Analysis and Improvement of an Authentication Scheme in Incremental Cryptography

Louiza Khati, Damien Vergnaud

To cite this version:

Louiza Khati, Damien Vergnaud. Analysis and Improvement of an Authentication Scheme in Incremental Cryptography. SAC 2018 - 25th International Conference on Selected Areas in Cryptography, Aug 2018, Calgary, Canada. pp.50-70, 10.1007/978-3-030-10970-7_3 . hal-01893905

HAL Id: hal-01893905

https://hal.inria.fr/hal-01893905

Submitted on 7 May 2020

Analysis and Improvement of an Authentication Scheme in Incremental Cryptography

Louiza Khati[1,2] and Damien Vergnaud[3,4]

[1] Département d'informatique de l'ENS
École normale supérieure, CNRS, PSL Research University
Paris, France
[2] ANSSI, Paris, France
[3] Sorbonne Université, CNRS
LIP6, Équipe Almasty, Paris, France
[4] Institut Universitaire de France, Paris, France

Abstract. Introduced in cryptography by Bellare, Goldreich and Goldwasser in 1994, *incrementality* is an attractive feature that enables to update efficiently a cryptographic output like a ciphertext, a signature or an authentication tag after modifying the corresponding input. This property is very valuable in large scale systems where gigabytes of data are continuously processed (e.g. in cloud storage). Adding cryptographic operations on such systems can decrease dramatically their performance and incrementality is an interesting solution to have security at a reduced cost.

We focus on the so-called *XOR-scheme*, the first incremental authentication construction proposed by Bellare, Goldreich and Goldwasser, and the only *strongly* incremental scheme (*i.e.* incremental regarding insert *and* delete update operations at *any* position in a document). Surprisingly, we found a simple attack on this construction that breaks the *basic security* claimed by the authors in 1994 with only one authentication query (not necessarily chosen). Our analysis gives different ways to fix the scheme; some of these patches are discussed in this paper and we provide a security proof for one of them.

1 Introduction

Bellare, Goldreich and Goldwasser initiate the study on *incremental cryptography* in [3] and then refined it in [4]. Cryptographic incremental constructions are meant to provide efficient updates compared to classical algorithms. Usually, the result of a cryptographic algorithm (such as encryption or authentication) over a document has to be re-computed entirely if any change is applied to the document (and this regardless of the modification size). Incremental cryptography enables to update a signature, a message authentication code (MAC) or a ciphertext in time proportional to the number of modifications applied to the corresponding document. This attractive feature leads to build many incremental cryptographic primitives such as encryption schemes [4, 2, 1], signature [3, 9, 16], MACs [4, 9, 14], hash functions [11, 6] and authenticated encryption constructions [8, 2, 18].

An algorithm is incremental regarding specific *update* operations such as inserting, deleting or replacing a data block inside a document. A desirable incremental algorithm should support all these operations for *any* positions: it should be possible to insert, delete or replace a data block of the document for all positions without breaking the security of the cryptographic algorithm. Most known algorithms only support replacement of data blocks and the algorithms that support insertion, deletion and replacement[1] are deemed *strongly incremental.*

Virus protection is the first application of incremental cryptography quoted in the seminal paper [3]. They consider the usage scenario where processor accesses files on a remote host and a virus can alter these files. A simple idea is to compute authentication tags for all files with a key stored securely by the processor and any modification by a virus will be detected by verifying the corresponding tag. Knowing that these files will be updated often enough, using an incremental authentication algorithm preserves the processor by performing a lighter computation.

Bellare *et al.* also introduced in [3] the corresponding security notions. In the *basic security* model, the adversary can obtain a valid authentication tag for any message it wanted (as in classical MAC security) and it can also update (with the supported update operations) valid pairs message/tag. This is a first security level but it is reasonable to consider a stronger adversary that can alter files *and* tags before applying update operations; it corresponds to the *tamper-proof security* notion introduced in [3].

Nowadays this use case can be extended to the "digital world". Large amount of data [10, 15] are processed every day by different services like cloud services, distributed networks and distributed storage. It is clear that all these data require integrity and/or privacy at a low computational cost otherwise going through gigabytes of data for minor changes without incremental primitives is really demanding in term of time and energy. A concrete example is the Cloud Bigtable by Google [7] that stores petabytes of data across thousands of commodity servers. This Bigtable has a particular data structure that links a unique index number to each block. In this case an incremental hashing that supports replacement and insertion operations is suitable as mentioned in [15]. A more critical usage is storage services in mobile cloud computing where a mobile client device is in addition limited in term of energy consumption. To solve this issue, Itani, Kayssi and Chehab provide an energy-efficient protocol in [13] that guarantee data integrity based on incremental MACs. Another use case is sensor networks and more specifically environmental sensors [12, 15]: several sensors are deployed at different physical positions and they record continuously data. At some point, all the data ends up in a big public database that has to be publicly checkable. The database is updated (insertion operation mainly) at a high frequency and if the hash value over all the database is entirely re-computed for each insertion it will be very consuming. All these use cases are examples among many others. Incremental cryptography is clearly an area to explore to solve practical issues.

[1] Actually supporting insertion and deletion is sufficient as replacement can be obtain by combining these two update operations.

For now incrementality is mainly investigated for hashing and signing even if it was also considered for encryption in [3, 2]. It is not surprising regarding all the practical use cases that need incremental authenticated constructions. Recently, the CAESAR[2] competition stimulates research on authenticated encryption algorithm. Sasaki and Yasuda analysed several candidates and found that none of them performs incrementality. That is why they designed their own authenticated encryption mode with associated data [18] based on existing constructions. This new mode is incremental for the replace, insert and delete operations, but the insert and delete operations of this mode concern only the last block of the authenticated data or the last block of the message (and it remains open to design a strongly incremental authenticated encryption algorithm).

Actually, as far as we know, the only authentication scheme that is strongly incremental is the *XOR-scheme* designed by Bellare, Goldreich and Goldwasser in [4] (*cf.* Figure 2). This strong property comes with a cost: only basic security is claimed in [3] and this algorithm needs to generate and store a lot of randomness. The MAC operation generates a random value for each data block and these random values are necessary for the verification and the update operations. The XOR-scheme is based on a pseudo-random function (PRF) and a pseudo-random permutation (PRP) and the incremental algorithms for (single block) insert and delete operations require only two applications of the underlying PRF and two applications of the underlying PRP. The XOR-scheme relies on the concept of *pair block chaining* (which was later used in [11] which involves taking each pair of two consecutive blocks of a message and feeding them into a pseudo-random function before chaining all the outputs of the PRF into the final hash. This scheme extends another scheme, called the *randomized XOR-scheme*, from [5] which is incremental only for replacement. Even if they share a similar name, these two algorithms are different: the randomized XOR-scheme is not based on a pair block chaining structure and requires actually much less randomness. To distinguish the two schemes, in this paper, we will call this second scheme the *unchained XOR-scheme*.

An analysis on some incremental hash functions was provided by Phan and Wagner [17]. They give, inter alia, patterns that could give collisions on a hash function based on pair block chaining. Two cases are of interest for the XOR-scheme: *non-distinct blocks* and *cycling chaining*. The first one considers repeated blocks messages like $A||B||C||B||A$ and $B||C||B||A||B$ that would have the same sum value if no randomness was used (*cf.* Figure 2) but as underlined by the authors the random values appended to each message block prevents these repetitions. The second one considers a variant of the XOR-scheme [11] where the first and the last block are chained then some repeated patters like $A||B||A$ and $B||A||B$ would have the same sum value but it not the case in the original version from [4]. Therefore, in the present state-of-the-art, no attacks are known against the original strongly incremental *XOR-scheme* proposed in [4].

[2] Competition for Authenticated Encryption: Security, Applicability, and Robustness.

1.1 Contributions of the Paper

In this paper, we analyse the security of the original XOR-scheme construction proposed by Bellare, Goldreich and Goldwasser in [4] and based on a chained structure as defined in [3].

ATTACKS. We provide an attack that breaks the XOR-scheme *basic security* claimed by the authors[3]. It succeeds with probability 1 using only one MAC query. It takes advantage of the chaining structure of this scheme and some xor function properties. This attack is very simple and it is surprising that it remained unnoticed until now (especially since the paper [4] appeared in a major computer science conference and was extensively quoted since 1994).

