Modeling Long-Term Signature Validation for Resolution of Dispute

Moez Ben MBarka[1,2], Francine Krief[2], and Olivier Ly[2]

[1] Cryptolog International, Paris, France
moez.benmbarka@cryptolog.com
[2] LaBRI, University of Bordeaux 1, 33400 Talence, France
{krief,ly}@labri.fr

Abstract. This paper considers the case where a dispute occurs between a verifier and a signer about the validity of a digital signature. In non-repudiation services such dispute may occur long after the signature creation and approval. We present a security model for digital signature validation with the notion of dispute. The first contribution of this paper is the definition of the semantics of a Resolution of Dispute Rule (RDR) in the scope of this model. The second contribution is a calculus for reasoning about the validation of digital signatures at a particular date which may be in the past (so-called long-term signature validation). This calculus is then used to implement the RDR. The usefulness of the calculus is demonstrated through modeling Evidence Record Syntax (ERS), one of the main protocols used in practice for long-term signature validation.

Keywords: long-term signature validation, resolution of dispute, formal calculus, Public Key Infrastructure.

1 Introduction

A digital signature is a piece of digital data that can be used to guarantee integrity, authenticity, and non-repudiation in electronic communications. It is nowadays based on public key cryptography. To see how it works, suppose that a signer Bob wants to sign a message that will be verified by Alice. Bob computes the hash of the message and then encrypts the output using his private key. Alice can then verify the signature using Bob's public key. This mathematical verification allows to prove the integrity of the message and the binding between the message and a public key. Authenticity and non repudiation come from verifying the binding between this public key and the signer using public key certificates. In the widely used X.509-based PKI model (PKIX [1]), this binding is signed by a Certification Authority (CA) which is trusted to verify the identity of the certificate's owner (called certificate's subject) before signing the certificate. The certificate's owner may be another CA (certification authority certificate) or an entity which is not a CA (end entity certificate). Various circumstances (e.g. key compromise) may cause a certificate to become invalid prior to its expiration. Under such circumstances, the CA needs to revoke the certificate [1, 2].

S. Mödersheim and C. Palamidessi (Eds.): TOSCA 2011, LNCS 6993, pp. 78–97, 2012.

1.1 Statement of the Problem

Digital signatures are gradually acquiring legal equivalence of handwritten signatures promoting their use to secure business and commercial digital communications. Nowadays digital signature schemes and the infrastructure within which they operate have time limitations which may be due to some inner expiration dates (e.g. the expiration of a certificate) or to the fact that the underlying cryptographic security relies on the presumed computational difficulty of certain theoretic problems (e.g. integer factorization) [3]. This situation is disturbing considering that there are many cases such as government records where the signatures are required by civil laws to be kept valid for a long period of time.

We address this issue by modeling long-term signature validation in the scope of a dispute between the typical actors: a verifier (Alice) who claims the validity of the signature at some date (possibly in the past) and a signer (Bob) who denies this claim. Imagine for a moment a judge trying to resolve a dispute between Alice and Bob in the year 2030. To prove their point, the parties invoke several records that were digitally signed by them in the year 2005. The judge needs to decide if the evidence presented is admissible in court. We use our model to define the semantics of a resolution of dispute rule that can be used by the judge to take a decision about the validity of the presented digital signatures.

1.2 Related Work

Long-Term Validation. It is already known that time-stamp services can be used to maintain digital document (in particular digital signatures) integrity and authenticity. The time-stamp is a digital attestation signed by a trusted third party, called Time-Stamping Authority (TSA), that a submitted document has been presented to the TSA at a certain time. For a document m, the time-stamp is a signature where the signer is a TSA and the signed message includes the pair $(T, h(m)))$ such as T is the current time retrieved from a reliable source of time and h is a cryptographic hash function.

Many technical frameworks for long-term signatures validation are defined in the literature [4,5,6,7,8] and are mainly based on time-stamping: the signature is time-stamped indicating that it was created at a moment before a subsequent key or algorithm compromise; this time-stamp is refreshed before the used algorithms or keys become compromised, ensuring its validity through the years. Most of these techniques rely on the use of Evidence Record Syntax (ERS) [8]. ERS uses a chain of time-stamps to prove the existence of some digital data at a certain date. Each time-stamp may protect a single data object or to a group of objects using a hash tree where only the root of the tree is time-stamped. ERS defines two renewal algorithms: the simple and the complex renewal methods. In the case of the simple renewal, only the last generated time-stamp has to be time-stamped again. In this scenario, it is not necessary to access the initially archived data objects themselves, since the algorithms used for the previous time-stamps are recognized as being still reliable. The simple renewal method is not sufficient when the hash functions used in the previous time-stamps become insecure. In this scenario, not only the initial time-stamps but also the initial data have to be time-stamped again using newer algorithms.

PKI Calculus. In [9], Maurer presented the pioneer deterministic calculus for PKI. The calculus is based on four statements (authenticity, trust, certificate, recommendation) and two inference rules. Starting from an initial set of statements (called Alice's initial view), Alice uses the inference rules to derive new statements. In this model, Alice can use a certificate issued by a CA X for a user Bob if and only if she can derive the following statements: she has a certificate signed by X for Bob, she knows X's public key and believes that it is authentic and she trusts X to correctly authenticate the owner of a public key. The objective of Alice is to derive these statements from her initial view using the inference rules. Later, many authors [10, 11] revisited this model to add additional concepts such as time, certificate expiration and revocation.

Signature Validity Models. In [12], the authors surveyed three validity models for digital signatures:

1. Shell model: in this model, the signature is always validated at the current date. The signature becomes invalid once the signer certificate gets expired.
2. Modified shell model: The signature is validated at the signing date and not at the verification date. Therefore, the expiration or revocation of the signer certificate does not affect the signature validity.
3. Chain model: in addition to the signing date, this model takes into account the issuance date of each certificate in the certificate validation chain. Not only, the expiration of the signer certificate does not affect the signature validity but also the signature remains valid even if an intermediate CA is revoked before the signing date.

The natural conclusions from these models are that:

– the shell model is only suitable for authentication services where the validity of the signature is required for short term.
– the modified shell and the chain models have the fundamental property: if a digital signature is valid at the signing date, it remains valid forever. Therefore, these models are suitable for long-term validation.

1.3 Motivations and Organization of the Paper

While many long-term signature validation frameworks [4, 5, 6] and standards [8, 7] are currently in use in practice, to the best of our knowledge, there is no formal model showing that they provide the evidences they claim. These frameworks miss formal definitions of long-term validation semantics. Defining such semantics is necessary to have a formal calculus. This is the first contribution of this paper. On the other hand, Maurer's calculus and its extensions showed above can be used to reason about public key authenticity but miss many features required to deal with long-term validation. The second contribution is the proposition of a formal calculus which captures these features (Figure 1).

Feature	our calculus	Maurer's calculus and extensions
Authenticity and revocation	yes	yes
Time aware signature validation	yes	no
Algorithm security	yes	no
Proof of existence	yes	no

Fig. 1. Our calculus vs previous calculus based on Maurer's model

The rest of the paper is organized as follows: Section 2 presents a simple security model for digital signature validation. Section 3 defines the resolution of dispute in the scope of this model. Section 4 presents our calculus which is used to implement the Resolution of Dispute Rule in Section 5. Section 6 uses the calculus to model ERS. Finally, Section 7 concludes the paper.

