Incremental Unforgeable Encryption

Enrico Buonanno[1], Jonathan Katz[2], and Moti Yung[3]

[1] Department of Computer Science, Columbia University, NY, USA.
eb659@columbia.edu
[2] Telcordia Technologies, Inc., NJ, USA and
Department of Computer Science, Columbia University.
jkatz@cs.columbia.edu
[3] CertCo, Inc., USA.
moti@cs.columbia.edu

Abstract. The recent selection of the AES block cipher to replace DES has generated interest in developing new modes of operation to supplement the modes defined as part of the DES standard [1,16,23]. We initiate the study of modes of encryption which are both *incremental* and *unforgeable*, and point out a number of applications for modes meeting these requirements. We also propose three specific modes achieving these goals, and discuss the strengths and weaknesses of each.

1 Introduction

1.1 Motivation

With the recent selection of the proposed AES, there has been an intensification of study on all aspects of private-key cryptography. In particular, there has been much interest in the design and analysis of new modes of operation for private-key encryption (indeed, NIST recently held a workshop focusing on this topic [22]). It is important to note that just as no block cipher is "best" for all applications, the same holds true for modes of encryption (hence the number of different modes proposed). In fact, each application determines a different set of requirements for the encryption scheme to be used.

It is often required to maintain an encrypted copy of data which undergoes frequent, yet small, changes [4]. One example is a user revising a file who wants to maintain an encrypted copy at all times. Other examples include the maintenance of an encrypted database or table throughout the course of many update operations during which isolated entries change while the bulk of the data remains the same. These examples may be taking place in an environment such as an encrypted file system [9] in which the underlying data is constantly changing yet encrypted versions must always be stored.

In such cases, using an incremental encryption scheme can lead to huge efficiency gains. The goal of incremental cryptography, introduced by Bellare, Goldreich, and Goldwasser [3], is to design cryptographic algorithms whose output can be updated very efficiently when the underlying input changes. For example, say we have a document D, and have applied cryptographic transformation T to this document to generate $\mathsf{T}(D)$. The document is then modified via update operation M (where, for example, $M =$ "delete block i") to give new document D'. An incremental update algorithm IncT is one such that $\mathsf{IncT}(D, M, \mathsf{T}(D))$ outputs a valid cryptographic transformation of D'. Note that computation of IncT can be potentially much faster than recomputing $\mathsf{T}(D')$ from scratch. Ideally, the running time of IncT should depend on the type of modification only (and possibly the security parameter), but, in particular, should be independent of the size of the document $|D|$. Even when this cannot be achieved, one might hope for an incremental update operation which runs in time $O(\log |D|)$ instead of time $O(|D|)$ which is required for re-computation from scratch. When documents change frequently, dramatic efficiency improvements are possible.

In many of the aforementioned scenarios, incrementality is not enough. Additionally, one typically requires a guarantee of data integrity. Consider the case of an encrypted file system or remote database. The resulting ciphertext may be stored on an insecure medium (this is the reason for encryption in the first place), and an adversary may be able to modify the ciphertext as he chooses. A malicious attacker should be prevented from modifying the ciphertext so that it will appear correct when it is later decrypted by the user. Data integrity is useful as a means of virus protection [4] if applications and data are always checked for validity before being used. We model this requirement via *unforgeability* [18, 19] (also known as ciphertext integrity [7]). Briefly, a malicious adversary (who views a sequence of encryptions and incremental update operations) should be unable to generate *any* new ciphertext which decrypts to a valid plaintext. This is the strongest notion of integrity for the case of encryption.

1.2 Previous Work

The joint importance of incrementality and integrity in the context of cryptographic file systems has been recognized elsewhere [4]. However, [4] focuses primarily on MAC and hashing algorithms. Achieving these goals simultaneously for the case of *encryption* has not previously been considered.

INCREMENTAL ENCRYPTION. Prior work dealing with incremental cryptography has focused mainly on hashing, signing, and message authentication [3,4,6,10,11, 21]. We are aware of only one previous work dealing with (among other things) incremental encryption [4]; the scheme there is based on encrypting a description of the modification and appending it to the end of the current ciphertext. A comparison of our work to [4] is worthwhile:

- A formal definition of security for incremental encryption does not appear in [4]. We provide appropriate definitions here.
- To achieve incrementality in the non-amortized sense, the scheme of [4] is complex and relatively inefficient. It requires $O(\log |D|)$ block cipher evaluations even for simple updates, and results in ciphertext which is as much as

four times longer than the plaintext. The schemes presented here are simple to implement, can be updated using a constant number of block cipher evaluations, and (in some cases) reduce the ciphertext expansion.
– The scheme of [4] does not achieve any measure of integrity (this was not the focus of their work). They did not consider active attacks, only semantic security. The schemes presented here are unforgeable under the strongest definition of the term.
– The basic scheme of [4] is not oblivious in the sense of hiding the revision history of the document (see Section **??**). The schemes presented here are all oblivious, without requiring additional complexity.

In fairness, the notion of security considered in [4] is stricter than that which we consider here. Specifically, we allow the adversary to determine the *location* of modifications made (although an adversary cannot determine the nature of the change). In practice, we believe this is not a serious concern. Discussion of this point appears in Section 2.3.

Unforgeability. The importance of integrity in the context of private-key encryption has been recognized for some time. Besides being important in its own right, integrity implies security against chosen ciphertext attacks [19] (see also [7]). Recently, there have appeared a flurry of definitions, detailed analyses, and modes of encryption all intending to more carefully address this issue [19, 17,7,8,12,24,22,15]. We use the notion of *unforgeability* [18,19] (also known as ciphertext integrity [7]), which is the strongest notion of integrity. Under this definition, an adversary may observe a sequence of encryptions (and incremental updates in our case) yet should be unable to generate *any* new ciphertext which decrypts to valid plaintext. The formal definition appears in Section 2.4.

