Authenticated On-Line Encryption

Pierre-Alain Fouque, Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette

DCSSI Crypto Lab
51, Boulevard de Latour Maubourg
75700 Paris 07 SP, FRANCE
`Pierre-Alain.Fouque@ens.fr`, `Antoine.Joux@m4x.org`
`Gwenaelle.Martinet@worldonline.fr`, `Fred.Valette@wanadoo.fr`

Abstract. In this paper, we investigate the authenticated encryption paradigm, and its security against blockwise adaptive adversaries, mounting chosen ciphertext attacks on on-the-fly cryptographic devices. We remark that most of the existing solutions are insecure in this context, since they provide a decryption oracle for any ciphertext. We then propose a generic construction called Decrypt-Then-Mask, and prove its security in the blockwise adversarial model. The advantage of this proposal is to apply minimal changes to the encryption protocol. In fact, in our solution, only the decryption protocol is modified, while the encryption part is left unchanged. Finally, we propose an instantiation of this scheme, using the encrypted CBC-MAC algorithm, a secure pseudorandom number generator and the Delayed variant of the CBC encryption scheme.

Keywords: Symmetric encryption, authenticated encryption, chosen ciphertext attacks, blockwise adversaries, provable security.

1   Introduction

An authenticated encryption scheme is a secret key scheme providing both privacy and integrity. In [7], Bellare and Namprempre have studied how to combine encryption scheme and message authentication code (MAC) to construct a secure composition against chosen ciphertext attacks. They have proved that the generically secure way is to first encrypt the data with a symmetric encryption scheme and then compute a MAC of the ciphertext. They called this method the "Encrypt-Then-MAC" paradigm. From the receiver point of view, the "Verify-Then-Decrypt" method is performed by first checking the MAC and if (and only if) the tag is correct, by decrypting.

Some other constructions for authenticated encryption have recently been proposed, [18, 21]. All these constructions ensure both integrity and confidentiality in a single pass and are thus more efficient than the Encrypt-Then-MAC composition. Furthermore, the decryption method is now slightly different: integrity is checked at the end of the decryption process. A lot of papers have also studied how to ensure integrity and confidentiality in a single pass (as in [8, 1]). The main result of these papers is that providing integrity with an encryption

scheme with redundancy and/or encoding is not generically secure. Some requirements are needed on the encoding scheme for the composition to be generically secure. For example, Vaudenay has shown in [23] that using a public redundancy is not a secure way to provide security against chosen ciphertext attacks. Indeed, some reaction attacks are possible: this shows that encoding schemes are usually not sufficient to provide authenticity. Another reaction attack has recently been proposed in [6] against the SSH binary Packet Protocol, proving that this scheme may be insecure in some contexts. However, lot of schemes are now well known to be secure in the strong sense, as the generic Encrypt-then-MAC composition, or single pass schemes as IACBC [18], OCB [21]... Thus constructing authenticated encryption schemes does not seem to be an open problem anymore.

However, in some practical applications the sender and the receiver of an authenticated ciphertext use a cryptographic device to perform the encryption, the checking and the decryption operations. In many cases, this cryptographic module uses a smart card with a limited storage. Due to memory restrictions, the card cannot store the whole ciphertext C to first check the tag τ and then decrypt C if τ was valid, as assumed in the standard model (or store the whole plaintext after decryption, check its integrity and output it only if valid).

To overcome this problem, interactions between the card and the rest of the world are usually performed on-the-fly. However, in [17], a new security flaw of on-line implementations was pointed out. The basic idea is to notice that with such implementations, messages are no longer atomic objects, as assumed in the usual security proofs. This idea leads to a new class of attacks called *blockwise adaptive*. Adversaries using such attacks see the kth ciphertext block before having supplied the $(k+1)$th plaintext block. Note that some techniques used in [6] to attack the SSH protocol are linked with this kind of attackers. In [17], only blockwise adaptive chosen plaintext attacks were considered. In this paper, we focus on *blockwise adaptive chosen ciphertext attacks* and look at the precautions that have to be taken, not only during the encryption phase, but also during the decryption phase, both executed on-the-fly.

Recently another work has been published, formalizing the remotely keyed authenticated encryption [11]. It solves the problem of authenticated encryption on a high bandwidth and insecure channel using a limited bandwidth and memory but highly secure cryptographic device. The work of [11] also proposes a generic way to solve this problem, but the solution they give transforms both the encryption and the decryption parts. In their solution, and in the previous work of [9], the main idea is to use a session key to encrypt the data. The cryptographic device just encrypts this session key and authenticates a hash of the ciphertext. However, in well-established standards such as SSH or SSL, designers cannot accept to change the protocol only for low memory devices. Consequently, our aim is to provide a solution that only modifies the decryption protocol.

Our Results. In this paper we study how authenticated encryption can be securely implemented when dealing with blockwise adaptive adversaries using chosen ciphertext attacks. We show how to securely implement on-line decryption.

In particular, an adversary should not be able to feed a decryption oracle with an invalid ciphertext and obtain the corresponding plaintext. We first describe the usual on-the-fly implementations of the Encrypt-Then-MAC composition and of authenticated encryption in one pass. We show how such implementations lead to totally insecure schemes when considering blockwise adversaries. Trivial attacks are possible: they provide to the adversary a decryption oracle without any check of the integrity. Since the basic encryption schemes are generally not secure against chosen ciphertext attacks, this leads to major security flaws.

