New Approaches to Digital Evidence

UELI MAURER, FELLOW, IEEE

Invited Paper

Digital evidence, such as digital signatures, is of crucial importance in the emerging digitally operating economy because it is easy to transmit, archive, search, and verify. Nevertheless, the initial promises of the usefulness of digital signatures were too optimistic. This calls for a systematic treatment of digital evidence. The goal of this paper is to provide a foundation for reasoning about digital evidence systems and legislation, thereby identifying the roles and limitations of digital evidence, in the apparently simple scenario where it should prove that an entity A agreed to a digital contract d.

Our approach is in sharp contrast to the current general views documented in the technical literature and in digital signature legislation. We propose an entirely new view of the concepts of certification, time stamping, revocation, and other trusted services, potentially leading to new and more sound business models for trusted services. Some of the perhaps provocative implications of our view are that certificates are generally irrelevant as evidence in a dispute, that it is generally irrelevant when a signature was generated, that a commitment to be liable for digital evidence cannot meaningfully be revoked, and that there is no need for mutually trusted authorities like certification authorities. We also propose a new type of digital evidence called digital declarations, based on a digital recording of a willful act indicating agreement to a document or contract.

Keywords*—*Digital declarations, digital evidence, digital signatures, expiration, nonrepudiation, public-key certificate, public-key infrastructure (PKI), revalidation, revocation, time stamping.

I. INTRODUCTION

Perhaps the main paradigm shift of the emerging information society is that digital information, such as software, digital multimedia content, or digital signatures, is becoming a key ingredient and resource of business and government processes and, more generally, of many activities in the society at large. The full consequences of this paradigm shift seem far from well understood and remain to be seen, but without doubt they will be far reaching.

Information differs radically from conventional resources, and it is not even clear what it means to possess, control, protect, archive, or let alone sell information. Digital rights management (DRM), including technical, legal, social, and business aspects, should provide a basis for controlling and protecting the new resource information, i.e., the digital assets of people, companies, and government organizations.

The types of digital assets receiving most attention today are software and digital content such as music, but in a broad view and definition of the term DRM, it includes all types of digital assets. An important new type of digital assets is the use of digital information as evidence. One can distinguish at least two types of such evidence. The first type is incriminating data (e.g., illegally copied software or a watermarked image) found under particular circumstances (e.g., on some user's computer hard disk).

This paper is concerned with the second type of digital evidence, actually generated for the purpose of being used as evidence, and whose value is independent of where and how it is stored. A main problem is to define the semantics of such evidence, which is also a DRM issue in the broad sense.

A. Digital Signatures: Promises and Obstacles

In a digital signature scheme, a user A has a secret key (or private key), which she keeps secret, and the corresponding public key, which is made publicly available. The digital signature for a digital contract can be generated only when given the secret key, but it can be verified by anybody using the public key. Hence, it can be interpreted as a proof that A agreed to the contract.

In view of the continuing automation and digitization of many business processes, the transmission, storage, and verification of physical evidence, like signed contracts, presents a major problem. In contrast to physical evidence, digital signatures are easy to transmit, archive, search, and verify. Moreover, digital signatures are generally unambiguous because their verification corresponds simply to the evaluation of a well-defined mathematical function, the signature verification predicate relative to a given public key. For these reasons, digital signatures promise to provide an elegant solution to the nonrepudiation problem in the digitally operating economy.

Manuscript received September 25, 2003; revised December 15, 2003.
The author is with the Department of Computer Science, ETH Zurich, Zurich CH-8092, Switzerland (e-mail: maurer@inf.ethz.ch).
Digital Object Identifier 10.1109/JPROC.2004.827358

Furthermore, due to the conjectured security of the underlying cryptographic mechanisms, digital signatures also promise substantially higher security compared to conventional signatures, and, hence, fewer disputes and simpler dispute resolution.

Despite the promises, in the context of nonrepudiation services digital signatures are currently used only in isolated applications.[1] We are still far from an internationally operational framework and infrastructure. Some of the obstacles are the lack of internationally applicable laws, the lack of standardization, the lack of viable business models for fostering the creation of a global public-key infrastructure (PKI), problems with the integration into business processes, and, last but not least, the abstractness and complexity of the subject matter, resulting in slow user acceptance.[2]

An even more fundamental problem is that the meaning of digital evidence is not well understood and often overestimated. The intrinsic problem that a digital signature is not linked to any event in the real world has no solution. It is inherently impossible to determine when, where, how, and by whom a digital signature was generated, even when secure hardware, biometric identification, and time stamping are used.

B. Contracts and Evidence for Nonrepudiation

A basic act in business and other contexts is to enter a formal agreement, often called a *contract*, between two (or more) entities. Such an agreement requires the clear mutual understanding of all relevant parameters, in particular the terms and conditions. A contract is valid only if both parties formally entered it. It is generally understood that a contract has been entered by a user or entity only if he or she (or an authorized representative) performed a well-defined conscious and willful act, for instance, by shaking hands and/or by signing a paper document, or by activating the generation of a digital signature on his smart card.

In order to prepare for a possible future dispute, each party to a contract wants to keep sufficient evidence for the claim that the other party agreed to the contract. This is a symmetric goal, although the evidence collected by each party may be different. Here we consider only one side of the symmetric problem:[3] how can an entity B obtain sufficient evidence that party A entered the contract? What sufficient means must be defined by the legal system.

One can distinguish at least three types of evidence, a combination of which may be used in a concrete setting:

- **physical evidence** (e.g., a signed paper document);
- **statements by witnesses**;
- **digital evidence** (digital signatures, time stamps, etc.).

Physical evidence and witnesses seem to contradict the new paradigm of digitized business processes. A widespread expectation for digital signatures is that they allow to avoid the need for physical evidence and witnesses.

C. Two Types of Digital Evidence

By *digital evidence* we mean a bitstring which can serve as evidence in a certain context, and which is generated for this purpose. One can distinguish two types of digital evidence.

- **Digital recordings**. A digital recording is any type of projection of physical reality, e.g., a digital image, video, or sound recording. An example is a digital image showing a person signing a contract. Such a digital recording is evaluated by converting it back to a physical form (e.g., an image on a computer screen) and then having it interpreted by human beings.
- **Digital evidence strings**. A digital evidence string (e.g., a digital signature) is verified by evaluating a well-defined (and unambiguous) mathematical function (e.g., the signature verification function relative to some public key). More generally, a digital evidence string can consist of several components, including several digital signatures, certificates, time stamps, etc. To forge a digital evidence string means to compute a bitstring which passes the verification function.

Digital evidence strings may perhaps appear to be more useful than digital recordings because they can be checked automatically and are, therefore, unambiguous. However, the main usefulness of digital recordings (see Section VI), even if they are easy or only moderately difficult to forge, is that they have a human-understandable interpretation as a physical reality and can, hence, meaningfully be confirmed or denied by a person, for instance, under oath.

D. The Context of This Paper

Digital evidence can be used in many different contexts, ranging from business disputes and e-government applications to criminal investigations. In this paper, we restrict ourselves to the following apparently simple and precise question.

> Which evidence is required to prove (e.g., in court) that an entity (say A) is liable for a digital document (or contract) d?[4]

Some examples of documents d are an online banking transaction, an online order, a software development contract (entered online), a testimony of some sort, or any other document that implies some type of liability, typically an obligation to pay a certain amount.

When a digital signature on document d relative to public key p_{A} is presented (e.g., in court), there are several objections that A might have.

1) p_{A} is not my public key.
2) I did not sign d (even though p_{A} is my valid public key).

[1]But see, for example, the Finread [10] project as a good example of an initiative for making digital signature technology fly in the financial sector.

[2]Technological problems (e.g., integrating PKI-technology mobile inyo devices), if any, seem to be only temporal.

[3]We leave out of consideration the theoretically interesting, but in many practical settings not very relevant, fair-exchange problem: namely, that the evidence should be exchanged simultaneously, with no party gaining a temporary advantage.

[4]An alternative formulation of the question is: Which evidence can A request to be presented (e.g., in court) before being declared liable?

3) The signature was generated after I had revoked p_A (or after p_A has expired).
4) I am liable for p_A, but only for transaction values much below that relevant in contract d.

The legal system must define precisely how objections by A are handled, i.e., which further evidence is required to conclude A's liability for d, since an entity B entering a contract with A wants to collect all the evidence required by the legal system.

E. Requirements for Evidence Systems and Legislation

In this section we briefly discuss some basic requirements for contract signing systems and the supporting legal framework.