ANALYSIS AND PATCHED CONSTRUCTIONS. We analyse our attack and the original XOR-scheme to find where its security breaks down. We show that the main flaw is that the XOR-scheme does not explicitly take into account the document length and we noticed that adding the number of data block to the construction prevents this kind of attacks. We analyse different ways to patch the scheme by introducing the document block length in the construction and found that the scheme can still be weak for some options.

We propose a modified version of the XOR-scheme and prove its basic security. Our security proof for the patched XOR-scheme uses tool from the unchained XOR-scheme security proof [5].

ORGANIZATION OF THE PAPER. We introduce some mathematical backgrounds, recall the security models for incremental MAC constructions and we give a detailed description of the XOR-scheme construction in Section 2. Then we present a general forgery attack and its analysis in Section 3. In Section 4, we discuss different solutions to patch efficiently the scheme without making it more complicated neither breaking the structure of the algorithm. We choose one construction and give its detailed description. Its security proof is given in Section 5 before the conclusion in Section 6.

2 Preliminaries

2.1 Notations

For any integer n, $\{0,1\}^n$ denotes the set of bit strings of length n and we let $\{0,1\}^*$ denote the set of all finite-length bit strings. For two bit strings X and Y, $X||Y$ denotes their concatenation. For a finite set S, we use $x \overset{\$}{\leftarrow} S$ to denote sampling x uniformly at random from S. For $X \in \{0,1\}^*$, we use $|X|$ to denote the bit length of X and $|X|_\ell$ denotes the number of ℓ-bit block in the bit-string X (and in particular $|X|_1 = |X|$).

RANDOM FUNCTIONS/RANDOM PERMUTATIONS. The set of all functions $\{0,1\}^\ell \to \{0,1\}^L$ is denoted $\mathcal{F}_{\ell,L}$. A *random function* F is a randomly chosen

[3] In [4, Theorem 3.1], Bellare, Goldreich and Goldwasser stated a security result for their scheme but no proofs are provided in their paper.

function in $\mathcal{F}_{\ell,L}$. The set of all permutations $\{0,1\}^{\ell} \rightarrow \{0,1\}^{\ell}$ is denoted $\mathcal{P}_{\ell}$. A *random permutation* P is a randomly chosen permutation in $\mathcal{P}_{\ell}$.

PSEUDO-RANDOM FUNCTIONS/PSEUDO-RANDOM PERMUTATIONS. Given a non-empty subset $\mathcal{K}$, a function family F_k: $\mathcal{K} \times \{0,1\}^{\ell} \rightarrow \{0,1\}^{L}$, where $k \in \mathcal{K}$, is a (t, ϵ)-pseudo-random function (PRF) if for any algorithm $\mathcal{A}$ running in a time at most t, the following holds:

$$|Pr[k \overset{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{F_k(\cdot)} = 1] - Pr[F \overset{\$}{\leftarrow} \mathcal{F}_{\ell,L} : \mathcal{A}^{F(\cdot)} = 1]| \leq \epsilon.$$

Given a non-empty subset $\mathcal{K}$, a permutation family P_k: $\mathcal{K} \times \{0,1\}^{L} \rightarrow \{0,1\}^{L}$, where $k \in \mathcal{K}$, is a (t, ϵ)-pseudo-random permutation (PRP) if for any algorithm $\mathcal{A}$ running in a time at most t, the following holds:

$$|Pr[k \overset{\$}{\leftarrow} \mathcal{K} : \mathcal{A}^{P_k(\cdot)} = 1] - Pr[F \overset{\$}{\leftarrow} \mathcal{F}_{L,L} : \mathcal{A}^{F(\cdot)} = 1]| \leq \epsilon.$$

2.2 Definitions

SYNTACTIC DEFINITION. We begin by describing the syntactic definition of strongly incremental MAC algorithms. In the following, we consider authentication of messages whose length is a multiple of an integer b (which is usually smaller than the block length of the underlying PRF or PRP) but is obviously possible to handle messages of arbitrary finite length using padding. A *document* $D \in \mathcal{D}$ with $\mathcal{D} = \bigcup_{i=1}^{\infty} \{0,1\}^{ib}$ is a sequence of n b-bit blocks, for some integer $n \geq 1$ denoted $D_{=}(D_1, D_2, \ldots, D_n)$ where D_i is the i-th b-bit block of D.

Definition 1. *A* strongly incremental MAC *scheme is a* 5*-tuple*

$$\Pi = (\mathcal{K},\ \mathtt{MAC},\ \mathtt{V},\ \mathtt{I},\ \mathtt{D})$$

in which:

$\mathcal{K}$ *is the key space. A key k is randomly chosen in the key space $\mathcal{K}$. The key k is a parameter for the* MAC, V, I *and* D *algorithms.*

MAC*, the MAC algorithm, is a probabilistic algorithm that takes as input k and a document D and returns an authentication tag T.*

V*, the verification algorithm, is a deterministic algorithm that takes as input a document D and a tag T and returns* 1 *if the tag is valid and* 0 *otherwise.*

I*, the incremental insert algorithm, is a probabilistic algorithm that takes as input the insertion position j, the inserted block D_i, the document D and a tag T to update.*

D*, the incremental delete algorithm, is a probabilistic algorithm that takes as input the suppression block position j, the document D the message block to add D'_j and a tag T to update.*

with the three following correctness properties:

- $(\forall k \in \mathcal{K})(\forall n \in \mathbb{N})(\forall D \in \{0,1\}^{nb})(\forall T \in \{\mathtt{MAC}(k, D)\})(\{\mathtt{V}(k, D,T)\} = \{1\})$

- $(\forall k \in \mathcal{K})(\forall n \in \mathbb{N})(\forall D = (D_1, D_2, \ldots, D_n) \in \{0,1\}^{nb})(\forall T \in \{\mathtt{MAC}(k, D)\})$ $(\forall j \in \{1, \ldots, n+1\})(\forall D^* \in \{0,1\}^b)(\forall T' \in \{\mathtt{I}(j, D^*, D, T)\})$ $(\{\mathtt{V}(k, (D_1, \ldots, D_{j-1}, D^*, D_j, \ldots D_n), T')\} = \{1\})$
- $(\forall k \in \mathcal{K})(\forall n \in \mathbb{N})(\forall D = (D_1, D_2, \ldots, D_n) \in \{0,1\}^{nb})(\forall T \in \{\mathtt{MAC}(k, D)\})$ $(\forall j \in \{1, \ldots, n\})(\forall T' \in \{\mathtt{D}(j, D, T)\})$ $(\{\mathtt{V}(k, (D_1, \ldots, D_{j-1}, D_{j+1}, \ldots D_n), T')\} = \{1\})$

Remark. In practice, the incremental algorithms I and D have to be more efficient than re-computing an entire authentication tag T and cryptographers are looking for scheme where these algorithms are constant-time (*i.e.* independent of the number of b-bit blocks of the document D).

Security Model. The adversary $\mathcal{A}$ is an algorithm (*i.e.* an oracle probabilistic Turing machine) playing in a computational security game denoted $G^{BS}_{\mathtt{MAC},\mathtt{V},\mathtt{I},\mathtt{D}}$ (*cf.* Figure 1). A key k is picked uniformly at random in the key space of the strongly incremental MAC and the adversary has access to all the following oracles

- **a MAC oracle:** the adversary can ask to compute a MAC (for k) on any document of its choice;
- **a verifying oracle:** the adversary can ask to verify (for k) the validity of any pair document/authentication tag;
- **an update oracle:** the adversary can use the incremental operations (for k) on chosen document/authentication tag pairs (in a way depending on the security models as defined below).

At the end of each oracle query (except verification queries), the corresponding authenticated document/authentication tag (D, r) is added to a list $\mathcal{L}$ and the adversary wins the game if it outputs eventually a pair $(D^\star, r^\star) \notin \mathcal{L}$ that is accepted by the verification algorithm.

Basic Security. As defined in [3], in basic security settings the adversary $\mathcal{A}$ is **not allowed** to do incremental operations on a couple (D, T) where the verification algorithm $\mathtt{V}(D, T)$ *will* fail. It can only apply incremental operations on couples (D, T) that belong to the list $\mathcal{L}$. As mentioned above, to win the security game, $\mathcal{A}$ must provide a forgery that is to say a document D^* and a tag T^* such that $\mathtt{V}(D^*, T^*)$ returns 1 and the couple (D^*, T^*) is not in the list $\mathcal{L}$.

