Space-efficient, byte-wise incremental and perfectly private encryption schemes

Kévin Atighehchi
ERISCS Research Group, Aix-Marseille Université
Email: kevin.atighehchi@univ-amu.fr

ABSTRACT
The problem raised by incremental encryption is the overhead due to the larger storage space required by the provision of random blocks together with the ciphered versions of a given document. Besides, permitting variable-length modifications on the ciphertext leads to privacy preservation issues. In this paper we present incremental encryption schemes which are space-efficient, byte-wise incremental and which preserve perfect privacy in the sense that they hide the fact that an update operation has been performed on a ciphered document. For each scheme, the run time of updates performed turns out to be very efficient and we discuss the statistically adjustable trade-off between computational cost and storage space required by the produced ciphertexts.
KEY WORDS: Incremental cryptography, Perfect privacy, Parallel cryptography, Secure cloud storage

I. Introduction

Incremental cryptography, introduced by Bellare, Goldreich and Goldwasser in [1], [2], [3], is used to maintain up-to-date outputs of cryptographic algorithms at low computational costs. Given the encryption of the current version of a file, it is preferable to avoid to recompute from scratch the encryption algorithm applied to the entire file whenever a modification, often minor, is performed on that file. Such a low computational efficiency for update operations finds application in various situations, as for example when we want to maintain constantly changing databases or editable documents on remote servers, such as for managing remote storage in secure clouds or mobile embedded networks. More precisely, when a document stored on a server is accessed concurrently by several users who perform some modifications on it, it is preferable for a given user that the running time to perform a local modification on the file depends as little as possible of the modifications performed by other users as well as the size of the updated document. Ideally, the run time of an update should be proportional to the amount of data changed. When a traditional, non-incremental cryptographic algorithm is used to perform a modification on the file, a critical access has to be considered and preserved for the entire file whenever a modification is performed. Note that having a "byte-wise" or "bit-wise" incremental scheme is primarily of importance. Even though this latter convention is more general, "byte-wise" incremental updates are adapted to changes in a document which often involves *replacements*, *insertions* or *deletions* of a bytes string. Moreover, an other essential interest of incremental cryptographic schemes is their inherent parallelism which will allow the use of multi-processors/cores for improving performances whenever required by the applications.

Related works: The standard authenticated encrytion mode GCM (Galois/Counter Mode [4]) does not support secure *replace* operations (although the standalone authentication scheme GMAC [5] does). Some other standard encryption or authenticated encryption schemes (such as XEX, XTS [6], OCB [7] or inc-IAPM [8]) support at best *replace* operations because they include a form of block indexation. A few incremental encryption schemes supporting efficient insert operations exist. We can distinguish one mode defined in [2] and two modes defined in [8]. Moreover, the first one [2] is not *oblivious*, that is to say, one can distinguish between a new ciphered document and an updated one. Computational efficiency is not really a problem for incremental encryption schemes. Indeed, in best cases the run time of an update is proportional to the amount of blocks changed (or inserted) in the document and remains constant when deleting blocks. The real problem of current incremental encryption schemes relates to the too large expansion of a ciphered document due to the provision of random bit strings. For instance, the best secured mode rECB [8] (for randomized ECB) produces ciphertexts that are twice larger than plaintexts and the authenticated encryption mode RPC [8] produces ciphertexts even much larger. Besides, all these modes do not allow insertion of arbitrary sized bytestrings without the need to re-align and recipher all subsequent data blocks. Some schemes supporting efficient variable-length data insertions (or deletions) by insuring the obliviousness property use a synchronisation scheme of random walks [9]. Nevertheless, the method described in [9] does not solve the problem of cryptographic form sizes.

Contributions: The AsiaCCS paper [9] has quickly introduced a generic construction to extend a block-wise incremental cryptographic scheme into a fully byte-wise one. The focus of the present paper is to deal with byte-wise incremental encryption schemes which produce smaller ciphertexts and still ensure the privacy of modifications:

- The first one is an incremental block-based encryption scheme having the ability to produce a size overhead of about (only) n bits for a document of n blocks. Its extension into a byte-wise incremental scheme can be done following the method described in [9].
- The method of [9] proposed to use a block-based incremental scheme as a black box without worrying about the sizes of produced cryptographic forms. We show that the same approach can be used to design byte-wise incremental cryptographic schemes that alleviate this problem. Contrary to the approach taken in [9], we describe here a scheme which relies on a stateless mode of encryption, which is used as a fine-grained primitive. For a particular distribution of probability, we give a new tight upper bound of the number of block-cipher evals needed when performing an update and discuss about the trade-off between average ciphertexts size and efficiency of the update.
- Then a most interesting third solution combining the advantages of the previous schemes is proposed. If we can not fully and extensively describe the algorithms by lack of space, the first constructions are presented in a logical, incremental manner that allows to easily deduce them.
- We also give proofs of their ind-CPA security. Since this paper focus on (non authenticated) encryption schemes, chosen plaintext attacks are the best attacks we can prevent against.
- For these schemes offering the same security levels, we give a brief comparison in terms of space and time efficiencies.
- We discuss their extensions into authenticated encryption schemes. In particular, we notice that when composing them with incremental MAC, the resulting incremental authenticated encryption schemes are more space-efficient than previous solutions [8].

Outline of the paper: The rest of this paper is organised as following. Required preliminaries such as, among others, precise security definitions, are given in Section 2. Our incremental modes of encryption with their efficiency analysis are each described in Section 3, 4 and 5 respectively. The proofs of security are given in Section 6. Finally, Section 8 concludes this paper.

II. Background and definitions

Encryption schemes take as input a document D which is usually divided into a sequence of fixed-size blocks $\sigma_1, ..., \sigma_n$. Cryptographic schemes were defined til now thanks to a so-called mode of operation over such blocks. Thus, documents were viewed as strings over an alphabet $\sum_B = \{0,1\}^{8N}$ where N is the given block-size in bytes.

Let us denote by T^* a set of probability measures ϕ on the set $S = \{1, ..., L\}$, where $\phi(i) > 0 \ \forall \ i \in S$. For byte-wise incremental schemes, the bytestring D is divided into variable sized blocks $(B_i)_{i=1..n}$ whose corresponding lengths $(u_i)_{i=1..n}$ follow a **strictly positive and discrete** probability distribution ϕ. Documents are then viewed as strings over an alphabet $\sum_b = \{0,1\}^{8l}_{l \sim \phi}$.

A. Updating documents

The space of modifications $\mathcal{M}_B$ defined so far was a block-wise space of modification [8]. This one allows operations such as $(delete, i)$, $(insert, i, P^*)$ or even $(replace, i, P^*)$ corresponding respectively to the deletion of the *i-th* block, the insertion of a block P^* just after index i or the replacement of the *i-th* block by P^*.

For our purpose, we have to define a byte-wise space of modification $\mathcal{M}_b$ allowing fine-grained operations $M = (substitute, i, j, \beta)$. Such an operation substitutes byte $i+1$ to byte $j-1$ (included) by β, a bytestring of any length (possibly empty). We use hereafter the following conventions:

- If $j = i+1$, this operation corresponds to an insertion just after byte i;
- If β is empty, this operation corresponds to a deletion from byte $i+1$ to byte $j-1$ (included);
- If $|\beta| = j-i-1$, this operations corresponds to a replacement from byte $i+1$ to byte $j-1$ (included).

Note that these conventions are not abusive, considering a n-byte string document $D = b_1b_2 \dots b_n$, a modification $(substitute, i, j, \beta)$ can be interpreted as taking the string $b_1b_2 \dots b_{i-1}b_ib_jb_{j+1} \dots b_{n-1}b_n$ and inserting β just after the byte index i so that we obtain the new document $D' = b_1b_2 \dots b_{i-1}b_i\beta b_jb_{j+1} \dots b_{n-1}b_n$. The resulting document after a modification operation M is denoted $D\langle M\rangle$. The resulting document after the ordered modification operations $M_1, M_2, ..., M_i$ is denoted $D\langle M_1, M_2, ..., M_i\rangle$ where $D\langle M_1, M_2, ..., M_i\rangle \equiv (((D\langle M_1\rangle)\langle M_2\rangle)...)\langle M_i\rangle$.

B. Incremental mode of encryption

Definition 1: An incremental encryption scheme is specified by a 4-tuple of algorithms $\Psi = (\mathcal{G}, \mathcal{E}, \mathcal{I}, \mathcal{D})$ in which:

- $\mathcal{G}$, the key generation algorithm, is a probabilistic polynomial time algorithm which takes as input a security parameter k and returns a symmetric key K.
- $\mathcal{E}$, the encryption algorithm, is a probabilistic polynomial time algorithm which takes as input K and a document $D \in \sum^+$ and returns the ciphertext $C = E_K(D)$.
- $\mathcal{I}$, the incremental update algorithm, is a probabilistic polynomial time algorithm which takes as input a key K, (a document D), a modification operation $M \in \mathcal{M}$, and the encrypted form C (related to D) and returns the modified ciphertext C'.
- $\mathcal{D}$, the decryption algorithm, is a deterministic polynomial time algorithm which takes as input a key K and a ciphertext $C = E_K(D)$ and returns a document D.

The behaviour of an incremental encryption scheme is depicted in the commutative diagram Figure 1. Considering a modified document $D' = D\langle M\rangle$, it is required that $\mathcal{D}_K(\mathcal{I}_K(M, D, \mathcal{E}_K(D))) = D'$. Note that the input document D is shown in brackets in the update algorithm of the above definition. The reason is that, depending on the fact whether we have a block-based or a byte-wise incremental encryption scheme, as well as the convention used in the implementation, $\mathcal{I}$ may or may not require access to the document D.

$$\begin{tikzcd}
D \arrow[r, bend left, "\mathcal{E}_K"] \arrow[d, "M"'] & C \arrow[l, bend left, "\mathcal{D}_K"] \arrow[d, "\mathcal{I}_K{,}M"] \\
D' \arrow[r, bend left, "\mathcal{E}_K"] & C' \arrow[l, bend left, "\mathcal{D}_K"]
\end{tikzcd}$$

Fig. 1: Incremental mode of encryption

For example, let us consider the use of a random permutation E_K, where a specific instantiation will be a 16-byte block cipher such as AES. We then recall the encryption phase rECB of a block-based incremental authenticated encryption scheme defined according to the *encrypt-then-MAC* composition in [8]. This scheme has the property to be perfectly private. Given a document D parsed as a sequence of 16-byte blocks $D_1, \dots, D_n$, the encryption algorithm $\mathcal{E}$ is constructed in the following way:

1) For each $i = 0, ..., n$ we pick r_i uniformly at random. The randomized input is $R = r_0r_1...r_n$.
2) Let $C_0 = E_K(r_0)$. For each $i = 1, ..., n$ let $C_i = (E_K(r_i \oplus r_0),\ E_K(r_i \oplus D_i))$. The ciphertext is $C = C_0C_1...C_n$.

The encrypted message is simply C. Concerning the incremental algorithm $\mathcal{I}$, an insertion (or replacement) of one block is simply done by generating a new random value r^* (or respectively regenerating the existing one) and performing two block-cipher operations.

Table[1] I summarises efficiency of schemes proposed by [8] supporting *delete* and *insert* operations. Note that RPC is an authenticated encryption scheme following the model of integrated checksum. Concerning rECB-XOR, if we extract the standalone encryption scheme rECB from rECB-XOR, one can notice that there is no need to cipher the random values, indistinguishability will be always ensured even without this overhead of encryption. The real problem with incremental cryptography is that we have to generate and supply a lot of random values to remove dependencies between the ciphered blocks and thus allowing efficient update operations. This can result, as in the case of rECB, in ciphertexts twice as long as plaintexts.

Algorithms	**Ciphertext size**	Block cipher evals	Indistinguishability
rECB-XOR (rECB composed with the incremental XORMAC [2])	$\mathbf{4n}$	$6n$	$\frac{\mu l}{2^{128}}$
rECB	$\mathbf{2n}$	$2n$	$\frac{\mu l}{2^{128}}$
rECB*	$\mathbf{2n}$	n	$\frac{\mu l}{2^{128}}$
RPC	$\mathbf{4n}$	$4n$	$\frac{q_{inc}}{2^{48}}$

TABLE I: Summary of incremental modes of operation proposed by Buonanno et *al*, instantiated with a 128-bit block cipher. The parameters n, l and μ are expressed in number of blocks. Queries (q_{inc}) to the update oracle concern only one block.

C. Indistinguishability

For a traditional encryption scheme, indistinguishability measures the un-ability for an adversary to distinguish ciphertexts. In the case of an incremental encryption scheme, we consider that an adversary must in addition be unable to distinguish modifications performed on a same ciphertext. Besides, for insuring indistinguishability of modifications, some stringent conditions have to be applied. Indeed, if the incremental algorithm $\mathcal{I}$ has a good running time complexity (for instance, linear in the amount of changes), some parts of a modified ciphertext remain unchanged. So, we require these modifications to be of the same type, same lengths and performed at the same location in the plaintext. Otherwise the adversary makes no effort to distinguish between them by looking the resulting updated ciphertext.

