When PKI (finally) met Provable Security

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—**Public Key Infrastructure (PKI) schemes were first proposed in 1978 and standardized in 1988, yet, unlike most cryptographic schemes, PKI schemes were never rigorously defined. Achieving provable security for PKI is necessary and long overdue, as PKI provides the foundation for important applications of public key cryptography, such as TLS/SSL. In response, we present the first precise specifications of a secure PKI scheme, suitable for a variety of PKI designs.**

PKI schemes have significantly evolved since X.509, with more complex goals, e.g., *transparency*, to ensure security against corrupt issuers. In addition to the basic PKI properties, our definitions encompass these more recent and advanced aspects.

Our results have important implications. First, our specifications allow a better scrutiny and comparison of the multitude of new PKI designs recently proposed, such as Google's Certificate Transparency (CT) and related PKIs, as well as future designs. Second, the specifications facilitate *proper* analysis of protocols and systems that use PKI, such as TLS/SSL, code signing, IPsec, DNSSEC, RPKI, BGPsec, permissioned blockchains, voting, recommendations, which is of critical importance as most real-world security schemes inherently rely on PKI. Finally, we use our specifications to formalize and prove X.509 version 2 PKI, showing that provable security is achievable for 'real' PKI designs.

***Index Terms*—**

I. Introduction

Public Key Infrastructure (PKI) provides an essential foundation for applications which rely on public key cryptography, and, specifically, it is crucial to ensure security in open networks and systems. Since the early PKI ideas were proposed in 1978 [1], the deployment of PKI has been dominated by the X.509 standard [2], whose first version was published in 1988. The practical importance of PKI grew after its adoption by Internet standards, most notably, the TLS/SSL protocol [3], which is 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 certificate authority (CA) is trusted to issue certificates for any domain [4], 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 [5], [6] and issued fake certificates for major websites. Furthermore, some CAs abused their powers by improperly delegating their certificate-issuing authority or even intentionally issuing rogue certificates [7]. Such PKI failures allow attackers to issue fake certificates, launch phishing and website spoofing attacks, and perform man-in-the-middle attacks, possibly leading to identity theft, surveillance, compromises of personal and confidential information, and other serious security breaches.

A basic property ensured by X.509 certificates is *accountability*, i.e., the CA issuing a given X.509 certificates is identifiable. For many years, accountability was considered a sufficient deterrent against rogue or negligent CAs; the issuing CA cannot deny having issued a rogue certificate - or, more precisely, that its private key was used to sign the rogue certificate. However, the many PKI failures brought the realization that *accountability is not a sufficient deterrent*, since the CA suffer repercussions only if and when the certificate in question is found - which may not occur, especially if abused 'stealthily'.

This realization 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) [8], [9], Enhanced-CT [10], Sovereign Key [11], CONIKS [12], AKI [13], PoliCert [14], ARPKI [15], DTKI [16], CoSi [17], [18], IKP [19], CertCoin [20], PB-PKI [21], Catena [22], CertLedger [23]*, among others.

As PKI schemes and their security properties become more complex, it becomes more and more important to clearly define and analyze their security. The current PKI situation stands in a sharp contrast to the accepted norms in cryptography, where it is expected to always develop rigorous specifications for cryptographic schemes. These norms began in the 1980s with the seminal papers defining secure encryption [24] and secure signature schemes [25], not long after these concepts were first proposed in [26]. The concepts of certificates and PKI were also developed in the 1980s, culminating with the release of the X.509 recommendation in 1988 [2]; however, until the current paper, there were no rigorous security specifications

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

A *specification* of a cryptographic scheme defines its functionality and security requirements, *independently* of a specific construction/design. For example, many designs of encryption and signature schemes were proven secure using the definitions of [24] and [25], respectively, or conjectured to satisfy them, as is often done for applied schemes such as RSA or DSA. Specifications of cryptographic schemes are needed to evaluate the security of different variants and designs, to facilitate the use of cryptographic schemes as building blocks for more complex schemes and protocols, and to evaluate the security of those composite schemes and protocols.

The concepts of certificates and (rudimentary) PKI were already introduced in 1978 [1], almost as soon as the concepts of public key encryption and signatures [26]. More than 40 years later, and after multiple PKI proposals and failures, it is time for our community to define security specifications for PKI schemes, and to properly analyze the security of the proposed PKI designs. The current situation is especially baffling as practical deployments of public key cryptography are, almost always, based on the underlying PKI. It is impossible to fully analyze the security of a cryptographic system when this critical component is not well defined and analyzed.

The goals of the new, 'post-X.509' PKI proposals, go far beyond the goals of X.509, and the 'post-X.509' designs are significantly more complex than the X.509 designs. However, so far, these goals have not been rigorously defined, and the security of PKI designs was not fully analyzed. Only few works present any analysis: [27], [28] analyze (only) the logging mechanism of CT, and both ARPKI [15] and DTKI [16] use automated symbolic analysis for system-specific properties.

In fact, even for the simple, 'classical' X.509 PKI, there are no definitions of security requirements and consequently, no proofs. Arguably, this may not be as critical, since for X.509, both definitions and proof are, arguably, relatively straightforward (see within). However, this lack of any definitions and proofs implies that works analyzing security of PKI-based protocols, e.g., IPsec/IKE [29], [30] and TLS [31], [32], mostly completely ignore the underlying PKI and simply assume the use of correct public keys. As one of the consequences, real-world deployments of such schemes often fail to properly implement the (relatively simple) requirements for a secure X.509 scheme. In particular, many browsers do not properly deploy the X.509 revocation mechanisms [33], such as CRLs and OCSP [34], [35], potentially allowing attackers to trick browsers into using revoked certificates.

A number of works [36]–[38] study security of cryptographic protocols based on a grossly-simplified notion of a PKI; however, they rely on extreme simplifications which even *ignore revocation*, a critical aspect of X.509 and other PKI schemes. Furthermore, some of these works define PKI using the ideal functionality approach of Universally Composable (UC) framework [36], and as such, there is no methodology to extend this notion and support other PKI requirements, such as revocation-related properties, transparency or other properties of post-X.509 PKI schemes.

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 or in the specific communication and attack models. As a result, if requirements are presented at all, they are informally defined and tailored to a specific construction.

The lack of *proper* definitions and proofs makes it challenging to build (provably) secure systems which use PKI, 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, due to the lack of formal specifications. Lastly, it is impossible to design and analyze schemes in a modular manner by provable reductions to simpler, already analyzed schemes.

In summary, the current situation, where there are no precise security specifications for PKI schemes, 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 an insecure PKI scheme. The concerns are even greater, considering that attacks against PKI are not only a theoretical threat, but are a major concern in practice.

In this work, we present rigorous security specifications for PKI schemes, allowing for reduction-based proofs of security. Our definitions support a wide range of PKI schemes, from X.509 to advanced, improved-security PKI schemes such as Google's Certificate Transparency, independently of their specific designs. We focus on the challenge of dealing with misbehaving parties, e.g., corrupt CAs, by *detecting* misbehavior and/or *preventing* damage due to misbehavior.

We do not address *trust-management* issues, such as the *decision to trust* a particular CA, currently dealt with in web-PKI by the defining root CAs in root stores of individual applications or systems, together with intermediary CAs certified by root CAs, and in X.509v3, by the basic, name and policy constraints. A model of the *trust decision* for PKI systems was proposed by Mauer [39], subsequently extended by [40], [41], and others [42]–[47]. These solutions are complementary but orthogonal to our results.

To define the security specifications, we reviewed and analyzed the existing PKI schemes and the properties they claim to provide. This analysis allowed us to identify the relevant requirements and define corresponding specifications for PKI systems in a way that *embraces*, *complements* and *reflects* the current PKI designs and *enables* future ones. These security specifications include: *accountability (*ACC*)*, Δ-*transparency (*$\Delta\mathsf{TRA}$*)*, *non-equivocation* (detection, $\Delta\mathsf{EQ\text{-}D}$, and prevention, $\mathsf{EQ\text{-}P}$), *revocation accountability (*ReACC*)*, *non-revocation accountability (*NReACC*)* and Δ-*revocation status transparency (*$\Delta\mathsf{ReST}$*)*. We map these security specifications to existing PKIs in Table I.

Contributions. This work:

1) Presents the first rigorous definition for a PKI scheme and its security specifications.
2) Reviews the PKI landscape, comparing the major proposed and deployed PKI schemes using the security specifications we identify.
3) Formally defines and proves the security of the X.509 version 2 PKI scheme.

Organization. §II informally introduces our security specifications and reviews the PKI landscape with respect to them; it also discusses related works. §III provides an overview of the *Modular Specifications Security Framework (MoSS) [48]*, which we use for specifications and analysis. §IV defines a PKI scheme and §V its security specifications. §VI discusses security of the X.509 version 2 protocol; the complete specifications and proofs appear in Appendix A and Appendix C . Finally, §VII concludes and discusses future work.

II. Security Goals of PKI Schemes

As a first step towards defining the PKI specifications, we analyzed the goals and properties of existing PKI schemes, focusing on security against corrupt CAs. In this section, we informally discuss these security goals and later formally define them as game-based specifications (§V-A).

We first discuss the goals of X.509 (§II-A) and the goals of post-X.509 PKI schemes (§II-B). We then compare the existing PKI schemes with respect to these goals (§II-C), summarizing the results in Table I.

A. X.509 Security Goals

PKI schemes define how to issue and use *public key certificates*. Certificates are issued by *Certificate Authorities (CAs)*, and typically contain an *identifier* and *public information*, including 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 receives a public key and verifies that the identity and other information in the certificate correspond to the entity requesting the certificate and providing the public key. The CA is accountable for any failure to properly perform this validation; *accountability* is the first property we discuss.