1.3 Summary of Results

We begin by presenting our definitions: a definition for incremental encryption, formal definitions of security (privacy and integrity) for the setting of incremental encryption, and a definition of obliviousness. We then introduce three modes of encryption achieving both incrementality and unforgeability, and give theorems indicating the exact security of each construction. We conclude with a discussion comparing the strengths and weaknesses of these modes.

2 Definitions

2.1 Notation

For probabilistic algorithm A, denote by $y \leftarrow A(x_1, x_2, \ldots)$ the experiment in which we generate random coins r for A and let y equal the output of $A(x_1, x_2, \ldots; r)$. Furthermore, let $\{A(x_1, x_2, \ldots)\}$ represent the probability distribution defined by execution of A on the specified input, with coins for A generated randomly. If S is a set, then $b \leftarrow S$ denotes assigning to b an element uniformly chosen from S. If $p(x_1, x_2, \ldots)$ is a predicate, the notation

$\Pr[x_1 \leftarrow S; x_2 \leftarrow A(x_1, y_2, \ldots); \cdots : p(x_1, x_2, \ldots)]$ denotes the probability that $p(x_1, x_2, \ldots)$ is true after ordered execution of the listed experiments.

Strong Pseudorandom Permutations. We follow [13,2,20] in defining a strong pseudorandom permutation family F as one for which the input/output behavior of F_{sk}, F_{sk}^{-1} "looks random" to someone who does not know the randomly selected key sk. We refer to the listed references for details.

2.2 Incremental Encryption

Document Modifications. We view the document D as a sequence of blocks $\sigma_1, \ldots, \sigma_n$, where the blocksize may depend on the security parameter k. For simplicity, we assume that all documents consist of an integral number of blocks (documents can be padded using standard methods if this is not the case). In the context of incremental cryptography, various modification operations have been considered. We represent a generic modification operation by M and denote the effect of M on document D by $D\langle M\rangle$. We define sequential modifications operations by: $D\langle M_1, M_2, \ldots, M_k\rangle \overset{\text{def}}{=} (\cdots((D\langle M_1\rangle)\langle M_2\rangle)\cdots)\langle M_k\rangle$. We consider the following *types* of modification operations:

- $M = (\mathsf{delete}, i)$ deletes block i of the document.
- $M = (\mathsf{insert}, i, \sigma)$ inserts σ between the i^{th} and $(i+1)^{\text{th}}$ blocks of the document.
- $M = (\mathsf{replace}, i, \sigma)$ changes the i^{th} block of the document to σ.

Other modifications of interest include the cut and paste operations, which divide a document into two or combine two documents together. Although some of the schemes presented here support these modifications, we leave the details for a future version of this paper.

The *location* of modification operation M is the block number i which is modified. We always implicitly assume that a modification operation is valid; that is, it represents some feasible[1] modification of D. For the operations considered above, we define $|M|$ to be the underlying blocksize.

Incremental Encryption Schemes. We recall the generic definition of an incremental algorithm as presented by Bellare, Goldreich, and Goldwasser [4], modified here for the case of encryption. (Although the definition below explicitly mentions the security parameter k, we omit this parameter when discussing concrete security definitions and theorems since, in practice, we are given a fixed-size block cipher only.)

Definition 1. *An incremental, private-key encryption scheme Π defined over modification space $\mathcal{M}$ is a 4-tuple of algorithms $(\mathcal{K}, \mathcal{E}, \mathsf{IncE}, \mathcal{D})$ in which:*

- *$\mathcal{K}$, the* key generation algorithm, *is a probabilistic, poly(k)-time algorithm that takes as input security parameter k (in unary) and returns secret key sk. The security parameter also fixes a block size b.*

[1] E.g., we do not call (delete, i) on D when D contains fewer than i blocks.

– $\mathcal{E}$*, the* encryption algorithm*, is a probabilistic, poly*$(k, |D|)$*-time algorithm that takes as input sk and document* $D \in (\{0,1\}^b)^+$ *and returns ciphertext* C*.*
– IncE*, the* incremental update algorithm*, is a probabilistic, poly*$(k, |C|, |M|)$*-time algorithm that takes as input secret key sk, document* D*, modification operation* $M \in \mathcal{M}$*, and ciphertext* C *and returns modified ciphertext* C'*.*
– $\mathcal{D}$*, the* decryption algorithm*, is a deterministic, poly*$(k, |C|)$*-time algorithm that takes as input secret key sk and ciphertext* C *and returns either document* D *or a special symbol* $\perp$ *to indicate that ciphertext* C *is invalid.*

We require that for all sk which can be output by $\mathcal{K}$*, for all valid* D*, we have* $\mathcal{D}_{sk}(\mathcal{E}_{sk}(D)) = D$*. Additionally, for all sk which can be output by* $\mathcal{K}$*, for all valid* D*, for all modifications* $M \in \mathcal{M}$*, we have* $\mathcal{D}_{sk}(\mathsf{IncE}_{sk}(D, M, \mathcal{E}_{sk}(D))) = D\langle M \rangle$*.*

Remark 1. We do not define the behavior of $\mathsf{IncE}_{sk}(D, M, C)$ in the case when it is given "garbage" input; i.e., when $\mathcal{D}_{sk}(C) \neq D$ or $\mathcal{D}_{sk}(C) = \perp$. We refer to this sort of input as *invalid* for the incremental update algorithm.