- **Practicality**. The procedures must be practical and efficient. For example, a basic requirement in e-commerce is that there be no need for the parties to meet physically.
- **Unambiguity**. The resulting evidence should be unambiguous and allow for the clear and efficient resolution of possible disputes.
- **Security**. If a user has not agreed to d, the risk that convincing evidence for this claim is produced is negligible.
- **Low cost** of infrastructure, technology, and processes.
- **Low trust requirements**. The need for trusted entities should be as low as possible. In particular, the need for *commonly* trusted authorities should be avoided if possible. The required level of trust should be minimal.
- **Precise and simple legislation**. Legislation should be unambiguous and simple.
- **Smooth integration** into existing technical and legal infrastructure, minimizing the necessary changes.
- **Wide usability and acceptance**. The system should be easy to use, even for only moderately educated people, if the mass market is addressed.

These requirements are inherently conflicting, and the goal of designing appropriate systems, procedures, and legislation must be to find a reasonable balance and tradeoff between the conflicting requirements.

Traditional business practices are fairly pragmatic. For example, conventional handwritten signatures are a very simple and pragmatic mechanism; they work well in practice, even though signatures offer only a moderate level of security against forgery. We discuss conventional handwritten signatures in Section VI-C. Similarly, the business model for credit cards still works despite the very low security and the substantial level of fraud. It can be expected that the best tradeoff for the digital evidence problem will also be fairly pragmatic, but this does not free us from understanding the meaning of evidence.

F. Ambiguous Versus Unambiguous Aspects of the Legal System

In order for life and business processes to be reasonably predictable, which is an important achievement of civilization, the society as a whole agrees to rely on a legal system that defines the general rules under which the society operates. In particular, the legal system defines under which circumstances a person or other entity must bear certain financial, physical (e.g., go to jail), or other consequences.

Here we understand the *legal system* as the abstraction of the set of rules used to make decisions. For our purpose, it does not matter whether these rules are thought to be fixed by legislation or established by juridical practice in interpreting the laws. In order to be able to formulate such rules and compare different types of legal systems, we need to introduce an appropriate level of formality. Since real legislation and juridical practice are always to some extent ambiguous, we need to clearly separate the unambiguous from the ambiguous issues. Our goal is then to reason precisely about the unambiguous issues, i.e., those issues for which the legal system must provide a clear understanding (rather than leaving it open for a judge's interpretation). In our context, four examples of clearly unambiguous issues are the following.

- Can A request to see a proof that she actually committed to the public key, or is a certificate by an accredited certification authority (CA) sufficient to conclude that she is liable for the public key?
- Given that A is found to be liable for the public key, can she request to see more evidence than her digital signature on d? If the answer is yes, what is this extra evidence she can request to be presented?
- Can A request that a time stamp on the signature be presented, "proving" that the signature was issued before the expiration of the public key?
- What does it mean to revoke a public key, i.e., which evidence must A present to prove that her public key is revoked?

The ambiguous aspects we will eliminate from our considerations are the following.

- The interpretation of the meaning of d, except possibly for certain well-defined parameters specified in d.[5]
- For a precise description e of a piece of physical (or other) evidence, the question whether the presented evidence matches e.

As an example of the latter, the law might require as evidence a paper document signed by A on which she declares liability for her public key p_A. This is a (sufficiently) precise description, but what is ambiguous (and left out of consideration in this paper) is the question whether a particular piece of paper actually meets the description, including whether the signature is actually A's.

G. Limitations of Scope

The legal system is discussed in this paper only at an abstract and generic level, without referring to the various national approaches and their differences. Current legislation is not our primary concern. We expect future legislation to adapt to new research results and business practices, including perhaps those proposed here.

Although we will propose to reason formally about the legal system, we are by no means advocating that legislation and dispute resolution procedures should be formalized

[5]For example, when a purchase price z is specified in d and liability is limited to a certain threshold t, then it is relevant whether $z \leq t$.

much more rigidly than today, except for certain aspects like whether certain pieces of evidence should or should not be considered by a judge.

Throughout the paper, one should keep in mind that the main purpose of collecting and storing evidence is not to resolve disputes, but to avoid them in the first place by deterring misbehavior. Actual disputes are the rare exception and the technical and legal procedures must, therefore, be pragmatic, simple, and as lightweight as possible, but sufficiently well defined to enable actual dispute resolution if needed.

We assume some limited familiarity with the basic cryptographic concepts and with basic security technologies. The interested reader is referred to [6] and [9] for introductory treatments of cryptography.

One point of the paper is to discuss the role and interpretation of public-key certificates and PKIs. It is important to mention that there are at least three entirely different types of public keys, namely signature, encryption, and authentication public keys (see Section II-F). The terms public-key certificate and PKI apply to all these types, although the semantics of certificates (and other information relevant in a PKI) are completely different for each type, the common property being that a certificate binds (in some sense) a public key to an identity. This paper is concerned exclusively with signature public keys used in the context of digital evidence.

To contrast our ideas with the state of the art, we describe the current views on digital evidence. Clearly, our presentation cannot capture all considerations that researchers and legislators have made, let alone give fair credit to all of them. We do not even attempt to review the entire literature on the subject, but we refer to the Web pages [11] and [12] as entry points into the PKI and certificate-related literature and into the digital signature legislation literature, respectively. However, we believe that our presentation nevertheless summarizes these views quite adequately to substantiate the claim that the new views are radically different. Problems with digital evidence are also discussed in [7].

H. Outline of the Paper

In Section II we briefly review the basic properties and functionality of several fundamental cryptographic concepts. In Section III we discuss current views and approaches, established in the literature and legislations, on how digital signatures, certificates, time stamps, etc., can be used as evidence. Section IV discusses the fundamental dilemma in digital evidence legislation. In Section V we present our new views on digital evidence, and in Section VI we propose so-called digital declarations as a pragmatic solution to the dilemma.

II. Preliminaries

A. Bitstrings

A *bitstring* (or simply string)[6] is a finite sequence of bits. For example, $a = 1001011$ is a bitstring of length $|a| = 7$. Let ϵ denote the empty string of length 0. Moreover, $\{0,1\}^n$ denotes the set of bitstrings of length n, and $\{0,1\}^*$ denotes the set of finite-length bitstrings, i.e.,

$$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\} = \bigcup_{n=0}^{\infty} \{0,1\}^n.$$

The concatenation of bitstrings a and b is denoted as $a\,||\,b$. Because splitting $a\,||\,b$ into a and b is ambiguous, as the separation point between a and b is not specified, we introduce the special notation $[a, b]$ to stand for a bitstring that uniquely represents the pair (a, b) of bitstrings.[7] This notation extends naturally to $[a_1, \ldots, a_k]$ denoting a bitstring that represents the list $(a_1, \ldots, a_k)$ of bitstrings.

B. One-Way Functions and Hash Functions

A *one-way function* $f : A \to B$ for a suitable domain A and range B is a function[8] such that

- f is easy to compute, i.e., there exists an efficient[9] algorithm that computes $f(x)$ for any x.
- f is infeasible to invert, i.e., there exists no efficient algorithm which on input $f(x)$, for a randomly selected x, computes an $x' \in A$ with $f(x') = f(x)$, with non-negligible probability of success.

One-way functions are perhaps the most basic cryptographic primitive capturing the notion of the infeasibility of a computation. Note that a one-way function is a fixed function, involving no secret key. It is widely believed but not proved that one-way functions exist.

A function $h : \{0,1\}^* \to \{0,1\}^n$, mapping bitstrings of arbitrary length to the n-bit strings, is called a *hash function*. A hash function h is called *collision resistant* if it is infeasible to find x and x' such that $h(x') = h(x)$.[10]

C. Bitstrings as Evidence

In order for a (purely digital) bitstring s to imply A's liability for document d, two conditions must be satisfied.

- A must have agreed to such liability, either explicitly (e.g., by a written declaration) or implicitly (by the law).[11]
- The string s must be unambiguously specified.

In this section we discuss what it means to specify s.

We first consider the simplest scenario where the document space $\mathcal{D}$ contains only one fixed element, namely, a digital check with a fixed value. User A wants to be able to write a digital check (only once, for simplicity) of a certain

[6]One can also think of a bitstring as a number, since there is a one-to-one correspondence between the bitstrings and the integer numbers.

[7]For example, $[a, b]$ could be the concatenation of the prefix-free encodings of a and b.

[8]Typically $A = \{0,1\}^m$ for some m or $A = \{0,1\}^*$, and $B = \{0,1\}^n$ for some n.

[9]We do not specify the terms "efficient," "infeasible," and "nonnegligible" precisely, as this is not necessary for the purpose of this paper. In the traditional cryptographic literature one usually considers an asymptotic setting, i.e., for an asymptotic family of functions with increasing parameter, where efficient (infeasible) means (not) computable in time polynomial in the parameter.

[10]For technical reasons, a precise definition requires that h is thought of as being chosen at random from a class of hash functions.