*1) Accountability (*ACC*):* Accountability is the ability to *identify the CA that issued a given certificate*. Accountability provides a reactive defense against a corrupt CA; a CA which issues a rogue certificate, once or repeatedly, can be ignored or otherwise punished. Typically, a CA is accountable for any certificate which is validated using the CA's public 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 issuing rogue certificate intentionally or due to negligence. Accountability motivates CAs to take precautions to protect their private keys and to ensure that the certificates they issue are authentic. Note that we use the term accountability as a technical goal, and do not refer to any specific legal or other repercussions of attribution of a rogue certificate to a CA.

*2) Revocation Accountability (*ReACC*):* Certificates' validity period starts from their *issue date* and ends on their *expiration date*, both of which are specified in the certificate; however, certificates can be *revoked*, i.e., invalidated before their expiration date. A certificate may be revoked for a variety of reasons, including a loss or compromise of the private key corresponding to the public key endorsed in the certificate, or when the CA determines that the certificate contains misleading information. *Revocation accountability* requires that each revoked certificate can be traced back to the revoking CA. This ensures accountability of the CA if it revokes a certificate without a legitimate reason, e.g., a request from the certificate owner.

*3) Non-Revocation Accountability (*NReACC*):* Since any certificate can be revoked at any time, relying parties need to verify that a certificate is still *non-revoked* when attempting to use it. PKI should ensure *non-revocation accountability* (NReACC) so that when a relying party checks a certificate and verifies it as non-revoked, then the information used to make this determination is attributable to a specific entity - typically, the issuing CA. This goal is important to detect rogue CAs who might provide relying parties with bad revocation information, e.g., stating that a certificate is non-revoked while it was actually revoked. If revocation information is attributable to a specific CA, then a combination of the non-revoked and revoked responses would provide a *Proof of Misbehavior* (*PoM*). In X.509, the non-revocation status is indicated by the *Certification Revocation List (CRL)* [49] or the *Online Certificate Status Protocol (OCSP)* [50] mechanisms.

Often, as done in X.509, only the certificate's issuer can revoke it; if the issuer is also the only entity providing the non-revocation responses, then this goal is very easy to achieve. However, there may be reasons to allow other entities to revoke certificates, e.g., to force revocation of certificates issued by a rogue CA, or to allow other entities to issue non-revoked responses, e.g., as supported by the X.509 OCSP protocol.

B. Post-X.509 Security Goals

We now discuss additional security goals, pursued by recent PKI schemes and designed to improve security against corrupt CAs. These goals include Δ-*transparency* (ΔTRA), Δ-*revocation status transparency* (ΔReST), Δ-*equivocation detection* (ΔEQ-D) and *equivocation prevention* (EQ-P).

Most of these security goals depend on *time*, i.e., their corresponding security guarantees are defined with respect to some pre-defined time bound Δ. To emphasize this fact, such goals include Δ in their names.

1) Δ-Transparency (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 rogue 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 rogue certificates. However, repeated cases of such certificates issued 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, punishment was often ineffective [51]–[53].

Furthermore, 'punishment' could only be applied *after* the damage was committed and *discovered* - if it was discovered at all. An attacker or corrupt CA could reduce the risk of discovery by minimizing the exposure of the rogue certificate. Except for efforts such as the Perspectives Project [54] or the EFF SSL Observatory [55] that aim to gather and inspect *all* SSL certificates used in practice, the burden of detecting and responding to rogue certificates is mostly on the clients that receive them; browsers typically cannot detect these certificates, much less to report them to a (non-existing) 'enforcement agency'.

This significant issue has motivated more recent PKI designs, e.g., Certificate Transparency (CT), where valid certificates must be *published*, allowing third parties (e.g., trusted 'monitors') to inspect and detect any discrepancies and suspect certificates. Transparency requires a certificate to be recognized (signed) by one or more parties committed to publish the certificate, making it *available* to interested parties, within a specified time frame Δ. In other words, transparency prevents a CA from 'silently' generating rogue yet validly-formed certificates and exposing them only to selected victims during an attack, and it facilitates detection of rogue certificates issued by a corrupt, compromised or negligent CA. By demanding that a certificate is transparent in order for it to be considered valid, the question is no longer whether rogue certificates will be detected, but rather *how soon*.

2) Δ-Revocation Status Transparency ($\Delta\mathsf{ReST}$): While $\Delta-$transparency forces certificates to be publicly available within some Δ, the transparency guarantee is not extended to the revocation status of the certificate. To illustrate why such a guarantee is important, consider a scenario where a private key of a $\Delta-$transparent certificate is compromised. Even after the subject of the compromised certificate has 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. Attackers can take advantage of this issue by withholding revocation information from the victims, thus convincing them that the revoked compromised certificate is transparent, as needed, and still valid since no information to the contrary is available.

This fundamental issue motivates the goal of $\Delta-$*revocation status transparency*. When an entity *attests* that a given certificate's revocation status, either as revoked or non-revoked, is transparent ($\Delta\mathsf{ReST}$), then that entity commits to making this *status* available to all interested third parties, again, within a predetermined Δ time period.

3) Equivocation Detection and Prevention: The aforementioned transparency goals ensure that interested parties *can learn* when certificates are issued and revoked. This allows detection of suspect certificates, e.g., certificates using misleading domain names. However, this does not *prevent* the issuing of such rogue certificates; furthermore, detection is not ensured as a part of the PKI scheme itself - it has to be done by some external process that inspects the published certificates. More specifically, it seems impossible to have an automated process that would reliably detect all forms of rogue or misleading domains names.

However, it may be possible for the PKI to detect, or even prevent, issuing of certificates which can be identified by a well-defined, computable process. We focus on one important category, *equivocating certificates*. An *equivocating certificate* is one which has the same identifier as a legitimate, previously-issued certificate, which was defined in advance as a certificate which should not be equivocated. We consider two non-equivocation goals:

- *Δ-equivocation detection ($\Delta\mathsf{EQ\text{-}D}$)*: ensures that any equivocating certificate is detected within Δ time of its issuance by at least one honest entity.
- *Equivocation Prevention ($\mathsf{EQ\text{-}P}$)*: prevents issuing of equivocating certificates, i.e., certificates containing an already-certified identifier.

$\Delta-$transparency implies $\Delta-$equivocation detection, but not equivocation prevention. Furthermore, we define $\Delta-$equivocation detection separately from $\Delta-$transparency, since it does *not imply* $\Delta-$transparency, i.e., $\Delta-$equivocation detection is not equivalent to transparency. In fact, some PKI schemes, notably CONIKS [12], offer $\Delta-$equivocation detection but *not $\Delta-$transparency* - indeed, transparency would conflict with some of CONIKS privacy goals. Other detection and prevention goals might be desirable for specific PKI systems. For example, equivocation prevention still allows issuance of misleading (but not identical) identifiers, e.g., misleading domain names which may be abused for phishing attacks such as g00gle.com or googleaccounts.com.

The aforementioned goals can have a global impact, i.e., apply to all the certificates in the system, or they can alternatively be relevant to only a subset of certificates.

C. Comparison of Existing PKI Schemes

One of the outcomes of rigorously defining PKI specifications is the ability to achieve provable security, for existing and future PKIs. It is therefore crucial to define a PKI scheme in a way that *embraces*, *complements* and *reflects* the 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 the results below. We compared all schemes with respect to the specifications formally presented in §IV; we also mention two additional properties, privacy and global name-spaces, which we do not address.

Notations in Table I. We use the $\mathsf{n/s}$ (not supported) symbol to indicate when a scheme does not seem to support a property. Otherwise, we use one of the three following symbols, ●, ⊙, or ◐, to indicate that a scheme supports the property. The ● symbol indicates that a scheme comes with a *rigorous*, reduction-based proof of the property. We indicate with an appropriate comment when a scheme is supported by an automated symbolic proof; note that such proofs are often of a property specific to that scheme, not properties defined

System [reference]	Safety requirements						Additional req.	
	ACC	ΔTRA	ΔEQ-D	EQ-P	ReACC/ NReACC	ΔReST	Privacy[1]	Global namespace
X.509v2 [2] (provably-secure, 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 [22]	⊙	⊙	⊙	⊙	⊙	⊙	n/s	✓
CertCoin [20]	n/s	⊙	⊙	⊙	n/s	⊙	n/s	✓
PB-PKI [21]	n/s	⊙	⊙	⊙	n/s	⊙	⊙	✓
CoSi [18]	◐	⊙	⊙	⊙	n/s	n/s	n/s	✓
Enhanced-CT [10] DTKI [16] [3]	◐	⊙	⊙	n/s	◐	⊙	n/s	✓
AKI [13]	◐	⊙	⊙	n/s	◐	⊙	⊙	✓
CONIKS [12]	◐	n/s	⊙	n/s	◐	⊙	⊙	✗
ARPKI [15] [4]	◐	⊙	⊙	⊙	◐	⊙	n/s	✓
CertLedger [23]	◐	⊙	⊙	⊙	⊙	⊙	n/s	✓
Certificate Transparency (CT) [8]	◐	⊙[5]	⊙[5]	n/s	◐	n/s[6]	n/s[7]	✓

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 [27], [28]. [6]CT IS EXTENDED TO INCLUDE REVOCATION TRANSPARENCY IN [9]. [7]CT: EXTENDED TO INCLUDE PRIVACY [56].

for an arbitrary PKI scheme. The ◐ symbol indicates that although no formal proofs were provided, it seems intuitively true that the system achieves the property; e.g., accountability in X.509 follows from the use of a signature scheme to sign the certificate. The ⊙ symbol indicates that the property seems justified, using an (informal) security argument; note that this may imply that additional assumptions or details may be needed to ensure security or to formally prove it.

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 [57], 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 CoSi is a general-purpose witnessing (logging) scheme and Catena is a witnessing 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, requiring additional assumptions.

Interestingly, many systems directly focus on more advanced properties, such as transparency and non-equivocation, and treat more 'basic' properties, such as accountability and revocation, as intrinsic to PKI, often without even stating them. This phenomenon is especially apparent in the 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 exceptions are CT, DTKI, and ARPKI, which have different types proofs or 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. [27] 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 [28], 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 [58]. 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.