We define an adversary A in a *find-then-guess* game where the incremental update algorithm is taken into account. This model of security was described for the first time in [8] and defines such an adversary as a two-phase algorithm:

1) "find" phase : A makes queries to its encryption oracle $\mathcal{E}_K(.)$ and updating oracle $\mathcal{I}_K(.,.,.)$ and eventually submits to the challenger either a pair of distincts chosen plaintext (D_0, D_1) of same length l, or a ciphertext C with a pair of modification operations (M_0, M_1) (of the same type and modifying the same location).
2) "guess" phase : The challenger selects a bit $b \in \{0, 1\}$ uniformly at random, encrypts the message D_b with $\mathcal{E}_K$ or applies the update M_b on C with $\mathcal{I}_K$, and the result is given to A which may then make more oracle queries. Finally A must output a guess for the value of b.

We are interested in the property of indistinguishability under an adaptive-chosen-plaintext attack (CPA). The adversary wins if it correctly identifies which of the two documents has been ciphered (or which one of the two modifications has been applied) in the challenge. The encryption scheme is said to be secure if reasonable adversaries cannot win significantly more than half the time. In the following, we define precisely two experiments, one for ciphertext indinstinguishability and an other one for modification indistinguishability.

[1] Since we will refer to these elements of comparison in the rest of the paper, please note that the number n can represent the size of the document either in number of 16-byte blocks or in number of bytes and change the number of cipher evals as appropriate.

Message secrecy: Let Ψ be an incremental encryption scheme whose security parameter is k. Let $A = (A_1, A_2)$ be an adversary that has access to the oracles $\mathcal{E}_K(.)$ and $\mathcal{I}_K(.,.,.)$. Now, let us consider the following random experiment: k being set to a fixed value, we run the algorithm $\mathcal{G}$ to obtain a secret key K. The algorithm A_1 takes as input k and outputs a triplet (D_0, D_1, s). The components D_0 and D_1 are two distinct plaintexts of same length l, the component s contains information about the system state, known by A_1, which will be passed to A_2. A bit b_0 is chosen uniformly at random in $\{0, 1\}$ and is kept secret from A_2. The encryption algorithm $\mathcal{E}_K$ is then launched over the plaintext D_{b_0} and returns a ciphertext C which constitutes the *challenge ciphertext* proposed to the algorithm A_2. Thus, A_2 has as inputs (D_0, D_1, s, C). Eventually, the algorithm A_2 outputs a bit b (whose the adversary hopes it equal to b_0).

Keeping in mind these notations, the random experiment described previously can be detailed in the following way:

```
$\mathbf{Expt}^{\mathbf{I\text{-}CPA}}_{\mathbf{\Psi,FG}}(\mathbf{A}, \mathbf{k})$
    $K \leftarrow \mathcal{G}(k)$
    $(D_0, D_1, s) \leftarrow A_1(k)$
($D_0$ and $D_1$ are distincts and of same length)
    $b_0 \leftarrow \{0, 1\}$
    $C \leftarrow \mathcal{E}_K(D_{b_0})$
    $b \leftarrow A_2(D_0, D_1, s, C)$
    if $b = b_0$ then return $1$ else return $0$
```

Definition 2: Let $\Psi = (\mathcal{G}, \mathcal{E}, \mathcal{I}, \mathcal{D})$ be an incremental encryption scheme over modification space $\mathcal{M}$, and let A be an adversary for an attack CPA. We define the adversary advantage $Adv_{\Psi}^{I\text{-}CPA}$ by :

$$Adv_{\Psi}^{I\text{-}CPA}(A, k) = \left| \Pr\left(Expt_{\Psi,FG}^{I\text{-}CPA}(A, k) = 1 \right) - \frac{1}{2} \right| .$$

We say that Ψ is $(t, q_e, \mu, q_{inc}, \epsilon)$-secure in the sense of $I\text{-}CPA$ if, for any adversary A which runs in time t, making q_e queries to the $\mathcal{E}_K(.)$ oracle and q_i valid queries to the $\mathcal{I}_K(.,.,.)$ oracle (with the total number of ciphered data[2] in all encryption and incremental update queries equal to μ), $Adv_{\Psi}^{I\text{-}CPA}(A, k)$ is less than ϵ.

Update secrecy: Let Ψ be an incremental encryption scheme whose security parameter is k. Let $A = (A_1, A_2)$ be an adversary that has access to the oracles $\mathcal{E}_K(.)$ and $\mathcal{I}_K(.,.,.)$. Now, let us consider the following random experiment: k being set to a fixed value, we run the algorithm $\mathcal{G}$ to obtain a secret key K. The algorithm A_1 takes as input k and returns a 4-tuple (C, M_0, M_1, s) where M_0 and M_1 are two possible modifications on D (of the same type and modifying the same location) and where s contains information about the system state, known by A_1, which will be passed to A_2. A bit b_0 is chosen uniformly at random in $\{0, 1\}$ and is kept secret from A_2. The algorithm of modification $\mathcal{I}_K$ is then launched over M_{b_0}, D, C and returns an updated ciphertext $C' = \mathcal{I}_K(M_{b_0}, D, C)$. The ciphertext C' constitutes the *challenge update* proposed to the algorithm A_2. Thus, the algorithm A_2 takes as inputs (M_0, M_1, s, D, C, C') and outputs a bit b (whose the adversary hopes it equal to b_0).

Keeping in mind these notations, the random experiment described previously can be detailed in the following way (**IM-CPA** stands for indistinguishability of modifications under CPA attack):

[2] An appropriate unit of measurement is selected, as the case might be.

```
$\mathbf{Expt}^{\mathbf{IM\text{-}CPA}}_{\mathbf{\Psi,FG}}(\mathbf{A},\mathbf{k})$
    $K \leftarrow \mathcal{G}(k)$
    $(C, M_0, M_1, s) \leftarrow A_1(k)$
($M_0 = (substitute, i_0, j_0, \beta_0)$ and
$M_1 = (substitute, i_1, j_1, \beta_1)$ are such
that $i_0 = i_1$, $j_0 = j_1$ and $|\beta_0| = |\beta_1|$)
    $b_0 \leftarrow \{0, 1\}$
    $C' \leftarrow \mathcal{I}_K(M_{b_0}, D, C)$
    $b \leftarrow A_2(M_0, M_1, s, D, C, C')$
    if $b = b_0$ then return $1$ else return $0$
```

Definition 3: Let $\Psi = (\mathcal{G}, \mathcal{E}, \mathcal{I}, \mathcal{D})$ be an incremental encryption scheme over modification space $\mathcal{M}$, and let A be an adversary for an attack CPA. We define the adversary advantage $Adv^{IM\text{-}CPA}_{\Psi}$ by :

$$Adv^{IM\text{-}CPA}_{\Psi}(A, k) = \left| \Pr\left(Expt^{IM\text{-}CPA}_{\Psi,FG}(A, k) = 1\right) - \frac{1}{2} \right|.$$

We say that Ψ is $(t, q_e, \mu, q_{inc}, \epsilon)$-secure in the sense of $IM\text{-}CPA$ if, for any adversary A which runs in time t, making q_e queries to the $\mathcal{E}_K(.)$ oracle and q_i valid queries to the $\mathcal{I}_K(., ., .)$ oracle (with the total number of ciphered data in all encryption and incremental update queries equal to μ), $Adv^{IM\text{-}CPA}_{\Psi}(A, k)$ is less than ϵ.

We provide in Table I the security of the schemes from [8]. As can be observed, the combined authenticated encryption rECB-XOR and the standalone encryption rECB are secure. However, this is not the case for RPC which, moreover, suffers from a high expansion of the ciphertext. If this expansion is parametrisable, trying to reduce it worsens the security of the scheme.

D. Perfect privacy

A simple way to design an incremental encryption algorithm is the following. Instead of applying modifications to the plaintext message and recipher the new one from scratch, take the current version of the ciphertext and append a ciphered description of modifications to obtain the new ciphertext. Such a scheme may be not acceptable since it is not *history free* in the sense that it can reveal to someone who knows the decryption key all previous versions of the document. Moreover, this method is not efficient and produces a ciphertext which is becoming larger at each modification.

Suppose Alice sends a ciphertext to Bob. The latter might be disappointed if he realizes that the ciphered document has been obtained by an incremental update. This problem could arise with documents such as commitments, contracts whose cryptographic forms must not disclose any information at all about modifications, not even the fact that update operations have been performed.

We will say that an incremental encryption scheme is *oblivious* (or *perfectly private*) if the behaviour of the composition of application of the encryption algorithm followed by incremental update operations is indistinguishable from the behaviour of the application of the encryption algorithm alone. A formal definition, taken from [10], [8], is the following:

Definition 4: Let Ψ be an incremental encryption scheme over modification space $\mathcal{M}$. We say that Ψ is oblivious (or perfectly private) if, for any two documents D, D_i, for any sequence of modifications M_1, ..., $M_i \in \mathcal{M}$ such that $D_i = D\langle M_1, \ldots, M_i \rangle$, and for all keys K, we have

$$\{\mathcal{E}_K(D_i)\} \equiv \{\mathcal{I}_K(M_i, D_i, ..., \mathcal{I}_K(M_1, D, \mathcal{E}_K(D))...)\}.$$

Note that we could be interested in computational indistinguishability between encrypted documents and updated ones. Let us consider a *privacy* property defined by the following simple game: the adversary

A interacts with its encryption oracle $\mathcal{E}_K(.)$ and updating oracle $\mathcal{I}_K(.,.,.)$ and eventually submits to the challenger a document D along with a modification $M \in \mathcal{M}$. The challenger selects a bit $b_0 \in \{0,1\}$ uniformly at random. If $b_0 = 0$ then the challenger returns $\mathcal{E}_K(D\langle M\rangle)$, otherwise it returns $\mathcal{I}_K(M, D, \mathcal{E}_K(D))$. The result is given to A which may then make more oracle queries. Finally A must output a guess b for the value of b_0. We consider that A wins if $b_0 = b$ and we say that the scheme is *private* if reasonable adversaries cannot win significantly more than half the time. We will denote by $\mathbf{Expt}_{\mathbf{\Psi}}^{\mathbf{Priv\text{-}CPA}}(\mathbf{A}, \mathbf{k})$ the corresponding experiment which returns 1 if $b_0 = b$ and 0 otherwise.

Definition 5: Let $\Psi = (\mathcal{G}, \mathcal{E}, \mathcal{I}, \mathcal{D})$ be an incremental encryption scheme over modification space $\mathcal{M}$, and let A be an adversary for an attack CPA. We define the adversary advantage $Adv_{\Psi}^{Priv\text{-}CPA}$ by :

$$Adv_{\Psi}^{Priv\text{-}CPA}(A,k) = \left| \Pr\left(Expt_{\Psi}^{Priv\text{-}CPA}(A,k) = 1 \right) - \frac{1}{2} \right|.$$

We say that Ψ is $(t, q_e, \mu, q_{inc}, \epsilon)$-secure in the sense of $Priv\text{-}CPA$ if, for any adversary A which runs in time t, making q_e queries to the $\mathcal{E}_K(.)$ oracle and q_i valid queries to the $\mathcal{I}_K(.,.,.)$ oracle (with the total number of ciphered data in all encryption and incremental update queries equal to μ), $Adv_{\Psi}^{Priv\text{-}CPA}(A,k)$ is less than ϵ. If $Adv_{\Psi}^{Priv\text{-}CPA}(A,k) = 0$ we say that Ψ is perfectly private.

The case of byte-wise incremental schemes: A simple way to construct a byte-wise incremental encryption scheme is to use a block-based incremental encryption scheme in which we assign only one byte of the document per block, but at the cost of both a large number of block-cipher evals and a very large ciphertext. A more interesting solution is to assign a variable number (chosen from a probability distribution) of contiguous bytes of the document per block in order to decrease both the computational and the size overheads. Concerning the privacy of modifications performed on a document, an adversary should not find a scenario of successive modifications which leads to a bias in the distribution of block lengths. Thus, an incremental update algorithm has to ensure that statistical tests will not reveal outlying regions in the sequence of variable-length parts and therefore the location of insertions. Secondly, concerning the size of the document, it has to conserve (on average) the overhead for both the ciphertext size and the number of operations to perform in the decryption algorithm. This should be implied by the perfect privacy property.

Definition 6: An incremental variable-length block-based encryption scheme is perfectly private if and only if: (i) The distribution of blocks' lengths does not depend on information about modifications performed, that is, update operations are implemented so that this distribution is preserved; (ii) The way to operate these blocks during an update is itself perfectly private, that is, after a modification, the set of relations between these blocks is indistinguishable from the set of relations between the blocks of a non-updated message.