[11]Here the term "implicitly agree" is used in the same sense as we (have to) implicitly agree to be liable for a conventional signature on a contract, without ever in our life explicitly agreeing to such liability. However, in the context of digital signatures, an explicit agreement appears to be reasonable and necessary.

fixed amount (say, \$100), payable by bank B, to anyone who presents this check to B. A does not know in advance whom she might later give the check.

The digital check is simply a bitstring c, which, when presented to bank B, constitutes the entitlement to receive the money (of course only once). As usual in banking, B would of course request A to sign a conventional contract specifying these instructions, including the string c, and stating that B has the right to withdraw \$100 from A's account, provided B can present the string c as proof. Such a declaration will be called a *commitment declaration*.

Of course, the string c, while being specified on the commitment declaration, must not appear on it nor be efficiently computable from it. (Otherwise B's employees would know c and could, hence, fraudulently cash the money, even if A never issued the check.) More precisely, the commitment declaration specifies a Boolean *verification function* (or verification predicate)

$$v : \{0,1\}^* \rightarrow \{0,1\}.$$

By definition, a string s is accepted as correct evidence if and only if $v(s) = 1$.[12] There may be several such strings s, and it is irrelevant which one is presented.

For this scheme to be secure (for A), it must be infeasible, when given v, to generate a string s satisfying $v(s) = 1$, with nonnegligible probability. Another requirement is that when given s and v one can efficiently check whether $v(s) = 1$.

This problem has an elegant solution. A selects c at random from $\{0,1\}^n$ for some large enough n, but instead of stating c explicitly on the declaration, A states $y = f(c)$, where f is an agreed one-way function. It is infeasible to derive c from y. For this scheme, the verification predicate is given by

$$v(s) = 1 \Longleftrightarrow f(s) = y.$$

D. General Document Spaces

The scheme described above is severely limited, as it allows only for the authorization of a single action by the bank. With increasing generality, one would want to issue several checks, checks with freely specifiable amounts, authorize general transactions (e.g., a money transfer to another account), or even sign any contract with any business partner, not just with a particular bank B. In other words, A wants to be able to sign a general digital document d from some *document space* $\mathcal{D} \subseteq \{0,1\}^*$, where the contract partner is specified as a parameter of d rather than on the commitment declaration.

Stated more formally, the commitment declaration specifies a Boolean verification predicate

$$v : \mathcal{D} \times \{0,1\}^* \rightarrow \{0,1\}$$

where a bitstring s implies liability for document d if and only if $v(d,s) = 1$. The following three requirements must be satisfied.

- **Security**. Given v, it must be infeasible to generate, strings $d \in \mathcal{D}$ and s with $v(d,s) = 1$, even when given arbitrary other such pairs (d', s') with $v(d', s') = 1$.
- **Efficient verifiability**. Given d, s, and v, one can efficiently check whether $v(d,s) = 1$.
- **Feasibility**. The legitimate party can efficiently generate, for any $d \in \mathcal{D}$, a bitstring c_d such that $v(d, c_d) = 1$.

To generalize the above solution based on a one-way function, a different string c_d for every $d \in \mathcal{D}$ would be required. Thus, the length of the commitment declaration would be proportional to $|\mathcal{D}|$, which is completely infeasible for a realistic context.

E. Digital Signature Schemes

Digital signatures, defined below, provide the solution to the above problem. In fact, a digital signature scheme can perhaps best be interpreted as a method for specifying a very large (exponential) number of values $f(c_i)$ in a compact form (by the public key), such that all bitstrings c_i can be efficiently computed from a compact value (the secret key), and such that even when given any subset $\mathcal{S}$ of the c_i it is nevertheless infeasible to compute any bitstring c_j not contained in $\mathcal{S}$.

A *digital signature scheme*[13] for document space $\mathcal{D}$ consists of three efficient algorithms.

- **Key generation algorithm**: Generates a random key pair, consisting of a signing key (the secret key) k and a verification key (the public key) p.
- **Signing algorithm**: Takes as inputs a document (or message) $d \in \mathcal{D}$ and a signing key k and computes the signature $\sigma = \mathrm{sig}(k, d)$ for d.
- **Signature verification algorithm**: Takes as inputs a document d, a signature σ, and a public key p and computes a binary predicate

  $$\mathrm{vsig}(p, d, \sigma)$$

  where one and zero are interpreted as "accept" and "reject," respectively.

The following conditions must be satisfied.

- **Correctness**: Valid signatures must always be accepted

  $$\mathrm{vsig}\,(p, d, \mathrm{sig}(k, d)) = 1.$$

- **Security**: It must be computationally infeasible to forge signatures for a given public key, i.e., to compute, without knowledge of the signing key, a document/signature pair that will be accepted.[14]

[12]Alternatively, but of course equivalently, one could use TRUE and FALSE instead of one and zero, respectively, as the values of the predicate. The term "predicate" is used for a binary-valued function whose output is interpreted as a logical value.

[13]The concept of a digital signature scheme was first proposed by Diffie and Hellman [1], in their seminal paper introducing the ingenious concept of public-key cryptography. However, they proposed no implementation of the concept, and it was only in 1978 when Rivest, Shamir, and Adleman (RSA) [8] proposed the first efficient instantiation, called the RSA system.

[14]This must be infeasible even if an attacker can obtain correct signatures for arbitrary chosen message, as long as the message for which the signature is forged is not asked. This strong type of security of a digital signature scheme is called existential unforgeability under an adaptive chosen-message attack.

Referring to Section II-D, the verification predicate can now be set to

$$v(d, s) = \text{vsig}(p, d, s)$$

for a specific public key p specified (by a declaration or by other means) as A's public key. The properties of the digital signature scheme assure that the requirements of Section II-D are satisfied.

The core of a cryptographic digital signature scheme typically works only for bitstrings of fixed length, i.e., the document space is usually $\mathcal{D} = \{0,1\}^n$ for some n (e.g., $n = 160$). Such a scheme can be converted into a signature scheme for documents of arbitrary length, i.e., for the document space $\mathcal{D}' = \{0,1\}^*$. This is achieved by hashing the document before signing it, using a cryptographic hash function $h : \{0,1\}^* \rightarrow \{0,1\}^n$. More precisely, the derived signature scheme for $\mathcal{D}'$ has the verification predicate

$$\text{vsig}'(p, d, s) = \text{vsig}\,(p, h(d), s)\,.$$

It must be prevented that B can prepare two different contracts with the same hash value, since A's signature would sign both contracts. This is guaranteed if h is collision resistant.

F. Public-Key Certificates and PKIs

Three settings in which public keys are used are:

- public-key encryption (used for key management, e.g., in SSL);
- bilateral message or entity authentication;
- digital signatures as evidence.

We are interested only in the third context, which differs substantially from the other two. For encrypting and authenticating communication between two entities, all that is needed is that they can establish a secure key, possibly with the help of trusted authorities. In contrast, the third context requires a common framework to which all users agree, defining what constitutes valid evidence.

In all these contexts, it is crucial that the public key one is using is authentic.[15] The authentication of public keys is one of the most crucial operations in information security. *Public-key certificates* are a possible solution for this problem, for all three contexts, but their semantics is different in the three contexts. The purpose of a public-key certificate, issued by some entity T for some other entity A, is to confirm that a certain public key "belongs to" A. The certificate consists of at least:

- A's unique identification information (e.g., the name, other parameters such as an e-mail address or possibly a digital picture);
- A's public key;
- further parameters like an expiration date, the specification of a context in which the certificate is valid, etc.
- T's digital signature on a digital document consisting of all the above information.

In order to be useful, a given certificate c must be checked using the public key p_{T} of the alleged issuer T of c. A certificate is useful only if one has an authenticated copy of p_{T}. Moreover, in the encryption and authentication context, user B checking the certificate must trust T. In the signature context, which is of interest in this paper, it remains to be discussed whether A and/or B must trust T. We define

$$\text{id}(p, c)$$

to be the identity string contained in the certificate c if checked with public key p. If the check fails, we could let $\text{id}(p, c) = \perp$, where $\perp$ is some failure symbol. Similarly, we define

$$\text{pk}(p, c)$$

to be the public key contained in the certificate c if checked with public key p. For example, if T has indeed issued the above described certificate for A and public key p_{A}, and if p_{T} is known to be T's public key, then we have $\text{id}(p_{\mathrm{T}}, c) = \mathrm{A}$ and $\text{pk}(p_{\mathrm{T}}, c) = p_{\mathrm{A}}$. Moreover, in the same spirit we can define

$$\text{exp}(p, c) \quad \text{and} \quad \text{lia}(p, c)$$

as the expiration time and a liability bound specified in c, and one could introduce similar expressions for other parameters of a certificate.