2 Security Model and Notations

In our model, a digital signature is denoted $Sig(X, m, sa, sv)$ where X identifies the signer, m is the signed message, sa identifies the signature algorithm, and sv is the signature value computed on m using sa and the private key corresponding to X's public key. The integrity and authenticity come from the cryptographic properties of the signature algorithm sa (namely, existential unforgeability against chosen message attacks [13]). A signature is said "cryptographically valid" if and only if sv verifies with X's public key on the message m using the signature algorithm sa. Non-repudiation relies on the authenticity of the signer's public key that can be derived from the certificate. The most widely PKI trust model is a hierarchy of CAs. In this model, the validation of a certificate at some date T consists in building and validating a certificate path from a trusted CA (called trust anchor) to the certificate to validate. Below, we combine the certificate validation algorithm defined in PKIX [1] with the cryptographic verification to define a Signature Validation Function (SVF). In the model, we accept SVF as the mean to prove integrity, authenticity and non-repudiation.

Definition 1 (Signature Validation Function (SVF)). SVF *takes as input the following: a signature $Sig(X, m, sa, sv)$, a set of trust anchors $\mathcal{K}$, a set of validation objects $\mathcal{O}$, and a date T. The function returns a pair $(status, \mathcal{P})$ such that $status = true$ if and only if:*

- *the signature is cryptographically valid.*
- *there exists a sequence of certificates $Cert_1, \ldots, Cert_n$ in $\mathcal{O}$ such that:*
  - *$\forall i \in [1, n-1]$, $Cert_i$'s subject is $Cert_{i+1}$'s issuer and $Cert_i$ is a certification authority certificate.*
  - *$Cert_1$ is issued by a trust anchor in $\mathcal{K}$.*
  - *$Cert_n$ is an end entity certificate and is issued to X.*
  - *$\forall i \in [1, n]$, $Cert_i$ is valid at T and there exists a revocation object $Rev_i \in \mathcal{O}$ signed by $Cert_i$'s issuer indicating that $Cert_i$ is not revoked at T.*

- $\forall i \in [1, n]$, *the signatures in* $Cert_i$ *and* Rev_i *are cryptographically valid.*

If $status = true$ *then the signature is said "valid" at* T *and* $\mathcal{P}$ *is the sequence* $\{Cert_1, Rev_1, \ldots, Cert_n, Rev_n\}$. *Otherwise, the signature said is "invalid" at* T. ◦

The function above requires a set of trust anchors that the verifier trusts without prior verification. In addition, the verifier needs to trust some assumptions about the security of the used algorithms. This yields a particular set that we will call Initial View of Trust ($\mathcal{IVT}$):

Definition 2 (Initial View of Trust ($\mathcal{IVT}(T)$)). $\mathcal{IVT}(T)$ *is composed of the following:*

- *a set of trust anchors* $\mathcal{K} = \{TA_1, \ldots, TA_n\}$ *such that* $\forall i \in [1, n]$, TA_i *is a CA identified by a certification authority certificate.*
- *a set of algorithm trust assertions* $\mathcal{A} = \{(a_1, T_{a_1}), \ldots, (a_k, T_{a_k})\}$, *such that* $\forall i \in [1, k], a_i$ *is a cryptographic hash function or a signature algorithm considered secure until at least the date* T_{a_i}.

The semantics of "secure" depends on the nature of the cryptographic algorithm. In our model, we will say that an algorithm is secure until at least T if it is commonly believed that the scientific knowledge and computing power will not allow to break the security properties of the algorithm until at least T.

3 Resolution of Dispute

In the model, Bob uses his private key to sign some digital data and Alice verifies the signature using Bob's public key. For instance, consider a scenario where Bob signs a contract with Alice to pay monthly some amount of money. If in the future, Bob's certificate is revoked, he may claim that he never signed the contract and stop paying Alice. This leads to a dispute situation: on one hand, Alice claims the validity of a signature and that this signature has been created at a date in the past (before revocation); on the other hand, Bob denies these claims. First, we define the dispute. Next, we define the resolution of dispute.

Definition 3 (Dispute). *If Alice (A) claims at a date* $T \geq T_d$ *that a signature* $Sig(B, m, sa, sv)$ *is signed by Bob (B) and is valid at* T_d*; and if Bob denies any of these claims, we say that Alice is in a dispute* $\mathsf{Dispute}(A, B, S, T_d)$ *with Bob.* T_d *is called the dispute date and* T *the resolution date.* ◦

Both Alice and Bob rely on a third party, called the Judge, to resolve the dispute. The Judge can resolve the dispute using a Resolution of Dispute Rule (RDR) at T based on validation objects (certificates, CRLs...) provided by Alice:

Definition 4 (Resolution of Dispute Rule). *If a dispute* $\mathsf{Dispute}(A, B, S, T_d)$ *occurs between Alice and Bob at* T, *the Judge executes the* RDR. *The* RDR *takes as input the following: the signature* S, *a set of validation objects* $\mathcal{O}$ *provided by Alice,* $\mathcal{IVT}(T)$, *the dispute date* T_d, *and the resolution date* T. *It outputs a pair*

$(status, \mathcal{P})$ such that $status \in \{true, false\}$ and $\mathcal{P} \subseteq \{S\} \cup \mathcal{O} \cup \mathcal{IVT}(T)$. If $status = true$, the Judge rules in favor of Alice. Otherwise, he rules in favor of Bob. In addition, if $status = true$ then $\mathcal{P} \neq \emptyset$ is the proof of the validity of the claims of Alice and is called the judgment proof. ◦

3.1 Threat Model

The Judge is trusted to honestly run the RDR and honestly output the result. However, any of the other actors can run the RDR and thus verify the output. Indeed, the Judge can be seen as a procedure (e.g. enforced by a court) whose output can be verified by other actors.

$\mathcal{IVT}$ is a public set trusted by all actors. This assumes that no attacker has significantly more computing power or cryptanalytic knowledge than what is widely known. $\mathcal{IVT}$ may naturally be updated to take account of new algorithms and authorities. We will assume that any update to $\mathcal{IVT}$ does not remove or reduce the validity interval of any algorithm or authority in the previous $\mathcal{IVT}$.

Finally, a traditional assumption in a PKI context is that honest involved authorities (namely TSAs and CAs) act honestly. In particular, we assume that the CAs conform to their Certification Policies (CP) that describe certificate issuance and revocation rules. We assume that the CPs of all involved CAs guarantee at least the following: a certificate can be revoked but not suspended and a signer can not access the signing key before the issuance date of her certificate. We will also assume that all honest CAs provide instant revocation for all honestly issued certificates during their validity intervals: there is no delay between effective compromise and the publication of the revocation information. We are aware that in the real world there is an inevitable delay between the compromise and the revocation. For the sake of simplicity, this delay will be first ignored but will be discussed later.

3.2 RDR Semantics

The objective of RDR is to build proofs of the validity of Alice's claims at the resolution date. If such proofs can be built in a verifiable way, the Judge rules in favor of Alice, otherwise, he rules in favor of Bob. Note that the latter case does not mean that Alice's claims are provably false, but only that these claims can not be proven using the provided inputs. We start by defining two properties that must be satisfied by the RDR. The first requirement is Correctness. This means that if the Judge rules in favor of Alice then her claims are correct. Naturally, we use the modified shell model: the signature is validated at the signing date.

Property 1 (Correctness). Given the the signature $S_d = Sig(B, m, sa, sv)$ and the $\mathsf{Dispute}(A, B, S_d, T_d)$; if $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T_d, T)$ outputs *true* then S_d is created at a date $T_s \leq T_d$ and is valid at T_s ◦

The second consideration is that a digital signature can not be repudiated. In other words, If a signature is accepted valid at some date is should remain valid forever: if a Judge rules in favor of the validity of a signature at a date T_d, he would rule in favor of the validity of the signature at any date after the date for which he has a proof of the existence and unforgeability of the signature.