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 proofs, 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.

III. The MoSS Framework

We now discuss the Modular Specifications Security (MoSS) [48] framework, which we use to define the PKI specifications and analyze PKI constructions.

We chose to use MoSS for several reasons. First, MoSS follows the *game-based approach* [59], [60], offering simple and intuitive definitions and proofs of security, both asymptotic and concrete. Further, it provides a well-defined execution process and well-defined and reusable, 'generic' models of delays, synchronization and faults, which we indeed reuse in our analysis (§VI).

Lastly, the other frameworks we considered [61]–[63] use *monolithic* specifications, e.g., UC's ideal functionalities. However, different PKI designs are designed for different security goals as illustrated in Table I. For example, most post-X.509 PKIs support transparency except for CONIKS, which supports detection of non-equivocation but not transparency, in order to provide certain privacy goals. As another example, CertCoin and PB-PKI do not require accountability and revocation accountability, two very basic PKI properties. PKI designs may also differ in their assumptions, e.g., different trust models.

Given these significant differences, it is infeasible to define a single PKI functionality that reflects *all* current PKIs. On the other hand, defining multiple PKI functionalities would be complex, repetitive, difficult to understand, and, most critically, difficult to compare, reuse and extend. In short, monolithic specifications are inappropriate for specification and analysis of PKI constructions.

In contrast, MoSS supports *modular* specifications, and cleanly separates *requirement specifications* (goals) which a protocol should achieve, from *model specifications* (assumptions) under which the protocol is analyzed; each specification is defined by an efficiently-computable *predicate* that reflects a single, focused assumption or requirement. These assumptions and requirements can then be combined in a modular fashion, facilitating their reuse and incremental design of protocols, an important feature for PKI schemes.

In the rest of this section, we briefly explain the three components of the MoSS framework: an *execution process*, *model* specifications and *requirement* specifications, focusing on the 'classical' asymptotic-security definitions. For the full description of the framework, see [48].

A. The MoSS Execution Process

The MoSS execution process $\mathbf{Exec}$ defines the process of executing a protocol $\mathcal{P}$ under an adversary $\mathcal{A}$, giving the adversary an extensive control over the environment, including communication, local clock values, inputs and faults. The execution process is a well-defined algorithm (Algorithm 15 in Appendix B) that outputs a *transcript* T containing all events in a randomly sampled run of protocol $\mathcal{P}$ with adversary $\mathcal{A}$. All events are serializable on a 'real-time' axis, and defined iteratively, as a sequence of invocations of specific operations on specific entities.

The execution process $\mathbf{Exec}_{\mathcal{A},\mathcal{P}}(params)$ is a randomized algorithm, which receives several inputs: parameters $params$ and two efficient (PPT) algorithms, $\mathcal{A}$ for the adversary and $\mathcal{P}$ for the protocol.

The execution process outputs a *transcript* $T \leftarrow \mathbf{Exec}_{\mathcal{A},\mathcal{P}}(params)$, 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 the adversary).
$T.\mathsf{F}$	Faulty (adversary-controlled) entities.
$T.ent[\hat{e}]$	Entity invoked in event $\hat{e} \leq T.e$.
$T.opr[\hat{e}]$	The operation that was 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 $\hat{e} \leq T.e$.

B. Specifications

MoSS uses a general notion of *specifications* in order to formally define *models* and *requirements*. Model specifications define the adversarial capabilities, communication and clock synchronization assumptions. Requirement specifications define the properties ensured by different PKI constructions.

A specification ξ is defined as a pair $\xi = (\pi, \beta)$, where $\xi.\pi$ is the *specification predicate* and $\xi.\beta$ is the *base function*[1]. A specification predicate takes as input an execution transcript T and input parameters $params$. Definition 1 specifies the advantage of an adversary with respect to a given specification.

Definition 1 (Advantage of adversary $\mathcal{A}$ against protocol $\mathcal{P}$ for specification ξ)**.** *Let $\mathcal{A}, \mathcal{P} \in PPT$ and let $\xi = (\pi, \beta)$ be a specification. The advantage of an adversary $\mathcal{A}$ against a protocol $\mathcal{P}$ for a specification ξ is defined as:*

$$\epsilon^{\xi}_{\mathcal{A},\mathcal{P}}(params) \overset{def}{=} \max\left\{0, \Pr\left[\begin{array}{c}\xi.\pi(T) = \bot, \text{ where} \\ T \leftarrow \mathbf{Exec}_{\mathcal{A},\mathcal{P}}(params)\end{array}\right] - \xi.\beta(params)\right\} \quad (1)$$

Definition 2 defines *model* specifications, which allow to restrict the capabilities of an adversary or the events that are allowed to happen in the execution process. §VI describes a model specification for our security analysis of X.509 version 2.

Definition 2 (Adversary $\mathcal{A}$ satisfies model $\mathcal{M}$ with negligible advantage)**.** *Let $\mathcal{A} \in PPT$, and let $\mathcal{M} = (\pi, \beta)$ be a model specification. Also, let $params \in \{0,1\}^*$, where $params$ includes a unary parameter $params.\mathcal{P}.1^\kappa$ and $|params| \leq 2 \cdot 1^\kappa$. We say that adversary $\mathcal{A}$ satisfies model $\mathcal{M}$ with negligible advantage, denoted as $\mathcal{A} \models_{poly} \mathcal{M}$, if for every protocol $\mathcal{P} \in PPT$ and $params$, the advantage of $\mathcal{A}$ against $\mathcal{P}$ for $\mathcal{M}$ and is negligible in $params.\mathcal{P}.1^\kappa$, i.e.:*

$$\epsilon^{\mathcal{M}}_{\mathcal{A},\mathcal{P}}(params) \in Negl(params.\mathcal{P}.1^\kappa)$$

Definition 3 defines requirement specifications, which describe what a protocol or scheme *ensures*, assuming a given model. §V presents PKI requirements.

Definition 3 (Protocol $\mathcal{P}$ satisfies requirement $\mathcal{R}$ with negligible advantage under model $\mathcal{M}$)**.** *Let $\mathcal{R} = (\pi, \beta)$ be a requirement specification. We say that protocol $\mathcal{P}$ satisfies requirement $\mathcal{R}$ under model $\mathcal{M}$, denoted $\mathcal{P} \models^{\mathcal{M}}_{poly} \mathcal{R}$, if for every PPT adversary $\mathcal{A}$ that satisfies $\mathcal{M}$ with negligible advantage and parameters $params \in \{0,1\}^*$, where $params$ includes a unary parameter $params.\mathcal{P}.1^\kappa$, the advantage of $\mathcal{A}$ against $\mathcal{P}$ for $\mathcal{R}$ is negligible in $params.\mathcal{P}.1^\kappa$, i.e:*

$$\begin{aligned}\mathcal{P} \models^{\mathcal{M}}_{poly} \mathcal{R} \quad &\overset{def}{=} \\ &\left(\forall\ \mathcal{A} \in PPT, params \in \{0,1\}^* \mid \mathcal{A} \models_{poly} \mathcal{M}\right) : \\ &\epsilon^{\mathcal{R}}_{\mathcal{A},\mathcal{P},}(params) \in Negl(params.\mathcal{P}.1^\kappa)\end{aligned} \quad (2)$$

IV. PKI Scheme

In this section, we formally define a PKI scheme. We start by describing entities involved in PKI and formally define what a certificate is (§IV-A). Then, we define the notion of certificate attribute attestation (§IV-B), i.e., how certificates can be attested to have different properties. Lastly, we provide an informal description of the operations that a PKI consists of (§IV-C) and then formalize them (§IV-D).

A. Entities and Certificates

A PKI scheme consists of protocols for a set of *authorities* such as *certificate authorities (CAs)*, which we denote as N. A PKI scheme also includes stateless functions, which may be invoked by any entity, (i.e., any entity not in N). Most significantly, these stateless functions are used by two types of users (clients) of the CAs: *subjects*, which typically use a CA to obtain and manage their public key certificates, and *relying parties*, which use the certificates of the subjects, to determine if they want to communicate with the subject (using the certified public key). We do not need to refer explicitly to the set of clients, since we never refer to the state of a particular client or to whether a particular client is honest or not.

A *certificate authority (CA)*, also referred to as an *issuer*, is an authority that issues *certificates* to subjects. 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. In some PKI schemes, e.g., Certificate Transparency (CT), the set of authorities N contains additional, non-CA authorities, such as *loggers* and *monitors*, which are used to ensure certain properties, e.g., transparency.

Definition 4 defines the elements of a typical certificate, which are used in the security specifications.

Definition 4 (Certificate)**.** *A* certificate *is a tuple:*

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

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.*
- *$\psi.\sigma$: the CA's signature over the certificate's fields.*

Note that the above elements are (intentionally) abstract and may be encoded in different ways. Certificates may contain additional data, such as *extensions*, to include further data that the issuer wishes to certify.

B. Attribute Attestations

A PKI scheme can use different methods to allow the certificate issuer, or other entities, to confirm that certificate possesses a certain attribute. For example, a certificate ψ can be attested to as non-revoked (or revoked) by the issuer who can produce an *attribute attestation*, i.e., a signed statement to that effect. An attribute attestation contains the necessary

[1]The base function is used for cases where the adversary has some 'trivial', allowed success probability, e.g., $\frac{1}{2^l}$ when using l bit MAC. Often, and throughout this paper, $\beta = 0$, so β can be essentially ignored.

information to identify the certificate and the corresponding property along with additional information needed to verify the statement. Similarly to our definition of a certificate, we define the elements of an attribute attestation in an abstract way, independent of the actual attestation process.