Remark 2. It may seem strange that we allow the running time of IncE to be polynomial in $|D|$. However, such schemes may be of interest; for example, when an update can be done in half the time it would take to re-encrypt from scratch. Optimally, the running time of IncE should be independent of the length of document D. Following [4], such schemes are called *ideal.* In the present schemes, IncE does not require access to the original document D; thus, execution of the incremental update algorithm is abbreviated by $\mathsf{IncE}_{sk}(M, C)$.

Remark 3. The underlying modification space $\mathcal{M}$ is included in the specification. As discussed in Section 2.2, various modification operations can be considered; not every incremental encryption scheme supports every such operation.

2.3 Indistinguishability of Incremental Encryption

Although security for incremental encryption has been discussed informally in [4], this is the first formal definition of which we are aware. The secrecy requirements (informally) are as follows: first, the basic encryption algorithm should be semantically secure. Second, the incremental update algorithm should not somehow "ruin" the semantic security of the encryption. Finally, the incremental update algorithm itself should not leak information (discussed in more detail below) about the underlying modification.

According to the previous definition [4], an adversary, upon observing an incremental update, should not be able to determine the location of the modification taking place or the symbol being modified. We relax this requirement and allow the possibility that an adversary can determine where a modification takes place (but still cannot determine the symbol being modified). For example, an adversary should be unable to distinguish between a $(\mathsf{replace}, i, \sigma)$ and a $(\mathsf{replace}, i, \sigma')$ modification. The case addressed by the previous definition is often not of practical concern (the location of a change may be known, anyway). Furthermore, the other benefits of our schemes outweigh this weaker security guarantee in many situations. In particular, our schemes are oblivious (hide details about the document modification history) and unforgeable, which are often more desirable goals. Finally, it is unclear how to extend the previous definition to handle modifications such as cut and paste, in which the "location" of the modification is trivially determined.

Formally, we model security using the notion of find-then-guess indistinguishability [2], which implies the standard notion [14] of semantic security. The adversary may interact with an *encryption oracle* $\mathcal{E}_{sk}(\cdot)$ and an *incremental update oracle* $\mathsf{IncE}_{sk}(\cdot,\cdot)$. Then, the adversary outputs either two documents (D_1, D_2) (which must have equal number of blocks ℓ) or a ciphertext C along with two modification operations (M_1, M_2) (which must be of the same type and modifying the same location). A bit b is chosen at random and kept hidden from the adversary. In the first case document D_b is encrypted, while in the second case the IncE algorithm is applied to M_b and C; in either case, the result is given to the adversary. The adversary may then continue to interact with the $\mathcal{E}_{sk}$ and IncE_{sk} oracles. We say the adversary *succeeds* if it correctly guesses b. We define the advantage of the adversary as twice the probability of success, minus 1/2.

Our concrete security definition follows the approach employed by [5]. We say that encryption scheme Π is $(t, q_e, \mu, q_{inc}, \ell; \epsilon)$-secure in the sense of indistinguishability if, for any adversary A which runs in time t, making q_e queries to the encryption oracle and q_{inc} queries to the IncE oracle (with the total number of blocks in all encryption and incremental update queries equal to μ), the advantage of A is less than ϵ.

2.4 Unforgeability of Incremental Encryption

We now consider unforgeability for incremental modes of encryption, extending the definition of [19]. The adversary is again allowed to interact with an encryption oracle and an incremental update oracle. Now, however, we allow the adversary to submit only *valid* queries (in the sense of Remark 1) to the IncE algorithm. At the end of its execution, the adversary outputs a ciphertext C which must be different from any ciphertext it received from either of its oracles. The adversary succeeds if C is valid. Formally:

Definition 2. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathsf{IncE}, \mathcal{D})$ be an incremental encryption scheme over modification space $\mathcal{M}$, and let A be an adversary. Let $\mathsf{Adv}^{\mathrm{unf}}_{A,\Pi} \overset{\mathrm{def}}{=}$*

$$\Pr\left[\mathrm{sk} \leftarrow \mathcal{K}; C \leftarrow A^{\mathcal{E}_{sk}(\cdot), \mathsf{IncE}_{sk}(\cdot,\cdot)} : \mathcal{D}_{sk}(C) \neq \perp\right].$$

We insist that A's queries to the IncE oracle are all valid, and that C was never received in response from either oracle. We say that Π is $(t, q_e, \mu, q_{inc}; \epsilon)$-secure in the sense of unforgeability if, for any adversary A which runs in time t, making

q_e queries to the encryption oracle and q_{inc} valid queries to the IncE *oracle (with the total number of blocks in all encryption and incremental update queries equal to μ),* $\mathsf{Adv}^{\mathrm{unf}}_{A,\Pi}$ *is less than ϵ.*

The above definition corresponds to the "basic security" of [3,4]. One may also consider the more complicated setting in which the adversary is given unrestricted access to the IncE oracle, and is allowed to submit invalid queries as well. Security in this setting is generally much more difficult to achieve [3,4,6]. In this paper, we concentrate on the case of basic security only[2] (which in typical applications is sufficient).