E. Relation among these notions

It is not difficult to see that if an incremental encryption scheme Ψ is perfectly private and ensures update secrecy then this same scheme ensures message secrecy as well. Intuitively, if Ψ is perfectly private an updated encryption is indistinguishable from an initial encryption and we could obtain a large-sized encrypted message by essentially making several updates on a initial small encrypted message[3]. In such a situation distinguishing the encryption of two distinct messages (of same length) is almost the same as distinguishing two distinct updates (of the same type and modifying the same location in the document). Note that this property is particularly usefull when providing proofs of security and constructing algorithms. Despite this, the paper stays conservative in the description of our incremental schemes by giving different algorithms for encryption and update and by providing proofs of security for perfect privacy, update secrecy and message secrecy while this latter appears to be redundant.

Theorem 1: Let $\Psi = (\mathcal{G}, \mathcal{E}, \mathcal{I}, \mathcal{D})$ be an incremental encryption scheme over modification space $\mathcal{M}$. If Ψ is $(t, q_e, \mu, q_{inc}, \epsilon)$-secure in the sense of $IM\text{-}CPA$ and $(t', q'_e, \mu', q'_{inc}, \epsilon')$-secure in the sense of $Priv\text{-}CPA$ then Ψ is also $(t+t', q_e+q'_e, \mu+\mu', q_{inc}+q'_{inc}, \epsilon+\epsilon')$-secure in the sense of $I\text{-}CPA$.

[3] To be more explicit in the underlying idea, we could imagine an intial empty message.

Proof of theorem 1: Consider the message secrecy experiment $\mathbf{Expt^{I\text{-}CPA}_{\Psi,FG}(A,k)}$ and change it a little as follows: the algorithm A_1 takes as input k and outputs the triplet (D, M_0, M_1, s) where M_0 and M_1 are two modifications (of the same type and modifying the same location) such that $D\langle M_0\rangle$ and $D\langle M_1\rangle$ are two distinct documents. Note that these modifications could change D completely. A bit b_0 is chosen uniformly at random in $\{0,1\}$ and is kept secret from A_2. A random ciphertext C of $D\langle M_{b_0}\rangle$ which constitutes the *challenge ciphertext* is then proposed to the algorithm A_2. This slightly changed experiment, denoted $\mathbf{Expt\text{-}b^{I\text{-}CPA}_{\Psi,FG}(A,k)}$, can be detailed in the following way:

```
$\mathbf{Expt\text{-}b^{I\text{-}CPA}_{\Psi,FG}(A,k)}$
    $K \leftarrow \mathcal{G}(k)$
    $(D, M_0, M_1, s) \leftarrow A_1(k)$
($M_0 = (substitute, i_0, j_0, \beta_0)$ and
$M_1 = (substitute, i_1, j_1, \beta_1)$ are such
that $i_0 = i_1$, $j_0 = j_1$ and $|\beta_0| = |\beta_1|$)
    $b_0 \leftarrow \{0,1\}$
    $C \leftarrow \mathcal{E}_K(D\langle M_{b_0}\rangle)$
    $b \leftarrow A_2(D, M_0, M_1, s, C)$
    if $b = b_0$ then return $1$ else return $0$
```

This new experiment being equivalent to the previous one, we therefore have:

$$\Pr\left(Expt\text{-}b^{I\text{-}CPA}_{\Psi,FG}(A,k) = 1\right) = \Pr\left(Expt^{I\text{-}CPA}_{\Psi,FG}(A,k) = 1\right).$$

On the basis of the last defined experiment, make the following change which consists to replace the challenge encryption $\mathcal{E}_K(D\langle M_{b_0}\rangle)$ by the composition of an encryption followed by an update $\mathcal{I}_K(M_{b_0}, D, \mathcal{E}_K(D))$. This new experiment, denoted $\mathbf{Expt\text{-}b^{IM\text{-}CPA}_{\Psi,FG}(A,k)}$, can be detailed in the following way:

```
$\mathbf{Expt\text{-}b^{IM\text{-}CPA}_{\Psi,FG}(A,k)}$
    $K \leftarrow \mathcal{G}(k)$
    $(D, M_0, M_1, s) \leftarrow A_1(k)$
($M_0 = (substitute, i_0, j_0, \beta_0)$ and
$M_1 = (substitute, i_1, j_1, \beta_1)$ are such
that $i_0 = i_1$, $j_0 = j_1$ and $|\beta_0| = |\beta_1|$)
    $b_0 \leftarrow \{0,1\}$
    $C' \leftarrow \mathcal{I}_K(M_{b_0}, D, \mathcal{E}_K(D))$
    $b \leftarrow A_2(D, M_0, M_1, s, C')$
    if $b = b_0$ then return $1$ else return $0$
```

Assume that we have a distinguishing algorithm $Dist$ which takes as input a bit a. If $a = 0$ $Dist$ runs $\mathbf{Expt\text{-}b^{I\text{-}CPA}_{\Psi,FG}(A,k)}$, otherwise it runs $\mathbf{Expt\text{-}b^{IM\text{-}CPA}_{\Psi,FG}(A,k)}$. By assumption of an incremental encryption scheme ensuring the privacy of modifications, the distinguishing advantage of $Dist$ is at most $Adv^{Priv\text{-}CPA}_{\Psi}(k)$ where $Adv^{Priv\text{-}CPA}_{\Psi}(k) = \max_A\{Adv^{Priv\text{-}CPA}_{\Psi}(A,k)\}$.

We note, moreover, that in the "guess" phase of the experiment $\mathbf{Expt\text{-}b^{IM\text{-}CPA}_{\Psi,FG}(A,k)}$ the intermediate ciphertext C (before update) is not known to A_2. Consequently, we have $\left|\Pr\left(Expt\text{-}b^{IM\text{-}CPA}_{\Psi,FG}(A,k) = 1\right) - \frac{1}{2}\right| \leq \left|\Pr\left(Expt^{IM\text{-}CPA}_{\Psi,FG}(A,k) = 1\right) - \frac{1}{2}\right|$.

We then conclude from the triangle inequality that: $Adv^{I\text{-}CPA}_{\Psi}(A,k) \leq Adv^{Priv\text{-}CPA}_{\Psi}(A,k)+Adv^{IM\text{-}CPA}_{\Psi}(A,k) \leq \epsilon' + \epsilon$. ■

III. A SPACE-EFFICIENT BLOCK-WISE INCREMENTAL ENCRYPTION SCHEME

In this section we describe our block-wise incremental encryption scheme that we will call *swr*ECB, as the idea is to use a *sliding window over the randomizers*. Depending on the parametrisation used, this scheme can be quite space-efficient. Let us consider a pseudorandom functions family F with input-length and output-length both equal to $8N$. Let us also consider an instance F_K of F and two fixed integers e, $d \in \mathbb{N}^{*2}$ such that $8N = ed$ with $d \geq 2$. We assume the use of a key generation algorithm that takes as input a security parameter k and returns a symmetric key K. In the following subsections we describe the three remaining algorithms.

A. Encryption and decryption algorithms

The algorithm 1 describes the encryption operation. It takes as input a key K and a document of n blocks of size $8N$-bits. First of all, it generates uniformly at random $n + d - 1$ blocks of size e-bits. Then it applies F_K on the concatenation of the first d random blocks, and XORes (Exclusive-Or) the result with the first plaintext block to obtain the first ciphered block. Then, it repeatedly performs the following steps: it considers the last $d - 1$ random blocks of the current window and the immediately following random block. F_K is applied on the concatenation of these d “shifted” random blocks and the result is Xored with the following plaintext block to obtain the corresponding ciphered block.

Algorithm 1 $\mathcal{E}$

```
Input: A blockstring $D = \{P_1 P_2 \dots P_n\}$, a key $K$
for $j = 1 \to d - 1$ do
    $r_j \leftarrow \{0,1\}^e$;
for $j = 1 \to n$ do
    $r_{d-1+j} \leftarrow \{0,1\}^e$;
    $C_j \leftarrow F_K(r_j \| r_{j+1} \| \dots \| r_{j+d-1}) \oplus P_j$;
return $((r_i)_{i=1 \dots n+d-1}, (C_i)_{i=1 \dots n})$;
```

The algorithm 2 describes the deterministic decryption operation. It takes as input a key K, the ciphered document and applies the same operations that the encryption algorithm by replacing the plaintext by the ciphertext.

Algorithm 2 $\mathcal{D}$

```
Input: A ciphertext $((r_i)_{i=1 \dots n+d-1}, (C_i)_{i=1 \dots n})$, a key $K$
for $j = 1 \to n$ do
    $P_j \leftarrow F_K(r_j \| r_{j+1} \| \dots \| r_{j+d-1}) \oplus C_j$;
return $(P_i)_{i=1 \dots n}$;
```

B. Incremental update algorithms

As seen before, by sliding a window over the sequence $(r_i)_{i=1 \dots n+d-1}$, we can obtain n blocks of size $8N$ bits. For convenience, we will refer to a window of index i as the block $r_i \| r_{i+1} \| \dots \| r_{i+d-1}$. The update operations (algorithms 3, 4 and 5) are described on a case-by-case basis due to different (cautious) approaches when dealing with the involved random blocks. As we can see, when a random block is affected, a certain number of upstream and downstream windows need to be reevaluated and the corresponding plaintext blocks reciphered. Besides, for security purposes in the algorithms 4 and 5, we mandate the redrawing of all the random blocks in the window which serves to cipher the replaced (or inserted) block.

C. Efficiency analysis

Certain assumptions, like the fact that algorithms 3, 4 and 5 describe modifications of only one block, have been taken for simplicity. In the followings, we discuss the efficiency of more general algorithms:

- The algorithm 3 assumes the presence in the document of $d-1$ blocks before the deleted one. If there is only k blocks before block m with $k<d-1$, only k evaluations of F_K is in fact required. Besides, none of the random blocks need to be regenerated. The extension of this algorithm to the deletion of any number q of contiguous blocks is clear. Whatever this number is, the number of windows affected is at most $d-1$. Consequently, the number of evaluations of F_K stays to (at most) $d-1$.

Algorithm 3 Deletion of a block P_m

```
Input: the ciphertext $((r_i)_{i=1\dots n+d-1}, (C_i)_{i=1\dots n})$, the document $D$, a key $K$
for $i = m-(d-1) \rightarrow m-1$ do
    $C'_i \leftarrow F_K(r_i \| r_{i+1} \| \dots \| r_{m-1} \| r_{m+1} \| \dots \| r_{i+d}) \oplus P_i$;
return $((r_i)_{i=1\dots m-1}, (r_i)_{i=m+1\dots n+d-1}, (C_i)_{i=1\dots m-d},$
       $(C'_i)_{i=m-(d-1)\dots m-1}, (C_i)_{i=m+1\dots n})$;
```

- In the algorithm 4 the d random blocks contained in the window of index m need to be regenerated. The algorithm described assumes the presence in the document of $d-1$ blocks before and after block m. If there is only k blocks before block m and only l blocks after it with $k<d-1$ and $l<d-1$, only $k+l+1$ evaluations of F_K are in fact required. The extension of this algorithm to the replacement of any number q of contiguous blocks implies the regeneration of at most $q+d-1$ random blocks, affecting at most $q+2(d-1)$ windows. We deduct a number of evaluations of F_K of at most $q+2(d-1)$.

Algorithm 4 Replacement of a block P_m by P'_m

```
Input: the ciphertext $((r_i)_{i=1\dots n+d-1}, (C_i)_{i=1\dots n})$, the document $D$, a key $K$
for $j = m \rightarrow m+d-1$ do
    $r'_j \leftarrow \{0,1\}^e$;
for $j = m-(d-1) \rightarrow m-1$ do
    $C'_j \leftarrow F_K(r_j \| \dots \| r_{m-1} \| r'_m \| \dots \| r'_{j+d-1}) \oplus P_j$;
$C'_m \leftarrow F_K(r'_m \| r'_{m+1} \| \dots \| r'_{m+d-1}) \oplus P'_m$;
for $j = m+1 \rightarrow m+d-1$ do
    $C'_j \leftarrow F_K(r'_j \| \dots \| r'_{m+d-1} \| r_{m+d} \| \dots \| r_{j+d-1}) \oplus P_j$;
return $((r_i)_{i=1\dots m-1}, (r'_i)_{i=m\dots m+d-1},$
       $(r_i)_{i=m+d\dots n+d-1}, (C_i)_{i=1\dots m-1},$
       $(C'_i)_{i=m-(d-1)\dots m+d-1}, (C_i)_{i=m+d\dots n})$;
```

- In the algorithm 5 the $d-1$ random blocks that follow r_m are regenerated and a new random block of size e-bits is inserted between r_m and r_{m+1}. Always with a view to simplification, the algorithm assumes the presence in the document of $d-2$ blocks before the block P_m and $d-1$ blocks after. Taking into account this insertion, a new window is appearing and $2d-1$ windows are affected so that at most $2d$ evaluations of F_K are required. The extension of this algorithm to the insertion of any number q of contiguous blocks implies the generation of q random blocks, bringing the number of evaluations of F_k to at most $q+2(d-1)$.