Property 2 (Eternal Validity). Given the signature $S_d = Sig(B, m, sa, sv)$ and the dispute $\mathsf{Dispute}(A, B, S_d, T_d)$; if $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T_d, T)$ outputs *true* and S_d is proven to exist at T_s and uses an algorithm trusted to be secure until at least T_s such that $T_s \leq T_d$, then $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T'_d, T)$ outputs *true* $\forall\ T'_d \geq T_s$. ◦

Now let go to the definition of the RDR semantics:

Definition 5 (RDR Semantics). *Given the signature $S_d = Sig(B, m, sa, sv)$, the set of the validation objects $\mathcal{O}$ provided by Alice and the* $\mathsf{Dispute}(A, B, S_d, T_d)$*;* $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T_d, T)$ *outputs true if and only if there exists a date $T_s \leq T_d$ such that the Judge can prove that all the following statements are valid at T:*

1. *there exists a date $T_x \geq T_s$ such that* $\mathsf{SVF}(S_d, \mathcal{K}, \mathcal{O}, T_x) = (true, \mathcal{P})$*.*
2. *each validation object V in $\mathcal{P}$ exists at a date $T_V \leq T_x$ and uses an algorithm trusted to be secure until at least T_V.*
3. *S_d exists at T_s and uses an algorithm trusted to be secure until at least T_s.* ◦

Proposition 1. *The* RDR *Semantics has both the Correctness and Eternal Validity properties.*

Proof.

- Correctness:The first part of the correctness is obvious: since S_d exists at T_s, then S_d is created at a date $T'_s \leq T_s \leq T_d$. It remains to prove that S_d is valid at T'_s. Let us denote $\mathcal{IVT}(T) = (\mathcal{K}, \mathcal{A})$ and $\mathcal{P} = \{Cert_1, Rev_1, \ldots, Cert_n, Rev_n\}$ the sequence of certificates and revocation objects returned by SVF at T_x and $\{CA_0, \ldots, CA_{n-1}\}$ the set of CAs such that $\forall i \in [1, n-1]$, CA_i is the CA identified in $Cert_i$, and CA_0 is the trust anchor from $\mathcal{K}$ that issued $Cert_1$. $\forall i$, we will denote $[T_{b_i}, T_{e_i}]$ the validity interval of $Cert_i$. First, we show by induction that $\forall i \in [1, n]$, $Cert_i$ and Rev_i are honestly generated.
  - for $i = 1$, $Cert_1$ and Rev_1 are honestly generated by CA_0 because they are created, using a secure algorithm, by a trust anchor which can not be compromised.
  - assume that $Cert_i$ and Rev_i are honestly generated. If $Cert_{i+1}$ or Rev_{i+1} is not honestly generated but forged (say by Alice), then Alice needs to access to the signing key (because $Cert_{i+1}$ and Rev_{i+1} are created at date before T_x using a secure algorithm). So, we have two cases:
    * The private key of CA_i is compromised (anyone including Alice may have access to this private key). So, Rev_i is also forged (because it indicates that $Cert_i$ is not revoked at T_x).
    * CA_i is not an honest CA. So $Cert_i$ is forged.

  Then, either $Cert_i$ or Rev_i is forged which contradicts the induction hypothesis. Hence, $Cert_{i+1}$ and Rev_{i+1} are honestly generated. Thus, $\forall i \in [1, n]$, $Cert_i$ and Rev_i are honestly generated by honest CAs.

On the other hand, a private key is not issued before the certificate issuance date. Since the creation of the signature at T'_s needs the private key (the signature algorithm is secure at $T'_s \leq T_s$), this gives $T'_s \in [T_{b_n}, T_d]$ and therefore $\forall i \in [1, n]$, $T'_s \in [T_{b_i}, T_{e_i}]$. Since $Cert_i$ is not revoked at T_x, then it is not revoked at any date in $[T_{b_i}, T_x]$ (a certificate can not be suspended) and in particular at T'_s. In addition, since $\forall i \in [1, n]$, $Cert_i$ is honestly issued by an honest CA, there exists a revocation object Rev'_i issued by CA_{i-1} that indicates that $Cert_i$ is not revoked at T'_s. Thus, the sequence $\mathcal{P}' = \{Cert_1, Rev'_1, \ldots, Cert_n = B, Rev'_n\}$ verifies the conditions of SVF and then $SVF(S, \mathcal{K}, \mathcal{P}', T'_s)$ outputs *true*. Hence, the Correctness property. $\square$

– Eternal validity:If it happens that S_d is proven to exist at T_s and uses an algorithm trusted to be secure until at least T_s and $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T_{d_1}, T)$ outputs *true* such that $T_{d_1} \geq T_s$, then there exists a date $T_x \geq T_s$ which verifies the first statements and a set of n dates $T_{VO_1} \ldots T_{VO_n}$ which verify the second statements. We can use the same dates for any date $T_{d_2} \geq T_s$ such that $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T_{d_2}, T)$ outputs also *true*. $\square$

3.3 Discussions

Role of Bob. Bob is not involved in the semantics above. One may think that this is unfair towards Bob. Due to the Correctness property, if it happens that the Judge rules in favor of Alice although Bob is not the actual signer, then this means that Bob's private key has been compromised before the date T_s but the revocation information was not yet published at T_x. This can not happen in the considered model as we assumed instant revocation. However, in the real world, there is always an inevitable delay between the effective compromise and the publication of the revocation information [14]. Therefore, it is important to wait some period of time after the signing date before considering the signature valid. This period (called caution period) is the estimated time duration required to process and publish the last revocation information. The value of the caution period must be established in agreement between all actors.

Using a Caution Period *C*. To consider a caution period, we need to update the Correctness property: the validity of the signature must be established at the signing date shifted (in the future) by the caution period:

Property 3 (Correctness with caution period). Given $S_d = Sig(B, m, sa, sv)$, a caution period C, and $\mathsf{Dispute}(A, B, S_d, T_d)$; if $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T_d, T)$ outputs *true* then S_d is created at a date $T_s \leq T_d$ and is valid at $T_s + C$. $\circ$

We also need to update the RDR semantics. The first statement in Definition 5 needs to be updated to the following: there exists a date $T_x \geq T_s + C$ such that $\mathsf{SVF}(S_d, \mathcal{K}, \mathcal{O}, T_x) = (true, P)$. For the sake of simplicity, the rest of the document will use the semantics without caution period.

4 Calculus for Long-Term Signature Validation

This section defines a formal calculus modeling the resolution of dispute. It uses a simplified subset of the sequent logic syntax which provides a formal setting to reason about logical truth based on deduction using inference rules. The formulas are a set of statements indicating trust relationship. Axioms are a set of particular statements, called Judge's Initial View, accepted as trustworthy by the Judge prior to deriving any judgments. In contrast with classical sequent calculus, values assigned to formulas are not *true* and *false*, but *valid* and *invalid*. For instance, an *invalid* formula is not necessary *false* but indicates that the Judge does not trust the statement inferred from this formula.