2.5 Obliviousness of Incremental Encryption

An incremental encryption scheme raises new security concerns. One such concern is that a ciphertext may reveal information about the revision history of the underlying plaintext. We say a scheme is *oblivious* if this revision history is hidden *even to someone who knows the secret key.* Motivation for this concern arises when the ciphertext is transmitted between two parties; the first party sending the ciphertext will not, in general, want the second party to be able to learn about the modifications made in the course of creating the document. This notion was first formally defined by Micciancio [21]; we modify his definition for the present context:

Definition 3. *Let Π be an incremental encryption scheme over modification space $\mathcal{M}$. We say that Π is* oblivious *if, for any two documents D, D', for any sequence of modifications $M_1, \ldots, M_i \in \mathcal{M}$ such that $D' = D\langle M_1, \ldots, M_i \rangle$, and for all keys sk, we have:*

$$\{\mathcal{E}_{sk}(D')\} \equiv \{\mathsf{IncE}_{sk}(M_i, \cdots, \mathsf{IncE}_{sk}(M_1, \mathcal{E}_{sk}(D)) \cdots)\}.$$

3 Incremental and Unforgeable Modes of Encryption

3.1 Encrypt-then-Incremental-MAC

One method of achieving both incrementality and unforgeability is to use an incremental mode of encryption together with an incremental MAC [3,4,6] of the ciphertext (the *encrypt-then-*MAC approach [7]). Let inc-$\mathcal{E}$ be an incremental encryption scheme, and let inc-MAC be a secure, incremental MAC algorithm. Then incremental, unforgeable encryption can be performed as follows:

$$\mathcal{E}_{k_1,k_2}(D) = C \circ \text{inc-MAC}_{k_2}(C), \tag{1}$$

where $C = \text{inc-}\mathcal{E}_{k_1}(D)$. Incremental updates are done in the straightforward way: first, perform the incremental update operation for C (based on the new

[2] The indistinguishability of our schemes, however, holds even when an adversary is given unrestricted access to the IncE oracle, as reflected in the definition.

document) to obtain C'; then, perform the incremental update operation for the MAC (based on the new ciphertext C').

Care must be taken with the details of inc-$\mathcal{E}$. First, note that not all incremental encryption algorithms will result in practical schemes. For example, if the incremental update operation for inc-$\mathcal{E}$ changes a large fraction[3] of the ciphertext C, there may not exist an efficient incremental update operation for the MAC. Furthermore, one must ensure that the incremental encryption scheme is indeed secure under the definition of Section 2.3. One possible solution is the following "randomized" ECB mode of encryption (rECB): given message $\sigma_1, \ldots, \sigma_n$, choose random $r_0, r_1, \ldots, r_n \leftarrow \{0,1\}^b$ (where b is the blocksize) and compute:

$$F_{sk}(r_0), F_{sk}(r_1 \oplus r_0), F_{sk}(\sigma_1 \oplus r_1), \ldots, F_{sk}(r_n \oplus r_0), F_{sk}(\sigma_n \oplus r_n).$$

This mode is a secure, incremental encryption scheme, with updates (replace, insert, and delete) done in the obvious way. Note that incremental updates result in only small changes to the ciphertext; thus, we can efficiently combine this mode with an incremental MAC algorithm (which handles replace, insert, and delete), as in (1). If the incremental MAC is secure, the result is a mode of encryption which is both incremental and unforgeable.

3.2 inc-IAPM Mode

The previous proposal is attractive for its simplicity. Unfortunately, in practice, incremental MAC algorithms are not very efficient (see Section 5). Thus, we propose other modes which improve the computational efficiency and ciphertext expansion rate.

The inc-IAPM mode described here is directly based on the IAPM mode of Jutla [17]. IAPM mode represents an advance over previous unforgeable modes of encryption, and is more efficient than the standard "encrypt-then-MAC" approach by a factor of two. Furthermore, it is parallelizable, which suggested to us the possibility that it could be adapted to give an incremental mode. However, it is non-trivial to modify this mode to achieve an *efficient*, incremental mode while completely satisfying our definitions of security.

We now present the details. Similarly to the IAPM mode, encrypting an n-block plaintext requires a sequence of pairwise independent blocks (labeled $S_0, S_1^\ell, S_1^r, \ldots, S_{n+1}^\ell, S_{n+1}^r, S^*$; see below) which will be used for "output whitening". These blocks are generated from a random seed K which is included with the ciphertext. Generation of these output whitening blocks is described in [17], and a more detailed analysis of the output whitening appears in [15]. It is important to note that generation of these blocks can be done very efficiently using $\mathcal{O}(\log n)$ block cipher calls [17] or even without using a block cipher at all [15]. In the analysis of our construction, we assume that blocks generated from the same seed are pairwise independent and blocks generated from different seeds are completely independent (this can be refined along the lines of [15]).

[3] Recall that an incremental encryption scheme might require time $O(|D|)$ for update operations, as long as this time is less than that required to re-encrypt from scratch.

Let F be a block cipher family with blocksize b. The key generation algorithm outputs a random key sk for F (in addition to the key necessary for generation of the whitening sequence). The plaintext is parsed as a sequence of b-bit blocks $\sigma_1, \ldots, \sigma_n$. Encryption is described in Fig. 1 (generation of the whitening sequence from seed K is as described above). Decryption is done by decrypting each ciphertext block, re-generating the pairwise independent sequence, and checking the integrity conditions on L_{n+1}, R_{n+1}, and σ^*. If the integrity checks succeed, D_i is computed as $L_i \oplus R_i$; if they fail, the output is $\perp$.

```
Algorithm $\mathcal{E}_{sk,sk'}(D;K)$
  Generate $S_0, \ldots$ using $K$ and $sk'$
  for $i = 1$ to $n$ :
     $L_i \leftarrow \{0,1\}^b; R_i = L_i \oplus \sigma_i$
  $L_0, R_0 \leftarrow \{0,1\}^b$
  $L_{n+1} = \bigoplus_{i=0}^n L_i$
  $R_{n+1} = \bigoplus_{i=0}^n R_i$
  $\sigma^* = L_0 \oplus R_0$
  for $i = 0$ to $n+1$ :
     $C_i^\ell = F_{sk}(L_i \oplus S_0^\ell) \oplus S_i^\ell$
     $C_i^r = F_{sk}(R_i) \oplus S_i^r$
  $C^* = F_{sk}(\sigma^* \oplus S^*) \oplus S_0$
  return $K, C_0^\ell, C_0^r, \ldots, C_{n+1}^\ell, C_{n+1}^r, C^*$
```

Fig. 1. inc-IAPM mode of encryption.

