

An Everlasting Secure Non-interactive Timestamping Scheme in the Bounded Storage Model

Assia Ben Shil^{1,3}(✉) and Kaouther Blibech Sinaoui^{2,3}

¹ Faculty of Sciences of Bizerte, University of Carthage, Tunis, Tunisia
`essia.benshil@gmail.com`

² ISTEUB, University of Carthage, Tunis, Tunisia
`kaouther.blibech@gmail.com`

³ LIP2 Laboratory, Tunis, Tunisia

Abstract. Digital timestamping is a cryptographic technique allowing to affix a reliable date to a digital document in order to prove that it exists and its integrity is kept since this date. However, there is a good chance that a lot of current timestamping systems will not be secure in the coming years. In fact, the security of most of the existing timestamping systems is based on the security of the used cryptographic techniques as hash functions. However, a hash function has a limited lifetime. In this context, we provide a non-interactive timestamping scheme in the bounded storage model (BSM). In this model, we assume that an adversary has a limited memory but his computing power can be unlimited. Thus, the security of our timestamping scheme does not depend on the lifetime of any cryptographic technique. We prove, in fact, that our timestamping scheme is eternally secure even against an adversary with unlimited computing power.

Keywords: Timestamping · Bounded storage model · Computing power · Eternal security

1 Introduction

Timestamping [16] is an important technique for proving the existence of digital documents and detecting their alterations. It is also important for the long-term preservation of some cryptographic tools as encryption keys and digital signatures.

Generally, a timestamping system provides a proof-of-existence of a document. This proof is called timestamp. To provide a timestamp for a given document there are two phases: a timestamping phase, and a verification phase. The timestamping phase allows one or more Timestamping Authority (TSA) to affix a reliable date to a document and generate the associated timestamp using a timestamping protocol. The verification phase allows any verifier to verify that the document was correctly timestamped by the TSA with respect to the used timestamping protocol.

Most of the existing timestamping systems [2, 3, 7–14, 16] are secure under the assumption that the cryptographic techniques they use are secure. However, with the evolution of computing power, there is a good chance that what is secure today will not be secure some years later. Therefore, since most of the existing schemes are based on some hash functions they assume to be one-way and without collisions, they cannot be considered secure forever. In fact, hash functions are not secure all the time [21].

To provide timestamping systems producing timestamps whose validity cannot be challenged, we decided to place our research work within the Bounded Storage Model (BSM) [17, 18]. In this model, instead of assuming that the users' computing power is limited, Maurer assumes that their memory is limited. Thus, the ciphers in this model are eternally secure [1].

The idea is that a very long random string called randomizer is transmitted at every round t ¹. When the randomizer is transmitted, no participant has sufficient storage space to entirely store it. Even if later their storage space becomes sufficient to entirely store the randomizer transmitted at the round t , the users would have already lost the access to this string. Thus, the performed encryptions are always valid.

In [19], Moran proved that timestamping is possible in the BSM and proposed a non-interactive timestamping scheme in this model. In this scheme, each stamper can timestamp his document locally without communicating with any other party [19]. Thus, the non-interactive timestamping ensures the confidentiality of the timestamped document and hides even the fact that a timestamping occurred. For these reasons, we consider that the system of Moran is very interesting, but it suffers from a lack of precision and practical details. In this context, we presented in [4] a non-interactive timestamping scheme in the BSM. In this paper, we provide a formal representation of our timestamping scheme and we prove that it is eternally secure.

This paper is organized as follows: In the following section, we present the existing timestamping systems and their weaknesses. Then, we introduce the BSM and we present Moran's timestamping system. In Sect. 5, we present Shamir's secret sharing scheme. Then, we present our non-interactive timestamping scheme [4] based on Shamir's secret sharing scheme. In Sect. 7, we detail more formally our timestamping scheme [5]². Finally, we prove the eternal security of our timestamping solution [5].

2 Existing Timestamping Schemes and their Weaknesses

There are two classes of timestamping systems described in [10]:

Simple timestamping systems [16] are centralized systems requiring a single TimeStamping Authority (TSA) and having an absolute trust in the TSA.

¹ A round t is the interval between the time t and the time $t + 1$.

² Notice that [5] is a short version of this paper that has been published in FMS: the Formal Methods for Security Workshop co-located with the PetriNets-2014 Conference.

In fact, the TSA receives a document's hash value and sends to the stamper a timestamp created within a dedicated timestamping protocol. In such a system, we cannot verify the timestamp reliability.