In general, T's authentic public key is not known *a priori* and must also be extracted from a certificate issued by some other trusted entity T'. This can result in certificate chains of (theoretically) arbitrary length. A certification topology often proposed is a tree-shaped hierarchy with a so-called root-CA at the root of the tree, with lower-level CAs at the inner nodes,[16] and with the certified users at the leaves. The certificate chain for user A corresponds to the path in the tree from the root-CA to A. In order for this to work, at least in the encryption and authentication context, B must hold an authentic copy of the root-CAs public key and must trust *all* entities on the chain [3]. The framework and mechanisms for managing public keys is often referred to as a *public-key infrastructure*.

In order to use certificates as digital evidence, their semantics (i.e., what "p belongs to A" means) must be made precise. We will discuss several such interpretations.

G. Causal Dependencies Between Bitstrings

Let h be a cryptographic hash function. If $b = h(a)$ for two strings a and b, then one can conclude that b was derived from a using h (unless h is insecure). More generally, the same is true if $b = h([c, a, d])$ for some arbitrary bitstrings c and d.

This is useful, for example, in the context of time stamping or when signing a hash value of a document rather than the document itself. If a is sufficiently long (say, at least

[15] A user B wants to make sure he is using A's correct encryption public key before encrypting a message (e.g., a session key) for A. Similarly, B must make sure to use A's correct authentication public key when authenticating a message from A. We refer to [3] and [4] for a formal treatment of these settings.

[16] Such CAs can also cross certify each other's public keys.

160 bits), then one can argue that a must have been known (to somebody, or internally in some system) before b was generated.

We introduce the following relations on bitstrings, relative to a fixed hash function h. We define

$$a <_h^w b \iff w = [c, d] \wedge b = h([c, a, d])$$

i.e., $a <_h^w b$ if and only if $b = h([c, a, d])$ for some bitstrings c and d given explicitly as $w = [c, d]$. Here w can be considered as a witness for the claim that b was derived from a, using hash function h.[17]

We generalize this notion by saying that b *is derived from* a (with respect to h), denoted $a \prec_h^w b$, if b can be derived from a in one or several steps. More precisely, either $a <_h^w b$ or $w = [w', x, w'']$ where $a <_h^{w'} x$ and $x \prec_h^{w''} b$

$$a \prec_h^w b \iff a <_h^w b \vee \left(w = [w', x, w''] \wedge a <_h^{w'} x \wedge x \prec_h^{w''} b\right).$$

This is a recursive definition.

III. Current Approaches to Digital Evidence

In this section, we describe, step by step, the basic ideas underlying the current approaches to digital evidence, by specifying the verification predicates (see Section II-D) that would be used to check the digital evidence.

In a totally naive view of using digital signatures, A's signature on d would imply liability for d. This is modeled by defining the verification predicate as

$$v(d, s) = \text{vsig}(p_\text{A}, d, s).$$

In this view, it would be considered understood outside of the context of the evidence that p_A is A's public key. In any realistic context, however, one must have evidence to prove that p_A is indeed A's public key.

A. Modeling Certificates and Hierarchical Certification

As described in Section II-F, a solution to this problem is to have A's public key certified by a trusted certification authority C whose public key p_C is (for now) assumed to be fixed and publicly known. This means that the digital evidence string s consists of two parts, the signature σ and the certificate c, which both need to be checked.[18] The verification predicate becomes

$$\begin{aligned} v(d, s) = \quad & s = [\sigma, c] \\ & \wedge \text{id}(p_\text{C}, c) = \text{A} \\ & \wedge \text{pk}(p_\text{C}, c) = p \\ & \wedge \text{vsig}(p, d, \sigma). \end{aligned}$$

[17]Note that for any b in the range of h and for every a, there generally exist (many) c and d such that $b = h([c, a, d])$. Therefore, it would not make sense to define $a <_h b$ as $\exists c, d : b = h([c, a, d])$. In the context of digital evidence, the main question is whether such c and d can actually be presented by an entity, not whether they exist.

[18]We do not distinguish between an entity and its name. In other words, A stands both for the entity A and for the bitstring representing the identity of A. The naming problem, i.e., assigning unique names to entities is important but beyond the scope of this paper.

Note that the string p appears in the above formula as existentially quantified, but this is only implicit. In an explicit notation one would have to write $\exists p$ in front of the entire formula (after "="). Here and below this is assumed to be understood from the context. Alternatively, we could have replaced the last two lines by

$$\text{vsig}(\text{pk}(p_\text{C}, c), d, \sigma)$$

avoiding the need to give a name to the string $\text{pk}(p_\text{C}, c)$.

If the public key of the CA C is not assumed to be publicly known, it also needs to be authenticated, for instance, by a certificate chain rooted in the so-called root-CA R. Here we model a two-level hierarchy where any CA certified by R is authorized to certify the users' public keys. In other words, a certificate by R not only confirms the authenticity of the CA's public key, it also authorizes the CA to issue certificates. Now the digital evidence string s includes both certificates. The name of the CA that is certified by R is not specified; i.e., it is a free parameter

$$\begin{aligned} v(d, s) = \quad & s = [\sigma, c, c'] \\ & \wedge \text{pk}(p_\text{R}, c') = p \\ & \wedge \text{id}(p, c) = \text{A} \\ & \wedge \text{pk}(p, c) = p' \\ & \wedge \text{vsig}(p', d, \sigma) \end{aligned}$$

or, shorter but equivalently

$$\begin{aligned} v(d, s) = \quad & s = [\sigma, c, c'] \\ & \wedge \text{id}(\text{pk}(p_\text{R}, c'), c) = \text{A} \\ & \wedge \text{vsig}(\text{pk}(\text{pk}(p_\text{R}, c'), c), d, \sigma). \end{aligned}$$

Certification chains of arbitrary lengths could be modeled similarly. However, note that the longer the chain, the higher the risk that one of the links in the chain fails, for instance, because a CA's signature key is compromised or because of fraud at a CA. In any case, the predicate $v(d, s)$ defines the public key of the root-CA and how many levels of certification are allowed.

B. Certificate Expiration and Time Stamping

It appears crucial that one can limit the period during which one is liable for signatures relative to a public key. One should be able to specify an expiration date of a public key, and one should also be able to revoke a public key before it expires, for instance, when the private key is lost or leaked or when the context changes (e.g., one changes the job).[19]

The general view is that as long as a certificate has not expired or been revoked, the user is liable for the corresponding signatures, but she is not liable for signatures issued after expiration or revocation. This implies that one must be able to determine *when* a digital signature was generated, or at least whether it was generated prior to a certain time.

[19]The literature usually refers to "certificate revocation," but actually what should be revoked is a public key or, more precisely, the commitment to a public key. This is obvious in a situation where there exist several certificates for a public key.

For this purpose, one can use a trusted *time-stamping authority* (TSA) T that confirms that a certain bitstring (e.g., a hash value of a signature) was presented at a certain time [2]. For this purpose, it issues a time stamp. A time stamp on a document is a signature by the TSA on the document together with the time when the time stamp was issued. A time stamp τ must be checked using the public key of the TSA. For a time stamp τ checked with public key p, let

$$\mathrm{time}(p,\tau) \quad \text{and} \quad \mathrm{str}(p,\tau)$$

be the time and the string, respectively, specified in τ, when checked with public key p.[20]

To model certificate expiration and time stamps, we consider for simplicity only single-level certification with a fixed and known public key p_{C}. Let p_{T} be T's public key, again assumed to be a fixed and authentically known parameter. The verification predicate is

$$\begin{aligned} v(d,s) = \quad & s = [\sigma, c, \tau] \\ & \wedge \mathrm{id}(p_{\mathrm{C}}, c) = \mathrm{A} \\ & \wedge \mathrm{vsig}\left(\mathrm{pk}(p_{\mathrm{C}}, c), d, \sigma\right) \\ & \wedge \mathrm{time}(p_{\mathrm{T}}, \tau) \leq \exp(p_{\mathrm{C}}, c) \\ & \wedge \mathrm{str}(p_{\mathrm{T}}, \tau) = \sigma. \end{aligned} \tag{1}$$

More generally [2], it can be considered sufficient as a time stamp for σ if the bitstring time stamped in τ is derived from σ, i.e., one could replace the clause $s = [\sigma, c, \tau]$ by

$$s = [\sigma, c, \tau, w]$$

and the clause $\mathrm{str}(p_{\mathrm{T}}, \tau) = \sigma$ by

$$\sigma \prec_h^w \mathrm{str}(p_{\mathrm{T}}, \tau).$$

Here h is assumed to be a known, standardized hash function.

To increase the security, one could require more than one time stamp, or more than one certificate. It is straightforward to give the corresponding verification predicates for these and other extensions.