Algorithm 5 Insertion of a block P' right after P_m

```
Input: the ciphertext $((r_i)_{i=1\dots n+d-1}, (C_i)_{i=1\dots n})$, the document $D$, a key $K$
$r' \leftarrow \{0,1\}^e$;
for $j = m+1 \rightarrow m+d-1$ do
    $r'_j \leftarrow \{0,1\}^e$;
$C'_{m-(d-2)} \leftarrow F_K(r_{m-(d-2)} \| \dots \| r_m \| r') \oplus P_{m-(d-2)}$;
for $j = m-(d-3) \rightarrow m$ do
    $C'_j \leftarrow F_K(r_j \| \dots \| r_m \| r' \| r'_{m+1} \| \dots \| r'_{j+d-2}) \oplus P_j$;
$C' \leftarrow F_K(r' \| r'_{m+1} \| \dots \| r'_{m+d-1}) \oplus P'$;
for $j = m+1 \rightarrow m+d-1$ do
    $C'_j \leftarrow F_K(r'_j \| \dots \| r'_{m+d-1} \| r_{m+d} \| \dots \| r_{j+d-1}) \oplus P_j$;
return $((r_i)_{i=1\dots m-1}, r', (r'_i)_{i=m+1\dots m+d-1},$
       $(r_i)_{i=m+d\dots n+d-1}, (C_i)_{i=1\dots m-(d-1)},$
       $(C'_i)_{i=m-(d-2)\dots m}, C', (C'_i)_{i=m+1\dots m+d-1},$
       $(C_i)_{i=m+d\dots n})$;
```

The space occupied by the random blocks $(r_i)_{i=1\dots n+d-1}$ is exactly $(ne + (d-1)e)$ bits. This is to be compared with the overhead of $8nN$ bits consumed by rECB. In other words, this means that the expansion of a ciphertext produced by swrECB is about $1 + \frac{1}{d}$ whereas it is exactly a factor 2 when using rECB. The ciphering of a n-block document requires n evaluations of F_K. Finally, the number of evaluations of F_K for update operations is summarised in the following table:

Deletion of q block	Replacement of q block	Insertion of q block
$d-1$	$q+2(d-1)$	$q+2(d-1)$

IV. A SPACE-EFFICIENT BYTE-WISE INCREMENTAL ENCRYPTION SCHEME

In this Section, we describe our space-efficient, byte-wise incremental and perfectly private incremental encryption scheme. Unlike the previous scheme, this one relies on a stateless mode of encryption, that is to say, a mode in which we do not maintain a state when ciphering the successive messages (the initialization

D	the document is now seen as a bytestring
β	bytestring to insert
$D[a, b]$	substring of D from the byte a up to the byte b (included)
$u_i \overset{\phi}{\leftarrow} \{1, \dots, L\}$	u_i is drawn from the set $\{1, \dots, L\}$ according to ϕ
B_i	i-*th* variable length block of size u_i
$u = (u_i)_{i=1..n}$	list/sequence of variable lengths u_i
$\overline{D} = (B_i)_{i=1..n}$	partitioned form of the bytestring D (sequence of parts B_i)
$\lvert . \rvert$	size of a data in bytes or number of elements in a sequence
$\lvert u \rvert$	number of parts in $\overline{D}$
$\lvert \beta \rvert$	size of the bytestring β, in number of bytes
k_0	index of a part after which the repartition starts
k_1	index of a part part before which the repartition terminates
u_0, $u_{\lvert u \rvert+1}$	sentinel lengths used by convention, and valued at 0
C_0, $C_{\lvert u \rvert+1}$	sentinel ciphered parts used by convention, whose contents do not matter
r_0, $r_{\lvert u \rvert+1}$	sentinel random vectors used by convention, whose values do not matter

TABLE II: Notations and conventions for mcXOR

vector is chosen at random each time). As previously, we assume the use of a key generation algorithm that takes as input a security parameter k and returns a symmetric key K. In the following subsections we describe the three remaining algorithms, namely, encryption, decryption and update algoritms.

A. Notations

The main notations used are described in Table II. The use of sentinel indices are necessary in the case of modification at the very beginning or the end of the document. For example, if we consider a n-byte document, they will allow us to consider modifications of type $(substitute, i, j, \beta)$ with $i = 0$ or $j = n+1$.

B. Encryption and decryption algorithms

The bytes string D is divided into variable sized blocks $(B_i)_{i=1..n}$ whose corresponding lengths $(u_i)_{i=1..n-1}$ (except the last term) follow a discrete probability distribution ϕ on a set $\{1, \ldots, L\}$. Obviously, the size S of D being fixed, an initially empty partition B is in practice built by repeating the following operations: (i) Draw a number a from $\phi([1, L])$ and set $S = S - a$; (ii) If $S > 0$ insert into B the part containing the next a bytes. Otherwise, terminate by inserting the part of the remaining $S + a$ bytes.

Stateless modes of encryption described in [11], such as OFB, CBC or XOR can be applied on variable numbers of contiguous bytes of D so that a randomizer, used as an initial vector for the considered mode, does not serve for one block but for several. The process is as follows. We partition the document into several groups of variable number of contiguous bytes, the number being choosen according to a discrete probability distribution (parameterized by a multiple lN of the block size, for instance $\mathcal{U}([1,\ lN])$). Then we cipher independently each group with a stateless mode of operation from [11], [12] as if they were different messages. Modes of operation that behave like a synchronous stream cipher (see stream cipher modes such as OFB or XOR [11]), for which the plaintext is masked with a generated keystream, are preferable for the reduction of the ciphertext expansion. We give an example of algorithm using the "stream cipher like" mode XOR (stateless version of CTR, sometimes called randomized CTR) instantiated with a block cipher, a good choice to keep a high degree of parallelism.

Let suppose a function $XOR.E_K$ which implements the encryption operation of XOR, takes as input a symmetric key K, a message M and returns the ciphered message C of the same size together with the associated random initialization vector IV. We do not recall the description of this well known algorithm [12] and we assume that the underlying pseudorandom functions family used is F. The encryption operation, denoted mcXOR (multiple calls to the XOR mode) is described in Algorithm 6.

Algorithm 6 $\mathcal{E}$

```
Input: A bytestring $D = \{b_1 b_2 \ldots b_n\}$, a key $K$
$n \leftarrow |D|$; $e \leftarrow 1$; $k \leftarrow 1$; $j \leftarrow 1$;
while $e \neq 0$ do
    $u_j \overset{\phi}{\leftarrow} \{1, \ldots, lN\}$;
    if $n - (k + u_j) > 0$ then
        $(r_j, C_j) \leftarrow XOR.E_K(D[k, k + u_j - 1])$;
        $k \leftarrow k + u_j$; $j \leftarrow j + 1$;
    else
        $(r_j, C_j) \leftarrow XOR.E_K(D[k, n])$;
        $u_j \leftarrow n - k + 1$; $e \leftarrow 0$;
$C \leftarrow C_1 || C_2 || \ldots || C_j$;
return $((r_i)_{i=1 \ldots j}, (u_i)_{i=1 \ldots j}, C)$;
```

If we denote j_0 the value of j at the termination of the algorithm, the resulting ciphertext is composed of the sequences of random vectors $(r_j)_{j=0 \ldots j_0}$, blocks lengths $(u_j)_{j=0 \ldots j_0}$ and ciphered bytes C. Let suppose a function $XOR.D_{K,IV}$ which implements the decryption operation of XOR, takes as input a symmetric key K, a ciphertext C together with the associated random IV and returns the corresponding plaintext D of the same size. The deterministic decryption is described in Algorithm 7.

Algorithm 7 $\mathcal{D}$

```
Input: A ciphertext $((r_i)_{i=1...j_0}, (u_i)_{i=1...j_0}, C)$, a key $K$
$k \leftarrow 1$;
for $i = 1 \rightarrow j_0$ do
    $B_i \leftarrow XOR.D_{K,r_i}(C[k, k + u_j - 1])$; $k \leftarrow k + u_j$;
$D \leftarrow B_1 || B_2 || \dots || B_{j_0}$;
return $D$
```

C. Incremental update algorithm

We recall that a random step in a walk corresponds to a random draw $u_i \sim \phi$. Let us assume that we have to insert a bytestring β somewhere between the first byte of B_{k-1} and the first byte of B_k in the sequence of variable length blocks $B_1, \dots, B_n$. The first approach which simply consists to partition β in the same manner as we partition the document and insert $\overline{\beta}$ at the good location in $\overline{D}$ (possibly decomposing B_{k-1} in two parts) is not sufficient. Indeed, this method leads to a bias in the resulting distribution of blocks' lengths after multiple insertions. For the same reasons, any other method which focus strictly on the partitioning of β is a losing proposition. The approach defined in [9] proposes to realise a synchronization of random walks so that we do not distub the probability distribution of blocks' lengths (u_i), leading to the ensurance of the perfect privacy property and the conservation of the average space and time overheads. Applying this method allows to perform a modification while respecting the lengths distribution but leads almost systematically to a repartitioning of an untouched but quite limited subpart of the document. Let the corresponding sequence of length $(u_i)_{i=1..n}$ be drawn i.i.d. (independently and identically distributed) from a discrete distribution. The problem is to generate a sequence of i.i.d. draws $(u'_i)_{i=k..l}$ from this distribution until we find a couple of indices (l, m) satisfying the following equality

$$\sum_{j=k}^{l} u'_j = \sum_{j=k}^{m} u_j + A$$

where $A = |\beta| + |B_{k-1}|$ if the insertion if performed between two bytes of B_{k-1} and $A = |\beta|$ otherwise. For instance, in this second case, the resulting sequence of variable sized data blocks is then $B_1, \dots,$ $B_{k-1}, B'_k, \dots, B'_l, B_{m+1}, \dots, B_n$ where the subsequence of blocks $(B'_i)_{i=k..l}$ (of respective lengths $(u'_i)_{i=k..l}$) which contains the inserted data replaces the subsequence $(B_i)_{i=k..m}$.

All the operations described (insertion, deletion and replacement) can be preformed thanks to a single one, a substitute function. The resulting scheme is secure provided that update operations are done by paying particular attention to the variable-length blocks that are changed (in content or length). Indeed, the associated random values need to be regenerated and these blocks reciphered. This function, described in Algorithm 8, takes as input an encrypted form $((r_i)_{i=1...j_0}, (u_i)_{i=1...j_0}, (C_i)_{i=1...j_0})$ augmented by sentinel values for the sake of algorithmic simplification, as they allow update modifications at the very beginning or the end of the document.

Algorithm 8 $\mathcal{I}$

```
Input:
    A ciphertext $((r_i)_{i=0...j_0+1}, (u_i)_{i=0...j_0+1}, (C_i)_{i=0...j_0+1})$,
    the document $D$, an operation $M = (substitute, i, j, \beta)$, a key $K$
1:  $v \leftarrow 1$; $c \leftarrow |\beta|$;
2:  $k_0 \leftarrow \text{argmin}_a \left(f(a) = \sum_{m=0}^{a} u_m | f(a) \geq i\right)$;
3:  if $f(k_0) > i$ then
4:      $\beta \leftarrow D[f(k_0-1)+1, i] \| \beta$;
5:      $c \leftarrow c + |D[f(k_0-1)+1, i|$;
6:      $k_0 \leftarrow k_0 - 1$;
7:  $k_1 \leftarrow \text{argmin}_a \left(f(a) = \sum_{m=0}^{a} u_m | f(a) \geq j-1\right)$;
8:  if $f(k_1) > j-1$ then
9:      $\beta \leftarrow \beta \| D[j, f(k_1)]$;
10:     $c \leftarrow c + |D[j, f(k_1)]|$;
11: $k_1 \leftarrow k_1 + 1$;
12: $x \overset{\phi}{\leftarrow} \{1, \ldots, lN\}$;
13: if $x = c$ then
14:     $u'_v \leftarrow x$;
15:     $(r'_v, C'_v) \leftarrow XOR.E_K(\beta)$;
16:     $v \leftarrow v + 1$;
17:     go to step 34;
18: if $x < c$ then
19:     $c \leftarrow c - x$; $u'_v \leftarrow x$;
20:     $(r'_v, C'_v) \leftarrow XOR.E_K(\beta[1, x])$;
21:     $\beta \leftarrow \beta[x+1, c]$;
22:     $v \leftarrow v + 1$;
23:     go to step 12;
24: else$[x > c]$
25:     if $k_1 = j_0 + 1$ then
26:         $u'_v \leftarrow c$;
27:         $(r'_v, C'_v) \leftarrow XOR.E_K(\beta)$;
28:         $v \leftarrow v + 1$;
29:         go to step 34;
30:     $c \leftarrow c + u_{k_1}$;
31:     $\beta \leftarrow \beta \| D[f(k_1-1)+1, f(k_1)]$;
32:     $k_1 \leftarrow k_1 + 1$;
33:     go to step 13;
34: $r \leftarrow ((r_i)_{i=1...k_0}, (r'_i)_{i=1...v-1}, (r_i)_{i=k_1...j_0})$;
35: $u \leftarrow ((u_i)_{i=1...k_0}, (u'_i)_{i=1...v-1}, (u_i)_{i=k_1...j_0})$;
36: $C \leftarrow ((C_i)_{i=1...k_0}, (C'_i)_{i=1...v-1}, (C_i)_{i=k_1...j_0})$
37: return $(r, u, C)$;
```

D. *Efficiency analysis*

Encryption efficiency: From now on we will consider a 16-byte block cipher. We estimate the average number of block cipher evals required to encrypt a document as follows: Given that a part size X follows a uniform distribution $\mathcal{U}([1,\ L])$ where L is a multiple of the block-size, according to the Wald's equation [13] the average number of parts in a ciphertext is tightly upper bounded by $\frac{|D|+L}{\mathbb{E}(X)}$. The number n_X of blocks in a part follows a distribution $\mathcal{U}([1,\ L/16])$. Subsequently, the Wald's equation allows us to upper bound the average number of block cipher evals by the product $\frac{|D|+L}{\mathbb{E}(X)} \cdot \frac{16+L}{32}$.