Secure timestamping systems are systems reducing the trust in the TSA. In fact, the latter has to provide a timestamping correctness proof. There are two kinds of secure timestamping systems:

- Centralized secure timestamping systems [2, 3, 7–10, 13, 14] requiring a single authority which has to prove the correctness of the generated timestamps. There are many techniques for doing that. For example, linking schemes [10] are used to link a set of timestamps. These links can be used as proofs at verification time.
- Distributed secure timestamping systems [11, 12] are based on duplication. In fact, each request is timestamped by n independent TSAs. Each TSA returns a timestamp fragment that the stamper uses to create the final timestamp.

The above timestamping systems are all based on some cryptographic techniques as digital signatures and hash functions. However, the lifetime of digital signatures is limited. In addition, if a document must be securely represented by its hash value, the used hash function has to be irreversible and collision-resistant. However, these properties cannot be proven for practical hash functions and many ones have been broken [21] (e.g. SHA-0³ in 2004 and SHA-1⁴ and MD5⁵ in 2005, which have been suggested as practical hash functions in several timestamping schemes).

To overcome the problems of the existing timestamping systems, we propose to adopt the secret sharing techniques [6, 22] whose security does not rely on the lifetime of any cryptographic technique. In [4], we provided a timestamping scheme in the bounded storage model. This scheme is based on Shamir's secret sharing scheme [22]. The aim of this paper is to prove the security of the mentioned timestamping scheme. In the following section, we introduce the bounded storage model since our timestamping scheme is conceived in this model.

3 The Bounded Storage Model

Generally, in cryptography, the proposed systems and the used functions are secure under the assumption that it exists a limit for the computational power of any user or adversary. However, there is no limit for his storage space. Thus, he can be able to store the whole of a transmitted ciphertext and decrypt it later when his computing power increases. In the bounded storage model, the proposed systems must be secure even against an adversary with an unlimited computational power. In fact, the adversary has a limited memory and is, consequently, not able to store all the transmitted ciphertext to try to decrypt it later.

³ Secure Hash Algorithm 0 published in 1993.

⁴ Secure Hash Algorithm 1 published in 1995.

⁵ Message-Digest Algorithm 5 published in 1992.

The BSM was proposed by Maurer in 1992 [17] for the development of encryption keys. It aims to generate, from a short secret key K , a key with a larger size X [15] that can be used as encryption key. The system operates as follows: In this model, it is assumed that the storage capacity is limited, and there is no assumption about the computing power. Let s be the assumed limit on a user's storage capacity. Ciphers in the BSM use a very long string R called randomizer. The latter may for example be a random sequence of bits transmitted by a satellite. If R is a random string of r bits, the space of R is $\{0, 1\}^r$. Notice that " $r \gg s$ " is required to ensure that no user can fully store R . Having a secret key K of size k in the space $\{0, 1\}^k$, we can use a known function F to generate the derived key $X = F(R, K)$ of size x bits. The function F must use only a small part of R so that we do not need to fully read R .

Example: Suppose that the adversary storage capacity is $s = 10^{15}$ bits, Alice and Bob share a secret key of 6000 bits and they (with the adversary Eve) access to a random source emitting 100 Gigabits per second (10^{11} bits/s) for about one day and a half ($1, 25 \cdot 10^{16}$ s), then they can derive a key length of 10 Gigabits ($x = 10^{10}$) and the adversary has absolutely no information about it. Alice and Bob do only need to read 10^{12} bits of the random source.

Maurer's system has been the subject of intensive studies. Indeed, many key generation systems have been proposed in the BSM [18]. The BSM was used for timestamping by Moran in [19]. In [4], we proposed an improvement of Moran's timestamping system. In the following section, we present the timestamping system of Moran and then our proposition is presented.

4 Timestamping Solutions in the BSM

In the BSM, we assume that a long random string R of r bits is transmitted during the round t (between t and $t + 1$). If s is the maximum storage capacity of any entity in t , then $s \ll r$. Notice the space of R is $\{0, 1\}^r$ where r is the size of the string R . Similarly, we consider that a document D of size d bits has a value in $\{0, 1\}^d$. In the timestamping scheme proposed by Moran, to timestamp a document D , its content is used to select a few blocks from R whose values will be inserted in the timestamp T of D .

For example, in practice, we can suppose that a randomizer emits at a fixed rate about 100 Gigabits/s (10^{11} bits/s) during a round of approximatively an hour (3600s). Therefore the maximum size r of the string transmitted during a round t is about $r = 3600 \cdot 10^{11}$ bits. This string will be divided into blocks of equal size in order to index the string. The block size is chosen so that its maximum value is adapted to the usual arithmetic calculations. Thus, we can define a maximum number of blocks for the transmitted string. For example, we can use blocks of 3072 bits.