C. Certificate Revocation

Next, we model certificate revocation. A signature is considered valid only if at the time of signing the certificate has not been revoked.

In the present approaches, this problem is taken care of by having the CA keep track of which public keys are revoked. A key can be revoked by contacting the CA. There are essentially two approaches (and combinations thereof) for providing this information to users.

- **Certificate revocation lists (CRLs)**. The CA maintains a CRL containing all revoked certificates. The CRL is digitally signed and published periodically by the CA. A problem with CRLs is that revocation becomes active only when the next CRL is published.
- **Online revalidation**. Each time a request for a certificate is received, the CA confirms, essentially by a new (revalidation) certificate, that the public key is still valid.

Typically, the CA C is trusted to manage the CRL correctly, to keep periodic backups of the old CRL, and, if needed, to confirm the CRL status at any requested time in the past. Perhaps surprisingly, in this case the verification predicate does not change at all, i.e., no further digital evidence is checked. The only modification, not reflected in the predicate, is that the recipient (say B) of A's signature on d checks the CRL status when receiving and time stamping the signature.[21]

If one does not fully trust the CA C, the recipient of A's signature would have to keep evidence for the fact that A's certificate was not revoked at the time of signing. For this purpose, C can sign a statement to this effect, which can be called a *revalidation certificate* or certificate status confirmation. For a given such revalidation certificate r, let

$$\mathrm{time}(p,r) \quad \text{and} \quad \mathrm{pk}(p,r)$$

denote the revalidation time and public key, respectively, stated in r when checked with public key p. This situation can, thus, be modeled by replacing $s = [\sigma, c, \tau]$ in (1) by $s = [\sigma, c, \tau, r]$ and including the clause

$$\mathrm{time}(p_{\mathrm{T}}, \tau) \leq \mathrm{time}(p_{\mathrm{C}}, r) + \Delta \wedge \mathrm{pk}(p_{\mathrm{C}}, c) = \mathrm{pk}(p_{\mathrm{C}}, r)$$

for some parameter $\Delta \geq 0$ which stands for the period after revocation during which signatures remain still valid.

In the above clause, one checks $\mathrm{pk}(p_{\mathrm{C}}, c) = \mathrm{pk}(p_{\mathrm{C}}, r)$, i.e., whether the public key in r is the correct one, namely, that of c. However, a valid question is why one needs the certificate c in the first place. If revalidation certificates are used and required (to prove A's liability), then they should by themselves be sufficient evidence, without certificate c. It is clear how this view could be modeled by an appropriate verification predicate.

D. Liability Limitations

A certificate c could contain as one of the parameters a bound, denoted $\mathrm{lia}(p,c)$, on the maximal liability implied by the signed contract d. In this case, one would add the clause

$$\mathrm{price}(d) \leq \mathrm{lia}(p,c)$$

in the verification predicate. Note that the bound $\mathrm{lia}(p,c)$ applies to each signature separately. One cannot meaningfully restrict the total liability of a public key, since the recipient of a signature has no way of knowing the total liability of all previously issued signatures for p. But we discuss a possible solution in Section V-H.

IV. The Fundamental Dilemma in Legislation

In the previous section, we have assumed that digital evidence alone implies liability, and we have modeled different degrees of sophistication of the digital verification predicate.

Of course, legislators have recognized the problem that if exclusively digital evidence were relevant in a dispute (as discussed in Section IV-A below), then a user A would have no possibility to defend herself in face of a correct combination of a digital signature, certificates, and time stamps. What one actually wants (see Section IV-B) is that the user's willful act

[20]As usual, these functions return a special value $\perp$ when τ is not a valid time stamp with respect to p.

[21]Since revocation is a monotone process, it suffices to check the absence of c on the CRL at an arbitrary time after the time stamp on the signature.

is what counts. But this leads to a fundamental dilemma, as discussed in Section IV-C.

Therefore, some legislations allow A to prove that she is not liable for a signature, without specifying precisely which evidence she could either present, or request to be presented by the other party. For example, what happens if A claims to have never in her life applied for a certificate, or if she has a perfect alibi (e.g., she was in prison) for the time when the signature was allegedly issued? This ambiguity is (today) left to the judge to decide. However, we argue that the general issue of which additional evidence would be considered in a dispute must be defined more precisely.[22] As a result of such an analysis, we point out that some problems cannot be resolved in the conventional view, forcing us to present a new view in Section V.

A. Approach I: Digital Evidence Implies Liability

Consider the simple and, therefore, attractive approach that digital evidence strings alone imply liability, as described in Section III. It is excluded that A can present further evidence to prove that she did not sign.[23]

In this approach, A is exposed to the abstract risk of being liable for a contract d if somebody manages to generate a correct signature. Such a forgery could happen for any one of many different reasons, some of which are discussed below.

1) The secret key could have leaked to a third party, for instance due to a security problem in the system or a timing or power attack on the user's smart card.
2) The signature could have been generated by the user's system, but without her consent, for instance, due to a virus or other malicious software on the system with the inserted smart card. The virus could either:
    a) call the smart-card without the user being aware;
    b) display a contract different from that actually signed.
3) The signature could have been generated by the user's system, without any influence from an outsider or a virus, but nevertheless without the user's consent and awareness. This could happen for instance if:
    a) the user interface is not sufficiently clear about which action (e.g., clicking "OK") initiates the signature generation;
    b) another person is using the user's system or secure signature device.

    Another possibility is that the user simply forgot that she actually completed a transaction.
4) The cryptographic signature scheme might be broken.[24]
5) The certificate could be false, for instance, because of a criminal CA employee or because the CA's private key is compromised.

When A is confronted with a signature (for her public key), she may have no clue which of the above reasons applies. Therefore, she cannot even meaningfully deny the digital signature.

B. Approach II: A Willful Act Implies Liability

An apparently completely different approach, much closer to the conventional understanding of contract legislation, is that A is considered liable for d only if she performed, in full awareness of the content of the contract d, the well-defined willful act (e.g., clicked "OK") which initiated the generation of the signature on d.

In this approach, the judge takes the digital signature only as one piece of supporting evidence, considering also other evidence. Many legislative approaches follow this line of reasoning. However, this approach, like the first, has also severe problems. Some relevant questions are the following. What kind of evidence should the judge consider to be more convincing than the digital signature? How should A be able to produce such evidence? Or should the judge request B to provide, in addition to the signature, further evidence? In the latter case, it is not clear what the purpose of the digital signature should be.

C. The Dilemma

The advantage of the first approach is that it is entirely unambiguous. The advantage of the second approach is that our current understanding—namely, that a willful act is required to enter a contract—is reflected. The dilemma is that one cannot have both advantages (but see Section VI).

The envisaged approach to solving this dilemma, followed in certain legislations, is as follows. By definition (i.e., by the law), it is indeed the user's *act* which implies liability, but the technical infrastructure and the involved processes are designed to satisfy very high security standards so that it appears virtually impossible that a signature is generated without the user's consent. Thus, the signature alone can indeed meaningfully be taken as convincing evidence that A has signed d. Some of the possible requirements are the following.

1) Very high security standards for the CA's technical infrastructure, processes, and personnel supervision.
2) High security margins in the choice of the cryptographic security parameters.[25]
3) The user interface is required to be highly unambiguous, essentially excluding any misunderstandings.
4) The user's private key is stored in a very secure device, without possibility to extract it.[26] Signatures are generated in the device. Because of virus attacks, the device should ideally have its own input and output mechanisms, for instance, a keyboard (at least a confirmation button) and a display.

[22]In fact, even if not defined by law, legal practice would have to be established, which could later become part of the law. And, hence, one must at least be able to argue what a reasonable such practice could be.

[23]There are obvious exceptions which a judge would have to consider—for example, A's claim that she was forced by B to sign d. Such exceptions are outside the scope of the model discussed in this paper.

[24]Current proposals for digital signature schemes depend on the assumed computational hardness of a very specific mathematical problem, for instance, factoring large integers. It is conceivable that a fast algorithm for solving this problem will be discovered. Even worse, such a discovery might not necessarily be reported to the public.

[25]For instance, one could use several signature schemes in parallel.

[26]Smart cards actually may not be sufficiently secure.

5) The security of a device could be increased further by a biometric identification mechanism, allowing only the designated user to activate the device.

There is an obvious tradeoff between the achieved level of security on one hand and the cost and practicality on the other hand. But even if the technical security is carried to an extreme (and impractical) level, it is impossible to eliminate all sources of uncertainty. For instance, no such solution can prevent a disaster in case the cryptographic signature scheme were broken.

It is important to point out that in this approach, it is indeed the digital evidence alone which implies liability. In other words, this actually corresponds to (a disguised version of) the first approach.