Remark 1. The verification $\mathtt{V}(D, T)$ is not applied before incremental operation to check authenticity otherwise the low computational cost is lost. It is simply **assumed** that $\mathcal{A}$ does not query incremental operations on altered couples. In this paper, we focus on this basic security notion only.

Definition 2. *Let Π = ($\mathcal{K}$, MAC, V, I, D) be a strongly incremental MAC scheme and let $\mathcal{A}$ be an adversary. Let* $\mathbf{Adv}^{BS}_{A,\Pi} :=$

$$\mathbf{Pr}[k \xleftarrow{\$} \mathcal{K}; \mathcal{L} \leftarrow \{\}; (D^*, T^*) \leftarrow \mathcal{A}^{\mathtt{MAC}^{\mathcal{L}}_k, \mathtt{V}^{\mathcal{L}}_k, \mathtt{I}^{\mathcal{L}}_k, \mathtt{D}^{\mathcal{L}}_k} : 1 \leftarrow \mathtt{V}^{\mathcal{L}}_k(D^*, T^*) \wedge (D^*, T^*) \notin \mathcal{L}].$$

Π is $(\lambda, q_m, q_v, q_{inc}; \epsilon)$-BS-secure in the basic sense if, for any adversary $\mathcal{A}$ which runs in time λ, making q_m queries to the `MAC` *oracle, q_v to the* `V` *oracle and q_{inc}* ***valid*** *queries to the incremental oracles* (`I`,`D`) *we have $\mathbf{Adv}^{BS}_{A,\Pi} < \epsilon$.*

```
Game $G^{BS}_{\mathtt{MAC},\mathtt{V},\mathtt{I},\mathtt{D}}$
$k \xleftarrow{\$} \mathcal{K}$
$\mathcal{L} \leftarrow \{\}$
If $\mathcal{A}^{\mathtt{MAC}^{\mathcal{L}}_k, \mathtt{V}^{\mathcal{L}}_k, \mathtt{I}^{\mathcal{L}}_k, \mathtt{D}^{\mathcal{L}}_k}$ makes a query $(D^*, T^*)$ such that
    - $\mathtt{V}^{\mathcal{L}}_k(D^*, T^*)$ returns 1 and
    - $(D^*, T^*) \notin \mathcal{L}$
Then Return 1 Else return 0.
```

Fig. 1. Game defining Basic Security (BS) for an incremental authentication scheme.

Tamper-Proof Security. As defined in [4]*tamper-proof security* is a stronger security notion since the adversary $\mathcal{A}$ is allowed to query incremental operation on **any** couple (D, T) even new couples (couples that do not belong to $\mathcal{L}$). Then $\mathcal{A}$ wins the security game if it provides a new couple (D^*, T^*) such that $\mathtt{V}(D^*, T^*)$ returns 1. It was already mentioned in [4], that the XOR-scheme does not achieve tamper-proof security and this is also the case of our modified XOR-scheme.

2.3 Description of the XOR-scheme

The XOR-scheme ($\mathcal{XS}$) as defined in [3] is an incremental authenticated algorithm based on pair-wise chaining as shown in Figure 2. Let ℓ and L be two positive integers and let $b < \ell$ be some positive integer. The $\mathcal{XS}$ scheme is based on a pseudo-random function $F : \mathcal{K}_F \times \{0,1\}^{2\ell} \longrightarrow \{0,1\}^L$ and a pseudo-random permutation $P : \mathcal{K}_P \times \{0,1\}^L \longrightarrow \{0,1\}^L$. The incremental algorithms for (single block) insert and delete operations require only two applications of the underlying PRF and two applications of the underlying PRP. The $\mathcal{XS}$ scheme generates an authentication tag for a document D by repeatedly applying the PRF to pairs of blocks – each made of a b-bit data block from the document D and an $\ell - b$ random block (pick uniformly at random and independently for each block). In the following, for simplicity, we consider only documents whose binary length is a multiple of b and we denote $\mathcal{D} = (\{0,1\}^b)^*$.

- The **key space** $\mathcal{XS}.\mathcal{K}$ is $\mathcal{K}_F \times \mathcal{K}_P$ the Cartesian product of the key space of the underlying PRF F and PRP P.
- The **MAC algorithm** $\mathcal{XS}$.`MAC` takes as input a document $D \in \mathcal{D}$ and outputs a tag $T := (r, \tau)$. For each document block D_i, an $(\ell - b)$-bits block r_i is randomly generated. The concatenation of these values is denoted

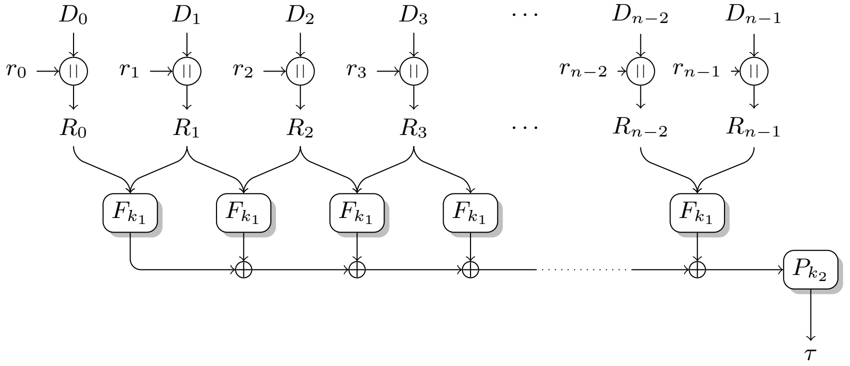

Fig. 2. Description of the XOR-scheme

$R_i := D_i||r_i$. Each couple (R_{i-1}, R_i) is processed by the function F_{k_1} and outputs a value denoted h_i then the bitwise XOR (eXclusive OR) of all the values (denoted Σ) is processed by the permutation P_{k_2} to give the value τ.

- The **Verification algorithm** $\mathcal{XS}.\texttt{V}$ takes as inputs the document D and a tag $T := (r, \tau)$. It re-computes the value τ from the inputs r and D. It returns 1 if this value is equal to the input τ and 0 otherwise.
- The **Insert operation** $\mathcal{XS}.\texttt{I}$ enables to insert a block value in a document. It takes as inputs the position j where the block value has to be inserted, the previous block value D_{j-1} [4], the block value D_j [5], the new block value D'_j and the tag T. It outputs the new tag.
- The **Delete operation** $\mathcal{XS}.\texttt{D}$ enables to delete a block from the document. It takes as inputs the position j where the block as to be deleted, the block value to delete D_j, the previous and next block values D_{j-1} and D_{j+1} and the tag T.

The update algorithms are intuitive and given in Fig. 8 (in Appendix) for update operations at a position different from the first block position. They can be adapted to be applied to the first block. In the original version, it is specified that a prefix and postfix are added to the document. For a document $D = D_1 \dots D_n$, the authentication tag is computed on $D_0||D_1 \dots D_n||D_{n+1}$ where D_0 and D_{n+1} are specific prefix and postfix values. In this paper, this specification is not taken into account: it does not prevent our attack and the repaired scheme is proven secure without it.

XOR-scheme limits. Supporting insert, delete and consequently replace operations should make the XOR-scheme very efficient in term of update time running. The fresh random values r_i generated by this scheme for each new document block are necessary for security. But generating so much randomness

[4] For the first position, there is no previous block.
[5] For the last position, there is no next block.

is *time consuming*: for an n-block document D, a $n(\ell - b)$-random bits value r is generated. Random generation also slows down the insertion operation. Another drawback is the tag *expansion*: the random value r is part of the tag and needs to be stored. For a n block document, random storage costs $n(\ell - b)$ bits. Even if today storage is not an issue, having short tags is desirable.