Definition 5 (Attribute attestation)**.** *An* attribute attestation *is a tuple:*

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

where:

- $\rho.attr$ *is the attribute attested, e.g.,* ACC *or* NREV.
- $\rho.serial$ *is the serial number of the corresponding certificate.*
- $\rho.\iota$ *is the identifier of the attesting entity.*
- $\rho.\tau$ *is the time of attestation.*
- $\rho.tbs$ *is data to be signed (tbs) by the attestation.*
- $\rho.\sigma$ *is a signature using the attesting entity's private key, over the* $\rho.tbs$ *data, the attribute* $\rho.attr$ *and the time of attestation* $\rho.\tau$.

C. Informal Overview of PKI Functionalities

We begin with the main PKI operation, issuing certificates.

1) Certificate issuance: A certificate is *issued* by some entity $\iota \in \mathsf{N}$ following a request generated from a client, the subject of a certificate. The subject provides the certificate's details, i.e., the subject identifier and public information (e.g., a public key). Then, the issuing entity uses a dedicated operation, which we denote as Issue, to produce the certificate. Issue creates a certificate following some prescribed format (e.g., X.509), by specifying the certificate information along with a proof (typically, a signature, and sometimes additional information) that the issuer has in fact issued the certificate.

The issuer's proof (signature) attests that the issuer has verified the validity of the certified information, typically, that the identifier corresponds to the endorsed public key and that the client knows the corresponding private key. We do not model nor analyze this verification process as it might be unique to each issuer and type of certificate.

2) Certificate revocation and attestations: Revocation is an important component of a PKI scheme, already addressed in X.509. To revoke a certificate, the issuer typically labels the certificate as revoked, using a Revoke operation, following a request from a client. A PKI scheme must include some mechanism to allow relying parties to check if a certificate was revoked. For example, X.509 defines two such mechanisms: *Certificate Revocation Lists (CRLs)* and the *Online Certificate Status Protocol (OCSP)*, both of which produce a signed statement by the issuing CA that attest to the revocation status of a certificate (OCSP) or a set of certificates (CRLs). More specifically, both mechanisms may be used to attest whether or not a certificate is revoked at a given time.

Post-X.509 PKIs use additional types of attestations, e.g., related to transparency and equivocation. We found it convenient to define a generic concept of an *attestation*, as well as a PKI operation, Attest, that can be used to produce an arbitrary attestation by "upgrading" or extending a set of attestations associated with a certificate.

Specifically, we define the following attribute attestations: three attestations related to revocation (revocation, non-revocation, and Δ-revocation status transparency), one transparency attestation (Δ-transparency), and two attestations related to equivocation (Δ-equivocation detection, and equivocation prevention). We discuss these attributes in detail in §V-A-§V-B.

The relying parties usually expect attestations to be made by specific entities; for example, the relying parties usually accept attestations of revocation and non-revocation only if signed (attested) by the issuer of the certificate. Similarly, relying parties would usually accept attestations of Δ-transparency and Δ-revocation status transparency only if signed by a trusted entity, e.g., a trusted logger.

The Attest operation might not always be able to *immediately* output the requested attestation because, for example, in some PKI systems advanced attributes might involve interaction with other authorities or take time to be processed (e.g., in CT, a signed certificate timestamp (SCT) is returned when a certificate is submitted for logging as a promise to actually log the certificate within a maximum merge delay (MMD). To support such non-immediate response mechanism, the Attest operation can output a *pending* attestation, which is a *commitment* to produce the requested attestation (or a failure indication), within some defined amount of time after which the Attest algorithm with the pending attestation is guaranteed to output the final response, which is either the expected *non-pending (final)* attribute attestation, or a failure indication ($\perp$).

3) Validating attestations with WasValid*:* 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 to allow *anyone*, specifically the relying parties, to perform this operation. Relying parties typically do not maintain any certificate-related state, thus, they must be able to validate certificates without any extra information beyond the information included in the certificate.

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

4) Audit and Proof-of-Misbehavior (PoM): Since the main PKI operations are performed by authorities, typically CAs, their actions need to be auditable to guard against any misbehavior. Consider the Δ-transparency specification, which requires that if a certificate is Δ-transparent, then it must be available to all interested entities within Δ time. To uphold this specification, we need a way to query the local state of those interested entities to make sure that Δ-transparent certificate were indeed made available to them. Accordingly, we define an Audit function, which can be invoked in a couple of ways:

- Audit may be invoked with a subject $subject$ and an attribute $attr \in \mathsf{AttrSet}$. In this case, Audit either returns a valid set of certificates Ψ issued for $subject$ along with their attribute attestations set P, according to the local state, or $\perp$ to signal that there are no such certificates.
- Audit may be invoked with a certificate ψ and an attribute attestation ρ. In this case, Audit outputs $\top$ if the current state indicates that ψ is valid with respect to ρ. 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. If the current state does not provide any relevant indication, then Audit returns $\perp$.

A *Proof-of-Misbehavior*, as referred to above, is any conflicting pair of attestations issued by an entity, such as a revocation attestation and a non-revocation attestation (for the same certificate and the same time). The PoM operation is used to verify purported proofs of misbehavior.

5) Monitoring: In most of the advanced PKI schemes, misbehavior is deterred and detected by having authorities' behavior *monitored*. The Monitor operation instructs entities to monitor other entities. Namely, an entity invoked with the Monitor operation with an input $\iota \in \mathsf{N}$, would monitor an entity ι. As a result, the monitoring entity would perform periodic Audit operations on the monitored entity to receive an update from the monitored entity, e.g., to receive from a logger new certificates issued (since previous Audit). Monitoring is required for some post-X.509 to achieve certain specifications, e.g., transparency and equivocation.

6) Additional operations: Finally, a PKI scheme also requires a few basic functions necessary to bind the entire scheme together. First, a PKI scheme (typically) requires an initialization operation, denoted as Init, to allow entities to perform any required initialization operations, e.g., generating cryptographic keys locally and exchanging them, exchanging other set up information with others, such as the set of all authorities in a specific system, etc. Second, all authorities need to be able to appropriately handle time-based events to ensure that repeating operations are performed as needed. The Wakeup operation is expected to be invoked periodically to execute repeated events. To illustrate, consider CT loggers which are required to periodically update and sign the log within each MMD. In this case, each CT log would invoke Wakeup once during an MMD to handle its periodic events. Lastly, to handle requests sent to each entity, either from other entities or from clients, we define a Receive operation to handle them.

D. Definition of a PKI Scheme

We now formally define a PKI scheme. Note that some schemes might require additional inputs or operations. In this case, our specifications should be interpreted as holding for any values of these additional inputs.

Most of the operations are *stateful*, i.e., they have access to the state and the local clock of the entity, and return a new state value; the only exceptions are the *stateless* WasValid and PoM operations. For simplicity, we abuse notation and omit the state and clock from the inputs to the operations (and the new state from the outputs), although, these values are actually always passed as parameters for stateful operations (in the PKI execution process) .

Definition 6 (PKI scheme)**.** *A PKI scheme $\mathcal{P}$ is a PPT algorithm, with the following minimal set of operations:*

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

where:

- $\mathsf{Init}(x) \rightarrow y$*: initialize the state using input x, with optional output y.*
- $\mathsf{Issue}(subject, pub, from, to) \rightarrow (\psi, P)/\perp$*: 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 ψ along with a set of attribute attestations P, or failure indicator $\perp$.*
- $\mathsf{Revoke}(\psi) \rightarrow P/\perp$*: Takes as input a certificate ψ, and outputs either a set of attribute attestations P, or failure indicator $\perp$.*
- $\mathsf{Attest}(\psi, attr) \rightarrow \rho/\perp$*: Takes as input a certificate ψ and an attribute attestation $attr$, and outputs either an attribute attestation ρ, or failure indicator $\perp$.*
- $\mathsf{Audit}(subject, attr) \rightarrow (\Psi, P)/\perp$ *or* $\mathsf{Audit}(\psi, \rho) \rightarrow \top/IA/\zeta/\perp$*: If the input is a subject identifier $subject$ and attribute $attr \in \mathsf{AttrSet}$, then the output is either a valid set of certificates Ψ along with a set of attribute attestations P, or failure indicator $\perp$. Otherwise, the input is a certificate ψ and attribute attestation ρ, and the output can be $\top$, i.e., the audit is successful, or one of three failure indicators: a* Proof-of-Misbehavior *(*PoM*) that shows misbehavior by the signer of the attestation, an* accusation *IA stating that the signer is faulty but without 'proof', or $\perp$ for invalid inputs.*
- $\mathsf{WasValid}(\psi, pk, \rho) \rightarrow \top/\perp$*: 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 $\perp$.*
- $\mathsf{Wakeup}(data)$*: Takes as input a wake-up event information $data$ and perform a time-based operation.*
- $\mathsf{Receive}(x)$*: Process input information x received via the network.*
- $\mathsf{PoM}(pk, \sigma) \rightarrow \top/\perp$*: a stateless function which takes as input a public key pk and proof σ, and outputs either $\top$ or $\perp$.*
- $\mathsf{Monitor}(\iota)$*: Takes as input an entity identifier ι and starts to periodically monitor ι.*

V. PKI Security Requirements

We now define PKI security requirements based on the security goals which we informally discussed in §II. We use the MoSS framework, introduced in §III, to do so.

The framework allows the requirements to be modular in two ways. First, we are able to separately define each requirement, e.g., accountability or transparency, which is necessary as different PKI schemes aim for different requirements and there is no single 'PKI functionality'. Second, the definitions of the requirements are independent of the definitions of the model, e.g., adversary or communication, assumed by a particular PKI construction. This approach is not only beneficial but also intuitive; indeed, defining what transparency *means* in the context of PKI is independent of whether or not transparency *holds* in a given construction under synchronous or asynchronous communication model or honest-but-curious or Byzantine adversary model, for example.

Once we have a specific PKI construction, then we can analyze and prove that it satisfies a specific subset of the requirements under a given model, per Definition 3. For example, in this work, we formalize and analyze X.509 version 2 PKI (see §VI-A for the model and §VI-B for the theorem and a proof sketch).