After having formally described the blockwise adaptive security model, both for confidentiality and integrity, we propose, as an alternative to existing solutions, a new generic construction and we prove its security. This solution requires an encryption scheme only secure against blockwise adversaries mounting chosen plaintext attacks, together with a new decryption protocol. The main idea is to blind the plaintext blocks obtained after decryption by XORing them with a pseudorandom sequence of bits. Then, when the integrity of the ciphertext is checked, the seed used to generate the pseudorandom sequence is returned. Thus the cryptographic device does not need neither memory to store the plaintext nor multiple sending of the ciphertext. Finally, section 6 of this paper presents a practical instantiation of this generic composition.

2   Blockwise Attacks against Naive Implementations

In smart cards or with online protocols, authenticated encryption schemes use on-the-fly interactions. In such a setting, the decryption process is not easy to implement, since both privacy and integrity should be provided. Most of the existing solutions are insecure since these interactive processes greatly facilitate chosen ciphertext attacks. Indeed, they do not prevent attackers from getting decryptions of invalid messages. The following paragraphs describe why classical constructions are insecure.

In practice, two naive methods are widely used. Both implement the Encrypt-Then-MAC composition or single pass authenticated encryptions. Let (C, τ) be an authenticated ciphertext. For the first decryption process, the ciphertext C is sent block by block to the smart card, which gradually returns for each block the corresponding plaintext block. In the same time, the crypto device updates the MAC computation. At the end, the tag τ is sent to the card which checks its validity. With this method, the user learns the plaintext even if the ciphertext was invalid, since integrity is checked after the plaintext has been returned. For example, with the OCB authenticated encryption scheme (see [21] for further details on this scheme), the authenticated ciphertext $C = C_1 || \ldots || C_m || \tau$ is sent block by block to the card: it decrypts each block C_i into M_i, sends it to the receiver and updates the Checksum for the MAC computation. When the card receives the tag τ it just checks its validity by comparing it with the value it has computed. The result of this comparison does not matter since the adversary has received the plaintext. As a consequence, the card provides a decryption oracle for the underlying encryption scheme. Since most of the existing encryption

modes are not secure against chosen ciphertext attacks, such a technique often leads to insecure schemes.

The second naive technique works as follows: an authenticated ciphertext (C, τ) is sent to the smart card, block by block, or buffer by buffer. The card just checks its validity: if τ is valid, in a second pass, the ciphertext C is sent again to the card. The transmission is made on-the-fly, *i.e.*, for each ciphertext block sent, the card outputs the corresponding plaintext block. The main drawback of this protocol is that no verification can be made to check whether the same ciphertext has been sent twice. This leads to a major security flaw: in a first pass, an adversary can send any valid ciphertext, in the sense of integrity, and in the second pass, another one, possibly invalid, which will be nonetheless decrypted.

Such security flaws call for a formal definition of security for authenticated encryption schemes secure against blockwise adaptive adversaries, and for schemes ensuring this security level.

3   Preliminaries

Recently in [17], some results appeared concerning the insecurity of some encryption schemes in a new adversarial model. This notion is meaningful when implementing digital signed streams as in [15], or when considering on-line encryption schemes which do not require the whole plaintext before beginning the encryption. In this section we formally describe these adversaries and the associated security model for both privacy and integrity against chosen plaintext and ciphertext attacks.

3.1   Privacy

Privacy in the Standard Model. In the standard model, privacy of an encryption scheme is viewed as ciphertext indistinguishability (IND), defined in [3]. This notion is modeled through a "left-or-right" (LOR) game: the adversary is allowed to feed the *left-or-right encryption oracle* $\mathcal{E}_K(\mathcal{LR}(\cdot, \cdot, b))$ with queries of the form (M_0^i, M_1^i) where M_0^i and M_1^i are two equal length messages. First, the oracle chooses a random bit b, and, for all queries (M_0^i, M_1^i), always encrypts the message M_b^i and returns a ciphertext C_b^i. The adversary's goal is to guess the bit b with non-negligible advantage in polynomial time. If an encryption scheme $\mathcal{SE}$ withstands an adversary performing a *chosen plaintext attack* (CPA), then we say that $\mathcal{SE}$ is IND-CPA secure. If the adversary has access, in addition, to a decryption oracle, taking as input a ciphertext C and outputting the corresponding plaintext M, the adversary is said to mount a *chosen ciphertext attack* (CCA). For a complete description of these notions the reader can refer to [3].

Privacy in the Blockwise Adversarial Model. In this setting, adversaries are adaptive during their queries to the different encryption oracles. The *blockwise left or right* encryption oracle $\mathcal{E}_K^{bl}(\mathcal{LR}(\cdot, \cdot, b))$ can be requested in an on-line manner. The adversaries send queries block by block, i.e. they submit

$(m_0^i[k], m_1^i[k])$. The oracle encrypts message block $m_b^i[k]$ according to the bit b chosen at the beginning of the game, and immediately outputs the ciphertext block $c_b^i[k]$, without waiting for the next plaintext block. The adversary can now adapt each plaintext block according to the previous ciphertext blocks. In the case of chosen ciphertext attacks, the adversary has also access to a blockwise decryption oracle, taking as input a ciphertext block $c[k]$ and returning the corresponding plaintext block $m[k]$. This better models implementations for which interactions are made on-the-fly between a user and his crypto device. Such adversaries are called *blockwise adaptive*. Note that when dealing with smart cards, more than one block can be stored and interactions could contain larger buffers. However adversaries can still adapt their queries between each buffer, and thus one can assume that single blocks are sent (here, the word block is redefined to cover a complete buffer). This leads to the notions of Blockwise Chosen Plaintext Attacks (BCPA) or Blockwise Chosen Ciphertext Attacks (BCCA).