V. The New Approach to Digital Evidence

A. Abstraction of the Legal System

As discussed in Section I-F, the actual legal systems in use today are quite ambiguous, leaving substantial room for interpretation of the meaning of evidence. Ambiguity is unavoidable because evidence is often ambiguous and legislation cannot foresee all possible cases that may occur in the future. Moreover, one can probably argue that this ambiguity is necessary to build common trust into the legal system, since the ultimate decisions are made by trustworthy people (rather than machines) after listening to all arguments.

However, in order to be able to reason precisely, we must separate the ambiguous from the unambiguous issues. Our goal is to be able to ignore the ambiguous issues and capture the unambiguous issues in a precise manner.

We are not advocating that the legal system should be much more formalized in general, trying to eliminate the ambiguities. But in view of the implicit promise of digital signatures to reduce ambiguity and allow for a substantial level of automation in handling evidence, we argue that the legal system should specify (more) clearly which types of evidence are relevant and will be considered in a dispute.

The resolution of a dispute in court is a process that may involve several phases during which new evidence can be presented and new witnesses can be asked to testify. At the end of this process, the judge's decision (e.g., on whether or not A is liable for d) is based on the total set of evidence that has been presented, including:

- physical evidence;
- statements by witnesses;
- digital evidence strings;
- digital recordings.

In order to separate the unambiguous from the ambiguous aspects of evidence, we assume to have an unambiguous *evidence description language*, with sufficiently well-defined syntax and semantics, for describing physical evidence, statements by witnesses, and digital recordings. However, such a concrete language will not be developed in this paper. Let $\mathcal{E} \subseteq \{0,1\}^*$ be the space of such evidence descriptions. For example, pieces of evidence that could be specified in such a language are the following.

- A physical declaration, signed by A, stating that A accepts liability for public key p_{A}, with expiration date t and liability bound (per signature) b.
- A witness testifying that he or she saw A sign a certain declaration on a particular date.
- A digital video recording showing A stating that she agrees to some contract d, mentioning the main parameters of d.

Here we only model evidence *description*, not the evidence itself. The ambiguous question whether a certain piece of evidence matches the description is left out of consideration as an issue to be resolved, if necessary, by a judge.[27] However, it is specified by the legal system (and not left to the judge's discretion) whether or not A can insist on having the physical declaration, the witness, and/or the video presented in court before being declared liable for d.

As discussed earlier, the digital evidence string s required in a specific setting to prove A's liability is specified by a verification predicate $v(d,s)$. Let, therefore, $\mathcal{V}$ denote the set of efficiently computable predicates

$$\mathcal{D} \times \{0,1\}^* \rightarrow \{0,1\}.$$

Abstractly, the formal decision about whether A is liable for d, after eliminating ambiguous issues, can be modeled as a *liability function*

$$\lambda : \mathcal{I} \times \mathcal{D} \times \mathcal{E} \times \mathcal{V} \rightarrow \{0,1\}$$

where $\mathcal{I}$ is the entity name space.[28] The meaning of this function λ is that it defines precisely which pieces of evidence are required to prove a user's liability, as follows. If

$$\lambda(\mathrm{A}, d, e, v) = 1$$

then A is liable for d if the following evidence is presented:

1) evidence satisfying description e.
2) a bitstring s satisfying $v(d,s) = 1$.

Typically, $\lambda(\mathrm{A}, d, e, v)$ depends on A, d and v only in a very simple manner. It depends on A only by the fact that A is a parameter of e (e.g., A must have signed a declaration).[29] It depends on d only through some relevant parameters, for example a price specified in d, denoted

$$\mathrm{price}(d).$$

Moreover, $\lambda(\mathrm{A}, d, e, v)$ typically depends on v only in that v is a parameter of e, e.g., v must be specified on a declaration signed by A.

The liability function separates the evidence into the two relevant parts: the digital evidence strings, which can be checked by a verification predicate, and the remaining evidence, which must be checked by human beings, for

[27]In the above examples, it is left to the judge to decide, for instance, whether a presented piece of paper, which might be partially burnt or otherwise destroyed, still counts as a valid declaration, whether the witness' testimony is satisfactory, and whether the presented video really shows A making the required statement.

[28]We are not concerned with the problem of efficiently representing the function λ, nor do we address the naming problem.

[29]This also follows from the requirement that all entities should be treated equally by the law. But $\lambda(\mathrm{A}, d, e, v)$ might possibly depend on the type of entity (e.g., a person or a legal entity).

instance, a judge, relative to a (sufficiently) precise description e. Conceptually, one can think of the digital evidence string s as being checked only after the remaining evidence has already been checked with respect to e. This makes the role of digital evidence more transparent. If $\lambda(\mathrm{A}, d, e, v) = 1$ and evidence matching description e has been presented, then A's liability for d depends solely on the presentation of a bitstring s with $v(d, s) = 1$. This is unavoidable. Any modifications one might propose to reduce this abstract risk of a bitstring implying liability (for example, time stamping s or allowing A to present further evidence) can be modeled within the described framework and yields simply a different function λ and a different predicate v. Our model is indeed completely general.

The choice of the function λ is defined by the legal system and must be chosen to balance the tradeoffs discussed in Section I-E. If very high security for A were the only goal, then λ should be defined such that $\lambda(\mathrm{A}, d, e, v) = 1$ only for e for which forging evidence is highly infeasible (e.g., requiring false testimonies by several witnesses), and for v such that forging a bitstring s with $v(d, s) = 1$ is also highly infeasible (e.g., requiring breaking several independent signature keys).

As discussed below, a typical case of evidence e is an explicit commitment declaration. The function λ could specify different security levels of such declarations. The required (by λ) security level could depend on the validity period and the liability bound specified in the declaration.

B. Delegated Signature Generation

One obvious method for increasing the unforgeability of a bitstring s with $v(d, s) = 1$ is to define the predicate v in a way that several signatures for different (explicit) public keys are required to satisfy v. One of the secret keys could be controlled by A, as her normal signature key, and the other secret keys could be controlled by some entities she trusts.

Such signatures will be called *delegate signatures*. A delegate authority T is trusted by A to generate the (in v) required signature only under well-defined circumstances defined by A in an agreement with T. A typical instruction for T could be to sign any signature by A upon request (by any entity), as long as A has not revoked this authorization from T. In fact we will argue in Section V-G that this is how revocation and revalidation should actually be interpreted.

Let p be A's public key and let p' be the public key T is using for user A. Then the predicate becomes

$$\begin{aligned} v(d, s) = \quad & s = [\sigma, \sigma'] \\ & \wedge \ \mathrm{vsig}(p, d, \sigma) \\ & \wedge \ \mathrm{vsig}(p', \sigma, \sigma'). \end{aligned} \tag{2}$$

Here we have assumed that T signs A's signature. It is left to the reader to model a somewhat different scenario when T also signs the document d, and to reason about whether this makes sense.

Note that the type of instructions for T do not show up in the verification predicate and are of no concern to B. They are only based on a bilateral agreement between A and T. If T does not follow the instructions, this is only A's problem. If T does not sign when it should, this prevents A from doing the desired business, and if T signs when it should not, then A risks to be liable for a contract she did not intend to enter.

The user's signature is typically generated by a mobile device like a smart-card. Since T can use advanced security technology and very strong cryptographic parameters not feasible with smart-cards, requiring T's signature adds substantially to the security of the overall system.[30] An obvious practical problem is to establish an authenticated channel from A to T, which itself relies on cryptographic mechanisms.

C. The Semantics of Certificates

Certificates are used to bind public keys to entities. In the context of digital evidence, there are two fundamentally different approaches to defining the semantics of a certificate.

1) A certificate by an accredited CA[31] confirms that p_{A} is A's public key and is sufficient evidence for A to be liable for p_{A}. No other evidence is considered. This view was modeled in Section III-A.
2) When registering the public key, A must explicitly commit to be liable for signatures with respect to p_{A}. Evidence confirming this commitment is generated and stored by the CA and can be presented by the CA if the need arises (i.e., if A denies liability for p_{A}). The semantics of the certificate is that the *CA holds such evidence*.

If the first interpretation is used, as modeled in Section III-A, then the CA must be trusted by all users and the security requirements are very high. But if the second approach is used, which we think is much more appropriate than the first, then we arrive at the following surprising conclusions.

- The certificate has absolutely no value as evidence in court, only the physical declaration does. The certificate is not an input to the verification predicate.
- The role of a CA is very different from that usually envisaged. The certificate only confirms that the CA keeps the physical (and possibly other) evidence needed to prove A's liability for p_{A}. The certificate also states other parameters of a declaration like the expiration date and liability limitations.[32]
- Only the recipient B of a signature from A must trust the CA, namely, that the CA would indeed be able to present the necessary evidence, quite possibly as a paid service.[33]

[30] One option is for A to operate herself a highly secure system T which she (i.e., her mobile system) can access remotely to obtain signatures, and which she can also shut down remotely.