Following the MoSS framework, we define requirements, and later also models, as *specifications* (§III-B). Namely, a requirement is a pair $\xi = (\pi, \beta)$, where $\xi.\pi$ is the *requirement predicate* and $\xi.\beta$ is the requirement *base function*, which gives the 'base' probability of success for an adversary. For the PKI requirements, we always use the trivial base function $\xi.\beta = 0$; hence, each requirement is defined by the corresponding predicate $\xi.\pi$. As per Equation 1, the input to the predicate is the execution transcript T, which is the output of a run of the execution process.

In the rest of this section, we define and discuss these requirement predicates (algorithms 1-7); we use the requirement names, e.g., ACC, to denote the corresponding predicates. We first present, in §V-A, the 'classical' requirements, as expected from 'X.509-like PKI' and then, in §V-B, the 'post-X.509' requirements.

A. X.509-PKI Security Requirements

We now present the 'basic' security requirements of a PKI scheme: accountability, revocation accountability and non-revocation accountability, which the 'classical' PKI schemes, such as X.509 [64], focus on. We define these requirements in the (commented) Algorithms 1-3 as predicates that return $\perp$ if the corresponding requirement does not hold, i.e., the adversary 'wins' in the game defined by the predicate. The input to the predicates is the transcript T from a run of the MoSS execution process (§III-A).

1) Accountability requirement $\mathsf{ACC}(T)$*:* The predicate (Algorithm 1) returns $\perp$ if accountability fails, namely, if there exists a valid, accountable certificate ψ, where its issuer $\psi.issuer$ is honest, yet during the execution, the 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 invoked on $\psi.issuer$ with these inputs. Notice the use of the IsValidAtt procedure, which is (also) a predicate, and returns $\top$ if the issuer is honest and the certificate is valid and accountable; this procedure is used also by other requirements to validate certificate attributes.

2) Revocation accountability requirement $\mathsf{REV}(T)$*:* The predicate (Algorithm 2) returns $\perp$ if revocation accountability fails, namely, if there exists a valid revocation attestation ρ for a certificate ψ with honest issuer $\psi.issuer$, yet, $\psi.issuer$ was *not* instructed to revoke ψ, or instructed to revoke after the specified revocation time.

3) Non-revocation accountability requirement $\mathsf{NReACC}(T)$*:* Similarly to revocation accountability, the *non-revocation accountability* predicate (Algorithm 3) returns $\perp$ if non-revocation accountability fails, namely, if there exists a valid non-revocation attestation ρ for a certificate ψ with honest issuer $\psi.issuer$, although $\psi.issuer$ *was* instructed to revoke ψ (before the revocation time specified in the attestation).

Next, we discuss the details and conventions used in Algorithms 1-3.

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'. Another example are threshold and proactive public-key systems, that 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 [65]. Some PKI schemes use 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 honest party outputs the *reserved tuple* ('PubKey', $role$, pk). See IsValidAtt (line 6).

2) Functions, e.g. IsValidAtt*:* We define a few functions, for operations used by multiple predicates; the definition of each function appears with the *first* predicate using this function. For example, the IsValidAtt function checks that the given attestation of a given certificate is valid for one of the attributes in a given set ATTR of attributes, and signed properly with the specified public key (given as one of the inputs). Furthermore, IsValidAtt validates that this public key, is indeed a public key associated with the *role* specified in the input to IsValidAtt. To validate this association, IsValidAtt uses the public-key association convention just introduced, i.e., it checks that a *honest entity* $\iota \in T.\mathsf{N} - T.\mathsf{F}$, has output the *reserved tuple* ('PubKey', $role$, pk). In most calls, $role$ is simply an identifier of an authority, i.e., $role \in \mathsf{N}$; in fact, usually $role = \iota$.

3) Issuer public key: The predicates rely on the (stateless) $\mathcal{P}.\mathsf{WasValid}$ function to check whether the certificate outputted by the adversary has the claimed attribute. $\mathcal{P}.\mathsf{WasValid}$ needs to be provided with the issuer's public key to verify the 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 aforementioned public-key association convention, and expect that each entity ι outputs its own public key in the form

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

```
1: $(\psi, \rho, pk, \iota) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho$, public key $pk$, and entity $\iota$
2: $AdvWins \leftarrow [$
       IsValidAtt$(T, \{\mathsf{ACC}\}, \psi, \rho, pk, \iota, \psi.issuer)$    ▷ $\psi.issuer$ 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 $AdvWins$ 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 ('PubKey', $role, pk) \in T.out[\hat{e}]$    ▷ Certificate $\psi$ has a valid attestation $\rho$ with an attribute from set ATTR, verified by the public key $pk$ of entity $role$. The mapping of $pk$ to $role$ was outputted by a honest entity $\iota$ (as such, should be correct).
7: end procedure
```

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

```
1: $(\psi, \rho, pk, \iota) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho$, public key $pk$ and entity $\iota$
2: $AdvWins \leftarrow [$
       IsValidAtt$(T, \{\mathsf{REV}\}, \psi, \rho, pk, \iota, \psi.issuer)$    ▷ $\psi.issuer$ 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 $AdvWins$ then return $\bot$ else return $\top$
```

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

```
1: $(\psi, \rho, pk, \iota) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificate $\psi$, attribute attestation $\rho$, public key $pk$, and entity $\iota$
2: $AdvWins \leftarrow [$
       IsValidAtt$(T, \{\mathsf{NREV}\}, \psi, \rho, pk, \iota, \psi.issuer)$    ▷ $\psi.issuer$ 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 $AdvWins$ then return $\bot$ else return $\top$
```

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

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

2: procedure EquivocationOccurred$(T, attr)$
3:     $(\ \psi, \psi', \rho_e, \rho'_e, pk_e, pk'_e, \iota, \iota'\ ) \leftarrow T.out_{\mathcal{A}}$    ▷ 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: $(\ \psi, \psi', \rho_e, \rho'_e, pk_e, pk'_e, \iota, \iota'\ ) \leftarrow T.out_{\mathcal{A}}$    ▷ Certificates $\psi$,$\psi'$, attribute attestations $\rho_e, \rho'_e$, public keys $pk_e, pk'_e$ and two (honest) entities $\iota, \iota'$
2: $AdvWins \leftarrow [$
       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 $AdvWins$ then return $\bot$ else return $\top$
```

('PubKey', ι, pk). Each time the adversary outputs an attribute attestation, it also outputs the public key of the attesting entity, and the ISVALIDATT function checks that this public key was indeed outputted by a honest entity (in the ('PubKey', ι, pk) form).

B. Post-X.509-PKI Security Requirements Specifications

We now define more advanced security requirements, relevant to 'post-X.509' PKI systems, such as Certificate Transparency (CT) [8]. We present in a rough order of complexity, beginning with requirements related to equivocation prevention and detection, and then transparency requirements.

*1) Equivocation prevention requirement (*EQ-P*):* The predicate (Algorithm 4) ensures that if equivocation prevention holds, then at no point in time will there exist two certificates with different public information issued for the same identifier with overlapping validity periods - yet *both* with the EQ-P attribute. Note that the requirement, as defined, allows the (legitimate) scenario where an unequivocal certificate ψ is revoked and an unequivocal replacement certificate ψ' is issued, still within the validity period of the (revoked) ψ. Forbidding such replacements would be simpler - but prevent support for this realistic scenario.

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; see line 7. This 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 requirement (ΔEQ-D*):* The predicate (Algorithm 5) checks that each equivocating certificate is detected within Δ time by (at least) one honest authority. Namely, we first confirm, using the EQUIVOCATIONOCCURRED procedure described in Algorithm 4, that the adversary produced two equivocating certificates then, we confirm that no honest entity detected the equivocation within Δ (line 3).

3) Δ-Transparency requirement (ΔTRA*):* The 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 by an authority, typically referred to as a *logger*.

Of course, we cannot 'force' transparency to hold; the logger may either just not send responses to the monitors, or may attempt to omit a specific certificates from its responses to the monitor. The requirement ensures that such behaviors would be detected in timely manner. The type of detection depends on the type of misbehavior:

- If the logger omits the certificate from its responses, this should be detected upon 'Audit' of the monitor with this certificate, and result in a *Proof-of-Misbehavior (PoM)*. This is confirmed in line 9 of Algorithm 6.
- If the logger stops responding, this should be detected upon the following request from the monitor. In this case, the monitor can only output an *Indicator of Accusation (IA)*. This is confirmed in line 8 of Algorithm 6.

4) Δ-Revocation status transparency requirement (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 predicate (Algorithm 7) ensures that if the specification holds, then whenever an entity $\rho_t.\iota$ produces a 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 (1) did not accuse $\rho_t.\iota$ during Δ as uncooperative and (2) upon audit, did not output a proof of $\rho_t.\iota$'s misbehavior.

VI. PROVABLY-SECURE PKI: X.509 VERSION 2

To demonstrate the feasibility of our approach, we formalized and analyzed X.509 version 2 (X.509v2), a simple yet realistic PKI scheme. In this section, we present the model specification we used to prove that X.509v2 satisfies accountability, revocation accountability and non-revocation accountability as well as the main theorem and its proof sketch. Due to the space constraints, we refer the reader to Appendix A and Appendix C for the complete construction and proofs.

A. Model Specification

We assume a simple model specification $\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}} = (\pi^{\text{X.509v2}}_{\Delta_{clk}}, 0)$, which is a conjunction of four model predicates, defined in [48], that formalize standard adversary, communication and synchronization assumptions.

$$\pi^{\text{X.509v2}}_{\Delta_{clk}} = \pi^{\mathsf{F}} \ \wedge \ \pi^{\text{2-rounds}}_{\text{SecInit}} \ \wedge \ \pi^{\text{Drift}}_{\Delta_{clk}} \ \wedge \ \pi^{\text{Wake-up}}_{\Delta_{clk}} \quad (3)$$

Specifically, π^{F} ensures that the adversary outputs all faulty entities in $\mathsf{T.F}$, $\pi^{\text{2-rounds}}_{\text{SecInit}}$ ensures two secure 'rounds' of initialization to securely exchange of the public keys of authorities, $\pi^{\text{Drift}}_{\Delta_{clk}}$ ensures bounded clock drift, and $\pi^{\text{Wake-up}}_{\Delta_{clk}}$ ensures reliable 'wake-up' service. Algorithm 8 shows the construction of the π^{F} to illustrate this approach. The remaining predicates can be found in [48].

B. Security Analysis

Theorem 1. *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 $\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}$.*

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 $\iota_M$
   $AdvWins \leftarrow [$
2:     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 $\iota_M \in T.\mathsf{N} - T.\mathsf{F}$
       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 $T.\tau[\tilde{e}] \leq \rho.\tau - \Delta$    ▷ $\iota_M$ is honest and $\iota_M$ monitors $\rho.\iota$ since (at least) $\Delta$ before the transparency-attestation.
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. $T.ent[\bar{e}] = \iota_M$
           and $T.opr[\bar{e}] =$ 'Audit'
           and $T.inp[\bar{e}] = \psi.id$
           and $\psi \notin T.out[\bar{e}]$
           and $T.\tau[\bar{e}] \geq \rho.\tau + \Delta$    ▷ $\iota_M$ is not aware of $\psi$ even though the $\Delta$ time period passed
   ]
6: if $AdvWins$ then return $\bot$ else return $\top$

7: procedure WasNotAccused$(T, \psi, pk, \rho, \iota)$
8:     return $\nexists \ddot{e}$ s.t. $T.ent[\ddot{e}] = \iota$
           and $T.out[\ddot{e}] = (\text{IA}, \rho.\iota)$
           and $T.\tau[\ddot{e}] \leq \rho.\tau + \Delta$    ▷ $\iota$ did not 'indicate/accuse' $\rho.\iota$, at least not until $\Delta$ time units after attestation
9:     and $\exists \check{e}$ s.t. $T.ent[\check{e}] = \iota$
           and $T.opr[\check{e}] =$ 'Audit'
           and $T.inp[\check{e}] = (\psi, \rho)$
           and $\mathcal{P}.\mathsf{PoM}(pk, T.out[\check{e}]) = \bot$
           and $T.\tau[\check{e}] > \rho.\tau + \Delta$    ▷ $\iota$ failed to produce 'proof' of $\rho.\iota$'s misbehavior, upon 'Audit' with input $\psi$ and $\rho$
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 $\rho_t$
   $AdvWins \leftarrow [$
3:     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 $\iota_M \in T.\mathsf{N} - T.\mathsf{F}$
       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 $T.\tau[\tilde{e}] \leq \rho_t.\tau - \Delta$    ▷ $\iota_M$ is honest and $\iota_M$ monitors $\rho_t.\iota$ since (at least) $\Delta$ before the transparency-attestation
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. $T.ent[\bar{e}] = \iota_M$
           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.
              $\rho'.attr = \mathsf{REV}$)
           )    ▷ 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** $\rho_t.\tau$
   ]
8: if $AdvWins$ then return $\bot$ else return $\top$
```