For this particular mode, we are only able to achieve security with respect to the replace operation[4]. The incremental (replace, i, σ) operation proceeds as follows. First, blocks L_i and R_i are updated to reflect the new value of block i. In addition, new random values L_0 and R_0 are selected. Finally, in order to satisfy the integrity check, the algorithm updates the values for L_{n+1}, R_{n+1}, and σ^*. Details follow; for simplicity, we describe the algorithm informally (the ciphertext is parsed as a sequence of b-bit blocks $K, C_0^\ell, C_0^r, \ldots, C^*$):

```
Algorithm $\mathsf{IncE}_{sk,sk'}((\mathsf{replace}, i, \sigma), C)$
  Compute $S_0, S_0^\ell, S_0^r, S_i^\ell, S_i^r, S_{n+1}^\ell, S_{n+1}^r, S^*$ from $K$
  Decrypt to obtain $L_0, R_0, L_i, R_i, L_{n+1}, R_{n+1}$, and $\sigma^*$
  Choose random $L_0'$, $L_i'$, and $R_0'$
  Set $R_i' = L_i \oplus \sigma$
  Update $L_{n+1}'$, $R_{n+1}'$, and $\sigma'^*$ to satisfy the integrity checks
  Re-encrypt $L_0', R_0', L_i', R_i', L_{n+1}', R_{n+1}'$, and $\sigma'^*$
```

[4] Incremental delete and insert modifications are known to be significantly more difficult to achieve in general [3,6].

3.3 RPC Mode

The inc-IAPM mode is more efficient than the encrypt-then-incremental-MAC approach, yet it only supports incremental replace operations. Here, we introduce RPC mode which is slightly less efficient than inc-IAPM, but supports replace, insert, and delete. The mode is specified by parameters b and r, where b is the block size and r is the amount of random padding. The document D is parsed as a sequence of $b - 2r$-bit blocks $\sigma_1, \ldots, \sigma_n$. Encryption is performed as follows (start is not part of the valid message space):

```
Algorithm $\mathcal{E}_{sk}^{b,r}(D)$
 for $i = 0$ to $n$ :
   $r_i \leftarrow \{0,1\}^r$
 $C_0 = F_{sk}(r_0, \mathsf{start}, r_1)$
 for $i = 1$ to $n-1$ :
   $C_i = F_{sk}(r_i, \sigma_i, r_{i+1})$
 $C_n = F_{sk}(r_n, \sigma_n, r_0)$
 $r^* = \bigoplus_{i=1}^{n} r_i$
 $C^* = F_{sk}(r^* \oplus r_0, 0^{b-2r}, r^*)$
 return $C_0 \ldots C_n C^*$
```

Decryption is done in the obvious way, by computing F_{sk}^{-1} for each block of the ciphertext and then checking that the first block contains an encryption of start, that the values $\{r_i\}$ are chained correctly, and that decryption of C^* gives the correct r_0 and checksum. If these integrity checks succeed, the computed document is output; if they fail, the output is $\perp$.

For lack of space, we describe (informally) the incremental delete, insert, and replace algorithms (a detailed description of these algorithms will appear in the full paper): The block i to be modified and adjacent blocks are decrypted to determine r_{i-1}, r_i, and r_{i+1}. When a new symbol σ is to be placed in position j (where $j = i, i+1$), σ is encrypted by choosing new, random r'_j and computing $C'_j = F_{sk}(r'_j, \sigma, r_{i+1})$. This causes the $\{r_i\}$ to remain chained correctly. Furthermore (and this occurs even during a delete modification), new r_0 and r_1 are chosen at random, and the checksum r^* is recomputed (and the resulting blocks are re-encrypted to give modified ciphertext blocks C'_0, C'_1, and C'^*).

4 Proofs of Security

The obliviousness of each mode is clear, by inspection. Thus, we focus on indistinguishability and unforgeability. We assume throughout this section that the block cipher family F is a family of strong pseudorandom permutations. The constructions can thus be analyzed by viewing F as a truly random permutation on b bits; "real-world" security bounds can be derived from the theorems below and exact security bounds on F. The theorems below give rough bounds on the security of the constructions. More detailed proofs, and tighter security bounds, will appear in the full version of this paper.

4.1 Encrypt-then-Incremental-MAC

A general composition theorem (along the lines of [7]) shows that a secure, incremental encryption scheme appended by a secure[5], incremental MAC of the result gives a secure, incremental encryption scheme which is also unforgeable. For the concrete security analysis, however, we choose the randomized ECB (rECB) mode (see Section 3.1) as the encryption algorithm, and the XOR-MAC of [4] (using r bits of random padding per block) as the MAC algorithm.

Theorem 1. rECB-XOR *over modification space* $\mathcal{M} = \{\mathsf{replace}, \mathsf{delete}, \mathsf{insert}\}$ *is* $(t, q_e, \mu, q_{inc}, \ell; \epsilon)$*-secure in the sense of indistinguishability, where:*

$$\epsilon = \mathcal{O}\left(\frac{\ell^2 + \mu\ell}{2^b} + \frac{\ell^2\mu^2}{2^{2b}}\right).$$

Proof (Sketch) Indistinguishability of the scheme is determined by rECB alone (since the MAC is computed on the ciphertext, it can be simulated by the adversary). Consider all blocks used as input to F during the course of the encryption oracle calls and incremental update oracle calls. Call such blocks *used.* First consider the case where the adversary outputs two documents, each containing ℓ blocks. Let r_0 and $\mathcal{B} = \{r_1 \oplus r_0, \sigma_1 \oplus r_1, \ldots, r_\ell \oplus r_0, \sigma_\ell \oplus r_\ell\}$ (cf. Section 3.1) be the set of blocks used as input to F during the encryption of the chosen document. The adversary has non-zero advantage in only the following cases: (1) For some $B_1, B_2 \in \mathcal{B}$, it is the case that $B_1 = B_2$; (2) for some i, $r_i \oplus r_0$, and $\sigma_i \oplus r_i$ are both *used.* The probability of (1) is bound by $\mathcal{O}(\ell^2/2^b)$, while the probability of (2) is bound by $\mathcal{O}(\ell^2\mu^2/2^{2b})$.