3 Forgery Attacks against the XOR-scheme

According to the basic security game described in Fig. 1, the adversary $\mathcal{A}$ wins the game if it finds a new pair (D^*, T^*) such that the verification operation returns 1. If an adversary has access to any tag T (such that $T = (r, \tau)$) returned by the $\mathtt{MAC}$ algorithm on a document D (for example $D_0||D_1||D_2$), it can forge a different document D^* having the same value τ. The value τ is computed as follows:

$$\tau = P_{k_2}[F_{k_1}(D_0||r_0, D_1||r_1) \oplus F_{k_1}(D_1||r_1, D_2||r_2)] \quad (1)$$

$$\Sigma = F_{k_1}(R_0, R_1) \oplus F_{k_1}(R_1, R_2) = h_1 \oplus h_2 \quad (2)$$

$\mathcal{A}$ can build a document $D^* \neq D$ and a value r^* such that the corresponding Σ^* value collides with Σ even if there is no weakness on F. A way to do so is inserting a specific block chain in document D in order to cancel all the new values h'_i introduced by these repetitions as shown in Figure 3. It seems that the chaining structure of the XOR-scheme should prevent this behavior because changing or inserting a block value will affect two values h_i then the tag τ will be different. These modifications have to be compensated: the values h'_i introduced has to be canceled by xoring the same value and all the original values h_i that are deleted has to be re-introduced. We use this trick to break the claimed *basic security.*

$$\begin{array}{cccccc} (R_0, R_1) & (.,.) & (.,.) & (R_1, R_2) & \\ \downarrow & \downarrow & \downarrow & \downarrow & \\ h_1 & \multicolumn{2}{c}{\underbrace{\oplus \ \ldots \ \oplus \ \ldots \ \oplus}_{=0}} & h_2 & = \Sigma \end{array}$$

Fig. 3. Xor cancellation strategy in the XOR-scheme

FORGERY ATTACK. Applying this strategy gives us an adversary $\mathcal{A}^{\mathtt{MAC}_{k_1,k_2}}$ wining the game $G^{BS}_{\mathtt{MAC},\mathtt{V},\mathtt{I},\mathtt{D}}$ with probability 1 and that requires only 1 $\mathtt{MAC}_{k_1,k_2}$ query.

Adversary $\mathcal{A}^{\mathtt{MAC}^{\mathcal{L}}_{k_1,k_2}}$

1 $\mathcal{A}$ asks the MAC of a short document D where $D = D_0||D_1||D_2$ and receives the corresponding authentication-tag $T = (r, \tau)$.

$$\begin{tikzcd}(R_0, R_1) \arrow[d] & & (R_1, R_2) \arrow[d] & \\ h_1 & \oplus & h_2 & = \Sigma\end{tikzcd}$$

Fig. 4. Σ computation for 3 block document

2 $\mathcal{A}$ builds a document D^* from D such that $D^* = D_0||D_1||D_2||D_1||D_2||D_1||D_2$ and a value r^* from r such that $r^* = r_0||r_1||r_2||r_1||r_2||r_1||r_2$.

$$\begin{tikzcd}(D_0||r_0, D_1||r_1) & & (D_1||r_1, D_2||r_2) & & (D_2||r_2, D_1||r_1) & & (D_1||r_1, D_2||r_2) & & (D_2||r_2, D_1||r_1) & & (D_1||r_1, D_2||r_2) & \\ (R_0, R_1) \arrow[d] & & (R_1, R_2) \arrow[d] & & (R_2, R_1) \arrow[d] & & (R_1, R_2) \arrow[d] & & (R_2, R_1) \arrow[d] & & (R_1, R_2) \arrow[d] & \\ h_1 & \oplus & h_2 & \oplus & \cancel{h_2} & \oplus & \cancel{h_2} & \oplus & \cancel{h_2} & \oplus & \cancel{h_2} & = \Sigma\end{tikzcd}$$

Fig. 5. Attack on the XOR-scheme

The document D^* is different from D but it has the same value τ. The document D^* given in Figure 5 is an example of a forgery and many other examples can be given. To be more general, for any $x \in \{0,1\}^b$, any $x' \in \{0,1\}^{(\ell-b)}$ and for any valid pair (D, T) such that $D = D_0 \dots D_i||D_{i+1} \dots D_n$, many forgeries $(D^*, (r^*, \tau))$ can be built by inserting the specific block chain $D_i||x||D_i||x$ in D (and the corresponding random value chain $r_i||x'||r_i||x'$ in r for any x') such that:

$$D^* = D_0 \dots D_{i-1}|| \underbrace{D_i||x||D_i||x||D_i} ||D_{i+1} \dots D_n$$
$$r^* = r_0 \dots r_{i-1}|| \underbrace{r_i||x'||r_i||x'||r_i||} r_{i+1} \dots r_n.$$

A variant of this forgery is to insert only a repeated document block D_i (and r_i) is the following:

$$D^* = D_0 \dots D_{i-1}|| \underbrace{D_i||D_i||D_i} ||D_{i+1} \dots D_n.$$
$$r^* = r_0 \dots r_{i-1}|| \underbrace{r_i||r_i||r_i||} r_{i+1} \dots r_n.$$

A more powerful forgery can be built from (D, T) by inserting any values x and y in D (and any values x' and y' in r) such that:

$$D^* = D_0 \dots D_{i-1}|| \underbrace{D_i||x||y||x||y||x||D_i||x||D_i} ||D_{i+1} \dots D_n.$$
$$r^* = r_0 \dots r_{i-1}|| \underbrace{r_i||x'||y'||x'||y'||x'r_i||x'||r_i||} r_{i+1} \dots r_n.$$

For all these attacks, the underbraced chains can be repeated many times. These three attacks are some of the possible attacks, following this *canceling* strategy, some exotic chains can be inserted in order to ends with a value τ that corresponds to a legitimate tag. A first observation is that all these attacks are performed by inserting blocks and providing a forgery D^* that has the same length that the original one looks impossible or at least harder.

4 Modification of the XOR-scheme

The previous section described an attack that breaks the basic security of the XOR-scheme by producing a document D^* using a MAC query (D, T) where $\tau = \tau^*$ and $|D|_b \neq |D^*|_b$. All the forgeries D^* produced are longer that the original document D. One can notice that if the adversary $\mathcal{A}$ is only allowed to MAC and verify documents that have the same length n then the attack presented in Section 3 will fail. A first naive idea is to force all documents to have the same length (documents that are too small can be padded in the MAC algorithm) but this solution is not realistic and the incremental property will be lost. A natural way to fix this flaw is to use the document length n for the computation of the value τ in order to make it *size dependent*. The size can be expressed according to any units: number of bits, bytes, blocks. Choosing the number of blocks n is sufficient. A postfix block containing the number of blocks n can be added at the end of the document then the computation of the value Σ will become:

$$\Sigma = F_{k_1}(D_0||r_0, D_1||r_1) \oplus F_{k_1}(D_1||r_1, D_2||r_2) \oplus \cdots \oplus F_{k_1}(D_{n-1}||r_{n-1}, r_n||n)$$

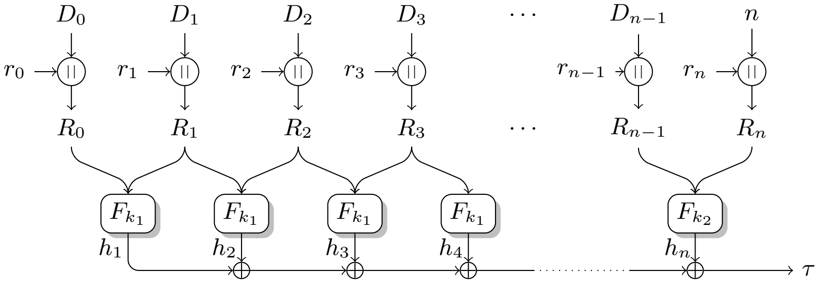

Fig. 6. Description of the fixed XOR-scheme

The last block works as a mask for each value τ: incremental operation will refresh the last random value r_n in order to have a different mask value for any modification. As a consequence, the pseudo-random permutation P is not necessary anymore ($\tau = \Sigma$), it is removed in the modified scheme (Figure 6). This version of the XOR-scheme is proven in Section 5.

The last random block value r_n (concatenated with the document length n) is necessary otherwise the corresponding h_n value can be canceled. The following attack is an example:

1. $\mathcal{A}$ asks the `MAC` of a document $D = D_0||D_1$ and receives the tag $T_1 = (r_0||r_1, \tau_1)$.
2. $\mathcal{A}$ asks to delete the first block of D by with the delete query $\mathtt{D}(0, ., D_0, D_1, \tau_1)$ and receives $T_2 = (r_1, \tau_2)$.