Algorithm 8 $\pi^{\mathsf{F}}(T)$ Predicate

```
1: return (
2:   $\forall \hat{e} \in \{0, \ldots, T.e\}$ :    ▷ 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 $T.\mathsf{F}$
   )
```

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

1) We first define a variation of $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ called $\overline{\mathrm{X.509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$, 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 $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$, where we execute an adversary $\mathcal{A}$ with the $\overline{\mathrm{X.509v2}}^{\mathcal{S},OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ scheme. Adversary $\mathcal{A}$ 'wins' if it outputs a message m and valid signature σ over m, using public verification key pk, without $\mathcal{A}$ receiving the signing key sk or asking the oracle to sign m.
3) After that, we 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 contradicts the security of $\mathcal{S}$.
4) We construct an adversary $\mathcal{A}$ and show it wins the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game with non-negligible probability, if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not satisfy each of $\xi_{\mathsf{ACC}}, \xi_{\mathsf{ReACC}}$ and ξ_{NReACC}.
5) The proof follows from $1-4$; see Appendix C.

□

VII. Conclusions and Future Work

In this work, we presented specifications for a secure PKI scheme. Our specifications can be applied to both classical, proposed and future PKI schemes. Future work includes an analysis of existing and proposed PKI schemes using these specifications; e.g., we hope to soon present proofs for X.509v3 and for secure variants of CT. We also expect future work to extend and refine our specifications, as well as to explore alternative methods for specifications of PKI schemes.

References

[1] L. M. Kohnfelder, "Towards a practical public-key cryptosystem." Bachelor's Thesis, Massachusetts Institute of Technology, 1978.

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

[3] 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

[4] 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.

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

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

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

[8] 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

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] S. Goldwasser and S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[25] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on computing*, vol. 17, no. 2, pp. 281–308, 1988.

[26] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

[33] A. Samoshkin, "SSL certificate revocation and how it is broken in practice," Medium, January 2018.

[34] D. Kim, B. J. Kwon, K. Kozák, C. Gates, and T. Dumitraș, "The broken shield: Measuring revocation effectiveness in the windows code-signing {PKI}," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 851–868.

[35] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, and C. Wilson, "An end-to-end measurement of certificate revocation in the web's pki," in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 183–196.

[36] 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.

[37] 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.

[38] 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.

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

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

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

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

[47] 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.

[48] "MoSS: Modular Specifications Security Framework," Anonymized. [Online]. Available: https://sites.google.com/view/provablysecure/moss

[49] 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.

[50] 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

[51] 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.

[52] 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.

[53] J. Hruska, "Apple, Microsoft buck trend, refuse to block unauthorized Chinese root certificates," ExtremeTech, April 2015.

[54] 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.

[55] Electronic Frontier Foundation (EFF). The EFF SSL Observatory. [Online]. Available: https://www.eff.org/observatory

[56] S. Eskandarian, E. Messeri, J. Bonneau, and D. Boneh, "Certificate Transparency with Privacy," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 329–344, 2017.

[57] "Namecoin." [Online]. Available: https://www.namecoin.org/

[58] 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.

[59] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 409–426.

[60] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004, https://eprint.iacr.org/2004/332.

[61] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. IEEE, 2001, pp. 136–145, online at https://eprint.iacr.org/2000/067.pdf, last updated Dec. 2018.

[62] U. Maurer, "Constructive cryptography–a new paradigm for security definitions and proofs," in *Joint Workshop on Theory of Security and Applications*. Springer, 2011, pp. 33–56.

[63] M. Backes, B. Pfitzmann, and M. Waidner, "A general composition theorem for secure reactive systems," in *Theory of Cryptography Conference*. Springer, 2004, pp. 336–354.

[64] 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

[65] 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.

Appendix A
X.509 Version 2: Construction

In this section, we formalize the design of the X.509 version 2 scheme (Algorithms 9-14). The resulting construction follows the common practices of the currently deployed X.509-based systems, except for allowing certain extensions, e.g., naming constraints for cross-certification, for simplicity and clarity of exposition, the construction supports two revocation mechanisms commonly used in practice, CRLs and OCSP.

A. System Entities and their Local State

In X.509v2, the set of authorities N consists of only *certificate authorities* (CAs), who perform the same tasks. Each CA 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.
- $s.\kappa$: the system's security parameter.
- $s.sk$: the secret signing key.
- $s.pk$: the public verification key.
- $s.certs$: the list of all certificate issued by the local 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.
- $s.interval$: when to produce an updated CRL proof.
- $s.\tau$: the value of the current local clock of the specific entity.
- $s.initRound$: a counter to track the initialization rounds; initialized to 0.

B. Algorithms

We now present the implementations of the PKI algorithms (Algorithms 9-14) specified in Definition 6 relevant to X.509v2. We do not present the $\mathsf{Receive}$, $\mathsf{Monitor}, \mathsf{PoM}$ and Audit algorithms as they are not required given X.509v2 design and goals. Specifically, there is no interaction between the CAs and consequently, no need for $\mathsf{Receive}$. X.509v2 does not provide advanced PKI security features, such as transparency and equivocation, and therefore does not need $\mathsf{Monitor}, \mathsf{PoM}$ or Audit algorithms.

1) The Init *Algorithm:* The initialization algorithm Init (Algorithm 9) operates in rounds. In the first round, each entity stores (locally) its identifier, security parameter and CRL update interval, and generates a private/public signing key pair. In addition, it outputs a 'Wake-up request', to make sure that the X.509v2.Wakeup algorithm will be executed at the inputted interval $s.interval$. In the second round, the algorithm outputs the local entity's public key, according to the convention discussed in §V-A. In the third and last round, the algorithm receives the public keys of all other entities and outputs those keys. The third steps is required to support secure communication assumptions described in the $\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}$ model.

2) The Issue *Algorithm:* The certificate issuing algorithm Issue (Algorithm 10) is used by a CA to issue an *accountable* certificate ψ. An honest CA invokes the algorithm only if the

Algorithm 9 X.509: initialization algorithm

```
1: procedure Init ($\iota, params, keys$)
2:   Switch $s.initRound$
3:     case 1:
4:       $s.initRound \leftarrow 2$                    ▷ Expect next round
5:       return (‘PubKey’, $s.\iota, s.pk$)            ▷ Output public key, see §V-A
6:     case 2:
7:       $s.initRound \leftarrow 3$                    ▷ Indicates init process done
8:       return $keys$                                 ▷ Output inputted public keys
9:     default:
10:      $s.\iota \leftarrow \iota$                    ▷ Store identifier
11:      $s.\kappa \leftarrow params.\kappa$           ▷ Store security parameter
12:      $s.interval \leftarrow params.interval$       ▷ Store CRL interval
13:      $(s.sk, s.pk) \leftarrow \mathcal{S}.\mathsf{Gen}(s.\kappa)$   ▷ Generate signature keys
14:      $s.initRound \leftarrow 1$                    ▷ Expect next round
15:      return (‘Wake-up request’, $s.interval$)      ▷ Next wake-up for CRL update
16:  end switch
17: end procedure
```

Algorithm 10 X.509: certificate issuance algorithm

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

Algorithm 11 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 12 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’, $s.interval$)          ▷ Set next wake-up for CRL update
6: end procedure
```

Algorithm 13 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 14 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
```

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 $\mathcal{S}.\mathsf{Sign}$ to produce a signature over the specific certificate fields (its serial number, subject, public information, validity period, issuer identity, and time of issuance). 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 11) is used by an *eligible* CA to revoke a certificate ψ. In X.509v2, an eligible CA is the one who initially issued the specific certificate. CAs revoke 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.509v2, 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 12) is used in X.509v2 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 the signature locally, so it can be retrieved via the Attest algorithm (Algorithm 13). 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 11), which will result in immediate execution of the X.509v2.Wakeup algorithm and immediate publish of the latest CRL.