Update efficiency: More precisely, let $(X_i)_{i\geq 1}$ and $(Y_i)_{i\geq 1}$ be the sequences of independent, identically distributed and strictly positive random variables with common distribution ϕ. We set the random walks S_n, T_m such that $S_n = \sum_{i=1}^{n} X_i$ and $T_m = \sum_{i=1}^{m} Y_i$ and the random subset Z of $\mathbb{N}^2$

$$Z = \{(n,m) \in \mathbb{N}^2 \ ; S_n - T_m = C \ ; C \in \mathbb{N}\}.$$

We denote by C the number of contiguous bytes to insert. If (n,m) and (n',m') are distinct in Z then either $n < n'$ and $m < m'$ or $n' < n$ and $m' < m$. In other words $n \leq n'$ and $m' \leq m$ leads to $(n,m) = (n',m')$. Indeed, if $n \leq n'$ and $m' \leq m$ then $\sum_{i=1}^{n} X_i = \sum_{i=1}^{m} Y_i + C$ and $\sum_{i=1}^{n'} X_i = \sum_{i=1}^{m'} Y_i + C$ imply $\sum_{i=n+1}^{n'} X_i = -\sum_{i=m'+1}^{m} Y_i$. Therefore Z has the form $\{(n_k,m_k); k \in \mathbb{N}^*\}$ and the sequences (n_k) and (m_k) are strictly increasing. Consider the algorithm described below that takes as input an initial value C_0 for C. First, we notice that we have two ways to terminate the algorithm, the traces of execution (5,1,2) or (9,2). We can reason about both the number n_1 of draws X and the number m_1 of draws Y.

```
1: $X \overset{\phi}{\leftarrow} \{1,\dots,L\}$;
2: If $X = C$ stop;
3: **if** $X < C$ **then**
4:     $C \leftarrow C - X$;
5:     go to step 1;
6: **if** $X > C$ **then**
7:     $Y \overset{\phi}{\leftarrow} \{1,\dots,L\}$;
8:     $C \leftarrow C + Y$;
9:     go to step 2;
```

We notice also the followings:

- If $C_0 > L$, the average number of consecutives executions of the step 3 is upper bounded by $C_0/\mathbb{E}(X)$, after which we have $C \leq L$.
- If $C_0 \leq L$, whenever step 1 (or 7) is executed we have $C \leq L$ and then a non-zero probability $P(x = C)$ (or $P(y = X - C)$ respectively) to terminate. Let us assume that ϕ is a uniform law and let us denote d_1 the total number of random draws ($d_1 = n_1 + m_1$). It turns out that d_1 follows a geometric law of parameter $p = \frac{1}{L}$, therefore we have the system of equations:
$$\begin{cases} \mathbb{E}(n_1) + \mathbb{E}(m_1) = L \\ \mathbb{E}(n_1) - \mathbb{E}(m_1) \leq \dfrac{2L}{L+1} \end{cases}$$
so that $\mathbb{E}(n_1) \leq 1 + L/2$. Similar but pessimistic upper bounds are possible for a binomial distribution $\mathcal{B}(L-1,p)+1$ or a geometric distribution $\mathcal{G}(p)$ with p a parameter to be set.

In more general terms, assuming a uniform distribution and $C_0 > L$, let d_0 denote the number of random draws needed to satisfy the predicate $C \leq L$ for the first time and d_1 the number of random draws needed to terminate the algorithm since the first satisfaction of this predicate. So, $d = d_0 + d_1$ and we have the upper bound $\mathbb{E}(n_1) \leq \frac{2C_0}{1+L} + \frac{L}{2} + 1$. This upper bound corresponds to the average number of parts to (re-)cipher when the insertion is performed between two parts. When the insertion is performed inside a part, this part has to be included in the partial repartition. Given that a part size is at most L, the upper bound becomes:

$$\mathbb{E}(n_1) \leq \frac{2C_0}{1+L} + \frac{L}{2} + 3. \qquad (1)$$

This latter constitutes a tight worst case upper bound for the average number of changes in the partition. Continuing, if we denote n_U the number of blocks to cipher during an update, we can give the following upper bound:

$$\mathbb{E}(n_U) = \mathbb{E}(n_1)\mathbb{E}(n_X) \leq \left(\frac{2C_0}{1+L} + \frac{L}{2} + 3\right)\frac{16+L}{32}.$$

Theorem 2: Assuming a 16-byte block cipher, a uniform distribution on the set $\{1, \dots, L\}$ where L is a multiple of the block-size and an operation $M = (substitute, i, j, \beta)$, the average number of block-cipher evals, when performing an update, is upper bounded by $\left(\frac{2|\beta|}{1+L} + \frac{L}{2} + 3\right)\frac{16+L}{32}$.

If we consider the case of a non-uniform distribution $\phi \in T^*$ with mean μ, by denoting $p_{min} = \min_{i \in \{1,\dots,L\}} \phi(i)$ we can show in the same way that $\mathbb{E}(n_1)$ is upper bounded by $C_0/\mu + 2/p_{min} + 3L/\mu$. This constitutes a loose bound. In the rest of the paper we consider the use of a uniform distribution.

Storage space: The average storage space required for the sequence u depends on the choice of the distribution and is upper bounded by $\frac{|D|+L}{\mathbb{E}(X)}\left\lceil\frac{\log L}{8}\right\rceil$. That represents, even for a basic distribution such as $\mathcal{U}([1,\ L])$, a small storage overhead. The average number n_r of random elements drawn from ϕ can be statistically parameterized such that $n_r \leq \frac{|D|+L}{\mathbb{E}(X)}$. As a result, the average total size (in bytes) of the encrypted message can be upper bounded by $|D| + \frac{|D|+L}{\mathbb{E}(X)}\left(N + \left\lceil\frac{\log L}{8}\right\rceil\right)$ where $X \sim \mathcal{U}([1,\ L])$.
Considering that n is the document size in bytes and L a multiple of the block-size such that $L = lN$, in the following Table III we present the storage space and the running time for an encryption corresponding with $N = 16$ and various values for l (note that we have rounded the factor of n and the constant to three significant digits and two significant digits respectively).

Mode	Distribution	Encryption or decryption	Synchronization	Average size
mcXOR	$\mathcal{U}([1,\ 128])$	$\leq 0.078n + 10$	≤ 302	$\leq 1.264n + 34$
	$\mathcal{U}([1,\ 256])$	$\leq 0.071n + 18$	≤ 1114	$\leq 1.133n + 34$
	$\mathcal{U}([1,\ 512])$	$\leq 0.067n + 34$	≤ 4274	$\leq 1.071n + 36$
rECB	constant	$0.063n$	block-wise	$2n$

TABLE III: Efficiency of mcXOR where the encryption/decryption and the random walks synchronization are expressed in average number of block-cipher evals. Note further that n is the document size in **bytes**.

As a result, our scheme is able to reduce drastically the storage space overhead consumed by ciphertexts. The synchronization complexity corresponds to the contribution $\left(\frac{L}{2} + 3\right)\frac{16+L}{32}$ and we observe that the more we want to decrease the storage overhead, the longer is the delay for syncronizing the random walks. For instance, if the distribution used is $\mathcal{U}([1,\ 512])$, the expansion of the ciphertext is slight but the synchronization needs to recipher approximately 70KB of data. Thus, such a high value for L is justified for large sized documents.

How to index efficiently: When the document is large and a user accesses only a portion of a ciphered document, one might wonder how to index efficiently the ciphered document, or in other words, how to evaluate efficiently a sum $\sum_{m=0}^{a} u_m$. A solution is to use an history independent data structure, such as the Oblivious Tree [10]. The use of such an oblivious data structure is important to keep the perfect privacy property. Since the interest of self-balancing data structures is well known, we describe quickly its use as following. The sequence of lengths $(u_i)_{i=1\dots|u|}$ are assigned to each leaf of the tree and the value assigned to a parent node is computed as the sum of the values of its childrens. Then an access to a byte index, an update or an adding of a length value are all done in $\mathcal{O}(\log|u|)$ add operations.

V. A NEW SCHEME COMBINING THE TWO APPROACHES

We have described two ways to improve the space efficiency: the first one consists in using a sliding window over a sequence of small randomizers, and then to apply a block cipher algorithm to each successive window to obtain a keystream; the second one consists in applying several times a randomized mode of operation to each sufficiently large part of a document so as to reduce the number of random input vectors. To provide a provable *perfect privacy* property at each update in this latter scheme we employ a synchronization of random walks (this remark could be true for the former if we extend it in a byte-wise incremental version in a black-box manner). This synchronization produces a ciphering overhead which increases with the average size of a part and any attempt to reduce this effect is welcome. For this purpose, it is not difficult to imagine a byte-wise incremental encryption scheme combining these two approaches: we partition the document in variable-sized parts and cipher each part with a slightly different version of XOR mode in which an initialization vector is no longer generated uniformly at random but corresponds to a window sliding over a sequence of randomizers. Such an operating mode, denoted swrXOR, is then parameterized with two parameters, the parts' lengths probability distribution and the randomizer size e. If the update algorithms are deducible from our previous schemes, it is worth stressing on the need for caution in the regeneration of the randomizers that should be involved in an update:

- The repartition of the document including the modification is composed of parts containing this modification but also of some other ones induced by the synchronization of random walks. Let us assume that this latter unchanged content corresponds to the parts $B_i, B_{i+1}, \ldots, B_{i+n-1}$ in the original partition. Let us denote by m the overall number of repartitioned parts and by $B'_i, B'_{i+1}, \ldots, B'_{i+m-1}$ the corresponding changed parts in the updated partition.
- In addition to these m parts, this variant of XOR has to be applyed to $2(d-1)$ adjacent parts. Indeed, for similar reasons as for swrECB, as regards the last part, all the randomizers of the associated input window need to be regenerated uniformly at random, implying the (re-)generation of a total of $m + d - 1$ randomizers, with the objective to conserve update indistinguishability. As a result, both the sequence of unchanged parts $B_{i-(d-1)}, B_{i-(d-2)}, \ldots, B_{i-1}$ and $B_{i+n}, B_{i+n+1}, \ldots, B_{i+n+d-2}$ need to be reciphered to maintain consistency with the sliding windows. Security and run times are discussed in Section VI and VII respectively.

VI. Security analysis

In the following subsections, we are interested in the security of our schemes concerning the message and update secrecy and finally the perfect privacy property. Note that proofs of indistinguishability are given assuming the use of a random function. They can be derived in a straightforward way in the pseudorandom function model, we refer to [12] to observe a reduction of security between a mode of operation and the underlying pseudorandom function used. These proofs being exactly the same, we do not present them in this paper.

A. *swrECB*

The perfect privacy of this scheme is obvious and we focus on the message and update secrecy properties. We prove that *swr*ECB is still secure despite the use of incremental operations and the rationale for regenerating at random all the blocks contained in the window associated with the inserted (or replaced) block will become clear.

Theorem 3: Let the size l be in number of blocks and suppose the use of a random function instead of F_K. *swr*ECB over modification space $\mathcal{M}_B$ is a $(t, q_e, q_{inc}, \mu^*, l, \epsilon)$-I-CPA-secure incremental encryption scheme in the *find-then-guess* sense where:

$$\epsilon \leq \frac{2l\mu^* + l(l-1)}{2^{8N+1}}$$

and μ^* is the greatest amount of ciphered blocks that the adversary can obtain during the queries phase, when performing only *replace* or *insert* operations, that is to say, $\mu^* = \sum_{i=1}^{q_e} n_i + q_{inc}(2d-1)$ where n_i is the block-length of the i-th document queried to the encryption oracle.

Theorem 4: Suppose the use of a random function instead of F_K. *swr*ECB over modification space $\mathcal{M}_B$ is a $(t, q_e, q_{inc}, \mu^*, \epsilon)$-IM-CPA-secure incremental encryption scheme in the *find-then-guess* sense where:

$$\epsilon \leq \frac{\mu^* + 2(d-1)}{2^{8N}}$$

and μ^* is the greatest amount of ciphered blocks that the adversary can obtain, as stated in the previous theorem.