3. $\mathcal{A}$ asks to insert the block D'_0 at the first position of the resulting document with the query $\mathtt{I}(0, D_1, ., D'_0, \tau_2)$ and receives $T_3 = (r'_0||r_1, \tau_3)$.
4. $\mathcal{A}$ asks to insert the block D_2 at the position 2 of the original document D with the query $\mathtt{I}(2, ., D_1, D_2, \tau_1)$ and receives $T_4 = (r_0||r_1||r_2, \tau_4)$.
5. $\mathcal{A}$ builds the document $D^* = D'_0||D_1||D_2$ and the tag $T^* = (r'_0||r_1||r_2, \tau_1 \oplus \tau_3 \oplus \tau_4)$.

The couple (D^*, r^*) is a forge: it is not in the list $\mathcal{L}$ of tagged document and it has a valid tag. To avoid such attacks, for incremental operations the last random block (concatenated with the document length) need to be always refreshed. To be sure that none of the previous attack will not be practical, an independent key is used to process the last couple (R_{n-1}, R_n). That way, it would be hard for an adversary to make a forgery from a linear combination of tagged documents.

Complexity. The modified XOR-scheme is slightly slower than the original one. For the MAC and the incremental algorithms the P call is removed but a call to the function F_{k_2} is added as shown in Figure 7. The delete D and insert I operations are slightly slower because of the last block update: the last value R_i depending on the document length has to be removed and the a new value R'_i with the new document length n' is added.

Functions	scheme	$(\ell - b)$-bits Generation	F	P	xor
MAC	XS	n	$n-1$	1	$n-1$
	M-XS	$n+1$	$(n-1)+1$	0	n
V	XS	0	$n-1$	1	$n-1$
	M-XS	0	$(n-1)+1$	0	n
D	XS	0	3	2	3
	M-XS	1	5	0	5
I	XS	1	3	2	3
	M-XS	2	5	0	5

Fig. 7. Complexity: XOR-scheme (XS) and Modified XOR-scheme (M-XS)

Other Solutions. In the original XOR-scheme (Fig. 2), the document length can be added differently in the algorithm (but still with a random value r_n):

1 Before the last operation $P_{k_2}(\Sigma)$, an intermediate operation $F_{k_3}(r_n||n, \Sigma)$ can be added such that $\tau = P_{k_2}[F_{k_3}(r_n||n, \Sigma)]$.

2 The block length can be processed individually as a last block such that $\tau = P_{k_2}[F_{k_3}(r_n||n) \oplus \Sigma]$.

5 Security Proof

The security proof follows the proof strategy used in [5] for proving the unchained XOR-scheme.

Information theoretic case. As in [5], we first consider the case where the two underlying PRFs F_{k_1} and F_{k_2} are replaced by two truly random functions F_1 and F_2 from $\{0,1\}^{2\ell}$ to $\{0,1\}^L$. We consider an unbounded adversary and the following theorem claims the security of this modified scheme in the information theoretic case. More precisely, it provides an absolute bound on the success of the adversary in terms of the number of oracle queries it makes.

Theorem 1. *Let $\mathcal{F}_{2\ell,L}$ be the family of random functions with input length 2ℓ and output length L. Let $\mathcal{A}$ be any (computationally unbounded) adversary, in the basic security settings, making a (q_m, q_v, q_{inc})-attack against the modified XOR-scheme with two functions picked uniformly at random from $\mathcal{F}_{2\ell,L}$. The probability that $\mathcal{A}$ is successful is at most*

$$q^2 \cdot 2^{b-\ell} + q_v \cdot (t^2 \cdot 2^{b-\ell} + \cdot 2^{-L}).$$

where $q = q_m + q_{inc}$ and t denotes the maximal block-length of the documents authenticated in the security game.

Remark. The modified XOR-Scheme is not tamper-proof secure.

Proof (Theorem 1 (Sketch.)). The proof follows closely the proof from [5]. The main difference is that we use two different random functions in the modified scheme and that we need the following simple lemma to prove that some specific matrix (close to the one used in [5]) is full-rank. For the reader familiar with [5], we use similar notations in the following.

Lemma 1. *Let X be some finite set and let $n \in \mathbb{N}$. Let $(R_0, R_1, \ldots, R_n) \in X^{n+1}$ with $R_i \neq R_j$ for all $i \neq j$ then if there exists $(R_0^*, R_1^*, \ldots, R_n^*) \in X^{n+1}$ such that*

$$\{(R_0, R_1), (R_1, R_2), \ldots (R_{n-1}, R_n)\} = \{(R_0^*, R_1^*), (R_1^*, R_2^*), \ldots (R_{n-1}^*, R_n^*)\}$$

then for all $i \in \{0, \ldots, n\}$, $R_i = R_i^*$.

Proof (Lemma 1). This lemma can be easily proved by induction over n. Let us denote S_n the first set $\{(R_0, R_1), (R_1, R_2), \ldots (R_{n-1}, R_n)\}$ where all R_i are distinct. In particular, the set S_n contains exactly n different couples. One can notice that the first member of each couple is the second member of the previous couple except the first and the last couples. In others words a value R_i appears in two couples: once as a first member and once as a second member except the first one R_0 and the last one R_n.
The case $n = 1$ is trivial. We consider the case $n = 2$ that provides greater clarity. Let assume that it exists $(R_0^*, R_1^*, R_2^*) \in X^3$ such that

$$\{(R_0, R_1), (R_1, R_2)\} = \{(R_0^*, R_1^*), (R_1^*, R_2^*)\}$$

and $\#\{R_0, R_1, R_2\} = 3$. As there are exactly two couples in each set, we have the following two cases:

- **case 1:** $(R_0, R_1) = (R_0^*, R_1^*)$ and $(R_1, R_2) = (R_1^*, R_2^*)$ then in this case, we get $R_0 = R_0^*, R_1 = R_1^*, R_2 = R_2^*$;
- **case 2:** $(R_0, R_1) = (R_1^*, R_2^*)$ and $(R_1, R_2) = (R_0^*, R_1^*)$. The first equality implies $R_1^* = R_0$ and the second equality implies $R_1^* = R_2$ and thus $R_0 = R_2$ which contradicts the statement $R_0 \neq R_2$.

Suppose now that Lemma 1 holds for all integers $k \leq n-1$ for some $n \in \mathbb{N}^*$. We will show that it holds for n.

Let us suppose that there exists $(R_0^*, R_1^*, \dots, R_n^*) \in X^{n+1}$ such that $S_n = S_n^*$ where S_n^* is the set $\{(R_0^*, R_1^*), (R_1^*, R_2^*), \dots (R_{n-1}^*, R_n^*)\}$. Again, as all the values R_i are different in S_n then the n couples are different. The equality of these two sets S_n and S_n^* implies that they contain exactly the same n couples and that in each set a couple appears only once. We have the following two cases:

- **case 1:** $(R_{n-1}, R_n) = (R_{n-1}^*, R_n^*)$ and $S_{n-1} = S_{n-1}^*$. From the induction hypothesis, for all $i \in \{0, \dots, n-1\}$, $R_i = R_i^*$.
- **case 2:** $(R_{n-1}, R_n) \neq (R_{n-1}^*, R_n^*)$ Then there exists $i \in \{0, \dots, n-1\}$ such that $(R_{n-1}, R_n) = (R_{i-1}^*, R_i^*)$. It implies $R_i^* = R_n$ and according to the structure of these sets, there is a couple in S_n^* that has a first member equal to $R_i^* = R_n$) and it has to be the case in S_n. But as mentioned above, R_n is a value that appears only in one couple of S_n and we get a contradiction.

□

We will use this lemma with $X = \{0,1\}^\ell$ at the end of the proof to show that different messages of the same block-length involve different input pairs for the underlying PRF F_1.