On the other hand, any verifier must save randomly some blocks of R (using a function named $Sketch(R)$) in order to verify, later, the validity of any timestamp made during the round t . Verifying the validity of a timestamp associated with a document D is performed at a later date by a verifier who has simply to

check that there are no conflicts between his sketch and the timestamp of D . However, in this solution, a timestamp includes some additional values of R . If the verifier cannot store during the round t more than s bits of R , he may at the verification time store s' with $s' > s$. Each verification of a timestamp generated in the round t can lead him to discover new blocks of R . After a number of verification processes, he can reconstruct partially, if not entirely R and backdate any document.

To overcome this problem, we propose a timestamping protocol allowing to verify the timestamp of a document D without learning additional values of R . To this aim, instead of inserting the blocks of R in the timestamp, these blocks will be the secret of the stamper. The verifier must then prove that he has the value of a given block in his sketch and the stamper has to prove that he has used this value to create the timestamp. In the verification process of this scheme, a verifier may accept or reject a timestamp without discovering any additional information about the string R transmitted during the round t .

Our timestamping scheme is presented with more details in the following sections. But, first of all, we will present Shamir's secret sharing scheme [22].

5 Shamir's Secret Sharing Scheme

Secret sharing allows sharing a secret S among the set of n participants such that a coalition of the n participants is required to determine the secret S . Threshold secret sharing [6, 22] allows sharing a secret S among a set of n participants such that a coalition of at most $k-1$ participants cannot determine the secret S , while any coalition of at least k participants can find out S .

Shamir's secret sharing scheme [22] is a k -out-of- n threshold secret sharing scheme based on polynomial interpolation. Shamir's secret sharing scheme is the most used one and most of the proposed secret sharing schemes are based on it. In this scheme, to share a secret among n participants P_1, P_2, \dots, P_n , the dealer:

- Sets the polynomial: $f(x) = S + \sum_{j=1}^{k-1} f_j x^j \bmod p$ where p is a prime ($p > n+1$) and the f_j are values chosen randomly and secretly in Z_p . He randomly selects n distinct elements of Z_p denoted x_i for $i = 1 \dots n$.
- For $i = 1 \dots n$, he secretly sends the point $(x_i, f(x_i))$ to the participant P_i .

Given only k points $(x_i, f(x_i))$, with x_i distinct values, there is only one polynomial $f(x)$ of degree $k-1$, which graphic representation passes through all these points. This polynomial can be effectively obtained from the points $(x_i, f(x_i))$. We can use the following interpolation formula for that purpose:

$$f(x) = \sum_{i=1}^k [f(x_i) \prod_{j=1, j \neq i}^k [(x - x_j)/(x_i - x_j)]]$$

Then, we compute the secret S as follows $S = f(0)$.

Notice that, this scheme is information theoretically secure. In fact, at least k participants have to pool their shares in order to discover the secret S . In the following section, we will describe our timestamping scheme conceived on the BSM and based on Shamir's secret sharing scheme.

6 Our Timestamping Scheme

The idea is to timestamp the document D locally using the secret sharing scheme of Shamir [22] in order to divide D into n shares D_i . Notice that this set of coefficients f_i of the polynomial f used to split D into n shares is public ($1 \leq i \leq k-1$)⁶. So, the stamper of a document D has to consider that D is his secret and compute the set of shares $D_i = f(i)$ for $1 \leq i \leq n$. Note $Share(D)$ the set of shares D_i . Assuming that the blocks of R are indexed by a number beginning from 1 till the number of blocks, the values R_{D_i} , indexed by the shares D_i for $1 \leq i \leq n$ are recovered. The polynomial that passes through the points $P_i(R_{D_i}, D_i)$ for $1 \leq i \leq n$ represents the timestamp of D . Note here that we choose to use the polynomial that passes through the points $P_i(R_{D_i}, D_i)$ and not the polynomial that passes through the points $P_i(D_i, R_{D_i})$. The idea is to hide the set of couples (index, value) of R used by the stamper. The fact of using a polynomial that passes through the "reverses" points allows to hide this association. This process is described in Fig. 1.

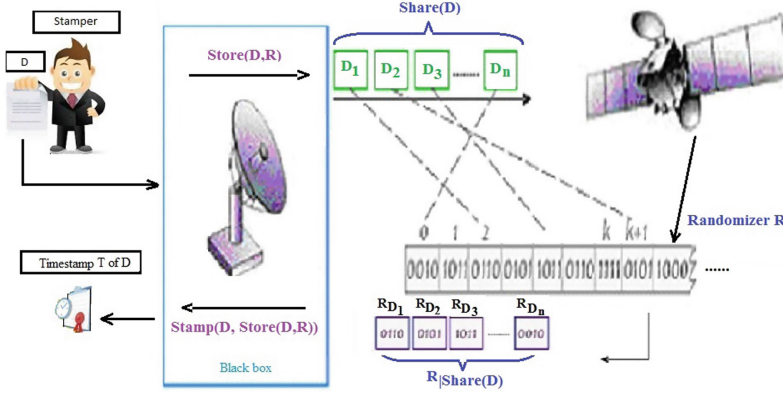


Fig. 1. Timestamping process

Any user of the system saves a random subset of R named $Sketch(R)$ formed by a number of couples of values (index of block, value of the block). This process is described in Fig. 2.