B. mcXOR

We assume the use of a distribution $\phi = \mathcal{U}([1,\ L])$ where L is a multiple of the block-size and the proof of security in appendix considers a particular stateless mode of encryption (XOR). In fact, we can generalize by replacing XOR by any stateless mode of encryption, the proof remains similar. As for *swr*ECB we prove that mcXOR is still secure despite the use of incremental operations.

Theorem 5: Let the size l be in number of bytes and suppose the use of a random function instead of F_K. *mc*XOR over modification space $\mathcal{M}_b$ is a $(t, q_e, q_{inc}, \mu_1^*, l, \epsilon)$-I-CPA-secure incremental encryption scheme in the *find-then-guess* sense where:

$$\epsilon \leq \frac{1}{N}\left(8\frac{l^2}{L} + 40l + 18L + 4\mu_1^*(l+L)\right)\frac{1}{2^{8N}}$$

and μ_1^* is the number of (re-)partitioned ciphered parts that the adversary can obtain during queries, that is to say:

$$\mu_1^* = \frac{2(\sum_{i=1}^{q_e} n_e^i + \sum_{i=1}^{q_{inc}} n_u^i)}{L+1} + 2q_e + (L/2+3)q_{inc}$$

where n_e^i is the byte-length of the i-th document queried to the encryption oracle and n_u^i is the byte-length of the i-th modification queried to the updating oracle.

Theorem 6: Suppose the use of a random function instead of F_K. *mc*XOR over modification space $\mathcal{M}_b$ is a $(t, q_e, q_{inc}, \mu_2^*, n_{uc}, \epsilon)$-IM-CPA-secure incremental encryption scheme in the *find-then-guess* sense where:

$$\epsilon \leq \frac{1}{N}\left(8\frac{n_{uc}^2}{L} + 40n_{uc} + 18L + 4\mu_2^*(n_{uc}+L)\right)\frac{1}{2^{8N}}$$

with μ_2^* the amount of (re-)partitioned ciphered parts that the adversary can obtain during the queries phase, augmented by the repartioned parts (of unmodified content) obtained during the challenge update, that is to say $\mu_2^* = \mu_1^* + L/2 + 3$, and n_{uc} the byte-length of the data to insert in the challenge update.

Theorem 7: For all document $D = b_1 \ldots b_n$ where $(b_i)_{i=1..n}$ are the ordered bytes of D and for all β (including the empty string), considering the encrypted document $C = \mathrm{mcXOR}.\mathcal{E}_K(D)$, the outputs of $\mathrm{mcXOR}.\mathcal{E}_K(b_1b_2 \ldots b_{i-1}b_i\beta b_j b_{j+1} \ldots b_{n-1}b_n)$ and $\mathrm{mcXOR}.\mathcal{I}_K((substitute, i, j, \beta), D, C)$ are perfectly indistinguishable.

C. swrXOR

We take the same assumptions about the distribution of parts' lengths. The proof of security in appendix describes only the differences when compared to the proofs supplied for *mc*XOR. It turns out that the upper bounds on insecurity of *swr*XOR stay close to this latter up to a constant factor.

Theorem 8: Let the size l be in number of bytes and suppose the use of a random function instead of F_K. *swr*XOR over modification space $\mathcal{M}_b$ is a $(t, q_e, q_{inc}, \mu_1^*, l, \epsilon)$-I-CPA-secure incremental encryption scheme in the *find-then-guess* sense where:

$$\epsilon \leq \frac{7L}{N}\left(8\frac{l^2}{L} + 40l + 18L + 4\mu_1^*(l+L)\right)\frac{1}{2^{8N-1}}$$

and μ_1^* is the number of (re-)partitioned ciphered parts that the adversary can obtain during queries, that is to say:

$$\mu_1^* = \frac{2(\sum_{i=1}^{q_e} n_e^i + \sum_{i=1}^{q_{inc}} n_u^i)}{L+1} + 2q_e + (L/2 + 3 + 2(d-1))q_{inc}$$

where n_e^i is the byte-length of the i-th document queried to the encryption oracle and n_u^i is the byte-length of the i-th modification queried to the updating oracle.

Theorem 9: Suppose the use of a random function instead of F_K. *swr*XOR over modification space $\mathcal{M}_b$ is a $(t, q_e, q_{inc}, \mu_2^*, n_{uc}, \epsilon)$-IM-CPA-secure incremental encryption scheme in the *find-then-guess* sense where:

$$\epsilon \leq \frac{7L}{N}\left(8\frac{n_{uc}^2}{L} + 40n_{uc} + 18L + 4\mu_2^*(n_{uc} + L)\right)\frac{1}{2^{8N-1}}$$

with μ_2^* the amount of (re-)partitioned ciphered parts that the adversary can obtain during the queries phase, augmented by the repartioned parts (of unmodified content) obtained during the challenge update, that is to say $\mu_2^* = \mu_1^* + L/2 + 3 + 2(d-1)$, and n_{uc} the byte-length of the data to insert in the challenge update.

Theorem 10: For all document $D = b_1 \ldots b_n$ where $(b_i)_{i=1..n}$ are the ordered bytes of D and for all β (including the empty string), considering the encrypted document $C = \text{swrXOR}.\mathcal{E}_K(D)$, the outputs of $\text{swrXOR}.\mathcal{E}_K(b_1 b_2 \ldots b_{i-1} b_i \beta b_j b_{j+1} \ldots b_{n-1} b_n)$ and $\text{swrXOR}.\mathcal{I}_K((substitute, i, j, \beta), D, C)$ are perfectly indistinguishable.

VII. Summary of results

For the extension of *swr*ECB into a byte-wise incremental scheme, we assume the use of a slightly different approach of [9] in which each variable-length part is ciphered thanks to the lowest possible number of block cipher calls. For instance, if a part is of length p bytes, only $\lceil p/N \rceil$ block-cipher evals is needed. Besides, as the construction of [9] was very generic, we discard the padding of the variable-length parts and encrypt them so that their lengths remain the sames[4]. Table IV summarizes the efficiencies of our schemes. We then notice that when we choose a very high parameter L for *mc*XOR the delay for the synchronization of the random walks becomes prohibitive, and we observe in this case that *swr*ECB becomes more attractive with respect to the *update* operations. Indeed we can choose a smaller parameter L^* for the random walk used in the extended *swr*ECB along with an appropriate choice for e so that the space consumption is equivalent to what we observe with *mc*XOR parametrised with L. This observation led us to consider the third scheme swrXOR. From a space/update time tradeoff perspective, by using just a little bit more than e bits of overhead by ciphered part, swrXOR is the most interesting. If we select correctly the involved parameters, combining the approach of the sliding window with mcXOR allows to significantly reduce the expansion of the ciphertext while avoiding a too large increase of the constant in the update average runtime (the contribution of the random walks synchronization). By varying the parameters of our encryption schemes, the following observations can be made about the coefficients in the upper bounds linear equations (Figure 2 depicts the lines for various parameters):

- If we increase the parameter L, or more generally, if we increase the average size of a part then the expansion of the ciphertext is reduced.
- By increasing the average size of a part we increase the value of the constant in the update average run time upper bound, making incremental updates of the ciphertext less efficient for small changes in the corresponding document. On the other hand, this is accompagned of a slight decrease of the line slope, making incremental updates more efficient for big changes.
- Using swrXOR instead of mcXOR with a smaller parameter L and an appropriate parameter e allows to obtain a reduction of both the ciphertext expansion and the constant part in the update average run time. In Figure 2, we can observe the behaviours of mcXOR with $L = 512$ and swrXOR with $L = 128$

[4] *swr*ECB is a XORing mask mode, so we can cipher each part of length s-bit with the correspondings s highest (or lowest, depending on the used convention) significant bits and discard the unused ones.

Mode	Encryption/decryption - Block-cipher evals in average	Insertion of β - Block-cipher evals in average	Average size
*mc*XOR	$\lvert D\rvert + \frac{2(\lvert D\rvert+L)}{L+1}\left(\mathbf{N} + \left\lceil \frac{\log L}{8} \right\rceil\right)$	$\left(\frac{2\lvert\beta\rvert}{1+L} + \frac{L}{2} + 3\right)\frac{16+L}{32}$	$\frac{2(\lvert D\rvert+L)}{(L+1)} \cdot \frac{(16+L)}{32}$
extended *swr*ECB	$\lvert D\rvert + \frac{2(\lvert D\rvert+L)}{(L+1)}\left(\frac{\mathbf{e(16+L)}}{\mathbf{256}} + \left\lceil \frac{\log L}{8} \right\rceil\right) + \frac{\mathbf{e(d-1)}}{\mathbf{8}}$	$\left(\frac{2\lvert\beta\rvert}{1+L} + \frac{L}{2} + 3\right)\frac{16+L}{32} + 2(d-1)$	
*swr*XOR	$\lvert D\rvert + \frac{2(\lvert D\rvert+L)}{(L+1)}\left(\frac{\mathbf{e}}{\mathbf{8}} + \left\lvert \frac{\log L}{8} \right\rvert\right) + \frac{\mathbf{e(d-1)}}{\mathbf{8}}$	$\left(\frac{2\lvert\beta\rvert}{1+L} + \frac{L}{2} + 3 + 2(d-1)\right)\frac{16+L}{32}$	

TABLE IV: Summary of tight upper bounds for the space and time efficiencies, by assuming the use of a uniform distribution $\mathcal{U}([1,\ L])$ where L is a multiple of the block-cipher size. The space consumption is in number of bytes.

and $e = 8$. From a space consumption standpoint, swrXOR is the more advantageous. As regards the update run times, swrXOR is more advantageous for insertion of less than 200KBytes. Finally, when performing a small change at a given location we would like to avoid reciphering completely the following bytes of the document (as frequently as possible). In this sense we emphasise that a smaller constant in the update average run time is preferable for small document.

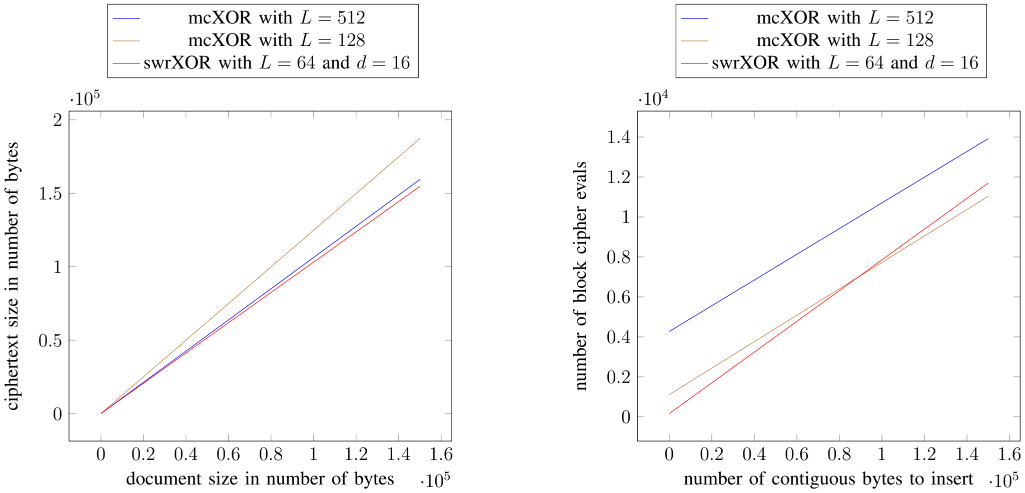

Fig. 2: Upper bounds for the average space consumptions and average update run times

Remarks about composition paradigms: Incremental authenticated encryption schemes can be obtained by the *encrypt-then-MAC* composition paradigm [14]: (*i*) The document is ciphered with an incremental encryption scheme (instantiated with a key k_1); (*ii*) The incremental MAC (message Authentication Code) is computed over the resulting ciphertext (instantiated with a key k_2). This composition is the most secure, if the symmetric encryption (instantiated with a key k_1) is IND-CPA and if the symmetric signature scheme (instantiated with a key $k_2 \neq k_1$) is INT-CTXT then the resulting authenticated encryption scheme is IND-CCA. Using one of our algorithm in such a composition scheme leads not only to very substantial savings on the space consumption but also sometimes to interesting gain in computational efficiency. Take, for example, a variant of mcXOR based on the stateless variant of CBC, say mcCBC. We design a byte-wise incremental authenticated encryption scheme in the following way: (i) Choose a distribution for the random walk that well reduces the expansion of the ciphertext produced by mcCBC, for instance, choose a distribution $\mathcal{U}([1,\ 512])$, and encrypt the document; (ii) Apply XOR-MAC [2] on the resulting ciphertext. Even though XOR-MAC is a block-wise incremental MAC, the resulting composed scheme still allows update operations whose run time is affine (in average) in the amount of data changed. This is due to the fact that mcCBC respects the alignement of the unchanged ciphered blocks. Finally the consumed space is two times the expansion of the ciphertext. Consequently, the overall consumed space by the resulting authenticated encryption is little more than two times the size of the document, which is much better than the factor 4 implied by the composition of rECB with the incremental XOR-MAC [8].