4.1 Statements

The statements use date instants T ($now()$ will denote the current date) and date intervals (e.g. $I = [T_b, T_e]$). All validation objects considered below are signed objects. In general, a validation object may be denoted V or $V(sa)$ where sa is the signature algorithm used to sign V. We will consider two types of statements:

Validation Object Statements. A validation object statement indicates that the Judge holds a validation object:

- $Sig(X, m, sa, sv)$ denotes that the Judge holds a signature signed by X. The signature value sv is computed on m using the signature algorithm sa.
- $EECert(X, Y, I, sa)$ denotes that the Judge holds an end entity certificate for Y's public key signed by X using sa. The certificate is valid in I.
- $CACert(X, Y, I, sa)$ denotes that the Judge holds a CA's certificate for Y's public key signed by X using sa. The certificate is valid in I.
- $NRev(X, Y, I, sa)$ denotes that the Judge holds an evidence signed by X using sa and indicating that Y's key is not revoked in I.
- $TSP(X, h(m), T_{tsp}, sa)$ denotes that the Judge holds a time-stamp signed by X at T_{tsp} using the signature algorithm sa on the hash of m using the cryptographic hash function h.

Assertion Statements. An assertion statement indicates that the Judge believes that an assertion is trustworthy:

- $CAAut(X, I)$ denotes that the Judge believes that X is a CA and that X's public key is authentic in I.
- $Aut(X, I)$ denotes that the Judge believes that X is an entity which is not a CA and that X's public key is authentic in I.
- $POE(V, T_{poe})$ denotes that the Judge believes in the existence of the validation object V at T_{poe}.
- $POI(V(sa), T_{poe})$ denotes that the Judge believes in the existence of the validation object $V(sa)$ at T_{poe} and that $V(sa)$ has not been forged.
- $Algo(a, T_a)$ denotes that the Judge believes that the algorithm a is trusted to be secure until at least T_a.

4.2 Initial View Construction

The Initial View IV is the set of initial statements considered trustworthy.

Definition 6 (Initial View Construction). *Given the trust anchors $\mathcal{K} = \{Cert_1, \ldots, Cert_n\}$, the algorithm assertions $\mathcal{A} = \{(a_1, T_{a_1}), \ldots, (a_p, T_{a_p})\}$, the signed validation objects $\mathcal{O} = \{V_1, \ldots, V_k\}$ and the signature $S = Sig(B, m, sa, sv)$; the initial view $IV = IV(\mathcal{K}, \mathcal{A}, \mathcal{O}, S)$ is constructed as follows:*

1. *initialize IV with the empty set.*
2. *if S is cryptographically valid, add S to IV.*
3. *$\forall i \in [1, k]$, if the signature in V_i is cryptographically valid, add V_i to IV.*
4. *For each certificate $Cert_i$ in $\mathcal{K}$, add $CAAut(X_i, I_{X_i})$ to IV such that X_i is the subject of the certificate $Cert_i$ and I_{X_i} is its validity interval.*
5. *For each pair (a_i, T_{a_i}) in $\mathcal{A}$, add $Algo(a_i, T_{a_i})$ to IV.* ◦

4.3 Inference Rules

In our model, we will consider the following inference rules:

Authenticity of public keys. Informally, Y is authentic if and only if the Judge holds a certificate for Y signed by X believed to be authentic and owned by a CA. Besides, the Judge must hold a revocation signed by X indicating that Y is not revoked. Finally, the judge must hold a POI for any used object. If Y is a CA, the Judge derives $CAAut$, otherwise, he derives Aut. $\forall T \in I = I_1 \cap I_2 \cap I_3 = [T_1, T_2]$:

$$\frac{\begin{array}{c} CAAut(X, I_1) \\ POI(CACert(X, Y, I_2, sa_2), T) \qquad POI(NRev(X, Y, I_3, sa_3), T) \end{array}}{CAAut(Y, [T, T_2])} \tag{1}$$

$$\frac{\begin{array}{c} CAAut(X, I_1) \\ POI(EECert(X, Y, I_2, sa_2), T) \qquad POI(NRev(X, Y, I_3, sa_3), T) \end{array}}{Aut(Y, [T, T_2])} \tag{2}$$

Immediate proof of existence. Informally, if the Judge holds a validation object V, then he holds a proof of existence of this object at the current date.

$$\frac{V}{POE(V, now())} \tag{3}$$

Proof of existence based on time-stamps. In our model, this is the only way to derive a proof of existence of an object at a date in the past. $\forall T \in I_X$:

$$\frac{POI(TSP(X, h(V), T_{TSP}, sa_{tsp}), T) \qquad Aut(X, I_X)}{POE(h(V), T_{TSP})} \tag{4}$$

Indirect proof of existence

Theorem 1. *If $m = h(d)$ is proven to exist at T_1 where d is a message proven to exist at a date $T \geq T_1$ and h is a preimage-resistant hash function secure until at least a date $T_h \geq T$, then d is indirectly proven to exist at T_1.*

The proof is straightforward and is relative to the resistance to preimage of the cryptographic hash function h [15]. Indeed, this guarantees that, given m, until the date T_h it is hard to find a message d' such that $m = h(d')$. This means that d exists at T_1. This gives the following inference rule: $\forall T_1 \leq T \leq T_h$:

$$\frac{POE(h(V), T_1) \qquad POE(V, T) \qquad Algo(h, T_h)}{POE(V, T_1)} \tag{5}$$

Proof of integrity. Informally, this rule means that if a signed object exists at T and the signature algorithm is trusted until at least $T_{sa} \geq T$, then the object exists at T and has not been forged at T. $\forall T \leq T_{sa}$:

$$\frac{POE(V(sa), T) \qquad Algo(sa, T_{sa})}{POI(V(sa), T)} \tag{6}$$

Eternal proof of integrity. Informally, this rule means that if a signed object has not been forged at T_0, then the signed object has not been forged for any later date. $\forall T \geq T_0$:

$$\frac{POI(V(sa), T_0)}{POI(V(sa), T)} \tag{7}$$

4.4 Statement Validity

Definition 7. *Given an initial view IV, a statement G is said valid at T if and only if $G \in IV$ or G can be derived at T by using the inference rules.* ◦

Definition 8 (Derivation tree and derivation proof). *Given an initial view IV and a statement G valid at T, the derivations used to derive G from IV can be represented by a tree rooted by G, where the leaves are statements from IV and each non-leaf node is derived from its children using one inference rule. This tree is called the derivation tree of G at T. The set of leaf nodes is called derivation proof of G at T and is denoted $P(IV, G, T)$.* ◦

Note that there may be several derivation trees for the same G, IV, and T which may yield different derivation proofs. A derivation proof is only unique given a particular derivation tree. Intuitively, a proof $\mathcal{P}$ is a minimal subset of IV that can allow the derivation of G at T using the same tree (or possibly another tree).

5 Implementing the Resolution of Dispute Rule

This section uses the calculus defined above to implement the RDR semantics. First we define an additional assertion statement: $Res_{A,B}(Sig(B, m, sa, sv), T)$ denotes that the Judge believes in the validity of Alice's claims for the dispute $\mathsf{Dispute}(A, B, Sig(B, m, sa, sv), T)$. The RDR semantics can be written using the following inference rule: $\forall T \geq T_s$ such that $T_s \leq T_{B_2}$:

$$\frac{POI(Sig(B, m, sa, sv), T_s) \qquad Aut(B, [T_{B_1}, T_{B_2}])}{Res_{A,B}(Sig(B, m, sa, sv), T)} \tag{8}$$

Definition 9 (Resolution of Dispute Rule). *Given* $\mathcal{IVT}(T) = (\mathcal{K}, \mathcal{A})$*, the signature* $S_d = Sig(B, m, sa, sv)$*, the set of the validation objects* $\mathcal{O}$ *provided by Alice and the* $\mathsf{Dispute}(A, B, S_d, T_d)$*;* $\mathsf{RDR}(S_d, \mathcal{O}, \mathcal{IVT}(T), T_d, T)$ *outputs true if and only if the statement* $Res_{A,B}(S_d, T_d)$ *is valid at* T *with the initial view* $IV = IV(\mathcal{K}, \mathcal{A}, \mathcal{O}, S_d)$*. In addition, if the Judge rules in favor of Alice, then* $P(IV, Res_{A,B}(S_d, T_d), T)$ *is a judgment proof.* ◦

Proposition 2 (Correctness). *Given a dispute* $\mathsf{Dispute}(A, B, S_d, T_d)$*; the Judge rules in favor of Alice at* T *according to Definition 5 if and only if he rules in favor of Alice at* T *according to Definition 9.*

The proof is depicted in Appendix A. This proposition will allow us to define some properties of the judgment proof using the calculus.

