A Timestamping Scheme with Eternal Security in the Bounded Storage Model

Assia Ben Shil[1]
Faculty of Sciences of Bizerte
University of Carthage
Email: essia.benshil@gmail.com

and

Kaouther Blibech Sinaoui[2]
ISSAT of Mateur
University of Carthage
Email: kaouther.blibech@gmail.com

[1,2] LIP2 Laboratory, Tunis, Tunisia

Abstract. Digital timestamping is a cryptographic technique allowing affixing a reliable date to a digital document. The security of most 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) whose security is not related to the life 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

In the last years, digital documents are beginning to replace paper documents in several areas. For this reason, it is important to locate a digital document in time to prove its existence and integrity since a given date. This task is achieved through a timestamping system that operates in two phases: a timestamping phase allowing one or more Timestamping Authority (TSA) to affix a reliable date to a document and generate the associated timestamp and a verification phase allowing any potential verifier to verify the correctness of the produced timestamp.

The security of most existing timestamping systems [2-10] is based on the security of the used cryptographic techniques as hash functions. These techniques are secure under the assumption saying that the users' computing power

is limited. However, nowadays, the computing power of computers grows exponentially. Therefore, the used hash functions are not secure all the time [17]. So, the existing timestamping systems cannot be considered secure forever. To propose timestamping systems producing timestamps with eternal validity, we are placed in the Bounded Storage Model (BSM) [13, 14]. In this model, Maurer assumes that users' storage capacity is limited, while their computing power can be unlimited. In this model, the ciphers are eternally secure [1]. The principle is that a very long random string is transmitted in every round t (the interval between the time t and the time $t+1$). At the time of the transmission of this string, no participant has sufficient storage capacity to store it fully.Even if later, his storage capacity becomes sufficient to fully store the string transmitted in t, the user has already lost he access to this string. Thus, the performed encryptions are valid forever. Moran proposed in [15] a non-interactive timestamping scheme in the bounded storage model. Indeed, in the scheme of Moran, each stamper or document owner can timestamp his document locally without communicating with any third party [15], unlike conventional timestamping systems that involve at least one TSA. Thus, the non-interactive timestamping ensures total confidentiality and hides even the fact that a timestamping occurred. These characteristics have made the proposition of Moran very interesting, but the timestamping system he proposed suffers from a lack of precision and practical details. In this context, we present a non-interactive timestamping scheme in the BSM. This paper is organized as follows: After introducing the BSM in section 2, we present firstly Moran's timestamping system, and then our non-interactive timestamping scheme in the BSM in section 3. Then, we detail more formally our timestamping scheme. Finally, we formally prove the eternal security of our timestamping solution.

2 The Bounded Storage Model

Generally, in cryptography, the proposed systems and the used functions are secure under the assumption that there is a limit of the computational power of any user or adversary. In the bounded storage model, the proposed systems must be secure even against an adversary with an unlimited computational power. The BSM was proposed by Maurer in 1992 [13] for the development of encryption keys. It aims to generate, from a short secret key K, a key with a large size X [14] that can be used as encryption key. The system operates as follows: In this model, it is assumed that the storage capacity is unlimited, no assumptions about the computing power was made. 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 $f : \{0,1\}^r \times \{0,1\}^k$, $x \rightarrow \{0,1\}$ 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. Maurer's system

has been the subject of intensive studies. Indeed, many key generation systems have been proposed in the BSM [11]. The BSM was used for timestamping by Moran in [15]. In [3], 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.

3 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 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. 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 to a document D is performed at a later date by a verifier who has simply to verify 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 remedy this problem, we propose a timestamping protocol allowing verifying 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. The idea is to timestamp the document D locally using the secret sharing scheme of Shamir [18] in order to divide D into n 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 $Share(D)$ the set of shares D_i. 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). 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 $Share(D)$ and $Sketch(R)$ he recovers the associated block R_i of R, computes its associated index by the polynomial given in the timestamp and verifies that the associated value in $Sketch(R)$ is R_i. In the following sections,

we provide a formal representation of our timestamping system and we prove its eternal security. To this aim, we will show that once the string R is transmitted, the probability of forging a fake timestamp for a document D is close to zero.

4 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 transmitted during a round t. This string is divided into n blocks of size b bits indexed from 1 to n and denoted $R_1, \ldots, R_n$. Knowing that the length of R is r bits and the size of a block is b bits, $r = n.b$. For each subset $S \subseteq I(R)$ where $I(R)$ is the set of indexes of blocks R_i $(1 \leq i \leq n)$, 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 c_1 and c_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 l shares such that the coalition of at least λ shares is necessary to reconstruct the secret while the coalition of at most $\lambda - 1$ shares do not reveal even partially the secret $(\lambda < l)$. A λ-out of-l threshold secret sharing scheme is denoted $SSS(\lambda, l)$.