To verify the validity of the timestamp T of D for a random string R , the verifier who stored $Sketch(R)$ proceeds as follows: For each index in both $Share(D)$

⁶ The fact that the set of f_i is public allows us to detect some kinds of attacks (see Theorem 3).

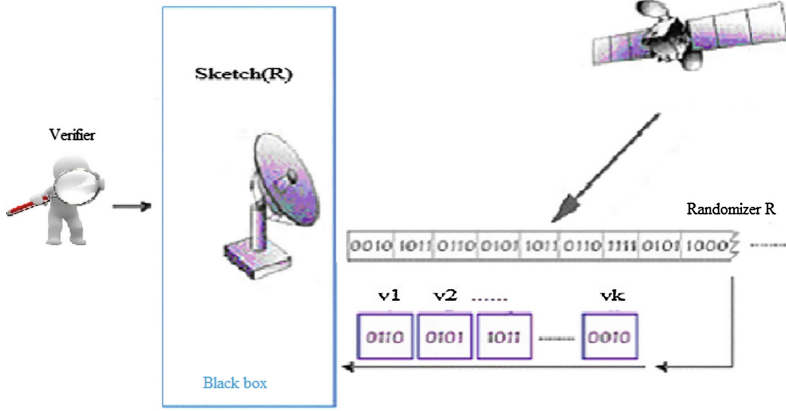


Fig. 2. Storage of a sketch of R

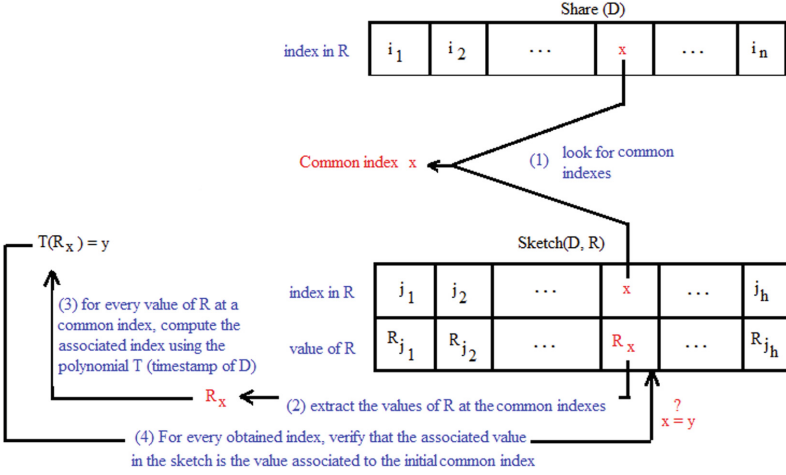


Fig. 3. Verification process

and $Sketch(R)$ he extracts the associated value R_i from $Sketch(R)$, computes its associated index by the polynomial given in the timestamp and checks that the computed index and the one of $Sketch(R)$ match. The verification process is described in Fig. 3.

In the following section we provide a more formal description of our scheme.

7 Formal Description of Our Timestamping Scheme

In the following sections we provide some definitions and notations and then we use them in order to prove the security of our timestamping scheme.

7.1 Definitions and Notations

In this section, we introduce some notations that we will use later in this paper.