5.1 Expiration of the Judgment Proof

Since derivations are deterministic, if the derivation of $Res_{A,B}(S_d, T_d)$ from IV yields the derivation proof $\mathcal{P}$ then $Res_{A,B}(S_d, T_d)$ can always be derived from $\mathcal{P}$ at T by building the same derivation tree. Therefore, replaying the RDR at the same judgment date using as input the judgment proof gives the same result. An important question is the following: if Alice comes later with this proof, will the Judge also rule in her favor? Each statement in the proof has an expiration date. For instance, certificate, revocation and authenticity statements have a validity interval from which we have the expiration date. An algorithm trust statement has also an expiration date which is the expiration date of the algorithm.

Definition 10 (Expiration date of a proof). *Given a judgment proof* $\mathcal{P}_p$ *obtained at a date* T_p*, let us denote* $E_1 \ldots E_n$ *the expiration dates extracted from the statements in* $\mathcal{P}$*. The expiration date of* $\mathcal{P}$ *is denoted* $E(\mathcal{P})$ *and is defined by the following date:*

$$E(\mathcal{P}) = min\{E_i, E_i > T_p\}_{1 \geq i \geq n}$$

◦

Proposition 3. *Given a proof* $\mathcal{P}_p$ *obtained at* T_p *for* $\mathsf{Dispute}(A, B, S_d, T_d)$*; if at* $T \in [T_P, E(\mathcal{P}_p)]$*, Alice provides* $\mathcal{O}$ *such that* $\mathcal{P}_p \subseteq IV(\mathcal{IVT}(T), \mathcal{O}, S_d)$*, then the Judge will rule in favor of Alice for* $\mathsf{Dispute}(A, B, S_d, T)$ *at* T*. In addition, if* $T_d < T_p$*, the Judge will rule in favor of Alice for* $\mathsf{Dispute}(A, B, S_d, T_d)$ *at* T*.*

The proof is depicted in Appendix B. Therefore, Alice has interest to archive the validation objects in $\mathcal{P}_p$. During $[T_p, E(\mathcal{P}_p)]$, favorable judgments are guaranteed to Alice provided she comes with these objects. Note that this works because we assumed that any update to $\mathcal{IVT}$ guarantees that any assertion statement derived from $\mathcal{IVT}(T_p)$ can be derived from any later $\mathcal{IVT}(T)$. If Alice needs favorable judgments beyond $E(\mathcal{P}_p)$, she needs to renew this proof. The proposition below is the foundation of renewal algorithms that will be showed in the next section.

Proposition 4. *Given a proof $\mathcal{P}_p$ obtained at T_p for* $\mathsf{Dispute}(A, B, S_d, T_d)$*; let us denote $\{V_1 \dots V_n\}$ the subset of validation objects in $\mathcal{P}_p$. If at T, Alice provides $\mathcal{O}$ such that $\mathcal{P}_p \subseteq IV(\mathcal{IVT}(T), \mathcal{O}, S_d)$ and that the Judge can derive $\{POE(V_1, T_{poe}) \dots POE(V_n, T_{poe})\}$ where $T_{poe} \in [T_p, E(\mathcal{P}_p)]$, then the Judge will rule in favor of Alice for the* $\mathsf{Dispute}(A, B, S_d, T_{poe})$ *at T. In addition, if $T_d < T_p$, the Judge will rule in favor of Alice for the* $\mathsf{Dispute}(A, B, S_d, T_d)$ *at T.*

This is a corollary of Proposition 3 and the proof is omitted.

Example. We consider the $\mathsf{Dispute}(A, B, S_d, now())$ where $S_d = Sig(B, m, sa, sv)$. The RDR is executed at $now()$ with the following: $\{S_d,\ EECert(X, B, I_b, sa),\ NRev(X, B, I_{rb}, sa)\}$. We will use $\mathcal{IVT}(now()) = (\{CAAut(X, I_x)\}, \{Algo(sa, T_{sa})\})$. We assume: $I_{rb} = [T_1, T_2]$ and $now() \in I_{rb} \subset I_b \subset I_x$ and T_{sa} is greater than the upper bound of I_x. First we build IV (assuming that all objects have cryptographically valid signatures): $\{Sig(B, m, sa, sv), EECert(X, B, I_b, sa), NRev(X, B, I_{rb}, sa), Algo(sa, T_{sa}),$ $CAAut(X, I_x, sa)\}$. From this IV, we can build the proof tree depicted in Figure 2. Leaf POIs are derived by successive application of Rules (3) and (6).

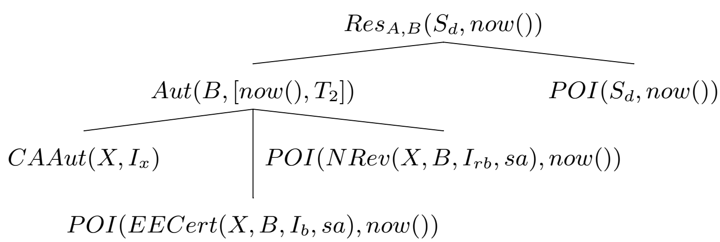

Fig. 2. Resolution proof tree

The judgment proof is: $\mathcal{P} = \{S_d, Aut(X, I_x), EECert(X, B, I_b, sa),$ $NRev(X, B, I_{rb}, sa), Algo(sa, T_{sa})\}$. This proof expires at $E(\mathcal{P}) = T_2$.

6 Using the Calculus in the Real World

One of the motivations of this work is to propose a calculus that can be used to model several frameworks [4, 5, 6] used in practice for long-term signature validation. Most of theses frameworks rely on the use of ERS [8]. In this section, we will model the use of ERS to renew the validity of a signature beyond the expiration date of a judgment proof. For this purpose, we will model the following operations: ERS creation, simple renewal and complex renewals.

ERS Creation. Prior to create an ERS, Alice should ensure that she has appropriate validation data to have a favorable judgment at the current date for $Dispute(A, B, S_d, now())$. With our calculus, this means that executing RDR at $now()$ outputs a proof $\mathcal{P}_0$. The objective of Alice is to continuously renew this proof to guarantee favorable judgments for the dispute $Dispute(A, B, S_d, T_0)$ where T_0 is as close as possible to the current date. For this purpose, Alice needs to create an ERS as soon as possible (before the date $E(\mathcal{P}_0)$).

ERS creation consists in requesting a time-stamp on the validation objects used to validate the signature. In our calculus, this means requesting time-stamps for the validation objects $\{V_1 \dots V_n\}$ in $\mathcal{P}_0$. Let us denote $ERS_0 = \{TSP_0 \dots TSP_n\}$ the obtained time-stamps using the authority X_0, the cryptographic function h_0 and the signature algorithm sa_0.