When the adversary outputs two modifications, the adversary has a non-zero advantage only if the modifications were of type $\mathsf{replace}$ or insert. In either case, a new, random r_i is chosen. The adversary can have non-zero advantage only if either (1) $r_i \oplus r_0$ is equal to $\sigma \oplus r_i$; or (2) $r_i \oplus r_0$ and $\sigma \oplus r_i$ are both *used.* The probability of either of these events is bound by $\mathcal{O}(\mu/2^b)$. ■

Theorem 2. rECB-XOR *over modification space* $\mathcal{M} = \{\mathsf{replace}, \mathsf{delete}, \mathsf{insert}\}$ *is* $(t, q_e, \mu, q_{inc}; \epsilon)$*-secure in the sense of unforgeability, where:*

$$\epsilon = \mathcal{O}\left(\frac{(q_e + q_{inc})^2}{2^b} + \frac{\mu^2}{2^r}\right).$$

Proof Unforgeability of the scheme is determined by the unforgeability of the MAC; taking the bounds from Theorem 3.1 of [4] gives the desired result. ■

[5] The security of the MAC must be that it is infeasible to forge a new, valid *pair* (M, tag); it is not sufficient that it be infeasible to forge a valid tag on a new message.

4.2 inc-IAPM

Theorem 3. *The inc-IAPM mode of encryption over modification space* $\mathcal{M} = \{\mathsf{replace}\}$ *is* $(t, q_e, \mu, q_{inc}, \ell; \epsilon)$*-secure in the sense of indistinguishability, where:*

$$\epsilon = \mathcal{O}\left(\frac{q_e + q_{inc}}{2^b} + \frac{\ell^4}{2^{2b}}\right).$$

Proof (Sketch) Consider all blocks x used as input to F during the course of the encryption oracle calls and incremental update oracle calls. Call all such blocks *used.* Note that a ciphertext block received by the adversary is computed as $C_i = F_{sk}(x_i) \oplus S_i$; in this case we say that x_i is *used with* S_i. First consider the case where the adversary outputs two documents, each containing ℓ blocks. Let $\mathcal{B} = \{L_1 \oplus S_0^\ell, R_1, \ldots L_n \oplus S_0^\ell, R_n\}$ be the set of blocks used as input to F during encryption of the chosen document. The adversary has non-zero advantage only in the following cases: (1) There exist $B_1, B_2, B_3, B_4 \in \mathcal{B}$ such that $B_1 = B_2$ and $B_3 = B_4$; (2) for some i, $L_i \oplus S_0^\ell$ is *used with* S_i^ℓ and R_i is *used with* S_i^r. This is similar to the analysis in Theorem 1, except that the pairwise-independent output whitening modifies conditions (1) and (2). The probability of (1) is $\mathcal{O}(\ell^4/2^{2b})$, and the probability of (2) is bound by $\mathcal{O}(q_e/2^b)$.

When the adversary outputs two modifications, the modification must (by definition) be of type $\mathsf{replace}$. In this case, a random block L_i is chosen. The adversary's success will be 0 unless $L_i \oplus S_0^\ell$ is *used with* S_i^ℓ and R_i is *used with* S_i^r. The probability of this occurring is bounded by $\mathcal{O}(q_{inc}/2^b)$. ■

Theorem 4. *The inc-IAPM mode of encryption over modification space* $\mathcal{M} = \{\mathsf{replace}\}$ *is* $(t, q_e, \mu, q_{inc}; \epsilon)$*-secure in the sense of unforgeability, where:*

$$\epsilon = \frac{1}{2^b} + \mathcal{O}\left(\frac{q_e^2 + q_{inc}^2}{2^b}\right).$$

Proof (Sketch) Assume that seeds used in encryption oracle calls never repeat; such repetitions occur only with probability $\mathcal{O}(q_e^2/2^b)$. Let $\mathcal{C}$ be the set of ciphertexts received by the adversary from all its oracle calls, and let C' be the new ciphertext output by the adversary. Clearly, the adversary's success will be bounded by 2^{-3b} if C' uses a seed which has never appeared in any of the ciphertexts in $\mathcal{C}$. Now, assume the adversary uses a seed K which was used previously. Denote by $\mathcal{C}_K \subset \mathcal{C}$ the set of all ciphertexts which use this seed. Note that since seeds do not repeat for encryption oracle calls and we are dealing with $\mathsf{replace}$ operations only, all ciphertexts in $\mathcal{C}_K$ have the same length. By a similar argument to that of [17], the adversary has advantage at most $1/2^b$ if the length of C' is not equal to this length.

For each position i $(0 \leq i \leq n+1)$, let $\mathcal{C}_i^\ell$ be the set of ciphertext blocks corresponding to position C_i^ℓ in all ciphertexts in $\mathcal{C}_K$. Define $\mathcal{C}_i^r$ and $\mathcal{C}^*$ analogously. The adversary has success bounded by 2^{-b} if, for any i, $C_i'^\ell \notin \mathcal{C}_i^\ell$ (with similar statements holding for $C_i'^r$ and C'^*). This holds by pairwise independence of the values in the output whitening sequence (and complete independence from sequences generated by other seeds).