- Randomizer: We call “randomizer” and we denote R the random string of size r transmitted during a round t . This string is divided into N blocks of size b bits indexed from 1 to N and denoted R_1, \dots, R_N . Thus, $r = N.b$. For each subset $S \subseteq I(R)$ where $I(R)$ is the set of indexes of R_i blocks ($1 \leq i \leq N$), we denote $R|_S$ the set of blocks of $R \forall i \in S$.
- Hamming Distance: A vector being a set of blocks, we define the Hamming Distance between two vectors V_1 and V_2 denoted DH as the number of blocks for which the two vectors differ.
- Threshold secret sharing: The threshold secret sharing is a technique for dividing a secret S into n shares such that the coalition of at least k shares is necessary to reconstruct the secret while the coalition of at most $k - 1$ shares does not reveal even partially the secret ($k < n$). A k -out of- n threshold secret sharing scheme is denoted $SSS(k, n)$.
- Shamir’s secret sharing scheme: The secret sharing scheme based on Shamir’s polynomial interpolation scheme is a $SSS(k, n)$. The principle is to fix a polynomial f of degree $k - 1$ and X a set of values ($X = X_1, \dots, X_n$). The secret sharing function denoted $Share(S)$ takes as input the secret S and generates the set of shares $Share(S) = (S_1, \dots, S_n)$ such that $\forall i, 1 \leq i \leq n, S_i = f(X_i)$. The Reconstruction function denoted $Share^{-1}$ takes as input a subset of X denoted $X_S = [X_{S_1}, \dots, X_{S_k}]$ such that $|X_S| = k$ and the k associated shares: $Share^{-1}(X_S, S_{X_{S_1}}, \dots, S_{X_{S_k}}) = f$, with f a polynomial such that $\forall X_i \in X_S$ and $S_i \in [S_{X_{S_1}}, \dots, S_{X_{S_k}}], f(X_i) = S_i$. The secret S is computed as follows: $S = P(0)$.

7.2 Presentation of Our Timestamping Scheme

Timestamping Phase. A “stamper” is represented by the two following functions:

- $Store(D, R)$ that uses Shamir’s secret sharing process to compute $Share(D)$ for a given document D . Then, It computes the vector $R|_{Share(D)}$ and stores it. More formally, $Store(D, R)$ consists in computing $Share(D) = (D_1, \dots, D_n)$, where D_i is the i^{th} index specified by D and the vector $(R_{D_1}, \dots, R_{D_n})$ where R_{D_i} is the block of R indexed by D_i . We call this vector $R|_{Share(D)}$, where the notation $R|_I$ means the values of blocks of R indexed by I_1, \dots, I_n , with $I = I_1, \dots, I_n$.

Definition 1. $Store(D, R) = R|_{Share(D)}$.

- $Stamp(D, Store(D, R))$ applies the interpolation formula to find the unique polynomial passing through the points $P_i(x, y)$ where x is a block of the vector $R|_{Share(D)}$ and y the associated index in $Share(D)$. This polynomial is the timestamp T .

Definition 2. $T = Share^{-1}(R|_{Share(D)}, Share(D)) = Stamp(D, Store(D, R))$.

Verification Phase. A “verifier” is represented by the two following functions:

- $Sketch(R)$ allows choosing, randomly, a set of indexes denoted H with $H \subset I(R)$, computing $R_{|H}$ and storing this vector.

Definition 3. $Sketch(R) = (H, R_{|H})$.

- $Verify(Sketch(R), D, T)$ allows to verify that there are no conflicts between $Sketch(R)$ and T (which is the timestamp of D , namely a polynomial).

Definition 4. $Verify(Sketch(R), D, T) = \text{success only if}$

$$DH(T(R_{|H} \cap R_{|Share(D)}, H \cap Share(D))) = 0$$

for a document D and a timestamp T .

The Behavior of an Adversary. An “adversary” consists in the two following functions:

- $Store^*(R)$ which saves a subset of R called C . The difference between $Sketch(R)$ and $Store^*(R)$ is that $Sketch(R)$ is computed randomly and “on line” while $Store^*(R)$ function may not be.
- $Stamp^*(D, C)$ that given a document D and a string C tries to produce a timestamp T^* of D .

Definition 5. $Stamp^*(D, C) = T^* = Share^{-1}(R_{|Share(D)}, Share(D))$.

Where $R_{|Share(D)}$ is the vector of blocks associated with $Share(D)$ according to T^* . If an adversary A produces for a document D and a randomizer R , a timestamp T^* that is equal to the timestamp T produced by $Stamp(D, Store(D, R))$, we say that he backdates “correctly” the document. More formally:

Definition 6. An adversary backdates a document D with a success probability γ for a given randomizer R if:

$$Pr[Verify(Sketch(R), D, Stamp^*(D, Store^*(R))) = \text{success}] \geq \gamma.$$

Definition 7. An adversary backdates “correctly” a document D for a given randomizer R if $DH(V_1, V_2) = 0$ with $V_1 = R_{|Share(D)}^*$ and $V_2 = R_{|Share(D)}$.

Definition 8. An adversary backdates “correctly” a document D for a given randomizer R with at most err errors, if $DH(V_1, V_2) \leq err$, with $V_1 = R_{|Share(D)}^*$ and $V_2 = R_{|Share(D)}$.

8 Security Proofs of Our Timestamping Scheme

Given the following parameters:

- s : the storage capacity of the most powerful adversary.
- r : the size of the random string R transmitted during a round t .
- b : the size of a block of the random string R transmitted during a round t .
- n : the number of indexes specified by a given document.
- N : the number of blocks of the random string R transmitted during a round t .
- $|H|$: the number of blocks of a sketch.