- Shamir secret sharing scheme: The secret sharing scheme based on Shamir's polynomial interpolation scheme is a $SSS(\lambda, l)$. We denote it SSSS (λ, l). The principle is to fix a polynomial P of degree $\lambda - 1$ and X a set of values $(X = X_1, \ldots, X_l)$. The secret sharing function denoted Share takes as input the secret S and generates the set of shares $Share(S) = (S_1, \ldots, S_l)$ such that $\forall$i, $1 \leq i \leq l$, $S_i = P(X_i)$. The Reconstruction function denoted $Share^{-1}$ take as input a subset of X denoted $X_S = [X_{S_1}, \ldots, X_{S_\lambda}]$ such that $|X_S| = \lambda$ and the λ associated shares: $Share^{-1}(X_S, S_{X_{S_1}}, \ldots, S_{X_{S_\lambda}}) = P$, with P a polynomial such that $\forall$ $X_i \in X_S$ and $S_i \in [S_{X_{S_1}}, \ldots, S_{X_{S_\lambda}}]$, $P(X_i) = S_i$. The secret S is computed as follows: $S = P(0)$.

5 Presentation of our Timestamping Scheme

5.1 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 $Share(D) = (D_1, \ldots, D_l)$, where D_i is the $i^{th} index specified by D. To the vector(R_{D_1}, \ldots, R_{D_l})$ is associated the vector $Share(D)$ where R_{D_i} is the block of R indexed by D_i. We denote this vector $R_{|Share(D)}$, where the notation $R_{|I}$ means the values of blocks of R indexed by $I_1, \ldots, I_n$, with $I = I_1, \ldots, I_n$.

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

- $Stamp(D, Store(D, R))$ uses Shamir's reconstruction process 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 blocks in $Share(D)$. This polynomial is the timestamp T.

Definition 2. $T = Share^{-1}(R_{|Share(D)}, Share(D))$.

5.2 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 verifying that there are no conflicts between $Sketch(R)$ and T.

Definition 4. *$Verify(Sketch(R), D, T)$ allows verifying the following equality :* $T(R_{|H \cap Share(D)}) = H \cap Share(D)$

In other words:

Definition 5. *$Verify(Sketch(R), D, T)$ allows verifying that* $DH(T(R_{|H} \cap Share(D)), H \cap Share(D)) = 0$ *for a timestamp T. If this equality is verified, the timestamp T is accepted by the verifier and is said "valid".*

5.3 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 "online" 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 6. $Stamp^*(D, C) = {R^*}_{|Share(D)}$.

Definition 7. $T^* = Share^{-1}({R^*}_{|Share(D)}, Share(D))$.

Where ${R^*}_{|Share(D)}$ is the vector of blocks associated to $Share(D)$ 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 8. *An adversary backdates a document D with a success probability γ for a given randomizer R if :* $Pr[Verify(Sketch(R), D, Stamp^*(D, Store^*(R)))] \geq \gamma$

Definition 9. *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 10. *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 $V_2 = R_{|Share(D)}$.*

6 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.
- l: the number of indexes specified by a given document.
- n: the number of blocks of the random string R transmitted during a round t.

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) $1 < n/|H| \ll l$: The first inequality means that the number of the blocks of R transmitted during a round t is greater than the number of blocks in a sketch saved by a potential user. The second inequality means that there exists an integer $u > 1$ such that $n = u.\ |H|$ with u much smaller than the number of indexes specified by a document.

(3) $2b \gg r/b$: 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) $r/b > l$: 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 negligible probability. Second, we prove that the timestamps provided by our timestamping scheme have an eternal validity.

6.1 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 If the adversary produces a timestamp of D such that the vector of blocks associated to $Share(D)$ according to this timestamp is far in Hamming distance from the vector of blocks associated to the timestamp produced by $Stamp(D, Store(D, R))$ then the verifier may with a high probability reject the timestamp of the adversary because the values indexed by $Share(D)$ according to the timestamp do not match the values indexed by $Share(D)$ according to his sketch. Given a correct timestamp T and a timestamp produced by an adversary A for the document D and the random string R denoted T^*, the adversary backdates the document D for R successfully, if he produces a timestamp T^* such that the vector of blocks associated to $Share(D)$ according to T^* denoted ${R^*}_{|Share(D)}$ is close in Hamming distance to $R_{|Share(D)}$. In this 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 $R_{successful(D)} = {R^\gamma}_{successful(D)}$ 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. *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. [16]*

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 indices. Denote $INCORRECT(D, R)$ the set of incorrect indices 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 indice 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:

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

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 $err \leq n/|H|.ln(1/\gamma)$.