A recent work of Fouque, Martinet and Poupard [14] has formalized this notion in a stronger way: in their work they have considered *concurrent blockwise adaptive adversaries*. In this setting, the adversaries may run a polynomial number (in the security parameter) of encryption and/or decryption sessions in parallel. Such a notion is clearly stronger than the one we need here. A scheme proved secure in their model is thus clearly also secure in the weaker model considered here. The formal definition and description of the experiments we consider are given in appendix A.1.

3.2   Integrity of Ciphertexts

The notion of *integrity of ciphertexts* (INT-CTXT) has been first introduced in [19] and independently in [7]. It formalizes the idea that it should be computationally infeasible for a polynomial-time adversary performing a chosen message attack (CMA), to produce an authenticated ciphertext, for a message, not previously queried to the oracle (weak unforgeability) or even already queried (strong unforgeability). The chosen message attack is modeled by giving access to an oracle, that takes as input a message M and returns an authenticated ciphertext (C, τ), and to a verification oracle, taking as input an authenticated ciphertext (C, τ) and returning a bit b depending on the validity of the tag. The strongest security notion used for integrity is the strong unforgeability. Of course this distinction with the weak unforgeability does not have sense in the case of deterministic MAC schemes. However, for probabilistic ones, this security notion is the stronger one we can expect for a MAC scheme.

In the case of integrity, the blockwise adversarial model modifies the encryption oracle: it is requested with on-line queries and it outputs the authenticated ciphertext blocks on-the-fly. During the game, the verification oracle does not output anything and just waits until the end of the query. If it is valid, it returns $b = 1$ and otherwise, it returns $b = 0$. The queries are made on-the-fly but this does not make any change in the formalism of the experiment, as no intermediate results are provided to the adversary. Thus, integrity in the blockwise adversarial model is only slightly different from the standard model, since the encryption

oracle is blockwise oriented. A description of the adversary experiment is given in appendix A.2. In the following we say that a scheme is B-INT-CTXT secure if any "reasonable" adversary cannot win this game with significant probability.

4   Authenticated Encryption in the Blockwise Setting

To construct an authenticated encryption scheme, the classical method is to use an encryption scheme along with a Message Authentication Code (MAC). In [7], Bellare and Namprempre have shown that if an authenticated encryption scheme is both IND-CPA secure and INT-CTXT secure, then it is also IND-CCA secure. They have also studied how to combine encryption and MAC schemes so that the construction is generically secure: the right way to securely combine them is to first encrypt the plaintext and then MACing the ciphertext: if the underlying encryption scheme is only IND-CPA secure and if the MAC scheme is strongly unforgeable, then the Encrypt-Then-MAC composition is IND-CCA secure.

If one considers now blockwise adaptive adversaries, one can see that the Encrypt-Then-MAC composition is no longer secure against chosen ciphertext attacks. Indeed, as we have seen above, such adversaries allow to break the naive implementations of the Encrypt-Then-MAC. Thus, not only the encryption part of the composition has to be studied against blockwise adversaries, but also the decryption part, which may lead to insecure schemes.

To secure these compositions, the first idea is to move the requirements on the underlying schemes to the blockwise setting. However, this is not sufficient since the MAC verification and the decryption process are no longer linked. Thus even if the scheme is IND-BCPA and B-INT-CTXT secure, the composition could be insecure against chosen ciphertext attacks. For example, we consider the composition of the CBC encryption scheme with delay, denoted by DCBC (fully described and proved secure in [14]), along with the EMAC authentication algorithm, strongly unforgeable ([20, 5]). This scheme, DCBC-Then-EMAC, provides IND-BCPA and B-INT-CTXT security. However, the already known chosen ciphertext attack on the CBC can be performed in the blockwise setting (see [16] for a detailed description of it) even if the DCBC is combined with the secure EMAC authentication scheme. This proves that the composed scheme is not IND-BCCA secure. This remark allows us to give the following proposition:

Proposition 1. *There exists a symmetric encryption encryption scheme providing both IND-BCPA and B-INT-CTXT security but which is not IND-BCCA secure. That is, we have:*

$$\textbf{IND-BCPA + B-INT-CTXT} \not\Rightarrow \textbf{IND-BCCA}$$

This shows that when dealing with blockwise adaptive adversaries some other and stronger requirements have to be made on the underlying schemes. The classical Encrypt-Then-MAC composition and all the known authenticated encryption schemes cannot be used as it to ensure both confidentiality and integrity. The protocol itself has to be modified to take into account blockwise adaptive adversaries.

5   The Decrypt-Then-Mask Secure Generic Composition

In this section we describe a practical solution to the Verify-Then-Decrypt paradigm. The composition simply modifies the decryption protocol and thus is backward compatible with existing schemes. The encryption phase is supposed to be secure in the blockwise model against chosen plaintext attacks. We also assume that the cryptographic device is memory limited (cannot store the whole plaintext), but stores the long term secret key. Moreover the receiver of an authenticated ciphertext is assumed to have access to a Pseudorandom Number Generator (PRNG), also implemented in the crypto device in charge of the decryption. In this context, we propose a new decryption process, generically secure against blockwise adaptive adversaries, mounting chosen ciphertext attacks.

5.1   Description

The main idea behind this construction is that the crypto device should not give any information on the plaintext before having checked the ciphertext integrity. However, because of its restricted storage capability, it cannot store the plaintext M, verify the MAC and output M only when valid. Instead, the crypto device will use a Pseudorandom Number Generator (PRNG) to mask the plaintext blocks so that they can be returned on-the-fly.