We assume that:

- (1) $r \gg s$: The size of the random string R transmitted during a round t is greater than the storage capacity of the most powerful user of the system.
- (2) $1 < N/|H|$: The number of blocks of R transmitted during a round t is greater than the number of blocks in a sketch saved by a potential user.
- (3) $2^b \gg r/b$: The number of possible values for a block of size b bits is greater than the number of blocks of the string R transmitted during a round t .
- (4) $r/b > n$: The number of blocks of the string R transmitted during a round t is greater than the number of shares used in the adopted Shamir’s secret sharing scheme.
- (5) $b \gg 1$: The size of a block of R is much greater than 1 bit.

In our security study we demonstrate mainly two important characteristics of our non-interactive timestamping scheme. First, we prove that backdating documents in our timestamping scheme has a negligible⁷ probability. Second, we prove that the timestamps provided by our timestamping scheme have an eternal validity.

8.1 Negligible Probability of Backdating Documents

In our timestamping scheme, the probability that an adversary backdates a document D for a string R already transmitted using his stored blocks of R is negligible. This proof is established in two steps. In the first step, we show that if an adversary A wants to backdate “successfully” a document D for a random string R , then he must backdate it “correctly” for this string R with an error err negligible. In the second step, we show that the probability of backdating “correctly” a document D for a random string R is negligible.

First Step. Given a correct timestamp T and a forged timestamp T^* produced by an adversary A for the document D and the random string R , A succeeds in backdating D for R if the vector of blocks associated with $Share(D)$ according to T^* denoted $R^*_{|Share(D)}$ is close in Hamming distance to $R_{|Share(D)}$. In this

⁷ An event that occurs with a negligible probability can be safely ignored.

case, we say that the adversary backdates “correctly” the document D for the random string R with err errors, where err is an integer very close to zero. More formally, let A be an adversary. Denote $R_{successful(D)} = R_{\gamma_{successful(D)}}^{\gamma}$ the set of strings R for which A has the necessary storage to try to backdate the document D with a probability of success greater than γ .

Lemma 1 ([20]). *If an adversary backdates a document D for a random string R with a probability of success γ then he backdates it “correctly” with at most $(N/|H|)\ln(1/\gamma)$ errors.*

Proof. Let us suppose that the adversary provides a timestamp for the document D for the string R such that the timestamp is made with $err^* > err$ incorrect indexes. Denote $INCORRECT(D, R)$ the set of incorrect indexes for D and R . If $H \cap INCORRECT(D, R) \neq \emptyset$ the verifier will reject the timestamp of the adversary.

Let i be an index of H , the probability that i be in $INCORRECT(D, R)$ is:

$$Pr[i \in INCORRECT(D, R)] = err^*/N.$$

The probability that i does not belong to $INCORRECT(D, R)$ is

$$1 - Pr[i \in INCORRECT(D, R)] = 1 - err^*/N.$$

The probability that all the elements of $|H|$ do not belong to $INCORRECT(D, R)$ is:

$$Pr[\forall i \in H, i \notin INCORRECT(D, R)] = (1 - err^*/N)^{|H|} \leq e^{-(err^*|H|)/N}.$$

If an adversary backdates a document D for a random string R with a probability of success $\gamma \leq e^{-(err^*|H|)/N} \leq e^{-(err|H|)/N}$ then he backdates it correctly with at most $err \leq (N/|H|)\ln(1/\gamma)$. \square

Denote $R_{correct}(D)$ the set of strings R for which A can “correctly” backdate D with at most $(N/|H|)\ln(1/\gamma)$ errors.

Theorem 1. *If $err \leq (N/|H|)\ln((1/\gamma))$ then, $R_{successful(D)}$ is a subset of $R_{correct}(D)$ [20].*

Proof. According to Lemma 1, if an adversary backdates a document D for a random string R with a probability of success γ then he backdates it correctly with at most $err \leq (N/|H|)\ln(1/\gamma)$. So, if a random string R belongs to $R_{successful(D)}$ then it belongs to $R_{correct}(D)$. Thus, $R_{successful(D)} \subseteq R_{correct}(D)$. In addition, the more the probability of success γ becomes close to 1, the more this error becomes close to 0. So, successfully backdating means “correctly” backdating with a “negligible” error. We prove, in the second step, that the probability that the random string R chosen by the adversary to backdate a document D be in $R_{correct}(D)$ is negligible. \square

Second Step. We now prove that for an adversary A , a document D and a string R : $\Pr[R \in R_{\text{correct}}(D)]$ is negligible.

