid *Algorithm:* The stateless certificate attestation validation algorithm WasValid (Algorithm 13) is used to check the validity of the following attestations: accountability, revocation and non-revocation. For accountability, the

Algorithm 12 X.509: certificate attestation algorithm

```
1: procedure Attest($\psi, attr, \alpha$)
2:     if $\psi \notin s.certs$ or        ▷ $\psi$ issued by another CA
3:       $attr \notin \{\mathsf{REV}, \mathsf{NREV}\}$ or        ▷ or $attr$ not supported
4:       $(attr = \mathsf{REV}$ and $\psi.serial \notin s.CRL)$ or        ▷ or wrong req. (attest non-revoked $\psi$ as revoked or vice-versa)
         $(attr = \mathsf{NREV}$ and $\psi.serial \in s.CRL)$ then
5:         return $\perp$
6:     end if
7:     if $\alpha =$ 'CRL' then
8:         $\rho \leftarrow (attr, \psi.serial, s.SignedCRL)$        ▷ $attr$ attestation
9:     else if $\alpha =$ 'OCSP' then
10:        $tbs \leftarrow \{attr, \alpha, \psi.serial, s.\iota, s.\tau\}$        ▷ OCSP data to be signed
11:        $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(s.sk, tbs)$        ▷ Sign OCSP data
12:        $\rho \leftarrow (attr, tbs, \sigma)$        ▷ $attr$ attestation
13:    end if
14:    return $\rho$
15: end procedure
```

Algorithm 13 X.509: stateless validation algorithm

```
1: procedure WasValid($\psi, pk, \rho$)
2:     if $\rho.attr =$ 'ACC' then        ▷ Check accountability
3:         return $\mathcal{S}.\mathsf{Ver}(pk, \psi, \rho.\sigma)$        ▷ Verify signature
4:     else if $\rho.attr =$ 'REV' then        ▷ Check revocation
5:         if $\rho.\alpha =$ 'CRL' then        ▷ If CRL is used
6:             return $\psi.serial \in \rho.tbs.CRL$        ▷ Ensure $\psi$ is in the CRL
7:                 and $\mathcal{S}.\mathsf{Ver}(pk, \rho.tbs, \rho.\sigma)$        ▷ Ensure signed CRL
8:         else if $\rho.\alpha =$ 'OCSP' then        ▷ If OCSP is used
9:             return $\psi.serial \in \rho.tbs$ and
                   $\mathcal{S}.\mathsf{Ver}(pk, \rho.tbs, \rho.\sigma)$        ▷ Ensure signed OCSP
10:        end if
11:    else if $\rho.attr =$ 'NREV' then        ▷ Check non-revocation
12:        if $\rho.\alpha =$ 'CRL' then        ▷ If CRL is used
13:            return $\psi.serial \notin \rho.tbs.CRL$        ▷ Ensure $\psi$ is not in the CRL
14:                and $\mathcal{S}.\mathsf{Ver}(pk, \rho.tbs, \rho.\sigma)$        ▷ Ensure signed CRL
15:        else if $\rho.\alpha =$ 'OCSP' then        ▷ If OCSP is used
16:            return $\mathcal{S}.\mathsf{Ver}(pk, \rho.tbs, \rho.\sigma)$        ▷ Ensure signed OCSP
17:        end if
18:    end if
19:    return $\perp$        ▷ In any other case
20: end procedure
```

algorithm verifies, using the provided public key pk, that the certificate ψ was correctly signed by $\psi.issuer$ at the time specified in the attribute attestation ρ.

For revocation, the algorithm checks whether the certificate is included in the (correctly) signed CRL provided in the attestation or that the attestation contains a (correctly) signed OCSP revocation statement. Correspondingly, for non-revocation, the algorithm checks whether the certificate *does not* appear in the signed CRL or that the attestation contains a signed OCSP non-revocation statement.

VII. X.509v2 Analysis

A. Model

We now define the model for X.509v2.

1) Adversarial assumptions: Given a set of entities N, the adversary might control any subset of entities F of N ($\mathsf{F} \subseteq \mathsf{N}$) it desires. Hence, since the definitions of the specifications satisfied by X.509v2 rely on the whether a given entity in N belongs in F or not, the adversarial model must enforce that the set of faulty entities $\mathsf{T.F}$ which was outputted by the adversary as part of the execution transcript T, matches the set of entities that the adversary actually controlled during T. We capture this using the $\mathcal{M}^{\mathsf{F}}$ predicate (Algorithm 14).

Algorithm 14 $\mathcal{M}^{\mathsf{F}}(T)$ Predicate

```
1: return (
2:   $\forall \hat{e} \in T.e$ :                                      ▷ For each event
3:     if $T.opr[\hat{e}] \in \{$‘Get-state’, ‘Set-state’, ‘Set-output’$\}$   ▷ If the operation means the adversary controls the entity
4:     then $T.ent[\hat{e}] \in T.\mathsf{F}$                          ▷ Then entity is in $T.\mathsf{F}$
   )
```

2) Communication assumptions: None of the entities in X.509v2 directly communicate with one another, thus, there are no communication assumptions.