Since the adversary $\mathcal{A}$ is computationally unbounded we may assume without loss of generality that it is deterministic. The probabilistic choices in $\mathcal{A}$'s attack on the scheme are thus the initial choice of F_1 and F_2 of the random functions in $\mathcal{F}_{2\ell,L}$ and the choices of random coins made by the authentication oracles in the security game. We assume (again without loss of generality) that $\mathcal{A}$ makes exactly $q = q_s + q_{inc}$ authentication queries (either as a direct MAC query or as an update query using the insert or the delete oracle). As in [5], there is no loss of generality to assume that $\mathcal{A}$ makes all its authentication queries and then makes exactly one verify query (for its purported forgery). We prove that in this case the probability of the event (denoted Succ) $\mathcal{A}$'s forgery is valid is upper-bounded by

$$q^2 \cdot 2^{b-\ell} + t^2 \cdot 2^{b-\ell} + 2^{-L}.$$

and using a classical argument (see e.g. [5]) we get the claimed bound for general adversaries.

We consider the simple case where all the random coins used in the last block of each authenticated document are different. Note that in all authentication queries (from a fresh MAC query or an update query), this random block is picked uniformly at random and independently of the previous blocks. To analyze the probability of this event (denoted $\mathsf{Distinct}$), we can therefore use the following simple lemma:

Lemma 2 ([5, Fact A.1]). *Let $P(m, t)$ denote the probability of at least one collision in the experiment of throwing t balls, independently at random, into m buckets. Then $P(m, t) \leq t^2/m$.*

We thus have

$$\begin{aligned}\Pr[\mathsf{Succ}] &= \Pr[\mathsf{Succ}|\mathsf{Distinct}] \cdot \Pr[\mathsf{Distinct}] + \Pr[\mathsf{Succ}|\overline{\mathsf{Distinct}}] \cdot \Pr[\overline{\mathsf{Distinct}}] \\ &\leq \Pr[\mathsf{Succ}|\mathsf{Distinct}] + \Pr[\overline{\mathsf{Distinct}}] \\ &\leq \Pr[\mathsf{Succ}|\mathsf{Distinct}] + P(2^{b-\ell}, q) \\ &\leq \Pr[\mathsf{Succ}|\mathsf{Distinct}] + q^2 \cdot 2^{b-\ell}.\end{aligned}$$

and it remains to upper-bound $\Pr[\mathsf{Succ}|\mathsf{Distinct}]$.

Let us fix a particular sequence of q documents $D^1, \ldots, D^q$ (each made of at most t blocks of b bits) corresponding to all documents authenticated in the security game by some authentication queries (either as a direct MAC query or as an update query using the insert or the delete oracle). We also fix $r^1, \ldots, r^q$ some bit-strings possibly used as random values in the modified XOR-scheme for these documents (*i.e.* r^i consists in $1 \leq t_i \leq t$ blocks of $\ell - b$ bits if D^i is made of t_i blocks of b bits) and we assume that the last blocks of all of them are all different. Finally we fix $\tau^1, \ldots, \tau^q$ some possible corresponding tags in $\{0,1\}^L$ for these documents. We consider only bit-strings $(D^1, \ldots, D^q)$, $(r^1, \ldots, r^q)$ and $(\tau^1, \ldots, \tau^q)$ for which the probability that there exists two functions F_1 and F_2 such that $T^i = (r^i, \tau^i)$ is a valid MAC for D^i (for all $i \in \{1, \ldots, q\}$) for F_1 and F_2 is non-zero.

We will compute the probability of the event that $\mathcal{A}$'s forgery is valid conditioned on the event that the authentication queries made by $\mathcal{A}$ are on the documents $D^1, \ldots, D^q$, use the random coins $(r^1, \ldots, r^q)$ and result in the tags $(\tau^1, \ldots, \tau^q)$. More precisely, we will show that this probability is upper-bounded by $t^2 \cdot 2^{b-\ell} + 2^{-L}$ (and since the bit-strings $(D^1, \ldots, D^q)$, $(r^1, \ldots, r^q)$ and $(\tau^1, \ldots, \tau^q)$ are arbitrary, we will get the result by standard conditioning arguments).

We consider a possible forgery output by $\mathcal{A}$ and we denote D^{q+1} the corresponding document, r^{q+1} the used randomness and τ^{q+1} the tag. It is worth noting that the pair (D^{q+1}, r^{q+1}) is different from all pairs (D^i, r^i) for all $i \in \{1, \ldots, q\}$ (since otherwise, this is not an actual forgery) but we cannot assume that the last block of r^{q+1} is different from the last blocks of all previous random values r^i for all $i \in \{1, \ldots, q\}$ (since $\mathcal{A}$ may choose it arbitrarily and it can reuse a value obtained in a previous authentication query).

For $i \in \{1, \ldots, q+1\}$, we denote D^i_j for all $j \in \{1, \ldots, t_i\}$, the j-th block of the document D^i and similarly r^i_j for all $j \in \{1, \ldots, t_i + 1\}$, the j-th block of the randomness r^i. As in [5], we consider the matrix B with $q+1$ rows and $2^{2\ell+1}$ columns over $\mathbb{F}_2 = \{0,1\}$ where the entry in row $i \in \{1, \ldots, q+1\}$ and column $j \in \{1, \ldots, 2^{2\ell+1}\}$ is defined as follows:

- for $j \in \{1, \ldots, 2^{2\ell}\}$, the entry is equal to 1 if j is the index of the 2ℓ-bit string $(D^i_{t_i}||r^i_{t_i}||t_i||r^i_{t_i+1})$ in lexicographic order (and 0 otherwise).

- for $j \in \{2^{2\ell}+1, \ldots, 2^{2\ell+1}\}$, the entry is equal to 1 if $j - 2^{2\ell}$ is the index of the 2ℓ-bit string $(D^i_k || r^i_k || D^i_{k+1} || r^i_{k+1})$ in lexicographic order for some $k \in \{1, \ldots, t_i - 1\}$ (and 0 otherwise).

In other words, the matrix B contains a 1 on the row i for $i \in \{1, \ldots, q+1\}$ only at positions corresponding to bit-strings of length 2ℓ used as inputs to the random functions F_1 and F_2 in the modified XOR-scheme (where the left part consisting of the first $2^{2\ell}$ columns of the matrix corresponds to the unique input of F_2 and the right part corresponds to all inputs to F_1).

We have the following lemma:

Lemma 3. *The matrix B has full rank with probability at least $1 - t^2 \cdot 2^{b-\ell}$.*

Proof (Lemma 3). The proof is similar to the proof of [5, Lemma A.3]. If the pair $(t_{q+1}, r^{q+1}_{t_{q+1}+1})$ is different from all $(t_i, r^i_{t_i+1})$ for $i \in \{1, \ldots, q\}$, then the matrix B is in echelon form (in its left part) and is thus trivially of full rank.

Otherwise, we assume that $r^{q+1}_{t_{q+1}+1}$ is equal to some $r^i_{t_i+1}$ and if $t_{q+1} = t_i$ (the last block of randomness of $\mathcal{A}$'s forgery is equal to the last block of randomness of the i-th authenticated message and the block-length of these two messages are equal). It is worth noting that there exists only one index $i \in \{1, \ldots, q\}$ such that this is the case (since we assume that these last blocks of randomness are all different). For this i-th document, the random blocks r^i_j for $j \in \{1, \ldots, t_i\}$ are all different with probability at least $1 - t_i^2 \cdot 2^{b-\ell} \geq 1 - t^2 \cdot 2^{b-\ell}$ by Lemma 2. Since the pair (D^{q+1}, r^{q+1}) is different from (D^i, r^i) and since the pairs (D^i_k, r^i_k) are all different for $k \in \{1, \ldots, t_i\}$ (with probability at least $1 - t^2 \cdot 2^{b-\ell}$), we can apply Lemma 1 to the sets (of the same length $t_i = t_{q+1}$):

$$\{D^i_1 || r^i_1 || D^i_2 || r^i_2, D^i_2 || r^i_2 || D^i_3 || r^i_3, \ldots, D^i_{t_i-1} || r^i_{t_i-1} || D^i_{t_i} || r^i_{t_i}\}$$

and

$$\left\{D^{q+1}_1 || r^{q+1}_1 || D^{q+1}_2 || r^{q+1}_2, D^{q+1}_2 || r^{q+1}_2 || D^{q+1}_3 || r^{q+1}_3, \ldots, D^{q+1}_{t_i-1} || r^{q+1}_{t_i-1} || D^{q+1}_{t_i} || r^{q+1}_{t_i}\right\}.$$