Now, how ERS_0 can allow to have a favorable judgment after $E(\mathcal{P}_0)$? At $T > E(\mathcal{P}_0)$, in addition to the validation objects $V_1 \dots V_n$, Alice needs to come with the set ERS_0 and the validation data that allow the Judge to derive $Aut(X_0, I_{X_0})$ at T. In this case, the Judge can derive at T, the statements $POE(V_i, T_0)$ for any $i \in [1, n]$ using the tree in Figure 3. Note that this requires that h_0 and sa_0 are trusted at T. Thus, he will rule in favor of Alice for $Dispute(A, B, S_d, T_0)$ (Proposition 4) with a judgment proof $\mathcal{P}_1$. During the interval $[T, E(\mathcal{P}_1)]$, $\mathcal{P}_1$ guarantees favor judgments for Alice for $Dispute(A, B, S_d, T_0)$. To go beyond $E(\mathcal{P}_1)$, Alice needs to continuously execute the renewal algorithms.

Simple Renewal. The simple renewal applies to the last time-stamps. i.e. the preceding time-stamps are time-stamped again together with complementary validation objects if needed. In our model, the simple renewal consists in requesting new time-stamps (ERS_1) for the time-stamps in ERS_0 and any validation object used to derive $Aut(X_0, I_{X_0})$. This renewal must occur at $T_1 < E(\mathcal{P}_1)$.

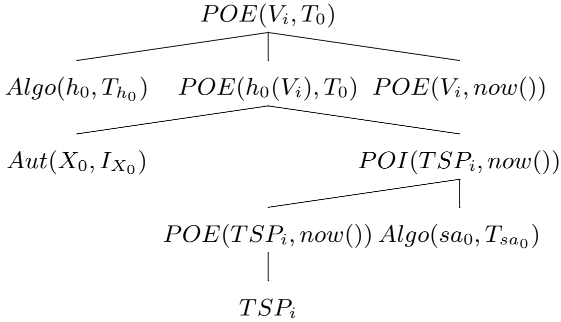

Fig. 3. Derivation of $POE(V_i, T_0)$ after ERS creation ($T = now()$)

Now, how ERS_1 allows the Judge to rule in favor of Alice after $E(\mathcal{P}_1)$? Let denote X_1, sa_1 and h_1 respectively the TSA, the signature algorithm and the cryptographic hash function used at T_1. To have favorable judgments at a date $T > E(\mathcal{P}_1)$, Alice needs to provide $\{V_1 \dots V_n\}$, ERS_0, ERS_1 and the validation objects that allow to derive $Aut(X_1, I_{X_1})$ at T. From ERS_1 and the derivation proof of $Aut(X_1, I_{X_1})$, the Judge can derive $POE(TSP(X_0, h_0(V_i), T_0, sa_0), T_1)$ for any $i \in [1, n]$ in the same way as in Figure 3. Also, since ERS_1 includes time-stamps on the validation objects used to derive the authenticity of X_0, this allows to derive $Aut(X_0, I_{X_0})$ at T. From that, the Judge can derive $POE(V_i, T_0)$ for any $i \in [1, n]$ and therefore he will rule in favor of Alice (Proposition 4) using a proof $\mathcal{P}_2$. Note again that this requires that h_0, h_1, and sa_1 are trusted at T. $\mathcal{P}_2$ guarantees favorable judgments to Alice until the date $E(\mathcal{P}_2)$. Before this date, Alice needs to run again the simple renewal algorithm.

Recursive simple renewals may allow Alice to postpone the expiration of the judgment proof. Naturally, Alice should use state of art algorithms and authorities at each renewal to be able to validate the last time-stamps. However, recursive simple renewals can not allow "eternal" validity. Indeed, for all the proofs $\mathcal{P}_i$ obtained after k simple renewals, $Algo(h_j, T_{h_j}) \in \mathcal{P}_i$, $\forall j \leq i$ (where h_j is the hash function used for the jth simple renewal). Thus, for all the proofs $\mathcal{P}_i$ we have $E(\mathcal{P}_i) < min\{T_{h_j}, j \leq i\}$. We can fairly assume that, at each renewal, Alice uses an up to date hash function such that $T_{h_{j+1}} \geq T_{h_j}$ $\forall j$. Therefore, T_{h_0} represents the maximum date for which Alice can have favorable judgments by recursive simple renewals. To have favorable judgments beyond this date, Alice needs to execute the complex renewal.

Complex Renewal. This renewal is performed when the probability of hash function compromise is foreseen. Not only time-stamps are re-timestamped but also data objects protected by the time-stamps. In our model, the need to execute this renewal occurs when the expiration date of a judgment proof $\mathcal{P}_{k+1}$ obtained after the kth simple renewal becomes close to the lowest expiration date (say T_{h_0}) of the hash functions used in the previous simple renewals. The renewal must be executed at a date $T_{k+1} < E(\mathcal{P}_{k+1})$. It consists in creating a new ERS on: $\{V_1, \dots, V_n\}$, ERS_k and the objects used at T_{k+1} to derive $Aut(X_k, I_{X_k})$ where X_k is the TSA used for the kth simple renewal.

Now, how the complex renewal allows the Judge to rule in favor of Alice at a date $T > T_{h_0}$? Let us denote h_{k+1}, X_{k+1} and ERS_{k+1} respectively the hash function, the TSA and the set of obtained time-stamps. At T, Alice needs to come with the validation objects in $\mathcal{P}_{k+1}$, ERS_{k+1} and the objects that allow to derive $Aut(X_{k+1}, I_{X_{k+1}})$. Since the complex renewal is also a simple renewal, it allows to derive $POE(TSP(X_0, h_0(V_i), T_0, sa_0), T_1)$ for any i in $[1, n]$ and $Aut(X_0, I_{X_0})$ at T. In addition, ERS_{k+1} includes a time-stamp on the object V_i which allows to derive $POE(V_i, T_{k+1})$ for any i in $[1, n]$. From that, $POE(V_i, T_0)$ can be derived at T for any $i \in [1, n]$. Thus, the Judge will rule in favor of Alice at T (Proposition 4) with a proof $\mathcal{P}_{k+2}$. If Alice needs favorable judgments after $E(\mathcal{P}_{k+2})$, she needs to continue to execute the simple renewal. Again, once the expiration date of a proof becomes close to the expiration of the hash functions used with a previous ERS, the complex renewal should be performed.

7 Conclusion and Future Work

We presented a security model for digital signature validation with the notion of dispute about the validity of a signature. The first contribution of this paper is the definition of the semantics of a resolution of dispute rule. The second contribution is a new calculus for reasoning about the validation of digital signatures at a particular date which may be in the past. The usefulness of our calculus is demonstrated through modeling ERS, one of the main techniques used in practice. The calculus allowed to model at which date the renewal becomes necessary, which data must be archived and how the renewal allows to have a favorable judgment after the expiration of the last proof.

One of the main components of the model is $\mathcal{IVT}$. The hypothesis that all actors trust a common set of trust anchors and algorithms seems very strong. If we can establish such agreement in a non-repudiable way, one may ask why this same way is not used to implement the resolution of dispute. On the other hand, this agreement requires that Bob agrees on the content of $\mathcal{IVT}$ at the signing date. We may assume that this $\mathcal{IVT}$ is part of the signed message which proves Bob's commitment. However, because we need to continuously update $\mathcal{IVT}$, we need Bob's commitment for each new $\mathcal{IVT}$. Bob's collaboration is then necessary for long-term validation which may be refused by Bob. In real world, this issue is resolved by the use of a third trusted party. The origin of this trust is usually legal frameworks [3, 16, 17] to which all other actors refer in case of dispute.