3) Synchronization assumptions: The only clock-synchronization assumption in X.509v2 is that the drift between clock values provided by the adversary (as input during the execution process), and the ‘real time’ values is *bounded*. We present this as the $\mathcal{M}^{\text{Drift}}_{\Delta_{clk}}$ predicate. It enforces two requirements on the execution: each local-clock value ($clk[\hat{e}]$) must be within Δ_{clk} drift from the real time $\tau[\hat{e}]$, and the real time values should be monotonously increasing. As a special case, when $\Delta_{clk} = 0$, this predicate corresponds to a model where the local clocks are fully synchronized, i.e., there is no difference between entities' clocks. See Algorithm 15.

Algorithm 15 $\mathcal{M}^{\text{Drift}}_{\Delta_{clk}}(T)$ Predicate

```
1: return (
2:   $\forall \hat{e} \in T.e$:                                  ▷ For each event
3:     $|T.clk[\hat{e}] - T.\tau[\hat{e}]| \leq \Delta_{clk}$          ▷ Local clock is within $\Delta_{clk}$ drift from real time
4:     and
5:       if $\hat{e} \geq 2$                                   ▷ And if $\hat{e}$ is not the first event
6:       then $T.\tau[\hat{e}] \geq T.\tau[\hat{e}-1]$             ▷ Then the real time is $\geq$ the real time at the previous event
   )
```

Finally, we get the $\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}$ model for the X. 509v2 protocol, which is a conjunction of the aforementioned assumptions, i.e.:

$$\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}} = \mathcal{M}^{\mathsf{F}} \ \wedge \ \mathcal{M}^{\text{Wake-up}}_{\Delta_{clk}} \ \wedge \ \mathcal{M}^{\text{Drift}}_{\Delta_{clk}} \tag{4}$$

In addition, the model verifies that the X.509v2.Wakeup algorithm was invoked in the beginning of the execution and at the correct time-based intervals. To that end, we define the $\mathcal{M}^{\text{Wake-up}}_{\Delta_{clk}}$ predicate, which ensures that if (‘Wake-up request’, x) was part of the output $out[\hat{e}]$ of any operation during the execution process and the execution did not terminate by 'real' time $\tau[\hat{e}] + x + \Delta_{clk}$, then at some event $\hat{e}' > \hat{e}$ (where $\tau[\hat{e}']$ was within Δ_{clk} from $\tau[\hat{e}] + x$), the same entity ($ent[\hat{e}]$) was indeed ‘Woken up’. The $\mathcal{M}^{\text{Wake-up}}_{\Delta_{clk}}$ predicate appears in Algorithm 16.

Algorithm 16 $\mathcal{M}^{\text{Wake-up}}_{\Delta_{clk}}(T)$ Predicate

```
1: return (
2:   and $\forall \hat{e} \in T.e$ :                                         ▷ For each event
3:     if ( (‘Wake-up request’, $x) \in T.out[\hat{e}]$                       ▷ If the output includes a (‘Sleep’, $x$) tuple
4:         and $T.\tau[T.e] \geq T.\tau[\hat{e}] + x + \Delta_{clk}$ )          ▷ And execution did not terminate yet after $x + \Delta_{clk}$ real time
5:     then $\exists \hat{e}' > \hat{e}$                                        ▷ Then there is a later event
6:         s.t. $|T.\tau[\hat{e}'] - T.\tau[\hat{e}] - x| \leq \Delta_{clk}$      ▷ With real time $x$ greater than at $\hat{e}$ (within $\Delta_{clk}$)
7:         and $T.ent[\hat{e}'] = T.ent[\hat{e}]$                              ▷ In which the entity is the same as in $\hat{e}$
8:         and $T.opr[\hat{e}'] =$ ‘Wakeup’                                  ▷ And the operation is ‘Wakeup’
   )
```

B. Security Analysis

Theorem 1 (X. 509v2 (asymptotically) satisfies accountability, revocation accountability and non-revocation accountability under model $\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}$)**.** *Let $\mathcal{S}$ be an existentially-unforgeable signature scheme and let N be a set of entities. Then* $\text{X. 509v2}^{\mathcal{S}}_{\mathsf{N}}$ *satisfies the accountability, revocation accountability and non-revocation accountability specifications under model* $\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}$.