We thus obtain that there exist an index $k \in \{1, \ldots, t_i - 1\}$ such that

$$(D^{q+1}_k || r^{q+1}_k || D^{q+1}_{k+1} || r^{q+1}_{k+1}) \neq (D^i_k || r^i_k || D^i_{k+1} || r^i_{k+1}).$$

Therefore in this case the left part of the last row (consisting of the first $2^{2\ell}$ columns) is identical to the left part of the i-th row but these rows differ in at least one position in the right part of the matrix B. By elementary operations on the rows, one can easily transform the matrix B in echelon form and it is therefore of full rank (with probability at least $1 - t^2 \cdot 2^{\ell-b}$). □

To conclude the proof, one can identify the functions F_1 and F_2 to their vector of values in $(\{0,1\}^{2\ell})^L$ by denoting $F_i(x) = (\varphi^{(x)}_{i,1}, \ldots, \varphi^{(x)}_{i,L})$ for $x \in \{0,1\}^{2\ell}$ and $i \in \{1, 2\}$, where $\varphi_{i,j} \in \{0,1\}^{2\ell}$ for $i \in \{1, 2\}$ and $j \in \{1, \ldots, L\}$. In this case by construction, τ^i is the authentication tag of the document D^i with randomness

r^i for all $i \in \{1, \ldots, q+1\}$ if and only if for all $j \in \{1, \ldots, L\}$, the j-th bit $\tau_i^{(j)}$ of τ_i is equal to the dot product of the i-th row of the matrix B and the vector $\varphi_{2,i}||\varphi_{1,i}$. Using the same argument as in [5], since B is of full rank, the number of vectors satisfying this $q+1$ equations is 2^L times smaller than the number of vectors satisfying only the first q equations (corresponding to the first q rows of B), and therefore we obtained that the forgery τ^{q+1} output by the adversary is valid with probability 2^{-L} if the matrix B is full rank.

We have thus proved that, in the simplified case, the probability that $\mathcal{A}$'s forgery is valid is upper-bounded by

$$q^2 \cdot 2^{b-\ell} + t^2 \cdot 2^{b-\ell} + 2^{-L}.$$

and thus the claimed bound for general adversaries. □

COMPUTATIONAL CASE. If we replace the (truly) random functions by pseudo-random functions in the previous result, we obtain readily the following computational security result:

Theorem 2. *Let F be the family of pseudo-random functions with input length $2 \cdot \ell$ and output length L. Let $\mathcal{A}$ be any adversary making a (q_m, q_v, q_{inc})-attack against the modified XOR-scheme with two functions picked uniformly at random from F and running in time λ.*
There exist an adversary $\mathcal{B}$ against the pseudo-randomness property of F that makes $q' = q \cdot t$ queries to F, runs in time $\lambda' = \lambda + O(q'(\ell + L))$ such that

$$\mathbf{Adv}^{PRF}_{\mathcal{B},F} \geq \mathbf{Adv}^{BS}_{\mathcal{A},\mathcal{XS}} - [q^2 \cdot 2^{b-\ell} + q_v \cdot (t^2 \cdot 2^{b-\ell} + \cdot 2^{-L})].$$

where $q = q_m + q_{inc}$ and t denotes the maximal block-length of the documents authenticated in the security game.

Proof (Theorem 2). The proof is identical to the proof of [5, Theorem 4.2] and is left to the reader. □

6 Conclusion

We showed that the XOR-scheme as described in [3] does not provide the claimed basic security: a forgery can be easily built from any tag by inserting specific document block chains to a legitimate document and the corresponding random value chains to the legitimate random value. We proposed a modified XOR-scheme that is not vulnerable to these attacks and we proved its security in the *basic sense*. Our modified XOR-scheme is the only secure strongly incremental algorithm but unfortunately it still has some drawbacks: the randomness generation slows down the algorithm and the tag length makes it unpractical because the random values have to be stored. But it is definitely worth analyzing its structure in order to improve it or to build another strongly incremental authenticated scheme (or prove a lower bound on the length of strongly incremental MAC algorithms). Another interesting open problem is to design a strongly incremental authenticated scheme that achieves tamper-proof security.

Acknowledgments. The authors are supported in part by the French ANR ALAMBIC Project (ANR-16-CE39-0006). The authors thank Mihir Bellare for helpful discussions and for pointing out references.

References

1. Atighehchi, K.: Space-efficient, byte-wise incremental and perfectly private encryption schemes. Cryptology ePrint Archive, Report 2014/104 (2014), http://eprint.iacr.org/2014/104
2. Atighehchi, K., Muntean, T.: Towards fully incremental cryptographic schemes. In: Chen, K., Xie, Q., Qiu, W., Li, N., Tzeng, W.G. (eds.) ASIACCS 13. pp. 505–510. ACM Press, Hangzhou, China (May 8–10, 2013)
3. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The case of hashing and signing. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 21–25, 1994)
4. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography and application to virus protection. In: 27th ACM STOC. pp. 45–56. ACM Press, Las Vegas, NV, USA (May 29 – Jun 1, 1995)
5. Bellare, M., Guérin, R., Rogaway, P.: XOR MACs: New methods for message authentication using finite pseudorandom functions. In: Coppersmith, D. (ed.) CRYPTO'95. LNCS, vol. 963, pp. 15–28. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 27–31, 1995)
6. Bellare, M., Micciancio, D.: A new paradigm for collision-free hashing: Incrementality at reduced cost. Cryptology ePrint Archive, Report 1997/001 (1997), http://eprint.iacr.org/1997/001
7. Bershad, B.N., Mogul, J.C. (eds.): 7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA. USENIX Association (2006), https://www.usenix.org/publications/proceedings/?f[0]=im_group_audience 3A137
8. Buonanno, E., Katz, J., Yung, M.: Incremental unforgeable encryption. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 109–124. Springer, Heidelberg, Germany, Yokohama, Japan (Apr 2–4, 2002)
9. Fischlin, M.: Lower bounds for the signature size of incremental schemes. In: 38th FOCS. pp. 438–447. IEEE Computer Society Press, Miami Beach, Florida (Oct 19–22, 1997)
10. Gantz, J., Reinsel, D.: The digital universe in 2010 : Big data, bigger digital shadows, and biggest growth in the far east. ems report (2013), https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf
11. Goi, B.M., Siddiqi, M.U., Chuah, H.T.: Incremental hash function based on pair chaining & modular arithmetic combining. In: Rangan, C.P., Ding, C. (eds.) INDOCRYPT 2001. LNCS, vol. 2247, pp. 50–61. Springer, Heidelberg, Germany, Chennai, India (Dec 16–20, 2001)
12. Hart, J.K., Martinez, K.: Environmental sensor networks: A revolution in the earth system science? Earth-Science Reviews 78(3), 177 – 191 (2006), http://www.sciencedirect.com/science/article/pii/S0012825206000511

13. Itani, W., Kayssi, A.I., Chehab, A.: Energy-efficient incremental integrity for securing storage in mobile cloud computing. 2010 International Conference on Energy Aware Computing pp. 1–2 (2010)
14. Micciancio, D.: Oblivious data structures: Applications to cryptography. In: 29th ACM STOC. pp. 456–464. ACM Press, El Paso, TX, USA (May 4–6, 1997)
15. Mihajloska, H., Gligoroski, D., Samardjiska, S.: Reviving the idea of incremental cryptography for the zettabyte era use case: Incremental hash functions based on SHA-3. Cryptology ePrint Archive, Report 2015/1028 (2015), http://eprint.iacr.org/2015/1028
16. Mironov, I., Pandey, O., Reingold, O., Segev, G.: Incremental deterministic public-key encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 628–644. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)
17. Phan, R.C., Wagner, D.A.: Security considerations for incremental hash functions based on pair block chaining. Computers & Security 25(2), 131–136 (2006), https://doi.org/10.1016/j.cose.2005.12.006
18. Sasaki, Y., Yasuda, K.: A new mode of operation for incremental authenticated encryption with associated data. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 397–416. Springer, Heidelberg, Germany, Sackville, NB, Canada (Aug 12–14, 2016)