Theorem 2. *If $n \gg 1$ and $b \gg 1$ then $\Pr[R \in R_{\text{correct}}(D)]$ is negligible, with b the size of a block of R and n the number of indexes in $\text{Share}(D)$.*

Proof. We proved in the first step, that if an adversary backdates a document “successfully”, he backdated it “correctly” with at most a negligible error. Then we proved in the second stage that the probability that the string R for which the adversary tries to backdate the document D be in $R_{\text{correct}}(D)$ is negligible.

In fact, knowing that the size of blocks of a random string R is b and the number of these blocks is N , the number of possible random strings is $(2^b)^N$.

Moreover, to backdate a document D , the adversary has to create a timestamp T^* for D such that at least $n - \text{err}$ blocks of R indexed by D are used to generate T^* .

In other words, he can try to backdate D only for random strings for which he knows the values of at least $n - \text{err}$ blocks from the n blocks indexed by D . Thus, since the adversary tries to correctly backdate D with at most err errors, the number of random strings he can use is at most $(2^b)^{N-n}$.

So, the probability that a random string R belongs to $R_{\text{correct}}(D)$ is:

$\Pr[R \in R_{\text{correct}}(D)] \leq (2^b)^{N-n} / (2^b)^N \leq 1/2^{nb}$. Since $n \gg 1$ and $b \gg 1$, this probability is negligible. \square

Theorem 3. *Given a sketch H , an adversary is not able to find a document D^* which is correctly timestamped within our timestamping system.*

Proof. If an adversary owns a sketch H , he could use the interpolation formula on H in order to find the polynomial that associated the blocks of H with their indexes. He has then a forged timestamp T^* . What he has to do later is to find a document D^* whose shares match those present in H and used to compute T^* . To do that, he has to use the interpolation formula to find a polynomial that matches a document D^* to H . But, since the sharing polynomial, in our scheme, is fixed and public he will not be able to correctly timestamp D^* . \square

8.2 The Eternal Security of Our Timestamping Scheme

In [20], Moran proves that in his non-interactive timestamping scheme, an adversary with a storage s can easily backdate $\delta = s/|T|$ documents by running the timestamping process on some δ documents and storing the generated timestamps (each of them has length of at most $|T|$). However, the probability that an adversary backdates more than δ documents is negligible. We show here that, in our timestamping scheme, after the transmission of R , it is very difficult to forge a fake timestamp for a given document. Moreover, we show that an adversary having a document D and δ correct timestamps can forge a fake timestamp for D only with a negligible probability. Thus, we prove the following theorem:

Theorem 4. *If $2^b \gg r/b$ and $r/b > n$ then the probability to forge a fake timestamp for a document D and a string R using δ correct timestamps related to R is negligible.*

Proof. The inequality $2^b \gg r/b$ means that the number of values for a block of size b bits is greater than the number of blocks of the string R transmitted during a round t .

n is the number of indexes specified by a given document, this number must always be less than the number of blocks of R . So, $r/b > n$.

It follows that $2^b \gg n$, which means that the number of possible values for a block of R is much greater than the number of indexes specified by the document D .

For each timestamp T_j ($1 \leq j \leq \delta$), if the adversary gives any value v from the 2^b possible values of a block of R , it will recover a given index i .

However, the fact that $i = T_j(v)$ does not mean that $R_i = v$. This means that i is the value associated with v by the polynomial T_j but the couple (i, v) does not necessarily belong to the string R . In other words, the string R may not associate the value v to the block indexed by i . Moreover, it may exist i' such that $R_{i'} = v$ and there is no way to verify if $i = i'$. The only points of T_j for which the adversary knows that they belong to R are the points whose indexes are specified by the document associated with T_j . The probability that the adversary chooses one of these points is $n/2^b \ll 1$.

To obtain the k points required to forge a fake timestamp, the probability is negligible since it is the product of the probabilities of selecting each of the points belonging to a valid timestamp.

So, the adversary can obtain the k points needed to forge a fake timestamp for a document D for a random string R with a negligible probability. \square

Thus, we proved that our non-interactive timestamping system is secure and that it produces timestamps with eternal validity. As improvement to our system, we can provide a multi-documents timestamping system. The idea is simple. To timestamp a set of n documents, we have just to give an index to every document. The couple (index, document) will then be represented as a point. By interpolation, we find out the unique polynomial passing through the set of points (eg. the set of documents). We sample the set of n other points of the obtained polynomial. These points will represent the shares of the set of documents to timestamp. Finally, we apply our timestamping system as presented in Sect. 6 of this paper. Thus, we reduce the cost of timestamping n documents one by one and we obtain shares and timestamps having the same size as in the timestamping system described in Sect. 6 of this paper.