Proof sketch. To prove that $\text{X. 509v2}^{\mathcal{S}}_{\mathsf{N}}$ satisfies accountability, revocation accountability and non-revocation accountability, we use the following methodology:

1) We first define a variation of $\text{X. 509v2}^{\mathcal{S}}_{\mathsf{N}}$ called $\overline{\text{X. 509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$, where a PPT oracle algorithm $OSign(sk, \cdot)$ is used to generate signatures using a secret key sk *instead* of entity $\iota \in \mathsf{N}$, where sk is the matching secret signing key of the verification key pk.
2) Then, we define a game called $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$, where we execute an adversary $\mathcal{A}$ with the $\overline{\text{X. 509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ scheme, and ask $\mathcal{A}$ to output a message m and signature σ over m, where σ can be verified using the public verification key pk; namely, without $\mathcal{A}$ knowing the matching signing key sk, nor $\mathcal{A}$ can use the oracle access to sign m.
3) We then formulate Lemma 1, showing that the existence of an adversary that ‘wins’ the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game with non-negligible probability means that $\mathcal{S}$ is not a secure signature scheme.
4) We then prove that if $\text{X. 509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not IA-satisfies $\xi_{\mathsf{ACC}}, \xi_{\mathsf{REV}}$ and ξ_{NReACC}, then we can construct an ad-

versary that wins the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{S}_{\mathrm{N}}}$ game with non-negligible probability.

5) Finally, we revisit Theorem 1 and complete its proof by combining steps $1-4$. For the detailed analysis, see Appendix B.

□

VIII. Conclusions and Future Work

In this work, we presented a PKI framework to achieve provably-secure PKI schemes. Our construction can be easily adopted by existing and future PKI schemes, and if necessary, can also be extended to provide additional properties not captured in this work. The next steps are to apply the framework into other PKI schemes, e.g., X.509 version 3 and CT, and provably analyze the specifications achieved by such schemes.

References

[1] B. B. CCITT, "Recommendations X. 509 and ISO 9594-8," *Information Processing Systems-OSI-The Directory Authentication Framework (Geneva: CCITT)*, 1988.

[2] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446 (Proposed Standard), RFC Editor, Fremont, CA, USA, pp. 1–160, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8446.txt

[3] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS Certificate Ecosystem," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 291–304.

[4] J. Prins, "DigiNotar Certificate Authority breach "Operation Black Tulip"," 2011.

[5] Comodo™, "Incident Report," Published online, http://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html, March 2011.

[6] J. Dyer, "China Accused of Doling Out Counterfeit Digital Certificates in 'Serious' Web Security Breach," VICE News, April 2015.

[7] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962 (Experimental), RFC Editor, Fremont, CA, USA, Jun. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6962.txt

[8] B. Laurie and E. Kasper, "Revocation Transparency," *Google Research, September*, 2012.

[9] M. D. Ryan, "Enhanced certificate transparency and end-to-end encrypted mail," in *NDSS*, 2014.

[10] P. Eckersley, "Sovereign Key Cryptography for Internet Domains," https://git.eff.org/?p=sovereign-keys.git;a=blob;f=sovereign-key-design.txt;hb=HEAD, 2012.

[11] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman, "CONIKS: Bringing Key Transparency to End Users," in *USENIX Security Symposium*, 2015, pp. 383–398.

[12] T. H.-J. Kim, L.-S. Huang, A. Perrig, C. Jackson, and V. Gligor, "Accountable Key Infrastructure (AKI): A Proposal for a Public-Key Validation Infrastructure," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 679–690.

[13] P. Szalachowski, S. Matsumoto, and A. Perrig, "PoliCert: Secure and Flexible TLS Certificate Management," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 406–417.

[14] D. Basin, C. Cremers, T. H.-J. 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*. ACM, 2014, pp. 382–393.

[15] J. Yu, V. Cheval, and M. Ryan, "DTKI: A New Formalized PKI with Verifiable Trusted Parties," *The Computer Journal*, vol. 59, no. 11, pp. 1695–1713, 2016.

[16] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, and B. Ford, "Certificate Cothority: Towards Trustworthy Collective CAs," *Hot Topics in Privacy Enhancing Technologies (HotPETs)*, vol. 7, 2015.

[17] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning," in *Security and Privacy (SP), 2016 IEEE Symposium on*. Ieee, 2016, pp. 526–545.

[18] S. Matsumoto and R. M. Reischuk, "IKP: Turning a PKI Around with Decentralized Automated Incentives," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 410–426.

[19] C. Fromknecht, D. Velicanu, and S. Yakoubov, "A Decentralized Public Key Infrastructure with Identity Retention," *IACR Cryptology ePrint Archive*, vol. 2014, p. 803, 2014.

[20] L. Axon and M. Goldsmith, "PB-PKI: A Privacy-aware Blockchain-based PKI," in *SECRYPT*, 2017.

[21] A. Tomescu and S. Devadas, "Catena: Efficient Non-equivocation via Bitcoin," in *2017 38th IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 393–409.

[22] M. Y. Kubilay, M. S. Kiraz, and H. A. Mantar, "CertLedger: A new PKI model with Certificate Transparency based on blockchain," *arXiv preprint arXiv:1806.03914*, 2018.

[23] B. Dowling, F. Günther, U. Herath, and D. Stebila, "Secure Logging Schemes and Certificate Transparency," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 140–158.

[24] M. Chase and S. Meiklejohn, "Transparency Overlays and Applications," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 168–179.