   A PRNG consists in two parts: a seed generation algorithm (initialization) taking as input a security parameter and outputting an initial state s_0 ; and a generation algorithm, taking as input the current state s_{i-1}, and outputting the random string r_i along with the next state s_i. Note that in the model presented in [10], the initialization algorithm also produces a key which selects one particular PRNG out of a whole family. However, we suppose here the PRNG is without key and that the only secret resides in the initial state. In the following we denote by $\mathcal{CC}$ the cryptographic device and by $\mathcal{R}$ the user of this device. We assume the existence of an authenticated encryption scheme $\mathcal{SE}$, taking as input a key K along with a plaintext M, and returning an authenticated ciphertext. Without loss of generality, it is denoted by (C, τ), where C is the ciphertext and τ the MAC on it, although privacy and authenticity could be provided in a single pass (with IACBC, OCB, ...).

   The user $\mathcal{R}$ will use $\mathcal{CC}$ to decrypt an authenticated ciphertext (C, τ). The main idea is to blind the plaintext blocks obtained after decryption of C by XORing them with a pseudorandom sequence of bits. Then, if the tag τ is valid, the seed used to generate the pseudorandom sequence is returned. Thus the cryptographic device does not need neither memory to store the plaintext nor multiple sending of the ciphertext. Formally, the decryption protocol between $\mathcal{CC}$ and $\mathcal{R}$, described in figure 1, works as follow:

Stage 0 $\mathcal{R}$ is given the ciphertext $C = c_1 \| \ldots \| c_n$ along with a tag τ. He wants to first check its integrity and, if valid, to decrypt it.
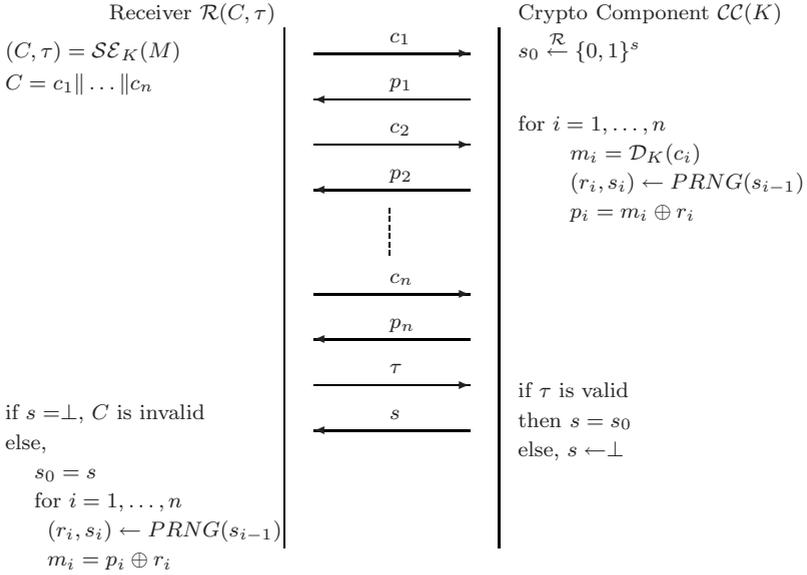
Fig. 1. The Decrypt-Then-Mask protocol

Stage 1 $\mathcal{CC}$ runs the seed generation algorithm to get a random seed $s_0 \in \{0,1\}^s$ for the PRNG (where s is a security parameter) and generates $(r_1, s_1) =$ PRNG(s_0). $\mathcal{R}$ sends to his crypto device $\mathcal{CC}$ the first ciphertext block c_1. $\mathcal{CC}$ initializes the tag computation, decrypts c_1 to obtain $m_1 = \mathcal{D}_K(c_1)$, computes $p_1 = m_1 \oplus r_1$ and returns p_1 to $\mathcal{R}$.

Stage 2 $\mathcal{R}$ sends the ciphertext on-the-fly. For each ciphertext block c_i he receives, $\mathcal{CC}$ updates the tag computation, decrypts c_i using the secret key, and masks it as $p_i = m_i \oplus r_i$ where $(r_i, s_i) =$ PRNG(s_{i-1}). $\mathcal{CC}$ finally outputs p_i. This process continues until the last ciphertext block c_n is sent and $\mathcal{CC}$ returns p_n.

Stage 3 $\mathcal{R}$ finally sends the tag τ to $\mathcal{CC}$ which checks its validity. If valid, $\mathcal{CC}$ returns to $\mathcal{R}$ the seed s_0 used as initial state for the PRNG. Otherwise it outputs the predefined symbol $\perp$.

Stage 4 If (C, τ) was valid, $\mathcal{R}$ can decrypt the ciphertext $P = p_1 \| \ldots \| p_n$ using s_0: for $i = 1, \ldots, n$, $(r_i, s_i) =$ PRNG(s_{i-1}) and $m_i = p_i \oplus r_i$. Then $M = m_1 \| \ldots \| m_n$ is the plaintext corresponding to C. Otherwise, if $\mathcal{R}$ receives $\perp$, he cannot recover the plaintext.

Even if the last stage requires a cryptographic operation (One Time Pad with a pseudorandom sequence of bits), it can be safely performed outside the crypto device, since no permanent secret is involved. Note that if the tag was valid, the user $\mathcal{R}$ recovers the message M by generating all the r_i values from the seed s_0. Otherwise, no information on the plaintext is given to the adversary, according to the security of the PRNG. Indeed, as the receiver has no information on the seed and since the one time pad blinds the message, any plaintext could have

been encrypted in P if the PRNG outputs are indistinguishable from random strings. Thus the security assumption on the PRNG is to be indistinguishable from truly random against Known Key Attacks[1], as defined in [10]: the adversary knows the key used to generate the r_i values but the seed s_0 and the states s_i are all hidden.