[31] More generally, a certificate chain rooted in an accredited root-CA.

[32] Such a confirmation is similar to the publication by a trade registry of the list of people authorized to sign on behalf of a company. The trade registry holds the paper documents with the authorization signatures by the company's representatives, tracing the history back to the foundation of the company. If the liability for a signature by one of its employees would be denied by a company, then it could request the authorization signatures for that employee to be presented by the trade registry.

[33] Whether B bases his trust on a certificate by a root-CA or on other evidence or experience is entirely left to him. The legal system is not relevant in this decision.

- As a consequence, the security requirements (and, hence, cost) for CA operations can be reduced significantly.
- There is an important new type of trusted authority whose task it is to store physical evidence and manage witnesses that can testify certain events. This fact may lead to new business models for setting up trusted services and may be useful in other contexts where physical evidence is unavoidable.

D. Commitment Declarations to Verification Predicates

In the traditional view, a user commits to a public key when applying for a certificate, meaning that she accepts to be liable for signatures relative to the public key.

In our view, a user's commitment, entered by signing (or otherwise confirming) a commitment declaration, is not to a public key, but to the entire verification predicate, which includes as input all the relevant digital evidence strings. Let

$$\mathrm{comm}_i(\mathrm{A}, v, t, b)$$

denote the description of appropriate (physical and other) evidence proving that A has committed to be liable for d if a string s satisfying $v(d, s)$ is presented, with expiration time t and liability bound b.[34]

The subscript i denotes the type of commitment declaration. Different types of commitment declarations may provide different security levels and involve different sets of evidence.[35] Here it will not matter how exactly type i is defined; we will not consider different types. As mentioned above, the required type of security level can depend on the validity period and the liability bound.

If a commitment declaration is required by the law, this would be modeled by setting $\lambda(\mathrm{A}, d, e, v) = 1$ only for $e = \mathrm{comm}_i(\mathrm{A}, v, t, b)$. In this view, as mentioned above, a certificate is a statement by which an authority confirms to hold evidence satisfying some description $e = \mathrm{comm}_i(\mathrm{A}, v, t, b)$ for some i, A, v, t, and b. But perhaps it is more appropriate to replace the terms certificate and certification authority by different terminology, for instance by "evidence confirmation." We do not propose such new terminology in this paper.

E. True Time Stamping Is Not Achievable

If $\lambda(\mathrm{A}, d, e, v) = 1$, then a user A is liable for d if the appropriate evidence e and a bitstring s satisfying $v(d, s) = 1$ is presented. There is no further condition. In particular, it is totally irrelevant when and how s was generated.[36] In other words, if one uses time stamps, they have a significantly different interpretation: namely, if the predicate $v(d, s)$ is defined such that s must include a time stamp of some form to yield $v(d, s) = 1$, then the time stamp can be interpreted as a special type of delegate signature. The authority's instruction is to sign any document, but only with the correct time attached. Only A, but not the recipient of A's signature, needs to trust this authority.

The basic point is that A has complete freedom to design the verification predicate v stated on the commitment declaration. In contrast to the use of time-stamping or revalidation services, which she is forced to trust, she can choose which entities she wants to trust and for which instructions she wants to trust them. Once this point of view is taken, time stamping does not make much sense anymore as one specific type of instruction for the delegation authority T. The important point is that T's signature is required to satisfy $v(d, s)$, and how the signature is interpreted and for which type of message it is required is much less important, actually irrelevant.

It should be mentioned that if a physically secure time-stamping mechanism could be devised, not relying on the trustworthiness of an authority, then an alternative view could be developed in which time stamping indeed makes sense.

F. Evidence Expires, Not Public Keys

The traditional view on expiration is that when a public key expires, signatures issued after the expiration date are not valid. For such public-key expiration to make sense, one must be able to determine when a signature was generated. But as explained in the previous section, this makes no sense.

This means that an expiration date stated on the commitment declaration must be interpreted differently. It specifies until when a string s can be *presented* as valid evidence, independently of when it was generated. In other words, evidence expires, not public keys. A commitment declaration becomes useless as evidence after its expiration, since all digital evidence relative to this declaration loses its value. One cannot argue that a string s was generated before the expiration and, hence, should still be valid.

This appears to be the only reasonable interpretation of expiration. As a consequence of this view, the validity period of evidence should be kept short. A possible approach is to require a commitment declaration of very high security, which can be long term, to authorize commitment declarations of a lower security level, which are restricted to the short term.[37]

[34]Note that the predicate v need not have a natural interpretation as, for example, the verification of the user's signature. It is just some arbitrary predicate the user specifies. Of course, there would have to be a language with predefined templates for specifying v. One would not actually write the program code on the declaration.

[35]For example, one type might require only a paper document signed by A while a more secure type might additionally require two witnesses confirming that A personally signed the document and was properly instructed about the meaning and consequences of the declaration.

[36]Even if not required by the legal system, it may perhaps appear safe in practice to have digital signatures time-stamped, as this could be useful in a dispute. However, in our view, we assume and demand that the legal system defines which evidence is required and/or admissible to prove or disprove A's liability for d. Hence, if time stamps are not required by the legal system, then they are useless. To object to this point of view one would have to describe a scenario in which time stamps are relevant; but such a scenario could then be modeled as part of λ, bringing us back to the same situation as before such an argument was made.

[37]This also shows that contracts with a long-term value, i.e., which cannot be settled by a payment within reasonable time, should perhaps not be entered by digital signatures.

G. Revocation is Impossible

The same line of reasoning also applies to public-key revocation. For revocation to make sense, one must be able to determine when a signature was generated. In addition, one must be able to prove that a public key was still valid at a certain time.

A commitment declaration cannot be revoked (e.g., by destroying it or declaring it invalid), as this would invalidate all the digital evidence generated so far. It is also not admissible that the expiration date on a commitment declaration is changed later for the purpose of earlier expiration, because this would mean that digital evidence collected by some entity B would expire earlier than legitimately expected by B.

This means that revocation of a public key is impossible. As with time stamping, revocation (more precisely revalidation) can be interpreted as a delegate signature. The authority T is instructed to sign only as long as it was not told to stop signing. User A can contact T by any agreed means of communication and authentication and instruct T to stop signing.

H. Controlling Accumulated Liability

In order to control liability, a user can specify a liability bound as a parameter of the commitment declaration. However, this liability bound applies separately to each new digital evidence string in reference to the declaration. In other words, the number of contracts d for which A becomes potentially liable as a consequence of the commitment declaration is not bounded. Therefore, despite the liability bound, the financial risk is unlimited. This is a major problem of the current view of digital evidence.

As a possible solution to this problem one could again use delegate signatures by an authority instructed to keep track of the total accumulated liability and to stop signing when a certain bound is reached. As with other delegate signatures, the security of such a liability control mechanism is only as secure as the authority's secret key.

VI. Digital Declarations

A. The Concept

In this section we describe a pragmatic solution to the digital evidence dilemma. On one hand, the evidence should be digital in order to fit smoothly into the digitized business processes, but on the other hand it should be linked to the physical reality, in particular to a user's willful act when agreeing to d. This appears to be a contradiction, but there exists a solution.

The solution, proposed in [5] and called *digital declarations*, is to use the second type of digital evidence discussed in Section I-C. The user performs some willful act related to the relevant contract or document, and this act is recorded digitally and combined with characteristic information of the digital document. In a typical implementation, the digital declaration can be signed together with the actual digital document.

B. Witnesses and Guaranteed Signer Awareness

In a general context of entering a contract, it is often too expensive or infeasible to generate evidence that is by itself sufficiently convincing. For example, a signed contract alone, if denied by one party, is usually not sufficient evidence (see the discussion in Section VI-C). This is why witnesses are needed to resolve disputes, at least as a last resort when the evidence and the parties' statements remain conflicting.

However, for most types of contracts (e.g., an online transaction) it is impossible, too expensive, or impractical to arrange for an external witness to be present when A performs an act (e.g., clicks "OK"). In this case, the only reasonable alternative for resolving a dispute is to let one (or both) of the two involved parties A and/or B testify as witnesses. This significantly reduces the chance of false statements because perjury is considered a significant crime and punished severely.

But using an involved party A as a witness is useful only if it is guaranteed that she *can* indeed testify, i.e., if there is a clear (preferably yes/no) question which she can definitely answer and which, under the assumption that it is answered correctly, would resolve the dispute. This requires that A is guaranteed to be aware of the correct answer because then answers like "I do not know" or "I am not sure" are unacceptable and can reasonably be held against her. Guaranteed awareness assures that in a dispute, one of the parties is necessarily consciously lying, and, hence, makes a request to testify meaningful. This fact helps prevent false denials in the first place, avoiding the need for actual dispute resolution.