[25] C. Cremers, "Key Exchange in IPsec revisited Formal Analysis of IKEv1 and IKEv2," in *European Symposium on Research in Computer Security*. Springer, 2011, pp. 315–334.

[26] R. Canetti and H. Krawczyk, "Security Analysis of IKE's Signature-Based Key-Exchange Protocol," in *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, ser. Lecture Notes in Computer Science, M. Yung, Ed., vol. 2442. Springer, 2002, pp. 143–161.

[27] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, "Authenticated Confidential Channel Establishment and the Security of TLS-DHE," *J. Cryptology*, vol. 30, no. 4, pp. 1276–1324, 2017.

[28] P. Morrissey, N. Smart, and B. Warinschi, "The TLS Handshake Protocol: A Modular Analysis," *Journal of Cryptology*, vol. 23, pp. 187–223, Apr. 2010.

[29] R. Canetti, D. Shahaf, and M. Vald, "Universally Composable Authentication and Key-exchange with Global PKI," in *Public-Key Cryptography–PKC 2016*. Springer, 2016, pp. 265–296.

[30] A. Boldyreva, M. Fischlin, A. Palacio, and B. Warinschi, "A Closer Look at PKI: Security and Efficiency," in *International Workshop on Public Key Cryptography*. Springer, 2007, pp. 458–475.

[31] S. Gajek, M. Manulis, O. Pereira, A.-R. Sadeghi, and J. Schwenk, "Universally Composable Security Analysis of TLS," in *International Conference on Provable Security*. Springer, 2008, pp. 313–327.

[32] U. Maurer, "Modelling a Public-Key Infrastructure," in *European Symposium on Research in Computer Security*. Springer, 1996, pp. 325–350.

[33] J. Marchesini and S. Smith, "Modeling Public Key Infrastructure in the Real World," in *European Public Key Infrastructure Workshop*. Springer, 2005, pp. 118–134.

[34] J. Braun, F. Kiefer, and A. Hülsing, "Revocation & Non-Repudiation: When the first destroys the latter," in *European Public Key Infrastructure Workshop*. Springer, 2013, pp. 31–46.

[35] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*. IEEE, 2000, pp. 2–14.

[36] D. Lekkas, "Establishing and managing trust within the Public Key Infrastructure," *Computer Communications*, vol. 26, no. 16, pp. 1815–1825, 2003.

[37] M. Zhou, P. Bisht, and V. Venkatakrishnan, "Strengthening XSRF Defenses for Legacy Web Applications Using Whitebox Analysis and Transformation," in *Information Systems Security*. Springer, 2011, pp. 96–110.

[38] W. A. Samer, L. Romain, B. Francois, and B. AbdelMalek, "A formal model of trust for calculating the quality of X. 509 certificate," *Security and Communication Networks*, vol. 4, no. 6, pp. 651–665, 2011.

[39] J. Braun, "Maintaining Security and Trust in Large Scale Public Key Infrastructures," Ph.D. dissertation, Technische Universität, 2015.

[40] J. Huang and D. M. Nicol, "An anatomy of trust in public key infrastructure," *International Journal of Critical Infrastructures*, vol. 13, no. 2-3, pp. 238–258, 2017.

[41] "The Modular Specifications Security Framework," Anonymized at the request of the authors.

[42] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Tech. Rep., 2008.
[43] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," RFC 6960 (Proposed Standard), RFC Editor, Fremont, CA, USA, Jun. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6960.txt
[44] S. B. Roosa and S. Schultze, "The "Certificate Authority" Trust Model for SSL: A Defective Foundation for Encrypted Web Traffic and a Legal Quagmire," *Intellectual property & technology law journal*, vol. 22, no. 11, p. 3, 2010.
[45] H. Asghari, M. Van Eeten, A. Arnbak, and N. A. van Eijk, "Security Economics in the HTTPS Value Chain," in *Twelfth Workshop on the Economics of Information Security (WEIS 2013), Washington, DC*, 2013.
[46] J. Hruska, "Apple, Microsoft buck trend, refuse to block unauthorized Chinese root certificates," ExtremeTech, April 2015.
[47] D. Wendlandt, D. G. Andersen, and A. Perrig, "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing." in *USENIX Annual Technical Conference*, vol. 8, 2008, pp. 321–334.
[48] Electronic Frontier Foundation (EFF). The EFF SSL Observatory. [Online]. Available: https://www.eff.org/observatory
[49] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 78–81, 2009.
[50] "Namecoin." [Online]. Available: https://www.namecoin.org/
[51] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 696–701.
[52] International Telecommunication Union, "Recommendation ITU-T X.509, OSI – The Directory: Public-Key and Attribute Certificate Frameworks," 2016. [Online]. Available: https://www.itu.int/rec/T-REC-X.509-201610-I/en
[53] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, May 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5280.txt
[54] R. Canetti, S. Halevi, and A. Herzberg, "Maintaining authenticated communication in the presence of break-ins," *J. Cryptology*, vol. 13, no. 1, pp. 61–105, 2000.
[55] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access control meets public key infrastructure, or: Assigning roles to strangers," in *IEEE Symposium on Security and Privacy*, 2000, pp. 2–14.
[56] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," RFC 2693 (Experimental), RFC Editor, Fremont, CA, USA, Sep. 1999. [Online]. Available: https://rfc-editor.org/rfc/rfc2693.txt

Appendix A
Secure Signature Scheme

We recall the definition and security game of a secure signature scheme, which $\mathrm{X.\,509v2}$'s security relies upon.