An interesting future work is the definition of other semantics for RDR. Ideally, in the scope of long-term validation, completely honest behaviors should not be an assumption. A potential interesting work is to capture other PKI models with potential malicious actors. This suggests the extension of the calculus to a model based on confidence valuation similar to the probabilistic model proposed by Maurer [9]. This will allow to define an evaluation metric for the judgment proofs based on the confidence levels of the statements composing the proofs. The Judge would select the proof with the highest confidence level and rule in favor of Alice provided this level goes beyond a determined threshold.

Acknowledgment. The authors would like to thank the anonymous reviewers for their valuable comments.

References

1. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Public key infrastructure: Certificate and CRL profile, IETF, Tech. Rep. RFC 5280 (2008)
2. Aarnes, A.: Public key certificate revocation schemes, Ph.D. dissertation, Norwegian University of Science (2000)
3. ETSI, Algorithms and parameters for secure electronic signatures; part 1: Hash functions and asymmetric algorithms, Tech. Rep. ETSI TS 102 176-1 V2.0.0 (2007)

4. Jerman Blaič, A., Klobučar, T., Jerman, B.D.: Long-term trusted preservation service using service interaction protocol and evidence records. Comput. Stand. Interfaces 29 (2007)
5. Huhnlein, D., Korte, U., Langer, L., Wiesmaier, A.: A comprehensive reference architecture for trustworthy long-term archiving of sensitive data. In: Third International Conference on New Technologies, Mobility and Security (2009)
6. Troncoso, C., De Cock, D., Preneel, B.: Improving secure long-term archival of digitally signed documents. In: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, pp. 27–36. ACM, New York (2008)
7. ETSI, CMS Advanced Electronic Signatures, Tech. Rep. ETSI TS 101 733 (2008)
8. Gondrom, T., Brandner, R., Pordesch, U.: Evidence Record Syntax (ERS), Tech. Rep. RFC4998 (2007)
9. Maurer, U.M.: Modelling a public-key infrastructure. In: Martella, G., Kurth, H., Montolivo, E., Hwang, J. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 325–350. Springer, Heidelberg (1996)
10. Marchesini, J., Smith, S.: Modeling public key infrastructures in the real world. In: Chadwick, D., Zhao, G. (eds.) EuroPKI 2005. LNCS, vol. 3545, pp. 118–134. Springer, Heidelberg (2005)
11. Bicakci, K., Crispo, B., Tanenbaum, A.S.: How to incorporate revocation status information into the trust metrics for public-key certification. In: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 1594–1598. ACM, New York (2005)
12. Baier, H., Karatsiolis, V.: Validity models of electronic signatures and their enforcement in practice. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 255–270. Springer, Heidelberg (2010)
13. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17, 281–308 (1988)
14. Walleck, D., Li, Y., Xu, S.: Empirical analysis of certificate revocation lists. In: Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, pp. 159–174. Springer, Heidelberg (2008)
15. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. Discrete Mathematics and Its Applications. CRC Press, Boca Raton (1997)
16. FNISA, Annexe B1 - règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographique, French Network and Information Security Agency, Tech. Rep. Version 1.20 du (janvier 26, 2010)
17. ETSI, Provision of harmonized trust service provider status information, Tech. Rep. ETSI TS 102 231 V3.1.2 (2009)

A Proof of Proposition 2

Let us first assume that the Judge rules in favor of Alice according to Definition 5. We want to prove that with the same inputs, the Judge will rule in favor of Alice according to Definition 9. Definition 5 means that there exists a date $T_s \leq T_d$ such that the three statements of the definition are satisfied at T.

1. The first statement means that there exists a date $T_x \geq T_s$ such that $\mathsf{SVF}(S_d, \mathcal{K}, \mathcal{O}, T_x) = (true, P)$, where P is a sequence of certificates and revocation objects. Let us denote this sequence by:

$$P = \begin{pmatrix} CACert(X_0, X_1, I_1, sa_1), NRev(X_0, X_1, I'_1, sa'_1) \\ \cdots \\ CACert(X_{n-2}, X_{n-1}, I_{n-1}, sa_{n-1}), NRev(X_{n-2}, X_{n-1}, I'_{n-1}, sa'_{n-1}) \\ EECert(X_{n-1}, X_n, I_n, sa_n), NRev(X_{n-1}, X_n, I'_n, sa'_n) \end{pmatrix}$$

From the definition of SVF, we have the following:

- X_0 is a trust anchor from $\mathcal{K}$. Therefore, there exits an interval I_0 such that $T_x \in I_0$ and $CAAut(X_0, I_0) \in IV$.
- the last certificate is issued to Bob ($X_n = B$).
- the signatures in S_d, the certificates and revocation objects in P are cryptographically valid. Therefore, $\{S_d, P\} \subset IV$.
- all certificates and revocation objects in P are valid at T_x. Therefore, $\forall i \in [1, n], T_x$ is included in I_i and I'_i.

2. The second statement means that any validation object in P exists at a date earlier than T_x and that at this date the signature algorithm used to protect the object is trusted to be secure. This gives us the following:
   - $\forall i \in [0, n-2]$, there exists a date $T_i \leq T_x$ (respectively a date $T'_i \leq T_x$) such that $POE(CACert(X_i, X_{i+1}, I_{i+1}, sa_{i+1}), T_i)$ (respectively $POE(NRev(X_i, X_{i+1}, I'_{i+1}, sa'_{i+1}), T'_i))$ can be derived at T.
   - There exists a date $T_n \leq T_x$ (respectively a date $T'_n \leq T_x$) such that $POE(EECert(X_{n-1}, X_n, I_n, sa_n), T_n)$ (respectively $POE(NRev(X_{n-1}, X_n, I'_n, sa'_n), T'_n))$ can be derived at T.
   - $\forall i \in [1, n]$, there exists a date $T_{sa_i} \geq T_i$ such that (sa_i, T_{sa_i}) is in $\mathcal{A}$ and then $Algo(sa_i, T_{sa_i}) \in IV$.

   With theses POE and $Algo$ statements, the Judge can derive the following statements using Rule (6) to have a POI with a date earlier than T_x then Rule (7) to have a POI with T_x:
   - $\forall i \in [0, n-2]$: $POI(CACert(X_i, X_{i+1}, I_{i+1}, sa_{i+1}), T_x)$ and $POI(NRev(X_i, X_{i+1}, I'_{i+1}, sa'_{i+1}), T_x)$
   - $POI(EECert(X_{n-1}, X_n, I_n, sa_n), T_x)$
   - $POI(NRev(X_{n-1}, X_n, I'_n, sa'_n), T_x)$

   Now, we will show that, with the previous statements, the Judge can derive $Aut(B, I_B)$ such that I_B is an interval that contains the date T_x. First, let us prove by induction the following: $\forall i \in [1, n-1]$, there exists an interval I''_i such that $T_x \in I''_i$ and the statement $CAAut(X_i, I''_i)$ is valid at T:
   - for $i = 1$, $CAAut(X_1, I''_1)$ can be derived with the following such that $T_x \in I''_1$ (Rule (1)):

$$\frac{CAAut(X_0, I_0) \qquad POI(CACert(X_0, X_1, I_1, sa_1), T_x) \qquad POI(NRev(X_0, X_1, I'_1, sa'_1), T_x)}{CAAut(X_1, I''_1)}$$

     This works because we showed that $CAAut(X_0, I_0) \in IV$, and T_x is included in all the intervals I_i and I'_i.