C. On the Role of Conventional Signatures

As a motivation for digital declarations, it is instructive to discuss the role of conventional handwritten signatures and why they are so useful in practice, despite the fact that their technical security is generally quite low. Conventional signatures are a pragmatic and flexible mechanism.

An idealized (but of course naive) view of the use of conventional signatures can be described as follows. A user's signature is well defined, for instance, by a master copy she has deposited. In case of a dispute, a signature allegedly issued by the user can be compared to her master signature. If one assumes that forged signatures can be recognized, then a signature is convincing evidence for the user's consent to the signed document.

In practice, however, things are quite different from this idealized view. First, most people's handwritten signatures are not very difficult to forge for a dedicated forger.[38] Second, in most settings (except, for example, in a bilateral business relationship with a bank), a person's master signature is neither deposited nor defined. Third, a person could use a signature different from the master copy in order to be able to later repudiate it.[39]

[38]In fact, some signatures like those often used by illiterates (e.g., "XXX") can trivially be forged.

[39]To avoid this last problem, the receiving party would have to have online access to a master signature registry.

The value of a handwritten signature is not primarily that it is difficult to forge, but rather that it creates a situation in which a person A *knows* whether or not she signed, thus guaranteeing her *awareness* of performing a conscious and willful act. Similarly, forging a signature also requires a conscious act. Due to this guaranteed awareness, the denial of having signed a document is a precise and meaningful claim, equivalent to the (serious) claim that the signature is forged.

The described view on conventional signatures is in sharp contrast to what digital signatures can achieve. The existence of a digital signature does not imply the guaranteed awareness of the alleged signer of the act that caused the signature generation. A signature could have been computed by a virus, because of another security problem, a flaw or ambiguity in the user interface, a flaw in the cryptographic mechanism, fraud or errors in the certification process, or any other of many possible reasons. Therefore, a user A cannot meaningfully deny that a signature was generated by her. Rather, a denial is equivalent to the quite useless claim that she is not aware of having issued a signature.

D. The Usefulness of Digital Declarations

Digital declarations are purely digital and are, hence, easy to transport and store. But they offer the same functionality as conventional handwritten signatures. In particular, a user is guaranteed to be aware of what he is doing and can, hence, meaningfully be forced to deny a digital declaration. Such a denial is equivalent to the accusation that the digital declaration is forged. We refer to [5] for a more thorough discussion.

Digital declarations can be embodied in many different ways. As an example, a user ordering a service or product online might be asked to speak a certain sentence referring to the product, the price, and the date of purchase. As another possibility, the willful act could be documented by a digital image, a video, or by any other recording device possibly invented in the future.

Digital declarations can be an essential feature of future digital transaction systems. Some of the reasons are:

- guaranteed user awareness;
- higher deterrence of misbehavior, hence, fewer disputes;
- improved security compared to conventional signatures;
- lower cost[40] due to reduced security requirements;
- improved acceptance of digital signature technology;
- usability by moderately educated people.

VII. Conclusion

We have presented a new view and interpretation of digital evidence which is in sharp contrast to the existing views, systems, and legislation.

A user commits, typically by a physical declaration that can also involve witnesses, to a verification predicate. In its simplest form, this could be the signature verification function for a particular public key. But by using a more complex verification predicate, possibly involving delegate signatures, the security can be increased substantially. Moreover, new functionality like a control of the accumulated liability becomes feasible.

Unless one accepts a legal system in which purely digital evidence, without any physical declaration, implies liability, certificates are irrelevant in a dispute. Instead of certificates one uses signed confirmations by certain authorities whose purpose it is to acquire, store, and present physical evidence.

Time stamps and revalidation certificates should be interpreted differently, namely, as delegate signatures. But once this view is taken, a more natural meaning, not related to time or revalidation of a public key, can be given to such delegate signatures. What counts is the verification predicate, not the interpretation a user may give to the parts of the digital evidence required to satisfy the predicate.

Public keys cannot expire, and they cannot be revoked. An expiration date on a commitment declaration refers to the time until when digital evidence can be presented, not to the time at which it must have been generated.

Another important point is that there is no need for *commonly* trusted authorities (except, of course, for the legal system itself). On one hand, the recipient B of A's digital signature must trust the digital statement by an authority C confirming that C holds A's physical commitment declaration. On the other hand, A relies on and must trust those entities that issue delegate signatures or otherwise are needed to contribute to the generation of sufficient digital evidence s required to satisfy the verification predicate. Since A can choose these entities she wants to trust rather than being forced to trust certain entities (e.g., a time-stamping service), the design freedom of evidence systems is substantially higher than generally believed.

We believe that the design of digital evidence systems and legislation needs to be revised in several respects. Based on the new interpretation of digital evidence and the concept of digital declarations, better solutions will hopefully be developed, for various application contexts that become relevant in the future.

The open problems include the analysis and formalization of such settings as well as the extension of the framework of which only an initial version was described in the paper. For example, an interesting and previously ignored context is that cryptographic schemes, for instance, the hash function or the signature scheme, need to be upgraded. This changes the predicates. How can one specify the predicates to allow for such upgrades?

We also propose the development of new business models for trusted services as an interesting open problem.

Acknowledgment

The author would like to thank T. Berson, D. Chaum, A. Curiger, J. Doekbrijder, N. Edwards, C. Ellison, S. Haber, M. Hauber, A. Hiltgen, M. Hirt, R. Kohlas, W. Mao, L. Meier, R. Oppliger, K. Paterson, B. Pfitzmann, B. Preneel, R. Renner, R. Rivest, A. Shamir, J. Sjödin, P. van

[40]There is no cost for extra hardware as one would use the recording hardware of the same device that people are using for other purposes, for example a next-generation mobile phone.

Oorschot, and M. Waidner for interesting discussions on digital evidence or for feedback on an earlier draft of the paper, and the anonymous referees for helpful comments and suggestions.

REFERENCES

[1] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. 22, pp. 644–654, Nov. 1976.

[2] S. Haber and W. S. Stornetta, "How to time stamp a digital document," *J. Cryptology*, vol. 3, no. 2, pp. 99–111, 1991.

[3] R. Kohlas and U. Maurer, "Reasoning about public-key certification: on bindings between entities and public keys," in *Lecture Notes in Computer Science, Financial Cryptography*, M. Franklin, Ed. Heidelberg, Germany: Springer-Verlag, 1999, vol. 1648, pp. 86–103.

[4] U. Maurer, "Modeling a public-key infrastructure," in *Proc. 1996 Eur. Symp. Research in Computer Security (ESORICS' 96)*, E. Bertino, Ed., pp. 325–350.

[5] ——, "Intrinsic limitations of digital signatures and how to cope with them," in *Lecture Notes in Computer Science, Information Security*, C. Boyd and W. Mao, Eds., 2003, vol. 2851, pp. 180–192.

[6] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 1997.

[7] R. Oppliger and D. Rytz, "Digital evidence: Dream and reality," *IEEE Security Privacy*, vol. 1, pp. 44–48, Sept.–Oct. 2003.

[8] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[9] B. Schneier, *Applied Cryptography*, 2nd ed. New York: Wiley, 1996.

[10] Finread Initiative [Online]. Available: http://www.finread.com

[11] SPKI/SDSI certificates [Online]. Available: http://world.std.com/~cme/html/spki.html

[12] [Online]. Available: http://rechten.uvt.nl/simone/ds-lawsu.htm

Ueli Maurer (Fellow, IEEE) was born in St. Gallen, Switzerland, in 1960. He received the Diploma degree in electrical engineering and the Ph.D. degree in technical sciences from Swiss Federal Institute of Technology (ETH) Zurich, Zurich, Switzerland, in 1985 and 1990, respectively.

From 1990 to 1991, he was DIMACS Research Fellow at the Department of Computer Science at Princeton University. In 1992, he joined the Department of Computer Science, ETH Zurich, where he is currently Professor of computer science and Head of the Information Security and Cryptography Research Group. He is also a Cofounder of Seclutions, a Zurich-based security software company. He is Editor-in-Chief of the *Journal of Cryptology* and Editor-in-Chief (with R. Rivest) of Springer-Verlag's book series *Information Security and Cryptography*. He holds several patents for cryptographic systems and has worked as a consultant for many companies and government organizations. His research interests include information security, the theory and applications of cryptography, information theory, theoretical computer science, and discrete mathematics.

Dr. Maurer is on the Board of Directors of the International Association for Cryptologic Research (IACR). He also serves on several management and scientific advisory boards. He was previously on the Editorial Board of the IEEE TRANSACTIONS ON INFORMATION THEORY.