Note that the main point here is to use the initialization algorithm for each decryption and then generate a new initial state each time a decryption is requested. Indeed, the final state of the generator should not be used as initial state in the next use of the PRNG, as often done in practice.

In the sequel, we prove the IND-BCCA security of this Decrypt-Then-Mask decryption process when using an authenticated encryption scheme IND-BCPA and B-INT-CTXT secure, along with a IND-KKA secure PRNG.

Remark 1. Hereafter, we assume that the decryption process is regularly clocked, *i.e.*, if k blocks of ciphertext have been sent, $k - l$ plaintext blocks have been returned, where l is a public parameter of the scheme. For example, the CBC encryption scheme is such that $l = 1$ since the first block corresponds to the IV. Note that this property is needed during the proof. Otherwise, insecure schemes can be built, with delayed decryption outputs depending on the plaintext block. For sake of simplicity, we assume in the sequel that $l = 0$, meaning that plaintexts and ciphertexts, excluding the tag, are of the same length. However, the proof holds for any value of l.

5.2   Security Proof

In this part we prove that when using the Decrypt-Then-Mask protocol proposed above, with a IND-BCPA and B-INT-CTXT secure scheme, then the scheme is also IND-BCCA secure.

Theorem 1 (With the Decrypt-Then-Mask protocol, IND-BCPA and B-INT-CTXT ⇒ IND-BCCA). *Let $\mathcal{SE}$ an authenticated encryption scheme using the decryption protocol described in the figure 1. If $\mathcal{SE}$ is B-INT-CTXT secure and IND-BCPA secure, and if the PRNG used in the decryption protocol is IND-KKA secure, then $\mathcal{SE}$ is IND-BCCA secure. Moreover, we have:*

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{IND-BCCA}}(k, t, q_e, q_d, \mu_e, \mu_d) \leq \mathbf{Adv}_{\mathcal{SE}}^{\mathrm{B-INT-CTXT}}(k, t, q_e, q_d, \mu_e, \mu_d)$$
$$+ \mathbf{Adv}_{PRNG}^{\mathrm{IND-KKA}}(k, t, q_d, \mu_d)$$
$$+ \mathbf{Adv}_{\mathcal{SE}}^{\mathrm{IND-BCPA}}(k, t, q_e, \mu_e)$$

Proof. We consider the adversary's IND-BCCA game defined in section 3.1. We recall that the adversary has access to a "left-or-right" blockwise oracle $\mathcal{LR}_{\mathcal{B}}(\cdot, \cdot, b)$ taking as input two plaintexts and encrypting one of them depending on a random bit b. The adversary has also access to a decryption oracle $\mathcal{D}^*(\cdot, \cdot)$ taking as input a ciphertext C and a candidate tag τ for it, block by block,

---

[1] Here the PRNG we use is supposed to be without any key. Thus one can assume the key is known and then use this security model.

and responding the queries as described in the protocol: first, it generates a seed s_0 for its PRNG. Then interactions with the adversary $\mathcal{A}$ begin. $\mathcal{A}$ sends to the decryption oracle the ciphertext C, on-the-fly. Each ciphertext block c_i is first decrypted into m_i using the decryption algorithm $\mathcal{D}_k$. Then the PRNG generates a random block r_i and the oracle outputs $p_i = m_i \oplus r_i$. At the end of the interactions, the decryption oracle receives the tag τ for the ciphertext. If it is valid, it outputs the seed s_0 for the PRNG. Otherwise, it sends a predefined symbol $\perp$, meaning that the ciphertext is invalid.

The adversary's goal is to guess with non negligible advantage the bit b used by the encryption oracle. The adversary $\mathcal{A}$ we consider runs in time t, submits at most q_e test messages of at most μ_e blocks, and q_d decryption queries, of at most μ_d blocks.

We start from the initial game G_0 and we will transform this game to obtain games G_1, G_2, and G_3. For $i = 0, 1, 2, 3$, we denote by S_i the event that $\mathcal{A}$ guesses the bit b in game G_i.

Game G_0. This is the original game, where encryption and decryption oracles are as described above. S_0 is defined to be the event that $\mathcal{A}$ correctly guesses the value of the hidden bit b, in the game G_0. Following the definition of security given in annex A.1, the relation holds:

$$\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCCA}}(k) = |\Pr(S_0) - 1/2|$$

Game G_1. This game is the same as G_0 except that the decryption algorithm never outputs the seed of the PRNG. Let E_1 be the event that some valid ciphertexts are rejected in game G_1. Note that in the security model, $\mathcal{A}$ is not allowed to feed the decryption oracle with outputs of the $\mathcal{LR}_\mathcal{B}$ oracle. Thus the decryption queries are necessarily forgeries: at least one ciphertext block or the tag have been modified in a query copied from an answer of the encryption oracle. Since $\mathcal{SE}$ is B-INT-CTXT secure, $\mathcal{A}$ cannot forge a valid ciphertext, except with probability at most $\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{B-INT-CTXT}}(k)$, depending on the scheme. Then it follows that:

$$\Pr(E_1) \leq \mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{B-INT-CTXT}}(k)$$

Using lemma 1 of [22] and after some probability manipulations, we get:

$$|\Pr(S_1) - \Pr(S_0)| \leq \mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{B-INT-CTXT}}(k)$$

Game G_2. This game is the same as G_1, except that the outputs r_i of the PRNG are replaced by purely random blocks R_i. As before, for all decryption blocks queries c_i, the value $p_i = m_i \oplus R_i$ is returned.