Denote by $\mathcal{L}_i$ the set of values corresponding to decryption of blocks in $\mathcal{C}_i^\ell$, and by L_i' the value corresponding to decryption of block $C_i'^\ell$ in the adversary's output ciphertext. From the argument above, we may assume $L_i' \in \mathcal{L}_i$. Note that the adversary will have success probability bounded by 2^{-b} unless it is that case that $L_0, \ldots, L_{n+1}$ were all used *together* in some ciphertext in $\mathcal{C}_K$. This is true since the values $\cup_i \mathcal{L}_i$ are completely independent except for the relation (defined by the checksum) which holds between L_i's used in the same ciphertext. In other words, the adversary's success is bounded by 2^{-b} unless the blocks C_i^ℓ were all used together in some ciphertext in $\mathcal{C}_K$. A similar argument holds for the R_i (and hence the C_i^r). The same argument also holds for L_0, R_0, and D^*, and hence for C_0^ℓ, C_0^r, and C^*. Thus, unless L_0 has repeated for some pair of ciphertexts in $\mathcal{C}_K$ (which happens only with probability $\mathcal{O}(q_{inc}^2/2^b)$), the adversary has advantage at most 2^{-b} if it outputs a ciphertext different from those in $\mathcal{C}_K$. ■

4.3 RPC Mode

Theorem 5. RPC^{b,r} *mode over modification space* $\mathcal{M} = \{\mathsf{replace}, \mathsf{delete}, \mathsf{insert}\}$ *is* $(t, q_e, \mu, q_{inc}, \ell; \epsilon)$*-secure in the sense of indistinguishability, where:*

$$\epsilon = \mathcal{O}\left(\frac{\ell^2 + \ell\mu}{2^{2r}} + \frac{q_{inc}}{2^r}\right).$$

Proof (Sketch) Consider all blocks used as input to F during the encryption and incremental update oracle calls. Call all such blocks *used*. We first consider the case when the adversary outputs two documents, each containing ℓ blocks. Let $\mathcal{B} = \{r_1|\sigma_1|r_2, r_2|\sigma_2|r_3, \ldots, r_\ell|\sigma_\ell|r_0\}$ be the set of blocks used as input to F during the encryption of the chosen document. The adversary has non-zero advantage in only the following cases: (1) For some $B_1, B_2 \in \mathcal{B}$, it is the case that $B_1 = B_2$; (2) some $B \in \mathcal{B}$ is *used*. The probability of (1) is $\mathcal{O}(\ell^2/2^{2r})$, while the probability of (2) is $\mathcal{O}(\ell\mu/2^{2r})$.

When the adversary outputs two modifications, the adversary has a non-zero advantage only if the modifications were of type replace or insert. In either case, a random r' is chosen. The adversary has non-zero advantage only when $r'|\sigma|r_{i+1}$ is *used*. But the probability of this is bounded by $\mathcal{O}(q_{inc}/2^r)$. ■

Theorem 6. RPC^{b,r} *mode over modification space* $\mathcal{M} = \{\mathsf{replace}, \mathsf{delete}, \mathsf{insert}\}$ *is* $(t, q_e, \mu, q_{inc}; \epsilon)$*-secure in the sense of unforgeability, where:*

$$\epsilon = \frac{1}{2^{2r}} + \mathcal{O}\left(\frac{(q_e + q_{inc})^2}{2^{2r}}\right).$$

Proof (Sketch) Denote by $\mathcal{C}$ the set of all ciphertexts received by the adversary from its oracles. Note that the initial blocks of all ciphertexts in $\mathcal{C}$ are distinct,

Table 1. Comparison of the various modes. See text for details.

	Block size	Expansion	Cipher evals	IND	Unf	Comments
rECB-XOR	128	4	$6n$	$\frac{\mu\ell}{2^{128}}$	$\frac{\mu^2}{2^{64}}$	
	256	2.7	$4.6n$	$\frac{\mu\ell}{2^{256}}$	$\frac{\mu^2}{2^{64}}$	
inc-IAPM	128	2	$2n$	$\frac{q_{tot}}{2^{128}}$	$\frac{q_{tot}^2}{2^{128}}$	only replace
RPC	128	4	$4n$	$\frac{q_{inc}}{2^{48}}$	$\frac{q_{tot}^2}{2^{96}}$	
	256	2	$2n$	$\frac{q_{inc}}{2^{64}}$	$\frac{q_{tot}^2}{2^{128}}$	

except with probability $\mathcal{O}((q_e + q_{inc})^2/2^{2r})$. Now, assume they are all distinct. Let the ciphertext output by the adversary be denoted C'.

Denote by $\mathcal{C}_0$ the set of values for the first block, for all ciphertexts in $\mathcal{C}$. Note that the adversary's success probability is bounded by 2^{-2r} if $C'_0 \notin \mathcal{C}_0$. Let $C \in \mathcal{C}$ be such that $C_0 = C'_0$ (by the assumption above, this defines a unique C). C and C' either consist of the same blocks (possibly in different order), or there exists a block which appears a different number of times in C and C'. In the first case, note that if the blocks are in different order, then the adversary's probability of success is bounded by 2^{-2r}. In the second case, since the random nonces used for the additional block are independent of all other nonces in the sequence (in particular, the value r_1 defined by block C_0), the adversary's probability of success is bounded by 2^{-2r}. ■

5 A Comparison of the Various Schemes

The encrypt-then-incremental-MAC approach is appealing because it is so simple. Unfortunately, incremental MAC algorithms are, in practice, difficult to design. The constructions of [3,6] are number-theoretic in nature and involve algebraic operations over groups instead of fast bit-wise operations such as XOR. A tree-based suggestion of [4] is practical, but update operations require $O(\log |D|)$ block cipher evaluations. The only truly practical incremental MAC (with constant update time) of which we are aware is the XOR-scheme of [4]. We call the resulting scheme (cf. Section 3.1) rECB-XOR.