Definition 6. *A signature scheme* $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *consists of the following probabilistic algorithms:*

- *Key generation* $\mathsf{Gen}(1^\kappa) \to (sk, vk)$*, with input security parameter* 1^κ *and output private signing key* sk *and public verification key* vk*.*
- *Signing* $\mathsf{Sign}(sk, m) \to (\sigma)$*, with input private signing key* sk *and a message* m*, and output signature* σ*.*
- *Verification* $\mathsf{Ver}(vk, m, \sigma) \to (\top/\bot)$*, with inputs public verification key* vk*, message* m *and signature* σ*, and output: true* $(\top)$ *if* σ *is a valid signature over* m*, otherwise false* $(\bot)$*.*

Definition 7. *A signature scheme* $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *is **existentially unforgeable** if for every PPT adversary* $\mathcal{A}$*:*

$$Pr\left[\mathbf{Exp}^{EU}_{\mathcal{S},\mathcal{A}}(1^\kappa) = 1\right] \in Negl(1^\kappa)$$

where $\mathbf{Exp}^{EU}_{\mathcal{S},\mathcal{A}}(1^\kappa)$*:*

1) $(sk, vk) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\kappa)$
2) *Adversary* $\mathcal{A}$ *receives* vk *and has an oracle access to* $\mathcal{S}.\mathsf{Sign}$ *to sign any message it desires.*
3) $\mathcal{A}$ *outputs message* m *and signature* σ*.*
4) *The experiment outputs* 1 *if* $\mathcal{S}.\mathsf{Ver}(vk, m, \sigma) = \top$ *and* $\mathcal{A}$ *did not use the oracle access on* m*, otherwise, the experiment outputs* 0*.*

Appendix B
X.509v2 Analysis Detailed Proofs

A. The $\overline{\mathrm{X.\,509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ scheme

We begin by defining a variation of the $\mathrm{X.\,509v2}^{\mathcal{S}}_{\mathsf{N}}$ scheme, denoted as $\overline{\mathrm{X.\,509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$.

Definition 8. *Let* $\mathcal{S}$ *be a signature scheme and let* $(sk, pk) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\kappa)$*, for a given security parameter* 1^κ*. Given a PPT oracle* $OSign(sk, \cdot)$*, let* $\overline{\mathrm{X.\,509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ *be a PKI scheme where one designated authority* $\iota \in \mathsf{N}$ *executes the* $\mathrm{X.\,509v2}^{\mathcal{S}}_{\mathsf{N}}$ *scheme with the following changes, and the rest of the authorities in* N *execute* $\mathrm{X.\,509v2}^{\mathcal{S}}_{\mathsf{N}}$ *without any changes:*

1) *The* $\overline{\mathrm{X.\,509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}.\mathsf{Init}$ *algorithm only executes the following code:*

   $$\textbf{\textit{return}}\ (\text{'}PubKey\text{'}, s.\iota, pk)$$

   where pk *is the public verification key of the* sk *signing key, given as input to* $\overline{\mathrm{X.\,509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ *and* $s.\iota$ *is the identifier of the local entity.*
2) *In all the algorithms that use the signing key,* replace *the use of the signing key with* $OSign(sk, \cdot)$ *oracle access. Namely, replace every:*

   $$\mathcal{S}.\mathsf{Sign}(s.sk, \cdot)$$

   with matching:

   $$OSign(sk, \cdot)$$

B. The $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ Game

We now define the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^\kappa, \mathsf{N})$ game:

1) Generate key pair $(sk, pk) \leftarrow \mathcal{S}.\mathsf{Gen}(1^\kappa)$.
2) Randomly choose an authority $\iota \overset{R}{\leftarrow} \mathsf{N}$.
3) Execute $\mathcal{A}$ with the $\overline{\mathrm{X.\,509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ scheme, i.e.,

   $$T \leftarrow \mathbf{Exec}_{\mathcal{A},\overline{\mathrm{X.509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(1^\kappa)$$
4) $\mathcal{A}$ outputs message m and signature σ, i.e., $(m, \sigma) \leftarrow T.out_{\mathcal{A}}$.
5) The experiment outputs 1 if:
   a) $\mathcal{S}.\mathsf{Ver}(pk, m, \sigma) = \top$
   b) $\mathcal{A}$ did not use the oracle access on m.
   c) $\mathcal{A}$ satisfies model $\mathcal{M}$ (see Def. 1).

d) ι is an honest authority, i.e., $\iota \in \mathsf{N} - \mathsf{F}$

Otherwise, the experiment outputs 0.