9 Conclusion

In this paper, we have formally presented a non-interactive timestamping solution in the bounded storage model. Our solution is non-interactive and hides even the fact that a timestamping occurred. It also ensures total confidentiality of the provided timestamps. In addition, our solution provides eternal security

for the provided timestamps. In fact, neither increasing the storage capacity of an adversary or the evolution of his computing power will compromise a provided timestamp. Thus, our solution is more secure than existing systems whose timestamps can be challenged when the computing power or storage capacity of users increases. In this context, we studied the security of our solution and proved that the possibility of cheating is negligible. In our future works, we plan to adopt new secret sharing schemes for timestamping.

References

1. Aumann, Y., Ding, Y.Z., Rabin, M.O.: Everlasting security in the bounded storage model. *IEEE Trans. Inf. Theory* **48**(6), 1668–1680 (2002)
2. Bayer, D., Haber, S., Stornetta, W.S.: Improving the efficiency and reliability of digital timestamping. In: Capocelli, R., Santis, A.D., Vaccaro, U. (eds.) *Sequences91: Methods in Communication, Security and Computer Science*, pp. 329–334. Springer, New York (1992)
3. Benaloh, J.C., de Mare, M.: One-Way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) *EUROCRYPT 1993*. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)
4. Ben Shil, A., Blibech, K., Robbana, R.: A new timestamping schema in the bounded storage model. In: *Proceedings of the 3rd Conference on Risks and Security of Internet and Systems, CRISIS* (2008)
5. Ben Shil, A., Blibech, K., Robbana, R.: A timestamping scheme with eternal security in the bounded storage model. In: *Proceedings of the Formal Methods for Security Workshop co-located with the PetriNets-2014 Conference* (2014)
6. Blakley, G.R.: Safeguarding cryptographic keys, *International Workshop on Managing Requirements Knowledge.*, 313–317 (1979)
7. Blibech, K., Gabillon, A.: A new timestamping scheme based on skip lists. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) *ICCSA 2006*. LNCS, vol. 3982, pp. 395–405. Springer, Heidelberg (2006)
8. Blibech, K., Gabillon, A.: A new totally ordered timestamping scheme. In: *5th Conference on Security and Network Architectures SAR* (2006)
9. Blibech, K., Gabillon, A.: CHRONOS: an authenticated dictionary based on skip lists for timestamping systems. In: *SWS*, pp. 84–90 (2005)
10. Blibech, K., Gabillon, A., Bonnetaze, A.: Etude des systèmes d’horodatage. *Tech. et Sci. Informatiques* **26**(3–4), 249–278 (2007)
11. Bonnetaze, A., Liardet, P., Gabillon, A., Blibech, K.: A distributed time stamping scheme, *4th Conference on Security and Network Architectures SAR 2005* (2005)
12. Bonnetaze, A., Trebuchet, P.: Threshold signature for distributed time stamping scheme. *Ann. des Telecommun.* **62**(11–12), 1353–1364 (2007)
13. Buldas, A., Laud, P., Lipmaa, H., Vilemson, J.: Time-stamping with binary linking schemes. In: Krawczyk, H. (ed.) *CRYPTO 1998*. LNCS, vol. 1462, p. 486. Springer, Heidelberg (1998)
14. Budas, A., Laud, P.R., Schoenmakers, B.: Optimally efficient accountable timestamping. In: *Public Key Cryptography* (2000)
15. Dziembowski, S., Maurer, U.M.: On generating the initial key in the bounded-storage model. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 126–137. Springer, Heidelberg (2004)

16. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. *J. Cryptology* **3**(2), 99–111 (1991)
17. Maurer, U.: Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptology* **5**(1), 53–66 (1992)
18. Maurer, U.: Secret key agreement by public discussion. *IEEE Trans. Inf. Theory* **39**, 733–742 (1993)
19. Moran, T., Shaltiel, R., Ta-Shma, A.: Non-interactive timestamping in the bounded storage model. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, pp. 460–476. Springer, Heidelberg (2004)
20. Moran, T., Shaltiel, R., Ta-Shma, A.: Non-interactive timestamping in the bounded storage model. *J. Cryptology* **22**(2), 189–226 (2009)
21. National Institute of Standards and Technology (NIST), Announcement of Weakness in the Secure Hash Standard, Technical report (1994)
22. Shamir, A.: How to share a secret. *ACM Commun.* **22**(11), 612–613 (1979)

Transactions on Petri Nets and Other Models of
Concurrency X

Koutny, M.; Desel, J.; Haddad, S. (Eds.)

2015, XV, 177 p. 41 illus. in color., Softcover

ISBN: 978-3-662-48649-8