Consider implementation with a 128-bit block cipher, and using $r = 64$ bits of padding for the MAC. This results in ciphertext which is more than four times the length of the plaintext (for an n-block document, rECB encryption results in $2n$ blocks, and the tag computed by the MAC is an additional $2n$ blocks). Encryption of a document from scratch is computationally expensive. An n-block document requires $2n$ block cipher evaluations for the encryption and an additional $4n$ block cipher evaluations to compute the MAC, for a total of $6n$ block cipher evaluations. Furthermore, the security guarantee (for unforgeability) is not as strong as one might like, since it is limited by the factor of $\mu^2/2^{64}$ even though the underlying block cipher is 128 bits long. On the other hand, the scheme supports incremental insert, delete, and replace operations, and each can be done in constant time. Performance of the mode is enhanced slightly when using a 256-bit block cipher. In this case, keeping $r = 64$ results in ciphertext only 2.7 times the length of the plaintext and only $4.6n$ block cipher evaluations are required for encryption of an n-block document. The security guarantee (for unforgeability) is still relatively poor.

The inc-IAPM mode results in ciphertext which is twice the length of the plaintext. Encryption of an n-block document requires only $2n + \mathcal{O}(1)$ block cipher evaluations (this is comparable to the gain in efficiency of IAPM vs. standard encrypt-then-MAC [17]). The incremental replace operation can be done using only a constant[6] number of block cipher evaluations independent of the document size. We note that this mode is efficient without requiring the larger, 256-bit blocksize. Finally, the security guarantees for inc-IAPM are stronger than that for the previous scheme, most strikingly in the case of unforgeability.

The biggest drawback of the inc-IAPM mode is that it does not support incremental delete and insert operations. This might be acceptable in some contexts; for example, when maintaining a database with a fixed number of entries. Since the number of entries does not change, delete and insert operations are not required. However, it is still important to support incremental replace operations, since this is what will be required when entries are modified.

RPC mode supports incremental replace, insert, and delete modifications in constant time. The ciphertext expansion and computational efficiency, however, are not as good as the inc-IAPM mode. When using a 128-bit block cipher, one might set $r = 48$ which results in ciphertext expansion by a factor of 4 and requires $4n$ block cipher evaluations for encryption of an n-block document. The security (especially in the sense of indistinguishability) is poor, being limited by $q_{inc}/2^{48}$. However, this mode performs much better when using a 256-bit block cipher. Here, we may set $r = 64$, resulting in ciphertext expansion of only 2 and requiring only $2n$ block cipher evaluations for encrypting an n-block document. The security may now be acceptable for many applications.

References

1. ANSI X3.106, "American National Standard for Information Systems—Data Encryption Algorithm—Modes of Operation," American National Standards Institute, 1983.
2. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. FOCS '97.
3. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental Cryptography: The Case of Hashing and Signing. Crypto '94.
4. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental Cryptography and Application to Virus Protection. STOC '95.

[6] Technically speaking, the replace operation may require a polylogarithmic number of *bit-wise* operations to compute members of the whitening sequence. However, these are bit-wise operations, which are about two orders of magnitude faster than block cipher evaluations.

5. M. Bellare, J. Kilian, and P. Rogaway. On the Security of Cipher Block Chaining. CRYPTO '94.
6. M. Bellare and D. Micciancio. A New Paradigm for Collision-Free Hashing: Incrementality at Reduced Cost. Eurocrypt '97.
7. M. Bellare and C. Namprempre. Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm. Asiacrypt 2000.
8. M. Bellare and P. Rogaway. Encode-then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. Asiacrypt 2000.
9. M. Blaze. A Cryptographic File System for Unix. 1st ACM Conference on Computer and Communications Security, 1993.
10. M. Fischlin. Incremental Cryptography and Memory Checkers. Eurocrypt '97.
11. M. Fischlin. Lower Bounds for the Signature Size of Incremental Schemes. FOCS '97.
12. V. Gligor and P. Donescu. Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. FSE 2001.
13. O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. JACM 33(4): 792–807 (1986).
14. S. Goldwasser and S. Micali. Probabilistic Encryption. JCSS, 28: 270-299, 1984.
15. S. Halevi. An Observation Regarding Jutla's Modes of Operation. Available at `http://eprint.iacr.org/2001/015`.
16. ISO 8372, "Information Processing—Modes of Operation for a 64-bit Block Cipher Algorithm," International Organization for Standardization, Geneva, Switzerland, 1987.
17. C. S. Jutla. Encryption Modes with Almost-Free Message Integrity. Eurocrypt 2001, to appear. Also available at `http://eprint.iacr.org`.
18. J. Katz and M. Yung. Complete Characterization of Security Notions for Probabilistic Private-Key Encryption. STOC 2000.
19. J. Katz and M. Yung. Unforgeability and Chosen-Ciphertext-Secure Modes of Operation. FSE 2000.
20. M. Luby. *Pseudorandomness and Cryptographic Applications*, Chapter 14. Princeton University Press, 1996.
21. D. Micciancio. Oblivious Data Structures: Applications to Cryptography. STOC '97.
22. See: `http://www.nist.gov/modes`.
23. National Bureau of Standards, NBS FIPS PUB 81, "DES Modes of Operation," U.S. Department of Commerce, 1980.
24. P. Rogaway. OCB Mode: Parallelizable Authenticated Encryption. Available at [22].