A Appendix

Original XOR-scheme Algorithms

```
Algo. $\mathtt{MAC}_{k_1,k_2}(D)$
$\Sigma \leftarrow 0^L$
$n \leftarrow |D|_b$
$r_0 \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
$r \leftarrow r_0$
$R_0 \leftarrow (D_0||r_0)$
For $i = 1$ to $n-1$ do
    $r_i \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
    $r \leftarrow r_{||}r_i$
    $R_i \leftarrow r_i||D_i$
    $h_i \leftarrow F_{k_1}(R_{i-1}, R_i)$
    $\Sigma \leftarrow \Sigma \oplus h_i$
$\tau \leftarrow P_{k_2}(\Sigma)$
$T \leftarrow (r, \tau)$
Return $(T)$
```

```
Algo. $\mathtt{V}_{k_1,k_2}(D, T)$
$\Sigma \leftarrow 0^L$
$n \leftarrow |D|_b$
$r \leftarrow T[0 : (n-1)(\ell-b)-1]$
$\tau \leftarrow T[(n-1)(\ell-b) :]$
$R_0 \leftarrow r_0||D_0$
For $i = 1$ to $n-1$ do
    $R_i \leftarrow r_i||D_i$
    $h_i \leftarrow F_{k_1}(R_{i-1}, R_i)$
    $\Sigma \leftarrow \Sigma \oplus h_i$
$\tau$'$\leftarrow P_{k_2}(\Sigma)$
If $\tau$' $= \tau$
    Return 1
Else
    Return 0
```

```
Algo. $\mathtt{D}_{k_1,k_2}(j, D, T)$
$n \leftarrow |D|_b$
$r \leftarrow T[0 : (n-1)(\ell-b)-1]$
$\tau \leftarrow T[(n-1)(\ell-b) :]$
$R_{j-1} \leftarrow r_{j-1}||D_{j-1}$
$R_j \leftarrow r_j||D_j$
$R_{j+1} \leftarrow r_{j+1}||D_{j+1}$
$h_j \leftarrow F_{k_1}(R_{j-1}, R_j)$
$h_{j+1} \leftarrow F_{k_1}(R_j, R_{j+1})$
$h'_j \leftarrow F_{k_1}(R_{j-1}, R_{j+1})$
$\Sigma \leftarrow h_j \oplus h_{j+1} \oplus h'_j$
$\Sigma \leftarrow \Sigma \oplus P^{-}_{k_2}(\tau)$
$\tau \leftarrow P_{k_2}(\Sigma)$
$r \leftarrow r_0 \ldots r_{j-1}||r_{j+1} \ldots r_n$
$T \leftarrow (r, \tau)$
```

```
Algo. $\mathtt{I}_{k_1,k_2}(j, D'_j, D,T)$
$n \leftarrow |D|_b$
$r \leftarrow T[0 : (n-1)(\ell-b)-1]$
$\tau \leftarrow T[(n-1)(\ell-b) :]$
$r'_j \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
$R'_j \leftarrow r'_j||D'_j$
$R_j \leftarrow r_j||D_j$
$R_{j-1} \leftarrow r_j||D_{j-1}$
$h'_j \leftarrow F_{k_1}(R_{j-1}, R'_j)$
$h'_{j+1} \leftarrow F_{k_1}(R_{j'}, R_j)$
$h_j \leftarrow F_{k_1}(R_{j-1}, R_j)$
$\Sigma \leftarrow h'_j \oplus h'_{j+1} \oplus h_j$
$\Sigma \leftarrow \Sigma \oplus P^{-}_{k_2}(\tau)$
$\tau \leftarrow P_{k_2}(\Sigma)$
$r \leftarrow r_0 \ldots r_{j-1}||r'_j||r_j \ldots r_n$
$T \leftarrow (r, \tau)$
```

Fig. 8. Original XOR-scheme Algorithm

Modified XOR-scheme Algorithms

Algo. $\mathtt{MAC}_{k_1,k_2}(D)$

```
$\Sigma \leftarrow 0^L$
$n \leftarrow |D|_b$
$r_0 \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
$r \leftarrow r_0$
$R_0 \leftarrow r_0||D_0$
**For** $i = 1$ **to** $n-1$ **do**
    $r_i \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
    $r \leftarrow r_{||}r_i$
    $R_i \leftarrow r_i||D_i$
    $h_i \leftarrow F_{k_1}(R_{i-1}, R_i)$
    $\Sigma \leftarrow \Sigma \oplus h_i$
$r_n \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
$R_n \leftarrow r_n||n$
$h_n \leftarrow F_{k_2}(R_{n-1}, R_n)$
$\tau \leftarrow \Sigma \oplus h_n$
$T \leftarrow (r, \tau)$
```

Algo. $\mathtt{V}_{k_1,k_2}(D, T)$

```
$\Sigma \leftarrow 0^L$
$n \leftarrow |D|_b$
$r \leftarrow T[0 : n(\ell-b)-1]$
$\tau \leftarrow T[n(\ell-b) :]$
$R_0 \leftarrow D_0||r_0$
**For** $i = 1$ **to** $n-1$ **do**
    $R_i \leftarrow r_i||D_i$
    $h_i \leftarrow F_{k_1}(R_{i-1}, R_i)$
    $\Sigma \leftarrow \Sigma \oplus h_i$
$R_n \leftarrow (n||r_n)$
$h_n \leftarrow F_{k_2}(R_{n-1}, R_n)$
$\tau \leftarrow \Sigma \oplus h_n$
**If** $\tau\text{'} = \tau$
    **Return** 1
**Else**
    **Return** 0
```

Algo. $\mathtt{D}_{k_1,k_2}(j, D, T)$

```
$n \leftarrow |D|_b$
$r \leftarrow T[0 : n(\ell-b)-1]$
$\tau \leftarrow T[n(\ell-b) :]$
$R_{j-1} \leftarrow r_{j-1}||D_{j-1}$
$R_j \leftarrow r_j||D_j$
$R_{j+1} \leftarrow r_{j+1}||D_{j+1}$
$h_j \leftarrow F_{k_1}(R_{j-1}, R_j)$
$h_{j+1} \leftarrow F_{k_1}(R_j, R_{j+1})$
$h'_j \leftarrow F_{k_1}(R_{j-1}, R_{j+1})$
$R_{n-1} \leftarrow r_{n-1}||D_{n-1}$
$R_n \leftarrow r_n||n$
$h_n \leftarrow F_{k_2}(R_{n-1}, R_n)$
$r'_n \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
$R'_n \leftarrow r'_n||(n-1)$
$h'_n \leftarrow F_{k_2}(R_{n-1}, R'_n)$
$\Sigma \leftarrow h_j \oplus h_{j+1} \oplus h'_j \oplus h_n \oplus h'_n$
$\tau \leftarrow \tau \oplus \Sigma$
$r \leftarrow r_0 \ldots r_{j-1}||r_{j+1} \ldots r_{n-1}||r'_n$
$T \leftarrow (r, \tau)$
```

Algo. $\mathtt{I}_{k_1,k_2}(j, D_j, D_{j-1}, D'_j, D_n, T)$

```
$n \leftarrow |D|_b$
$r \leftarrow T[0 : n(\ell-b)-1]$
$\tau \leftarrow T[n(\ell-b) :]$
$r'_j \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
$R'_j \leftarrow r'_j||D'_j$
$R_j \leftarrow r_j||D_j$
$R_{j-1} \leftarrow r_j||D_{j-1}$
$R_{n-1} \leftarrow r_{n-1}||D_{n-1}$
$R_n \leftarrow r_n||n$
$h_n \leftarrow F_{k_2}(R_{n-1}, R_n)$
$r'_{n+1} \xleftarrow{\$} \{0,1\}^{(\ell-b)}$
$R'_{n+1} \leftarrow r'_{n+1}||(n+1)$
$h'_{n+1} \leftarrow F_{k_2}(R_{n-1}, R'_{n+1})$
$h'_j \leftarrow F_{k_1}(R_{j-1}, R'_j)$
$h'_{j+1} \leftarrow F_{k_1}(R_{j'}, R_{j+1})$
$h_j \leftarrow F_{k_1}(R_{j-1}, R_j)$
$\Sigma \leftarrow h'_j \oplus h'_{j+1} \oplus h_j \oplus h_n \oplus h'_{n+1}$
$\tau \leftarrow \tau \oplus \Sigma$
$r \leftarrow r_0 \ldots r_{j-1}||r'_j||r_j \ldots r_{n-1}||r'_{n+1}$
$T \leftarrow (r, \tau)$
```

Fig. 9. Modified XOR-scheme Algorithm.