C. The Relation Between $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ and the Security of the Signature Scheme $\mathcal{S}$

We now show that the existence of an adversary that 'wins' the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game with non-negligible probability means that $\mathcal{S}$ is not a secure signature scheme.

Lemma 1. *If there is a PPT adversary $\mathcal{A}$ that satisfies*

$$Pr\left[\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^\kappa,\mathsf{N})=1\right]\notin Negl(1^\kappa) \tag{5}$$

then $\mathcal{S}$ is not a secure signature scheme.

Proof. Assume to the contrary that such adversary $\mathcal{A}$ exists, yet $\mathcal{S}$ is a secure signature scheme.

Following §B-B, if $\mathcal{A}$ 'wins' the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game with non-negligible probability, then this means that $\mathcal{A}$ can output a message m and a valid signature σ over m, where $\mathcal{A}$ has only access to the verification key and oracle accesses to the signing key, without requesting the oracle to sign m.

Therefore, according to definition of existential unforgeability (Def. 7), the following holds for $\mathcal{A}$

$$Pr\left[\mathbf{Exp}^{EU}_{\mathcal{A},\mathcal{S}}(1^\kappa)=1\right]\notin Negl(1^\kappa) \tag{6}$$

thus contradicting the security of $\mathcal{S}$. □

D. Linking Accountability, Revocation Accountability, and Non-Revocation Accountability to the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}}$ Game

We now show that if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not ensures accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then we can construct an adversary that wins in the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game.

Claim 1. *If $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not ensures accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{ACC}}$ such that*

$$Pr\left[\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A}_{\mathsf{ACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^\kappa,\mathsf{N})=1\right]\notin Negl(1^\kappa) \tag{7}$$

Proof. From Equation 3, if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not ensures accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{ACC}}$ that satisfies

$$Pr\begin{bmatrix}(T)\leftarrow\mathbf{Exec}_{\mathcal{A}_{\mathsf{ACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^\kappa)\\ \mathbf{Exp}^{\mathsf{ACC}}_{\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(T,1^\kappa)=\top\end{bmatrix}\notin Negl(1^\kappa) \tag{8}$$

Therefore, all that is left is to show that if Eq. 8 holds then Eq. 7 also holds.

First, according to the description of the security experiment $\mathbf{Exp}^{\mathsf{ACC}}_{\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ (Alg. 1), the return value of the experiment is TRUE only if, among other criteria, the following is TRUE:

$$\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}(\psi,pk,\rho) \tag{9}$$

for a certificate ψ outputted by the adversary, ρ is accountability attestation ($\rho.attr=\mathsf{ACC}$), and pk is the public key of $\psi.issuer$ (the issuer of the certificate), which is an honest authority that did not issue ψ by executing the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Issue}$ algorithm.

Second, according to the implementation of $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}$, as described in Alg. 13, the algorithm executes

$$\mathcal{S}.\mathsf{Ver}(pk,\psi,\rho.\sigma) \tag{10}$$

Lastly, the only place in $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ where an honest authority ι computes its keys is in the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Init}$ algorithm (Algorithm 8); specifically the sign/verify key pair is generated in line 2, using the $\mathcal{S}.\mathsf{Gen}$ algorithm. Furthermore, the signing key is only used in algorithms: $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Issue}$, $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Wakeup}$, and $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Attest}$, and only with the $\mathcal{S}.\mathsf{Sign}$ algorithm; however, certificates can only be issued in X.509v2 using the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Issue}$ algorithm.

Thus, following Eq. 8, the value described in Eq. 9 must be TRUE, and as a result, Eq. 10 must also equal TRUE. Accordingly, with accordance to X.509v2's implementation, adversary $\mathcal{A}_{\mathsf{ACC}}$ is a PPT adversary that for a message $m=\psi$ was able to generate a signature $\sigma=\rho.\sigma$ that is validated with non-negligible probability with the verification key pk, without access to the signing key, and without ever having the honest authority $\psi.issuer$ sign m. Hence, such $\mathcal{A}_{\mathsf{ACC}}$ adversary satisfies Eq. 7. □

We now show that if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not ensures revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then we can construct an adversary that wins in the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game.

Claim 2. *If X.509v2 does not achieves revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{ReACC}}$ such that*

$$Pr\left[\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A}_{\mathsf{ReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^\kappa,\mathsf{N})=1\right]\notin Negl(1^\kappa) \tag{11}$$

Proof. From Equation 3, if X.509v2 does not ensures revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{ReACC}}$ that satisfies

$$Pr\begin{bmatrix}(T)\leftarrow\mathbf{Exec}_{\mathcal{A}_{\mathsf{ReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^\kappa)\\ \mathbf{Exp}^{\mathsf{ReACC}}_{\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(T,1^\kappa)=\top\end{bmatrix}\notin Negl(1^\kappa) \tag{12}$$

Therefore, all that is left is to show that if Eq 12 holds then Eq 11 also holds.