The crucial point here is that the adversary knows the random values returned by the decryption box. Indeed, assume that $\mathcal{A}$ wants to have the encryption of message M. Then he queries the left-or-right oracle for the pair of equal messages (M, M) so that the ciphertext (C, τ) he receives necessarily encrypts M. Since the adversary cannot request the decryption oracle directly with

(C, τ), either one block of C or the tag τ has to be modified to be sent. Thus, one can suppose that $\mathcal{A}$ feeds $\mathcal{D}^*(\cdot, \cdot)$ with the legitimate query (C, τ'), where $\tau' \neq \tau$. As a consequence, according to the encryption scheme, the adversary gets, in the best case, the values $p_i = m_i \oplus r_i$. Since he already knows m_i, he can deduce r_i. Therefore, we assume that $\mathcal{A}$ always knows the random values generated by the decryption box.

In this game, $\mathcal{A}$ can detect the modification in two different ways:

– if $\mathcal{A}$ obtains the seed s_0, then he can detect the oracle's misbehavior. Indeed, he runs the PRNG to obtain the values r_i that would have been generated, and detects that the oracle is cheating, since $r_i \neq R_i$ except with negligible probability. However, since we specified in game G_1 that the oracle never outputs s_0, this event cannot occur.
– if $\mathcal{A}$ can distinguish the PRNG outputs from random bits sequences, he can also detect the oracle's misbehavior. However, the PRNG is supposed to be indistinguishable from random, and thus this bad event occurs with probability at most $\mathbf{Adv}_{PRNG,\mathcal{A}}^{\mathrm{IND-KKA}}(k)$.

Finally, since the adversary never obtains the seed for the PRNG, the adversary's advantage between the two games G_1 and G_2 depends only on the security of the PRNG. Thus, we have:

$$|\Pr(S_2) - \Pr(S_1)| \leq \mathbf{Adv}_{PRNG,\mathcal{A}}^{\mathrm{IND-KKA}}(k)$$

where $\mathbf{Adv}_{PRNG}^{\mathrm{IND-KKA}}(k, t, q_d, \mu_d) = \max_{\mathcal{A}}\{\mathbf{Adv}_{PRNG,\mathcal{A}}^{IND-KKA}(k)\}$ is defined to be the maximum adversary's advantage in distinguishing the output of the PRNG from a truly random string, with at most q_d queries of length at most μ_d, and in time t, when the secret key is known and all the states are hidden. A precise definition is given in the full version paper [13].

Game G_3. We start from G_2 and we modify the decryption oracle's behavior as follows: instead of decrypting the ciphertext block c_i into m_i, and then masking it with a random block R_i, the decryption oracle generates a random block P_i and outputs it directly. Here the adversary gains no information from the decryption oracle since he receives random values independent of the previous computations and oracle calls. Furthermore since the one time pad is unconditionally secure in the information theoretic sense, and since the outputs of the PRNG are indistinguishable from random strings, no information leaks about the plaintext in the previous game. Thus the adversary gains no advantage between these two games and we have:

$$\Pr(S_3) = \Pr(S_2)$$

Moreover the decryption oracle clearly gives no information to the adversary. So the game of the adversary is the same as for a chosen plaintext attack in the blockwise sense. Thus, his advantage of guessing the bit b in this game implies:

$$|\Pr(S_3) - 1/2| \leq \mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCPA}}(k)$$

Finally, adding all advantages involved in the different games gives the theorem.

6   Practical Instantiation

In this section, we propose a practical implementation of the Decrypt-Then-Mask composition with the Delayed CBC, proposed in [17] and proved secure in [14], along with the FIPS 186 Pseudorandom Number Generator (see [12, 10]).

The Delayed-CBC encryption scheme, denoted by DCBC, has been proposed as a countermeasure against the blockwise adaptive attack on the classical CBC, in [17]. In the DCBC encryption mode, the encryption process is on-line : when it receives the kth input block, it computes the kth output block and returns the $(k-1)$th one. Finally, when it receives the stop command, the encryption process returns the last block. The security proof for the DCBC has been recently made in [14]. In that paper, the authors prove the DCBC secure against blockwise chosen plaintext attacks, in the sense of the concurrent left-or-right security. As mentioned in section 3.1, this notion is stronger than the one we need here.

The generic composition Decrypt-Then-Mask uses a symmetric authenticated encryption scheme secure in the sense of indistinguishability and integrity. As shown above, the Delayed CBC can be used to ensure BCPA security. Combined in the Encrypt-Then-MAC composition with the encrypted MAC, called EMAC [20], it also provides integrity. Thus, the composition DCBC-Then-EMAC is a secure instantiation of the scheme in the Decrypt-Then-Mask setting.

However, to implement the Decrypt-Then-Mask protocol, a Pseudorandom Number Generator should be used. Such generators have been proposed in the literature with different security proofs, depending on the model we consider. The requirements we have here on the generator are very strong. Indeed, since the receiver of the ciphertext should also have an implementation of it, we should assume that the key is known to the adversary and that the only secret is the initial state chosen by the crypto device. Such security notion has been defined in [10]. In that paper the authors studied the security of two popular Pseudorandom Number Generators, the ANSI X9.17 and the FIPS 186. In our setting, the ANSI X9.19 is not appropriate since it is totally insecure when the key is known to the adversary. However, the FIPS 186 generator is proved secure against Known Key Attacks, when the states of the generator are hidden, and when the inputs are not under control of the adversary. This is exactly our setting and thus we propose to use it in our scheme. The full version of this paper [13] recalls this security framework and the theorem of [10] for the security of the FIPS 186 generator. One of the constructions provided by the FIPS 186 specifications [12] is based on the core SHA-1 function with the underlying assumption that it can be considered as a random function. This assumption is not new and has already been used in some papers as in [4] and [2]. This seems realistic in practice since no attack that could suggest such a weakness has been found for this function.