– Now we assume that there exists an interval I''_i such that $T_x \in I''_i$ and $CAAut(X_i, I''_i)$ is valid at T. The application of Rule (1) gives $CAAut(X_{i+1}, I''_{i+1})$ such that $T_x \in I''_{i+1}$:

$$\frac{CAAut(X_i, I''_i) \qquad POI(CACert(X_i, X_{i+1}, I_{i+1}, sa_{i+1}), T_x) \quad POI(NRev(X_i, X_{i+1}, I'_{i+1}, sa'_{i+1}), T_x)}{CAAut(X_{i+1}, I''_{i+1})}$$

Therefore, there exists an interval I''_{n-1} such that $T_x \in I''_{n-1}$ and $CAAut(X_{n-1}, I''_{n-1})$ is valid at T. Now, we use Rule (2) to derive $\boldsymbol{Aut(B, I_B)}$ such that $T_x \in I_B$:

$$\frac{CAAut(X_{n-1}, I''_{n-1}) \qquad POI(EECert(X_{n-1}, X_n, I_n, sa_n), T_x) \quad POI(NRev(X_{n-1}, X_n, I'_n, sa'_n), T_x)}{Aut(X_n = B, I''_n = I_B)}$$

3. The last statement in Definition 5 gives the following: $POE(S_d, T_s)$ is valid at T and there exists a date $T_{sa} \geq T_s$ such that (sa, T_{sa}) is in $\mathcal{A}$ and then in $Algo(sa, T_{sa}) \in IV$. Therefore, Rule (6) allows to derive $\boldsymbol{POI(S_d, T_s)}$.

In conclusion, we showed that the Judge can derive the statements $POI(S_d, T_s)$ and $Aut(B, I_B = [T_{B_1}, T_{B_2}])$ at T where $T_d \geq T_s$ and $T_s \leq T_{B_2}$ (because $T_s \leq T_x$ and $T_x \in I_B$). Then, Rule (8) allows to derive $Res_{A,B}(S_d, T_d)$ at T. Therefore, the Judge will rule in favor of Alice according to Definition 9. $\square$

The other part of the proof is similar and is omitted.

B Proof of Proposition 3

We want to prove that Alice will have a favorable judgment at any date $T \in [T_P, E(\mathcal{P}_p)]$ provided she comes with the objects $\mathcal{O}$ such that $\mathcal{P}_p \subseteq IV = IV(\mathcal{IVT}(T), \mathcal{O}, S_d)$. If the Judge rules in favor of Alice at T_p, this means that $Res_{A,B}(S_d, T_d)$ can be derived at T_p from $\mathcal{P}_p$. Then, there exist an interval $I_B = [T_{B_1}, T_{B_2}]$ and a date T_s verifying $T_s \leq T_{B_2}$ and $T_s \leq T_d$ such that the statements $POI(S_d, T_s)$ and $Aut(B, I_B)$ can be derived at T_p. $Aut(B, I_B)$ means that the Judge believes at T_p that B's key is authentic within the interval I_B. Let us extract the expiration dates from the statements in $\mathcal{P}_p$ and order them chronologically in the following dates $E_1 ... E_k$. With T_p, this gives us $k + 2$ intervals. Within each interval, the validity status of all statements in $\mathcal{P}_p$ are constant. Therefore, the validity status of any Aut and $CAAut$ is constant within each interval. We consider the date $E_j = E(\mathcal{P}) = min\{E_i, E_i > T_p\}_{1 \geq i \geq k}$. Since $Aut(B, I_B)$ can be derived at T_p from $\mathcal{P}_p$, then $Aut(B, I_B)$ can be derived at any date $T \in [T_p, E(\mathcal{P}_p)]$. In addition, since S_d and $Algo(sa, T_{sa})$ are in $\mathcal{P}_p$, then Rule (3) gives $POE(S_d, T)$ and Rule (6) gives $POI(S_d, T)$ $(T \leq E(\mathcal{P}_p) \leq T_{sa})$. Finally, Rule (8) gives $Res_{A,B}(S_d, T)$ $(T \leq E(\mathcal{P}_p) \leq T_{B_2})$. Therefore, the Judge will rule in favor of Alice at T for the dispute $\mathsf{Dispute}(A, B, S_d, T)$.

- $POI(S_d, T_s)$
  - $POE(S_d, T_s)$
    - $Algo(h_1, T_{h_1})$
    - $POE(h(S_d), T_s)$
      - $POI(TSP_1, T_{TSP_1})$
      - $Aut(X_{TSP_1}, I_{TSP_1})$
    - $POE(S_d, T'_s)$
  - $Algo(sa, T_{sa})$

Fig. 4. Derivation of $POI(S_d, T_s)$ at T_p ($T_s \leq T'_s \leq T_{h_1}$ and $T_{TSP_1} \in I_{TSP_1}$)

- $POI(TSP_1, T_{TSP_1})$
  - $POE(TSP_1, T_{TSP_1})$
    - $Algo(h_2, T_{h_2})$
    - $POE(h_2(TSP_1), T_{TSP_1})$
      - $POI(TSP_2, T_{TSP_2})$
      - $Aut(X_{TSP_2}, I_{TSP_2})$
    - $POE(TSP_1, T'_{TSP_1})$
  - $Algo(sa_1, T_{sa_1})$

Fig. 5. Derivation of $POI(TSP_1, T_{TSP_1})$ at T_p

Now, we consider the case $T_d < T_p$. This means that at T_p, the Judge derived a POI for S_d with a date in the past ($T_s \leq T_d < T_p$). The only way is using Rule (4) with a time-stamp $TSP_1 = TSP(X_{TSP_1}, h_1(S_d), T_s, sa_1)$ as in Figure 4. First, using the same reasoning as above for $Aut(B, I_B)$, $Aut(X_{TSP_1}, I_{TSP_1})$ can be derived at any date $T \in [T_p, E(\mathcal{P}_p)]$. Second, we have two cases:

- If $T_{TSP_1} = T_p$, then $[T_p, E(\mathcal{P}_p)] \subset I_{TSP_1}$ and therefore $T \in I_{TSP_1}$. Thus, $POI(S_d, T_s)$ can be derived at T with the derivation tree in Figure 4 using T instead of T_{TSP_1} and T'_s (recall that $T \leq E(\mathcal{P}_p) \leq T_{h_1}$).
- If $T_{TSP_1} < T_p$, then the Judge derived at T_p a POI statement for TSP_1 with a date $T_{TSP_1} < T_p$. The only way is using Rule (4) with $TSP_2 = TSP(X_{TSP_2}, h_2(TSP_1), T_{TSP_1}, sa_2)$ (Figure 5). We have again two cases:
  - If $T_{TSP_2} = T_p$, then $POI(TSP_1, T_{TSP_1})$ can be derived at T using $T_{TSP_2} = T'_{TSP_1} = T$ and thus, $POI(S_d, T_s)$ can be derived at T.
  - If $T_{TSP_2} < T_p$, then $POI(TSP_2, T_{TSP_2})$ was derived at T_p using Rule 4 with a time-stamp $TSP_3 = TSP(X_{TSA_3}, h_3(TSP_2), T_{TSP_2}, sa_3)$.

  Recursively, we end with a time-stamp TSP_n such that $T_{TSP_n} = T_p$ and then $POI(TSP_{n-1}, T), \ldots, POI(TSP_1, T)$ can be derived at T which allows to derive $POI(S_d, T_s)$ at T.

Therefore, in both cases $POI(S_d, T_s)$ can be derived at T. This allows the Judge to derive $Res_{A,B}(S_d, T_d)$ at T and then to rule in favor of Alice at T for the dispute $\mathsf{Dispute}(A, B, S_d, T_d)$. $\square$