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 $R_{correct}(D)$. [16]*

Proof. According to the 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, more the probability of success γ become close to 1, more this error become 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.

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

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

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 $R_{correct}(D)$ 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 $l - err$ 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 $l - err$ blocks from the l 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 $(2^b)^{n-l}$.

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

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

6.2 The Eternal Security of our Timestamping Scheme

In [16], Moran proves that in his non-interactive timestamping scheme, an adversary with a storage M can easily backdate $\delta = M/T$ documents by running the timestamping process on some k documents and storing the generated timestamps (each of which has length at most T). However, the probability that an adversary backdates more than k 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 3. *If $2^b \gg r/b$ and $r/b > l$ 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.

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

It follows that the $2^b \gg l$, which means that the number of possible values for a block of R is much greater than the number of indices 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 to 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 indices are specified by the document associated to T_j. The probability that the adversary chooses one of these points is $l/2^b \ll 1$.

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.

Thus, 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.

7 Conclusion

In this paper, we have presented a non-interactive timestamping solution in the bounded storage model. Our solution is not 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 increase. In this context, we studied the security of our solution and formally proved that the possibility of cheating is negligible. In our future work, we plan to adopt new secret sharing schemes for timestamping.

References

1. Y. Aumann and Y. Z.Ding and M. O. Rabin, *Everlasting security in the bounded storage model*, IEEE Transactions on Information Theory. 2002.
2. D. Bayer and S. Haber and W. S. Stornetta, *Improving the efficiency and reliability of digital timestamping*, In: Sequences91: Methods in Communication, Security and Computer Science. 1992.

3. A. Ben Shil and K. Blibech and R. Robbana, *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.
4. K. Blibech and A. Gabillon, *A New Timestamping Scheme Based on Skip Lists*, In: ICCSA (3). pp. 395-405, 2006.
5. K. Blibech and A. Gabillon, *A New Totally Ordered Timestamping Scheme*, In: 5th Conference on Security and Network Architectures SAR. . 2006.
6. K. Blibech and A. Gabillon, *CHRONOS: an authenticated dictionary based on skip lists for timestamping systems*, In: SWS. pp. 84-90, 2005.
7. K. Blibech and A. Gabillon and A. Bonnecaze, *Etude des systèmes d'horodatage*, Technique et Science Informatiques 26(3-4). pp. 249-278, 2007.
8. A. Bonnecaze and P. Liardet and A. Gabillon and K. Blibech, *A distributed time stamping scheme*, 4th Conference on Security and Network Architectures SAR 2005. 2005.
9. A. Budas and P. R. Laud and H. Lipmaa and J. Willemson, *Timestamping with Binary Linking Schemes*, In: CRYPTO. ICICS. pp. 486-501, 1998.
10. A. Budas and P. R. Laud and B. Schoenmakers, *Optimally efficient accountable timestamping*, In: Public Key Cryptography. 2000.
11. S. Dziembowski and U. Maurer, *On Generating the Initial Key in the Bounded-Storage Model*. In: Advances in Cryptology- EUROCRYPT. pp.126-137, 2004.8
12. S. Haber and W. S. Stornetta, *How to Time-Stamp a Digital Document*, J. Cryptology 3(2). pp. 99-111, 1991.
13. U. Maurer, *Conditionally-perfect secrecy and a provably-secure randomized cipher*, Journal of Cryptology. pp. 53-66, 1992.
14. U. Maurer, *Secret key agreement by public discussion*, IEEE Transaction on Information Theory, 39. pp. 733-742, 1993.
15. T. Moran and R. Shaltiel and A. Ta-Shma, *Non-interactive Timestamping in the Bounded Storage Model*, In: In Advances in Cryptology. CRYPTO'04. LNCS, Springer, 3152. pp. 460-476, 2004.
16. T. Moran and R. Shaltiel and A. Ta-Shma, *Non-interactive Timestamping in the Bounded Storage Model*, In: J. Cryptology. pp. 189-226, 2009.
17. National Institute of Standards and Technology (NIST), *Announcement of Weakness in the Secure Hash Standard*, Technical report. 1994.
18. A. Shamir, *How to share a secret*, Commun. ACM 22(11). pp. 612-613, 1979.