References

[1] J. H. An and M. Bellare. Does encryption with redundancy provide authenticity. In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt'01*, volume 2045 of *LNCS*, pages 512 – 528. Springer-Verlag, 2001.   145

[2] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom Functions Revisited: The Cascade Construction and its Concrete Security. In *Proceedings of the 37th Symposium on Foundations of Computer Science*. IEEE, 1996. 156

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of operation. In *Proceedings of the 38th Symposium of Foundations of Computer Science*, pages 394 – 403. IEEE Computer Security Press, 1997. 148

[4] M. Bellare, R. Guérin, and P. Rogaway. XOR-MACs: New Methods for Message Authentication using Finite Pseudorandom Functions. In D. Coppersmith, editor, *Advances in Cryptology – Crypto'95*, volume 963 of *LNCS*, pages 15 – 28. Springer-Verlag, 1995. 156

[5] M. Bellare, J. Kilian, and P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. In Y. Desmedt, editor, *Advances in Cryptology – Crypto'94*, volume 839 of *LNCS*, pages 341 – 358. Springer-Verlag, 1994. 150

[6] M. Bellare, T. Kohno, and C. Namprempre. Authenticated Encryption in SSH: Provably Fixing the SSH Binary Packet Protocol. In *Ninth ACM Conference on Computer and Communications Security*, pages 1 – 11. ACM Press, 2002. 146

[7] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology – Asiacrypt'00*, volume 1976 of *LNCS*, pages 531 – 545. Springer-Verlag, 2000. 145, 149, 150

[8] M. Bellare and P. Rogaway. Encode then encipher encryption: How to exploit nounces or redundancy in plaintexts for efficient cryptography. In T. Okamoto, editor, *Advances in Cryptology – Asiacrypt'00*, volume 1976 of *LNCS*, pages 317 – 330. Springer-Verlag, 2000. 145

[9] M. Blaze, J. Feigenbaum, and M. Naor. A Formal Treatment of Remotely Keyed Encryption. In K. Nyberg, editor, *Advances in Cryptology – Eurocrypt'98*, volume 1403 of *LNCS*, pages 251 – 265. Springer-Verlag, 1998. 146

[10] A. Desai, A. Hevia, and Y. L. Yin. A Practice-Oriented Treatment of Pseudo-random Number Generators. In L. Knudsen, editor, *Advances in Cryptology – Eurocrypt 2002*, volume 2332 of *LNCS*, pages 368 – 383. Springer-Verlag, 2002. 151, 153, 156

[11] Y. Dodis and J. H. An. Cancelment and its Applications to Authenticated En-cryption. In E. Biham, editor, *Advances in Cryptology – Eurocrypt'03*, LNCS. Springer-Verlag, 2003. 146

[12] FIPS PUB 186-2. Digital Signature Standard. Technical report, National Institute of Standards and Technologies, 2001. 156

[13] P.-A. Fouque, A. Joux, G. Martinet, and F. Valette. Authenticated On-line Encryption, 2003. Full version of this paper. Available at `http://www.di.ens.fr/~fouque`. 155, 156

[14] P.-A. Fouque, G. Martinet, and G. Poupard. Practical Symmetric On-line En-cryption. In T. Johansson, editor, *Proceedings of the Fast Software Encryption Workshop 2003*, LNCS. Springer-Verlag, 2003. 149, 150, 156

[15] R. Gennaro and P. Rohatgi. How to Sign Digital Streams. In B. Kaliski, editor, *Advances in Cryptology – Crypto'97*, volume 1294 of *LNCS*, pages 180 – 197. Springer-Verlag, 1997. 148

[16] S. Goldwasser and M. Bellare. Lecture Notes on Cryptography, 2001. Available at `http://www-cse.ucsd.edu/users/mihir`. 150

[17] A. Joux, G. Martinet, and F. Valette. Blockwise-Adaptive Attackers. Revisiting the (in)security of some provably secure Encryption Modes: CBC, GEM, IACBC.

In M. Yung, editor, *Advances in Cryptology – Crypto'02*, volume 2442 of *LNCS*, pages 17 – 30. Springer-Verlag, 2002. 146, 148, 156

[18] C. Jutla. Encryption Modes with Almost Free Message Integrity. In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt'01*, volume 2045 of *LNCS*, pages 529 – 544. Springer-Verlag, 2001. 145, 146

[19] J. Katz and M. Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In B. Schneier, editor, *Proceedings of the Fast Software Encryption Workshop 2000*, volume 1978 of *LNCS*, pages 284 – 299. Springer-Verlag, 2000. 149

[20] E. Petrank and C. Rackoff. CBC-MAC for Real-Time Data Sources. *Journal of Cryptology*, 13(3):315 – 338, 2000. 150, 156

[21] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In *Proceedings of the 8th Conference on Computer and Communications Security*, pages 196 – 205. ACM Press, 2001. 145, 146, 147

[22] V. Shoup. OAEP reconsidered (Extended Abstract). In J. Kilian, editor, *Advances in Cryptology – Crypto'01*, volume 2139 of *LNCS*, pages 239 – 259. Springer-Verlag, 2001. 154