5) The Attest *Algorithm:* In X.509v2, the certificate attestation algorithm Attest (Algorithm 13) 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 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 14) is used to check the validity of the following attestations: accountability, revocation and non-revocation. For accountability, the 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.

Appendix B
MoSS's Execution Process

See Algorithm 15 for the pseudo-code definition of MoSS's execution process. For more details about the framework, see [48].

Appendix C
X.509v2 Detailed Security Analysis

A. Preliminaries

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

Definition 7. *A signature scheme* $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *consists of the following probabilistic algorithms:*

- *Key generation* $\mathsf{Gen}(1^\kappa) \rightarrow (sk, vk)$*, with input security parameter* 1^κ *and output private signing key* sk *and public verification key* vk*.*
- *Signing* $\mathsf{Sign}(sk, m) \rightarrow (\sigma)$*, with input private signing key* sk *and a message* m*, and output signature* σ*.*
- *Verification* $\mathsf{Ver}(vk, m, \sigma) \rightarrow (\top/\bot)$*, 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 8. *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*.*

Algorithm 15 Adversary-Driven Execution Process $\mathbf{Exec}_{\mathcal{A},\mathcal{P}}(params)$

```
1: $(s_{\mathcal{A}}, \mathsf{N}) \leftarrow \mathcal{A}(params.\mathcal{A}, params.\mathcal{P})$    ▷ Initialize adversary with $params.\mathcal{A}, params.\mathcal{P}$
2: $\forall i \in \mathsf{N}:\ s_i \leftarrow \mathcal{P}[\text{'Init'}](\perp, (i, params.\mathcal{P}), \perp)$    ▷ Initialize entities' local state
3: $s \leftarrow params$    ▷ Initial exec state
4: $e \leftarrow 0$    ▷ Initialize loop's counter
5: repeat
6:     $e \leftarrow e + 1$    ▷ Advance the loop counter
7:     $(ent[e], opr[e], type[e], inp[e], clk[e], \tau[e]) \leftarrow \mathcal{A}(s_{\mathcal{A}})$    ▷ $\mathcal{A}$ selects entity $ent[e]$, operation $opr[e]$, operation type $type[e] \in \{\text{`'}, \text{`}\mathcal{P}\text{'}\}$, input $inp[e]$, clock $clk[e]$, and real time $\tau[e]$ for event $e$.
8:     $(s_{ent[e]}, out[e], sec\text{-}out[e][\cdot]) \leftarrow \mathcal{P}[opr[e]](s_{ent[e]}, inp[e], clk[e])$
9:     $(s_{\mathcal{A}}, out_{\mathcal{A}}, \mathsf{F}) \leftarrow \mathcal{A}(s_{\mathcal{A}}, out[e])$    ▷ $\mathcal{A}$ decides when to terminate the loop ($out_{\mathcal{A}} \neq \perp$), based on $out[e]$
10:    $sLog[e] \leftarrow (s_{\mathcal{A}}, s_{ent[e]}, s)$    ▷ Save state of all entities to return as part of $T$
11: until $out_{\mathcal{A}} \neq \perp$
12: $T \leftarrow (out_{\mathcal{A}}, e, \mathsf{N}, \mathsf{F}, ent[\cdot], opr[\cdot], type[\cdot], inp[\cdot], clk[\cdot], \tau[\cdot], out[\cdot], sec\text{-}out[\cdot][\cdot], sLog[\cdot])$
13: Return $T$    ▷ Output transcript of run
```

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.

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

The first step towards the proof is to define a variation of the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ scheme, denoted as $\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$, where the cryptographic signing operations of entity $\iota \in \mathsf{N}$ are performed via an oracle instead by ι itself.

Definition 9. *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* $\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ *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) *In the* $\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}.\mathsf{Init}$ *algorithm, for entity* ι*, replace the following line:*
$$(s.sk, s.pk) \leftarrow \mathcal{S}.\mathsf{Gen}(s.sec)$$
with:
$$(s.sk, s.pk) \leftarrow (\perp, pk)$$
where pk *is the public verification key of the* sk *signing key, given as input to* $\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$.
2) *When* ι *runs an algorithm 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:
$$OSign(sk, \cdot)$$

C. 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, where an adversary $\mathcal{A}$ is executed with the $\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ scheme defined in §C-B. $\mathcal{A}$ wins if it was able to produce a message m and signature σ, where σ is a valid signature over m according to the public key pk, which the adversary has no access to its matching secret key sk, and $\mathcal{A}$ did not ask the oracle to sign m. The motivation here is that if there exists an adversary $\mathcal{A}$ that wins the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^{\kappa}, \mathsf{N})$ game, then such adversary could be used to contradict the security of the signature scheme itself (as we show below).

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 $\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}$ scheme, i.e.,
$$T \leftarrow \mathbf{Exec}_{\mathcal{A},\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params)$$
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 $\mathcal{M}$.
   d) ι is an honest authority, i.e., $\iota \in \mathsf{N} - \mathsf{F}$

   Otherwise, the experiment outputs 0.

D. 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*

$$Pr\left[\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^{\kappa},\mathsf{N})=1\right]\notin Negl(1^{\kappa}) \quad (4)$$

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 §C-C, 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, with non-negligible probability, $\mathcal{A}$ outputs a message m and a valid signature σ over m, for a random (sk, pk) and where $\mathcal{A}$ has only the verification key pk and oracle access to $OSign(sk,\cdot)$ (with sk being the signing key). To 'win', $\mathcal{A}$ must not request the oracle to sign m.

Therefore, according to definition of existential unforgeability (Def. 8), the following holds for $\mathcal{A}$

$$Pr\left[\mathbf{Exp}^{EU}_{\mathcal{A},\mathcal{S}}(1^{\kappa})=1\right]\notin Negl(1^{\kappa}) \quad (5)$$

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

We will now show that if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not satisfy any of the requirements (accountability, revocation accountability and non-revocation accountability), then an adversary $\mathcal{A}$ may win, with non-negligible advantage, the $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ game.

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

We next show that if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not ensure 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 satisfy accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{ACC}}$ such that*

$$Pr\left[\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A}_{\mathsf{ACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^{\kappa},\mathsf{N})=1\right]\notin Negl(1^{\kappa}) \quad (6)$$

Proof. From Definition 3, if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not satisfy accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{ACC}}$ that satisfies $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, i.e., $\mathcal{A}_{\mathsf{ACC}} \models_{\text{poly}} \mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, yet

$$\epsilon^{\mathsf{ACC}}_{\mathcal{A}_{\mathsf{ACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)\notin Negl(params.\mathcal{P}.1^{\kappa}) \quad (7)$$

In other words, from Definition 1, $\mathcal{A}_{\mathsf{ACC}}$ satisfies

$$\Pr\left[\begin{array}{c}\mathsf{ACC}\,(T)=\bot,\ \text{where}\\ T\leftarrow \mathbf{Exec}_{\mathcal{A}_{\mathsf{ACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)\end{array}\right]\notin Negl(1^{\kappa}) \quad (8)$$

Note that for $\mathsf{ACC}\,(T)=\bot$, the adversary must have shown an accountability failure for some non-corrupt entity; let us denote this party by ι'.

Now, the inputs and outputs (and T) of an execution of $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ with the a non-faulty entity ι, are identical to these resulting from an execution of the 'regular experiment' $\mathbf{Exec}_{\mathcal{A}_{\mathsf{ACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)$ (with the same protocol, adversary and random strings). This is since the $OSign(sk,\cdot)$ oracle returns exactly the same result as the signature function it replaces. In particular, the transcript T of the execution is independent of the random choice of the party ι in line 2 of $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$. Since the behavior of ι is the same as any other honest entity, this holds also when $\iota'=\iota$; and since ι was selected randomly, this occurs with probability at least $\frac{1}{|\mathsf{N}|}$. Hence:

$$\epsilon^{\mathsf{ACC}}_{\mathcal{A}_{\mathsf{ACC}},\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params)\geq\frac{1}{|\mathsf{N}|}\cdot\epsilon^{\mathsf{ACC}}_{\mathcal{A}_{\mathsf{ACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params) \quad (9)$$

which means that

$$\epsilon^{\mathsf{ACC}}_{\mathcal{A}_{\mathsf{ACC}},\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params)\notin Negl(params.\mathcal{P}.1^{\kappa}) \quad (10)$$

Therefore, following Definition 1, $\mathcal{A}_{\mathsf{ACC}}$ satisfies

$$\Pr\left[\begin{array}{c}\mathsf{ACC}\,(T)=\bot,\ \text{where}\\ T\leftarrow \mathbf{Exec}_{\mathcal{A}_{\mathsf{ACC}},\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params)\end{array}\right]\notin Negl(1^{\kappa}) \quad (11)$$