First, according to the description of the security experiment $\mathbf{Exp}^{\mathsf{ReACC}}_{\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ (Alg. 2), the return value of the experiment is TRUE only if, among other criteria, the following is TRUE:

$$\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}(\psi, pk, \rho) \tag{13}$$

for a certificate ψ outputted by the adversary, ρ is non-revocation accountability attestation ($\rho.attr = \mathsf{REV}$), and pk is the public key of $\rho.issuer$ (the issuer of the certificate), that did not revoke ψ by executing the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Revoke}$ algorithm.

Second, according to the implementation of $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}$, as described in Alg. 13, the algorithm executes

$$\mathcal{S}.\mathsf{Ver}(pk, \rho.tbs, \rho.\sigma) \tag{14}$$

Lastly, the only place in $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ where an honest authority ι computes its keys is in the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Init}$ algorithm (Algorithm 8); specifically the sign/verify key pair is generated in line 2, using the $\mathcal{S}.\mathsf{Gen}$ algorithm. Furthermore, the signing key is only used in algorithms: $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Issue}$, $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Wakeup}$, $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Attest}$, and only with the $\mathcal{S}.\mathsf{Sign}$ algorithm; however, certificates can only be have the non-revocation accountability attribute in X.509v2 using the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Attest}$ algorithm.

Thus, following Eq. 12, the value described in Eq. 13 must be TRUE, and as a result, Eq. 14 must also equal TRUE. Accordingly, with accordance to $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$'s implementation, adversary $\mathcal{A}_{\mathsf{NREV}}$ is a PPT adversary that for a message $m = \rho.tbs$ was able to generate a signature $\sigma = \rho.\sigma$ that is validated with non-negligible probability with the verification key pk, without access to the signing key, and without ever having the honest authority $\rho.\iota$ sign m. Hence, such $\mathcal{A}_{\mathsf{NREV}}$ adversary satisfies Eq. 15. □

We now show that if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not ensures non-revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then we can construct an adversary that wins in the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game.

Claim 3. *If* X.509v2 *does not achieves non-revocation accountability under model* $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$*, then there exists a PPT adversary* $\mathcal{A}_{\mathsf{ReACC}}$ *such that*

$$Pr\left[\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A}_{\mathsf{NREV}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^{\kappa}, \mathsf{N}) = 1\right] \notin Negl(1^{\kappa}) \tag{15}$$

Proof. From Equation 3, if X.509v2 does not ensures non-revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{NREV}}$ that satisfies

$$Pr\left[\begin{array}{l}(T) \leftarrow \mathbf{Exec}_{\mathcal{A}_{\mathsf{NREV}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^{\kappa}) \\ \mathbf{Exp}^{\mathsf{NREV}}_{\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(T, 1^{\kappa}) = \top\end{array}\right] \notin Negl(1^{\kappa}) \tag{16}$$

Therefore, all that is left is to show that if Eq 16 holds then Eq 15 also holds.

First, according to the description of the security experiment $\mathbf{Exp}^{\mathsf{NREV}}_{\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ (Alg. 3), the return value of the experiment is TRUE only if, among other criteria, the following is TRUE:

$$\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}(\psi, pk, \rho) \tag{17}$$

for a certificate ψ outputted by the adversary, ρ is non-revocation accountability attestation ($\rho.attr = \mathsf{NREV}$), and pk is the public key of $\rho.issuer$ (the issuer of the certificate), is an honest authority that did not revoke ψ by executing the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Revoke}$ algorithm.

Second, according to the implementation of $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}$, as described in Alg. 13, the algorithm executes

$$\mathcal{S}.\mathsf{Ver}(pk, \rho.tbs, \rho.\sigma) \tag{18}$$

Lastly, the only place in $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ where an honest authority ι computes its keys is in the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Init}$ algorithm (Algorithm 8); specifically the sign/verify key pair is generated in line 2, using the $\mathcal{S}.\mathsf{Gen}$ algorithm. Furthermore, the signing key is only used in algorithms: $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Issue}$, $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Wakeup}$, $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Attest}$, and only with the $\mathcal{S}.\mathsf{Sign}$ algorithm; however, certificates can only be have the non-revocation accountability attribute in X.509v2 using the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Attest}$ algorithm.

Thus, following Eq. 16, the value described in Eq. 17 must be TRUE, and as a result, Eq. 18 must also equal TRUE. Accordingly, with accordance to $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$'s implementation, adversary $\mathcal{A}_{\mathsf{NREV}}$ is a PPT adversary that for a message $m = \rho.tbs$ was able to generate a signature $\sigma = \rho.\sigma$ that is validated with non-negligible probability with the verification key pk, without access to the signing key, and without ever having the honest authority $\rho.\iota$ sign m. Hence, such $\mathcal{A}_{\mathsf{NREV}}$ adversary satisfies Eq. 15. □

E. Completing the Proof

Now, we revisit Theorem 1, and complete its proof.