[23] S. Vaudenay. CBC Padding: Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS,... In L. Knudsen, editor, *Advances in Cryptology – Eurocrypt 2002*, volume 2332 of *LNCS*, pages 534 – 545. Springer-Verlag, 2002. 146

A   Formal Definitions

A.1   Privacy

Here are described the experiments we consider:

$\mathsf{Expt}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCPA}}(k)$
   Pick a random bit $b \in \{0,1\}$
   $K \xleftarrow{\mathcal{R}} \mathcal{K}(k)$
   $d \leftarrow \mathcal{A}^{\mathcal{E}_K^{bl}(\mathcal{LR}(\cdot,\cdot,b))}$
   if $d = b$, return 1, else return 0

$\mathsf{Expt}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCCA}}(k)$
   Pick a random bit $b \in \{0,1\}$
   $K \xleftarrow{\mathcal{R}} \mathcal{K}(k)$
   $d \leftarrow \mathcal{A}^{\mathcal{E}_K^{bl}(\mathcal{LR}(\cdot,\cdot,b)),\mathcal{D}_K^{bl}(\cdot)}$
   if $d = b$, return 1, else return 0

where $\mathcal{E}_K^{bl}(\mathcal{LR}(\cdot,\cdot,b))$ is an encryption oracle taking as input two blocks of messages and returning the encryption of one of them depending on the bit b, and where $\mathcal{D}_K^{bl}(\cdot)$ is a decryption oracle taking as input a ciphertext block and returning the corresponding plaintext. We denote by IND-BCPA (respectively by IND-BCCA) the security in the blockwise model against chosen plaintext attacks (respectively chosen ciphertext attacks). The adversary's advantage in winning the IND-BCPA and the IND-BCCA games are defined as:

$$\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCPA}}(k) = \left| \Pr[\mathsf{Expt}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCPA}}(k) = 1] - 1/2 \right|$$

$$\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCCA}}(k) = \left| \Pr[\mathsf{Expt}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCCA}}(k) = 1] - 1/2 \right|$$

Therefore, we define the security bound of the scheme in the IND-BCPA and in the IND-BCCA senses by:

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{IND-BCPA}}(k, t, q_e, \mu_e) = \max_{\mathcal{A}}\{\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCPA}}(k)\}$$

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{IND-BCCA}}(k, t, q_e, q_d, \mu_e, \mu_d) = \max_{\mathcal{A}}\{\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{IND-BCCA}}(k)\}$$

where the maximum is over all legitimate $\mathcal{A}$ having time-complexity t, asking to the oracle at most q_e encryption queries totaling μ_e blocks, and possibly q_d decryption queries totaling μ_d blocks. The time complexity is defined to be the worst case total execution time of the experiment plus the size of the code of the adversary. We consider that the time complexity is polynomially bounded in the security parameter k. A secret-key encryption scheme is said to be IND-BCPA (respectively IND-BCCA) secure, if for all polynomial time, probabilistic adversaries, the advantage in the respective games is negligible as a function of the security parameter k.

A.2   Integrity

The experiment we consider is as follows:

$\mathsf{Expt}_{\mathcal{SE},\mathcal{A}}^{\mathrm{B-INT-CTXT}}(k)$

   $K \xleftarrow{\mathcal{R}} \mathcal{K}(k)$

   If $\mathcal{A}^{\mathcal{E}_K^{bl}(\cdot), \mathcal{D}_K^*(\cdot,\cdot)}(k)$ makes a query (C, τ) to the oracle $\mathcal{D}_K^*(\cdot, \cdot)$ such that
      - $\mathcal{D}_K^*(C, \tau)$ returns 1, and
      - (C, τ) was never an output of $\mathcal{E}_K(\cdot)$
   then return 1, else return 0.

where $\mathcal{E}_K^{bl}(\cdot)$ is a blockwise authenticated encryption oracle taking as input a plaintext M on-the-fly and returning a pair (C, τ) of ciphertext and tag, and where $\mathcal{D}_K^*(\cdot, \cdot)$ is a decryption oracle taking as input a ciphertext C along with a candidate tag τ for it, and returning a bit b such that $b = 1$ if valid and $b = 0$ otherwise. Note that the adversary $\mathcal{A}$ is not allowed to feed the decryption oracle $\mathcal{D}_K^*(\cdot, \cdot)$ with outputs of the encryption oracle. Otherwise, he could trivially win the game. So, if (C, τ) is an output of the encryption oracle, then the adversary should modify at least one block of C or the tag τ before calling the decryption oracle.

The adversary's advantage in winning the B-INT-CTXT game is defined as:

$$\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{B-INT-CTXT}}(k) = \Pr[\mathsf{Expt}_{\mathcal{SE},\mathcal{A}}^{\mathrm{B-INT-CTXT}}(k) = 1]$$

Therefore, we define the security bound of the scheme in the B-INT-CTXT sense by:

$$\mathbf{Adv}_{\mathcal{SE}}^{\mathrm{B-INT-CTXT}}(k, t, q_e, q_d, \mu_e, \mu_d) = \max_{\mathcal{A}}\{\mathbf{Adv}_{\mathcal{SE},\mathcal{A}}^{\mathrm{B-INT-CTXT}}(k)\}$$

where the maximum is over all legitimate $\mathcal{A}$ having time-complexity t, asking to the oracle at most q_e encryption queries totaling at most μ_e blocks, and at most q_d decryption queries totaling at most μ_d blocks.