VIII. Applications and conclusion

One can deduce a variant of the protocol of [15] which does make use of incremental encryption and incremental MAC to update at low costs the cryptographic forms of outsourced documents. For instance, one could choose a solution in which a user $\mathcal{U}$ can update documents stored on a server $\mathcal{S}$ while their contents are kept secret from this latter but are authenticated by the two entities. Considering that $\mathcal{U}$ uses a secret key k_1 to encrypt documents and a key k_2 (shared with $\mathcal{S}$) to authenticate them, such a solution is obviously possible with the *encrypt-then-MAC* composition: $\mathcal{U}$ updates himself (herself) a ciphered document on his (her) local workspace with his (her) key k_1 and sends the changed ciphered parts to $\mathcal{S}$, which in turn updates the corresponding MAC with the shared key k_2.

Many modern applications have to deal with the handling of changes in large secure electronic documents on remote servers for which the development of incremental cryptographic algorithms is required. Indeed this is mandatory in order to reduce delays to maintain consistency. Block-wise incremental algorithms defined so far have the advantage of being parallelisable and already allow interesting operations such as block deletion/insertion. Unfortunately, their extension towards byte-wise incremental schemes [9] does not solve the ciphertext expansion problem. In this paper, we propose new byte-wise incremental and perfectly private encryption schemes that can be parametrised in order to produce smaller ciphertexts. Such a characteristic is as critical as computational resources are in cloud storage.

A first approach is to design a space-efficient block-wise encryption scheme and extend it into a byte-wise incremental one. Its security is proved assuming a good pseudorandom function. A second approach is to use a stateless encryption mode as a fine-grained primitive. In this latter case, its security relies only on the correct use of this primitive. The run time of the incremental update operations is very efficient but the contribution of the random walks synchronization increases with the parameters of the distribution and thus lessens its interest for small documents and updates. The question then arises as to how decrease this synchronization overhead. A possible solution, our third approach denoted swrXOR, is a variant of the *mc*XOR mode in which the input random blocks are replaced by sliding windows over a sequence of small random blocks. In other words this means combining the advantages of *mc*XOR and *swr*ECB. In such a solution, by choosing a small value for L and the appropriate value for e for approaching the consumed space of mcXOR, we can take advantage of the same space-efficiency while lowering the delay for the random walks synchronization. Finally, it would be worthwhile to estimate performances and security of these schemes by considering the use of a real synchronous stream cipher [16], [17].

References

[1] M. Bellare, O. Goldreich, and S. Goldwasser, “Incremental cryptography: The case of hashing and signing,” in *INTERNATIONAL CRYPTOLOGY CONFERENCE ON ADVANCES IN CRYPTOLOGY*. Springer-Verlag, 1994, pp. 216–233.

[2] ——, “Incremental cryptography and application to virus protection,” in *Proc. of the 27th Ann. ACM Symp. on the Theory of Computing*. ACM Press, 1995, pp. 45–56.

[3] M. Bellare and D. Micciancio, “A new paradigm for collision-free hashing: incrementality at reduced cost,” in *In Eurocrypt97*. Springer-Verlag, 1997, pp. 163–192.

[4] D. A. McGrew and J. Viega, “The security and performance of the galois/counter mode (gcm) of operation,” in *INDOCRYPT*, 2004, pp. 343–355.

[5] D. A. McGrew, “Efficient authentication of large, dynamic data sets using galois/counter mode (gcm),” in *IEEE Security in Storage Workshop*, 2005, pp. 89–94.

[6] L. Martin, “Xts: A mode of aes for encrypting hard disks,” *IEEE Security & Privacy*, vol. 8, no. 3, pp. 68–69, 2010.

[7] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, “Ocb: a block-cipher mode of operation for efficient authenticated encryption.” ACM Press, 2001, pp. 196–205.

[8] E. Buonanno, J. Katz, and M. Yung, “Incremental unforgeable encryption,” in *FSE’01*, 2001, pp. 109–124.

[9] K. Atighehchi and T. Muntean, “Towards fully incremental cryptographic schemes,” in *ASIACCS*, 2013, pp. 505–510.

[10] D. Micciancio, “Oblivious data structures: Applications to cryptography,” in *In Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*. ACM Press, 1997, pp. 456–464.

[11] N. F. P. . National Bureau of Standards, “DES modes of operation,” U.S. Department of Commerce, Tech. Rep., 1980.

[12] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, “A concrete security treatment of symmetric encryption,” in *Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE*, 1997, pp. 394–403.

[13] A. Wald, “Some generalizations of the theory of cumulative sums of random variables,” *The Annals of Mathematical Statistics*, vol. 16, no. 3, pp. pp. 287–293, 1945. [Online]. Available: http://www.jstor.org/stable/2235707

[14] M. Bellare and C. Namprempre, “Authenticated encryption: Relations among notions and analysis of the generic composition paradigm,” *J. Cryptol.*, vol. 21, no. 4, pp. 469–491, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1007/s00145-008-9026-x

[15] W. Itani, A. Kayssi, and A. Chehab, “Energy-efficient incremental integrity for securing storage in mobile cloud computing,” in *Energy Aware Computing (ICEAC), 2010 International Conference on*, 2010, pp. 1–2.

[16] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, “Rabbit: A new high-performance stream cipher,” in *Proc. Fast Software Encryption 2003, volume 2887 of LNCS*. Springer, 2003, pp. 307–329.

[17] E. Biham and J. Seberry, “Py (roo): A fast and secure stream cipher using rolling arrays,” *IACR Cryptology ePrint Archive*, vol. 2005, p. 155, 2005.

APPENDIX

A. *swrECB*

Proof of theorem 3: Let G_1 be the multiset of windows involved during encryption and updating oracle accesses and let G_2 be the multiset of the l windows involved during the encryption of the challenge. Let D_1 be the event that the windows of G_2 are distinct from the windows of G_1, and let D_2 be the event that the windows of G_2 are all distinct. For simplicity, we also define S to be the event of success of A in the game $\mathbf{Expt}^{\mathbf{I\text{-}CPA}}_{\mathbf{\Psi,FG}}(\mathbf{A},\mathbf{k})$.

Let us now consider the event $\mathrm{D} = \mathrm{D}_1 \cap \mathrm{D}_2$. When the event D occurs, the challenge ciphertext is obtained by evaluating l times the random function to *distinct and new* points, new in the sense that they have not appeared during queries. Thus, the corresponding outputs of the random function are l new random draws from $\{0,1\}^{8N}$. By XORing such a random value with a plaintext block of the challenge, we obtain a ciphered block which is still a new random draw. The l ciphered blocks of the challenge are then independently and identically distributed over $\{0,1\}^{8N}$. Besides, this distribution is independent of the previous ciphered blocks obtained during queries. Consequently, the adversary can at best make a random guess. We have $\Pr(\mathrm{S} \mid \mathrm{D}) = 1/2$ and it follows that

$$|\Pr(\mathrm{S}) - 1/2| \leq \left|\Pr(\mathrm{S} \mid \overline{\mathrm{D}}) - \frac{1}{2}\right| \Pr(\overline{\mathrm{D}}) \leq \frac{1}{2}\Pr(\overline{\mathrm{D}}).$$

Before continuing, consider the windows $(w_i)_{i=1\ldots n}$ in the same ciphertext, the probability that $w_i = w_{i+k}$ for all $i, k > 0$ is exactly $1/2^{ed}$. In other words, the fact that two windows within a same ciphertext share a common (but shifted) sequence of randomizers or not, the probability that they collide is still the same. Now it remains to estimate the probability of occurrence of the event $\overline{D}$:

- Let us denote $(w_i^c)_{i=1\ldots l}$ the windows used in the challenge ciphertext. Let us denote $(w_i^{e1})_{i=1\ldots n_1}$, $(w_i^{e2})_{i=1\ldots n_2}$, ..., $(w_i^{eq_e})_{i=1\ldots n_{q_e}}$ the windows used during encryption queries and $(w_i^{u1})_{i=1\ldots 2d-1}$, $(w_i^{u2})_{i=1\ldots 2d-1}$, ..., $(w_i^{uq_{inc}})_{i=1\ldots 2d-1}$ the windows used during update queries. We assume update queries of type replacement or insertion (only) since they bring more information to the adversary. For $i \in [\![1,l]\!]$, $j \in [\![1,q_e]\!]$ and $k \in [\![1,n_j]\!]$ let us denote the event $\overline{d_e^{ijk}} = \{w_i^c = w_k^{ej}\}$ and for $i \in [\![1,l]\!]$, $j \in [\![1,q_{inc}]\!]$ and $k \in [\![1,2d-1]\!]$ let us denote the event $\overline{d_u^{ijk}} = \{w_i^c = w_k^{uj}\}$. The event $\overline{D_1}$ is included in $\displaystyle\bigcup_{i=1\ldots l}\left(\bigcup_{j=1\ldots q_e}\bigcup_{k=1\ldots n_j}\overline{d_e^{ijk}} \quad \bigcup_{j=1\ldots q_{inc}}\bigcup_{k=1\ldots 2d-1}\overline{d_u^{ijk}}\right)$. We therefore have

$$\Pr(\overline{\mathrm{D}_1}) \leq \frac{l\mu^*}{2^{ed}}$$

where $\mu^* = \sum_{i=1}^{q_e} n_i + q_{inc}(2d-1)$. The contribution $q_{inc}(2d-1)$ corresponds to the most favourable update queries for the adversary (*insertion* or *replacement* only).

- We define the event $\overline{d_2^{ij}} = \{w_i^c = w_j^c\}$ for all $i, j \in [\![1, l]\!]$ with $i \neq j$. The event $\overline{D_2}$ being included in $\bigcup\limits_{i=1...l} \bigcup\limits_{j=i+1...l} \overline{d_2^{ij}}$, we can subsequently upper bound its probability of occurence as follows:

$$\Pr(\overline{\mathrm{D_2}}) \leq \sum_{i=1}^{l} \sum_{j=i+1}^{l} \frac{1}{2^{ed}} \leq \frac{l(l-1)}{2^{ed+1}}.$$

■

Proof of theorem 4: Now let us consider the update secrecy game. *Deletions* at a same location being obviously indistinguishable, the adversary will return in the “*find*“ *phase* operations of type *insertion* or *replacement*.

Let us now consider the *”guess” phase*. Notice that when the challenger applies one of the two operations on the ciphertext, a certain amount (at most $2(d-1)$) of adjacent ciphered blocks change while the corresponding plaintext blocks are unchanged. It is important to make clear that this is only due to the changing of the $2(d-1)$ adjacent input windows to the random function. Indeed, considering a challenge update of type $(insert, i, P^*)$, the returned updated ciphertext C' contains at most $2d-1$ new windows. One of them, w', serves to cipher the inserted block and we call the other ones “the adjacents“. Let G^u be the multiset of windows of G_1 augmented by the new adjacent windows $(w'_j)_{j=i-d+1...i}$ and $(w'_j)_{j=i+1...d-1}$. We denote D^u the event that w' is distinct from the windows of G^u.

Now, let us consider the event D^u. When the event D^u occurs, whatever the operation choosed by the challenger is, this new ciphered block is indistinguishable from a new random draw from $\{0,1\}^{8N}$. This is due to the fact that the output of the random function is really a new random draw, and not a reused one. Just as in the previous proof, by denoting again S the event of success of A in the game $\mathbf{Expt}^{\mathbf{IM\text{-}CPA}}_{\mathbf{\Psi,FG}}(\mathbf{A}, \mathbf{k})$, we have

$$|\Pr(\mathrm{S}) - 1/2| \quad \leq \quad \frac{1}{2}\Pr(\overline{\mathrm{D^u}})$$

We fix an arbitrary order for the elements of G^u, that is to say, we write $\mathrm{G}^u = (w_i)_{i=1...u^*+2(d-1)}$. We denote $\overline{d^i}$ the event that $w' = w_i$ for $i \in [\![1, u^* + 2(d-1)]\!]$. The events $(\overline{d_i})_{i=1...u^*+2(d-1)}$ are not mutually exclusive, but we can upper bound the probability of occurence of $\overline{\mathrm{D}^u}$ as follows:

$$\Pr(\overline{\mathrm{D^u}}) \leq \frac{\mu^* + 2(d-1)}{2^{ed}}.$$

■

Why regenerate all the random blocks in the window w'*:* Taking the example of a *replace* operation, assume that we do not regenerate all of them but only one, the first one. Consider the value $w = r_i \| r_{i+1} \| \ldots \| r_{i+d-1}$ before an update and the corresponding value $w' = r'_i \| r_{i+1} \| \ldots \| r_{i+d-1}$ after an update. The probability that w' equals w is exactly $1/2^e$. In this case update queries are more advantageous for the adversary than encryption queries and annihilates the indistinguishability of the scheme. We conclude that all the random blocks that serve to encrypt the inserted (or replacement block) need to be regenerated, implying a change in the $2(d-1)$ adjacent windows. This leads to the reencryption of $2(d-1)$ unchanged plaintext blocks, a necessary overhead of encryption.