Theorem 1 (X.509v2 (asymptotically) satisfies accountability, revocation accountability and non-revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$)**.** *Let* $\mathcal{S}$ *be an existentially-unforgeable signature scheme and let* N *be a set of entities. Then* $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ *satisfies the accountability, revocation accountability and non-revocation accountability specifications under model* $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$.

Proof. The proof for all four properties is essentially identical; we present the argument for *accountability* and later discuss the (trivial) adaptions for the other three properties.

Assume, therefore, that $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not achieve *accountability*, and we will show that this implies that $\mathcal{S}$ is not a secure signature scheme. According to Claim 1, this means there exists a PPT adversary $\mathcal{A}$ that wins the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game with non-negligible probability. Note that Claim 1 refers to this adversary as $\mathcal{A}_{\mathsf{ACC}}$; the argument follows by substituting $\mathcal{A}$ in Eq. (7).

Similarly, from Claims 2-3, if X. 509v2 does not achieve *revocation accountability or non-revocation accountability*, then there a PPT adversary that wins the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game with non-negligible probability.

However, Lemma 1 shows that if there exists a PPT adversary $\mathcal{A}$ that wins the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game with non-negligible probability, then $\mathcal{S}$ is not a secure signature scheme. □

Appendix C
Execution Process

Algorithm 17 Adversary-Driven Execution Process $\mathbf{Exec}_{\mathcal{A},\mathcal{P}}(p)$

```
1: $(s_{\mathcal{A}}, \mathsf{N}) \leftarrow \mathcal{A}(p)$    ▷ Initialize adversary
2: $\forall i \in \mathsf{N}:\ s_i \leftarrow \mathcal{P}(\perp, \text{‘Init’}, (i, p), \perp)$    ▷ Initialize entities’ local state
3: $e \leftarrow 0$    ▷ Initialize loop’s counter
4: repeat
5:     $e \leftarrow e + 1$    ▷ Advance loop’s counter
6:     $(ent[e], opr[e], inp[e], clk[e], \tau[e]) \leftarrow \mathcal{A}(s_{\mathcal{A}})$    ▷ $\mathcal{A}$ selects entity $ent[e]$, operation $opr[e]$, input $inp[e]$, clock $clk[e]$, and real time $\tau[e]$ for event $e$
7:     if $opr[e] =$‘Set-state’ then    ▷ When $\mathcal{A}$ wants to change the local state of entity $ent[e]$
           $s_{ent[e]}, out[e], sec\text{-}out[e][\cdot] \leftarrow inp[e], \perp, \perp$
8:     else if $opr[e] =$‘Set-output’ then    ▷ When $\mathcal{A}$ wants to change the output of entity $ent[e]$
           $out[e], sec\text{-}out[e][\cdot] \leftarrow inp[e]$
9:     else if $opr[e] =$‘Get-state’ then    ▷ When $\mathcal{A}$ wants to get the current local state of entity $ent[e]$
           $out[e], sec\text{-}out[e][\cdot] \leftarrow s_{ent[e]}, \perp$
10:    else if $opr[e] =$‘Sec-in’ then    ▷ When $\mathcal{A}$ wants entity $ent[e]$ to receive secure output
                                          ▷ $\mathcal{A}$ specifies a previous event in $inp[e]$ and then the process uses $(ent[inp[e]], sec\text{-}out[inp[e]][ent[e]])$ as input to $\mathcal{P}$
           $\left(s_{ent[e]}, out[e], sec\text{-}out[e][\cdot]\right) \leftarrow \mathcal{P}\left(s_{ent[e]}, \text{‘Sec-in’}, (ent[inp[e]], sec\text{-}out[inp[e]][ent[e]]), clk[e]\right)$
                                          ▷ $\mathcal{P}$ returns the state of entity $ent[e]$, the output, and the secure output
11:    else    ▷ Otherwise, $\mathcal{A}$ wants to execute operation $opr[e]$ of $\mathcal{P}$ with input $inp[e]$ over entity $ent[e]$ with local clock of $clk[e]$
           $\left(s_{ent[e]}, out[e], sec\text{-}out[e][\cdot]\right) \leftarrow \mathcal{P}\left(s_{ent[e]}, opr[e], inp[e], clk[e]\right)$
12:    end if
13:    $(s_{\mathcal{A}}, out_{\mathcal{A}}, \mathsf{F}) \leftarrow \mathcal{A}(s_{\mathcal{A}}, out[e])$    ▷ Inform $\mathcal{A}$ of the value of $out[e]$ and allow $\mathcal{A}$ to decide whether to continue ($out_{\mathcal{A}} = \perp$), or to terminate the loop ($out_{\mathcal{A}} \neq \perp$)
14: until $out_{\mathcal{A}} \neq \perp$
15: $T \leftarrow (out_{\mathcal{A}}, e, \mathsf{N}, \mathsf{F}, ent[\cdot], opr[\cdot], inp[\cdot], clk[\cdot], \tau[\cdot], out[\cdot], sec\text{-}out[\cdot][\cdot])$    ▷ Output
16: Return $T$
```