Now, all that is left to show is that if Eq. 11 holds then Eq. 6 also holds. First, the return value of the accountability predicate (Alg. 1) is $\bot$, i.e., $\mathcal{A}$ wins, only if $\mathsf{WasValid}$ holds, among other criteria. More precisely, $\mathcal{A}$ wins only if it outputs a valid certificate ψ with an honest authority listed as issuer $\psi.issuer$, an accountability attestation ρ (i.e., $\rho.attr=\mathsf{ACC}$), and the public key pk of $\psi.issuer$, although $\psi.issuer$ did not issue ψ (by invoking the $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Issue}$ algorithm). Namely, it holds:

$$\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}(\psi,pk,\rho)=\top \quad (12)$$

Second, according to the implementation of $\mathsf{WasValid}$ (Alg. 14), when given an accountability attestation ($\rho.attr=\mathsf{ACC}$), $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}$ returns $\mathcal{S}.\mathsf{Ver}(pk,\psi,\rho.\sigma)$, i.e., returns $\top$ iff:

$$\mathcal{S}.\mathsf{Ver}(pk,\psi,\rho.\sigma)=\top \quad (13)$$

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 9); specifically the sign/verify key pair is generated in line 13, 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. 11, the value described in Eq. 12 holds with non-negligible probability, and as a result, Eq. 13 also holds with non-negligible probability. 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. 6. ☐

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

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

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

$$Pr\left[\overline{\textbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A}_{\mathsf{ReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^{\kappa},\mathsf{N})=1\right]\notin Negl(1^{\kappa}) \qquad (14)$$

Proof. From Definition 3, if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not satisfy revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{ReACC}}$ that satisfies $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, i.e., $\mathcal{A}_{\mathsf{ReACC}}\models_{\text{poly}}\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, yet

$$\epsilon^{\mathsf{ReACC}}_{\mathcal{A}_{\mathsf{ReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)\notin Negl(params.\mathcal{P}.1^{\kappa}) \qquad (15)$$

In other words, from Definition 1, $\mathcal{A}_{\mathsf{ReACC}}$ satisfies

$$\Pr\left[\begin{array}{c}\mathsf{ReACC}\,(T)=\bot,\ \text{where}\\ T\leftarrow\textbf{Exec}_{\mathcal{A}_{\mathsf{ReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)\end{array}\right]\notin Negl(1^{\kappa}) \qquad (16)$$

Note that for $\mathsf{ReACC}\,(T)=\bot$, the adversary must have shown a revocation accountability failure for some non-corrupt entity; let us denote this party by ι'.

Now, the inputs and outputs (and T) of an execution of $\overline{\textbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ with the a non-faulty entity ι, are identical to these resulting from an execution of the 'regular experiment' $\textbf{Exec}_{\mathcal{A}_{\mathsf{ReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)$ (with the same protocol, adversary and random strings). This is since the $OSign(sk,\cdot)$ oracle returns exactly the same result as the signature function it replaces. In particular, the transcript T of the execution is independent of the random choice of the party ι in line 2 of $\overline{\textbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$. Since the behavior of ι is the same as any other honest entity, this holds also when $\iota'=\iota$; and since ι was selected randomly, this occurs with probability at least $\frac{1}{|\mathsf{N}|}$. Hence:

$$\epsilon^{\mathsf{ReACC}}_{\mathcal{A}_{\mathsf{ReACC}},\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params)\geq\frac{1}{|\mathsf{N}|}\cdot\epsilon^{\mathsf{ReACC}}_{\mathcal{A}_{\mathsf{ReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params) \qquad (17)$$

which means that

$$\epsilon^{\mathsf{ReACC}}_{\mathcal{A}_{\mathsf{ReACC}},\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params)\notin Negl(params.\mathcal{P}.1^{\kappa}) \qquad (18)$$

and therefore, following Definition 1, $\mathcal{A}_{\mathsf{ReACC}}$ satisfies

$$\Pr\left[\begin{array}{c}\mathsf{ReACC}\,(T)=\bot,\ \text{where}\\ T\leftarrow\textbf{Exec}_{\mathcal{A}_{\mathsf{ReACC}},\overline{\mathrm{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params)\end{array}\right]\notin Negl(1^{\kappa}) \qquad (19)$$

Now, all that is left to show is that if Eq. 19 holds then Eq. 14 also holds. First, according to the description of the revocation accountability requirement (Alg. 2), the return value of the revocation accountability predicate is $\bot$, i.e., $\mathcal{A}$ wins, only if, among other criteria, holds:

$$\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}(\psi,pk,\rho) \qquad (20)$$

for a certificate ψ outputted by the adversary, ρ is non-revocation accountability attestation ($\rho.attr=\mathsf{ReACC}$), 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. 14, the algorithm executes

$$\mathcal{S}.\mathsf{Ver}(pk,\rho.tbs,\rho.\sigma) \qquad (21)$$

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 9); specifically the sign/verify key pair is generated in line 13, 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. 19, the value described in Eq. 20 holds with non-negligible probability, and as a result, Eq. 21 also holds with non-negligible probability. Therefore, adversary $\mathcal{A}_{\mathsf{ReACC}}$ 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 $\mathcal{A}_{\mathsf{ReACC}}$ adversary satisfies Eq. 14. ☐

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

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

$$Pr\left[\overline{\textbf{Exp}}^{Forge,\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A}_{\mathsf{NReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(1^{\kappa},\mathsf{N})=1\right]\notin Negl(1^{\kappa}) \qquad (22)$$

Proof. From Definition 3, if $\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ does not satisfy non-revocation accountability under model $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{NReACC}}$ that satisfies $\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, i.e., $\mathcal{A}_{\mathsf{NReACC}}\models_{\text{poly}}\mathcal{M}^{\mathrm{X.509v2}}_{\Delta_{clk}}$, yet

$$\epsilon^{\mathsf{NReACC}}_{\mathcal{A}_{\mathsf{NReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)\notin Negl(params.\mathcal{P}.1^{\kappa}) \qquad (23)$$

In other words, from Definition 1, $\mathcal{A}_{\mathsf{NReACC}}$ satisfies

$$\Pr\left[\begin{array}{c}\mathsf{NReACC}\,(T)=\bot,\ \text{where}\\ T\leftarrow\textbf{Exec}_{\mathcal{A}_{\mathsf{NReACC}},\mathrm{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)\end{array}\right]\notin Negl(1^{\kappa}) \qquad (24)$$

Note that for $\mathsf{NReACC}\,(T) = \bot$, the adversary must have shown a non-revocation accountability failure for some non-corrupt entity; let us denote this party by ι'.

Now, the inputs and outputs (and T) of an execution of $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ with the a non-faulty entity ι, are identical to these resulting from an execution of the 'regular experiment' $\mathbf{Exec}_{\mathcal{A}_{\mathsf{NReACC}},\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params)$ (with the same protocol, adversary and random strings). This is since the $OSign(sk,\cdot)$ oracle returns exactly the same result as the signature function it replaces. In particular, the transcript T of the execution is independent of the random choice of the party ι in line 2 of $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$. Since the behavior of ι is the same as any other honest entity, this holds also when $\iota' = \iota$; and since ι was selected randomly, this occurs with probability at least $\frac{1}{|\mathsf{N}|}$. Hence:

$$\epsilon^{\mathsf{NReACC}}_{\mathcal{A}_{\mathsf{NReACC}},\overline{\text{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params) \geq \frac{1}{|\mathsf{N}|}\cdot\epsilon^{\mathsf{NReACC}}_{\mathcal{A}_{\mathsf{NReACC}},\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}(params) \tag{25}$$

which means that

$$\epsilon^{\mathsf{NReACC}}_{\mathcal{A}_{\mathsf{NReACC}},\overline{\text{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params) \notin Negl(params.\mathcal{P}.1^{\kappa}) \tag{26}$$

and therefore, following Definition 1, $\mathcal{A}_{\mathsf{NReACC}}$ satisfies

$$\Pr\left[\begin{array}{c} \mathsf{NReACC}\,(T) = \bot,\ \text{where} \\ T \leftarrow \mathbf{Exec}_{\mathcal{A}_{\mathsf{NReACC}},\overline{\text{X.509v2}}^{OSign(sk,\cdot)}_{\mathsf{N},\iota,pk}}(params) \end{array}\right] \notin Negl(1^{\kappa}) \tag{27}$$

Now, all that is left to show is that if Eq. 27 holds then Eq. 22 also holds.

First, according to the description of the non-revocation accountability requirement (Alg. 3), the return value of the non-revocation accountability predicate is $\bot$, i.e., $\mathcal{A}$ wins, only if, among other criteria, holds:

$$\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}(\psi, pk, \rho) \tag{28}$$

for a certificate ψ outputted by the adversary, ρ is non-revocation accountability attestation ($\rho.attr = \mathsf{NReACC}$), 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 $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Revoke}$ algorithm.

Second, according to the implementation of $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{WasValid}$, as described in Alg. 14, the algorithm executes

$$\mathcal{S}.\mathsf{Ver}(pk, \rho.tbs, \rho.\sigma) \tag{29}$$

Lastly, the only place in $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ where an honest authority ι computes its keys is in the $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Init}$ algorithm (Algorithm 9); specifically the sign/verify key pair is generated in line 13, using the $\mathcal{S}.\mathsf{Gen}$ algorithm. Furthermore, the signing key is only used in algorithms: $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Issue}$, $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Wakeup}$, $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Attest}$, 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 $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}.\mathsf{Attest}$ algorithm.

Thus, following Eq. 27, the value described in Eq. 28 holds with non-negligible probability, and as a result, Eq. 29 also holds with non-negligible probability. Therefore, adversary $\mathcal{A}_{\mathsf{NReACC}}$ 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 $\mathcal{A}_{\mathsf{NReACC}}$ adversary satisfies Eq. 22. □

G. Completing the Proof

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

Theorem 1. *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. Assume that $\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}$ 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 $\overline{\mathbf{Exp}}^{Forge,\mathcal{M}^{\text{X.509v2}}_{\Delta_{clk}}}_{\mathcal{A},\text{X.509v2}^{\mathcal{S}}_{\mathsf{N}}}$ 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. (6).

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

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