B. mcXOR

Stochastic processes background: Let X_1, X_2, ... be *i.i.d.* random variables drawn according to $\mathcal{U}([1,\ L])$ and define the sum $S_T = \sum_{i=1}^{T} X_i$. Consider the first time T where $S_T \geq l$. We call T the stopping time for the stopping rule $\min_T(S_T \text{ s.t. } S_T \geq l)$. The following hold:

- Wald’s first equation: $\mathbb{E}(S_T) = \mathbb{E}(T)\mathbb{E}(X_1)$,
- Wald’s second equation: $\mathbb{E}((S_T - T\mathbb{E}(X_1))^2) = V(X_1)\mathbb{E}(T)$.

We deduce the followings: (i) **Fact 1**. According to the first equation, we have $\frac{2l}{1+L} \leq \mathbb{E}(T) < \frac{2(l+L)}{1+L}$. ($ii$) **Fact 2**. By developing the second equation and noticing that $Tl \leq TS_T < T(l+L)$ we have the following rough upper bound $\mathbb{E}(T^2) \leq 4\left(\frac{l}{L}\right)^2 + 20\frac{l}{L} + 9$.

Proof of theorem 5: The reasoning being almost the same that for *swr*ECB we give only the important steps. First of all, we suppose negligible the time taken to partition the bytestrings. Let us denote by r' and r'' the random vectors used to encrypt respectively the parts of length l' and l'', where r', r'' are drawn randomly from $\{0,1\}^{8N}$ and l', l'' are drawn randomly from $\{1,\ldots,L\}$. We say that r' and r'' overlap in values if there exist k', k'' such that $r'+k' = r''+k''$ where $k' \in \left\{0,1,\ldots,\lceil\frac{l'}{N}\rceil - 1\right\}$ and $k'' \in \left\{0,1,\ldots,\lceil\frac{l''}{N}\rceil - 1\right\}$. Let us denote by $\mathrm{Pr}_{\text{overlap}}$ the probability of occurence of such an event. It is clear that $\mathrm{Pr}_{\text{overlap}} \leq \frac{2L/N}{2^{8N}}$. Continuing, we need to use a certain number of notations:

- $T_{e,i}$ is the number of parts in the i-th encryption query,
- $T_{u,i}$ is the number of parts in the i-th update query,
- T_l is the number of parts in the challenge ciphertext,
- G_1 is the multiset of random vectors involved during encryption and updating oracle accesses. We notice that G_1 is of size $\sum_{i=1}^{q_e} T_{e,i} + \sum_{i=1}^{q_{inc}} T_{u,i}$.
- G_2 is the multiset of random vectors involved during the encryption of the challenge. This one is then of size T_l.

Let D_1 be the event that none of the vectors of G_2 overlaps a vector of G_1, and let D_2 be the event that none of the vectors of G_2 overlap. As previously we only need to be concerned with $\overline{D} = \overline{D_1} \cup \overline{D_2}$:

- We use the update run time bound (inequation (1) Section IV-D) and fact 1 to evaluate the probability of occurence of $\overline{D_1}$. By applying several times the union bound it follows that:

$$\begin{aligned}\Pr(\overline{D_1}) &\leq \left(\sum_{i=1}^{q_e}\mathbb{E}(T_{e,i}) + \sum_{i=1}^{q_{inc}}\mathbb{E}(T_{u,i})\right)\mathbb{E}(T_l)\,\mathrm{Pr}_{\text{overlap}} \\ &\leq \frac{4\mu_1^*(l+L)}{N2^{8N}}.\end{aligned}$$

- We use the fact 1 to evaluate the probability of occurence of $\overline{D_2}$. By the union bound it follows that:

$$\Pr(\overline{D_2}) \;\leq\; \mathbb{E}(T_l^2)\,\mathrm{Pr}_{\text{overlap}} \leq \frac{(8\frac{l^2}{L} + 40l + 18L)}{N2^{8N}}.$$

■

Proof of theorem 6: In the update secrecy game, the adversary will return in the "*find*" *phase* operations such as $(substitute, i, j, \beta_b)_{b=0,1}$ where $|\beta_0| = |\beta_1| > 0$. Indeed, the case $|\beta_0| = |\beta_1| = 0$ yields the same updated message in both cases. Let us now denote by T_{uc} the number of new parts produced during this update. According to inequation (1) Section IV-D we have $\mathbb{E}(T_{uc}) < \frac{2|\beta_0|}{1+L} + \frac{L}{2} + 3$. We need to split T_{uc} in two times $T_{uc,1}$ and $T_{uc,2}$ where $T_{uc,1}$ corresponds to the time taken to produce the parts containing the modified portion and $T_{uc,2}$ the time taken to resynchronize the random walks, that is, the parts containing unmodified data in the message. Considering the challenge update, let G_3 be the multiset G_1 augmented by the random vectors used to encrypt the parts that serve to resynchronize the random walks and let G_4 be the multiset of the random vectors used to encrypt the parts containing the modified portion of the message. G_1 is of size $\sum_{i=1}^{q_e} T_{e,i} + \sum_{i=1}^{q_{inc}} T_{u,i} + T_{uc,2}$ and G_4 is of size $T_{uc,1}$.

Let D_3 be the event that none of the vectors of G_3 overlaps a vector of G_4, and let D_4 be the event that none of the vectors of G_4 overlap. As previously we only need to be concerned with $\overline{D} = \overline{D_3} \cup \overline{D_4}$:

- We note that $\mathbb{E}(T_{uc,2})$ is maximized when $\mathbb{E}(T_{uc,1})$ is minimized so that $\mathbb{E}(T_{uc,2}) \leq L/2 + 3$. Then, upper bounding the various average times, we deduce the following:

$$\begin{aligned}\Pr(\overline{D_3}) &\leq \left(\sum_{i=1}^{q_e}\mathbb{E}(T_{e,i}) + \sum_{i=1}^{q_{inc}}\mathbb{E}(T_{u,i})\right)\mathbb{E}(T_{uc,2})\,\mathrm{Pr}_{\text{overlap}} \\ &\leq \frac{4\mu_2^*(l+L)}{N2^{8N}}.\end{aligned}$$

- Here again, using the fact 2 and applying the union bound:

$$\Pr(\overline{D_4}) \quad \leq \quad \mathbb{E}(T^2_{uc,1}) \Pr_{\text{overlap}} \leq \frac{(8\frac{n^2_{uc}}{L} + 40n_{uc} + 18L)}{N2^{8N}}.$$

■

Proof of theorem 7:

The assertion of this proposition is obvious. If we consider the blocks' lengths in the partition of the plaintext $b_1b_2\dots b_{i-1}b_i\beta b_jb_{j+1}\dots b_{n-1}b_n$ performed by $\mathrm{mcXOR}.\mathcal{E}_K(b_1b_2\dots b_{i-1}b_i\beta b_jb_{j+1}\dots b_{n-1}b_n)$ or $\mathrm{mcXOR}.\mathcal{I}_K((substitute, i, j, \beta), D, C)$, we see a sequence of terms independently and identically distributed from a discrete probability distribution, except the last one (due to the termination condition in the algorithm). Its probability of occurrence is conditioned by the length of the document and the summation of all previous drawn terms, what still allows to say that the probability distribution of this latter term is the same in both cases. Finally, in either case, each variable-length block is ciphered thanks to the mode of operation XOR independently of the others.

■

C. *swrXOR*

Proofs of theorems 8 and 9: Let $w_i = r_ir_{i+1}\dots r_{i+e-1}$ be the 2^d-bit encoding of the *i-th* sliding window, that is, the window used to encrypt the *i-th* part. Let us denote by l_i and l_j the lengths of the i-th and j-th parts, where l_i and l_j are drawn randomly from $\{1, \dots, L\}$ and $i \neq j$. Let us also denote by S' the set $\left\{0, 1, \dots, \lceil \frac{l_i}{N} \rceil - 1\right\}$ and by S'' the set $\left\{0, 1, \dots, \left\lceil \frac{l_j}{N} \right\rceil - 1\right\}$. We say that w_i and w_j overlap (in values) if there exist k', k'' such that $w_i + k' = w_j + k''$ where $k' \in S'$ and $k'' \in S''$. Let us denote overlap such an event. It follows that $\Pr\left[\text{overlap}\right] \leq \Pr\left[\exists\ k' \in S', k'' \in S'' \text{ s.t. } |w_i - w_j| = |k'' - k'|\right] \leq \Pr\left[|w_i - w_j| \leq L/N\right]$. Two cases may occur:

- $|j - i| \geq e$, it is then clear that

$$\Pr\left[|w_i - w_j| \leq L/N\right] \leq \frac{L}{N2^{8N-1}}. \tag{2}$$

- $1 \leq |j - i| \leq e - 1$ then let us denote $o = |j - i|$. Assuming that $j > i$, let the following 2^d-bit encodings $A_i = r_ir_{i+1}\dots r_{i+o-1}$, $A_{i+e} = r_{i+e}\dots r_{i+o+e-1}$ and $Z = r_{i+o}\dots r_{i+e-1}$. The string Z corresponds to the $(2^d)^{e-o}$ lower significants bits of w_i, while this is the $(2^d)^{e-o}$ most significants bits of w_{i+e}. We can rewrite w_i and w_{i+o} in the following way:

$$w_i = A_i\left(2^d\right)^{e-o} + Z,$$

$$w_{i+o} = Z\left(2^d\right)^{o} + A_{i+e}$$

so that $|w_{i+o} - w_i| = \left|A_{i+e} - A_i2^{d(e-o)} + Z\left(2^{do} - 1\right)\right|$. Note that the support of A_i and A_{i+e} is $[\![0, \left(2^d\right)^o - 1]\!]$. We are interested in the support of the random variable $X_{i+e} = A_{i+e} - A_i\left(2^d\right)^{e-o}$. One can observe the followings: (i) if $o \leq e-o$ its support is a set of exactly 2^{2do} values on which X_{i+e} is uniformly distributed; (ii) if $o > e-o$ its support is a set of at least 2^{do} values. More precisely, some of the elements taken by X_{i+e} can indeed be described by two distinct couples (A_i, A_{i+e}), (A'_i, A'_{i+e}) with $A_i \neq A'_i$ or $A_{i+e} \neq A'_{i+e}$. Consequently, for any value x of this set we can upper bound its probability of occurence by 2^{do-1}. Since our interest is to upperbound $\Pr\left[|w_i - w_j| \leq L/N\right]$, we continue in the following way by denoting $U = 2^{do} - 1$ for a convenient display:

$$\begin{aligned}
\Pr[|w_{i+o} - w_i| \leq L/N] \leq & \Pr\left[|X_{i+e} + ZU| \leq \frac{L}{N} \;\middle|\; \frac{L}{N} < U\right] + \\
& \Pr\left[|X_{i+e} + ZU| \leq \frac{L}{N} \;\middle|\; \frac{L}{N} \geq U\right] \\
\leq & \sum_{Z=0}^{2^{d(e-o)}-1} \Pr\left[|X_{i+e} + ZU| \leq \frac{L}{N} \;\middle|\; \frac{L}{N} < U\right] \Pr\left[Z\right] + \\
& \Pr\left[\left|\frac{X_{i+e}}{U} + Z\right| \leq \left\lceil \frac{L}{NU} \right\rceil \;\middle|\; \frac{L}{N} \geq U\right] \\
\leq & \Pr\left[|X_{i+e}| \leq \frac{L}{N} \;\middle|\; \frac{L}{N} < U\right] \Pr\left[Z = 0\right] + \\
& \Pr\left[|X_{i+e} + U| \leq \frac{L}{N} \;\middle|\; \frac{L}{N} < U\right] \Pr\left[Z = 1\right] + \\
& \Pr\left[\left|\frac{X_{i+e}}{U} + Z\right| \leq \frac{2L}{NU} \;\middle|\; \frac{L}{N} \geq U\right] \\
\leq & \frac{2L/N}{2^{de-1}} + \frac{4L/N}{(2^{do} - 1)2^{d(e-o)}} \leq \frac{6L}{N2^{8N-1}}.
\end{aligned} \tag{3}$$

Whatever the case is, using 2 and 3 we conclude that $\Pr\left[\text{overlap}\right] \leq \frac{7L}{N2^{8N-1}}$.The rest of the proof is similar to that provided for mcXOR except that the multisets G_1, G_2, G_3 and G_4 are composed of the sliding windows used during the queries phase. In addition to the overhead of encryption implied by the synchronization of random walks during an update, we need to take into account the overheads implied by the sliding windows which force us to reencrypt $2(d-1)$ unchanged parts. Since these additionnal parts bring more information to the adversary, we just have to reestimate the cardinal of G_1 to deduct an upper bound of insecurity for the message secrecy. In the same way, by reestimating the cardinal of G_3 we deduct an upper bound of insecurity for the update secrecy. ■

Proof of theorem 10: As regards the distribution of blocks' lengths, the proof is exactly the same that for mcXOR. The exception is, whether we consider the encryption or the update, each variable-length part is ciphered with an input vector which actually is a sliding window (d-wise chain) over a string of random elements. The way in which the parts are ciphered is therefore the same in both cases. Consequently, encryption and update output ciphertexts that are perfectly indistinguishable. ■