

# The Order of Encryption and Authentication for Protecting Communications (Or: How Secure is SSL?)\*

Hugo Krawczyk\*\*

**Abstract.** We study the question of how to generically compose *symmetric* encryption and authentication when building “secure channels” for the protection of communications over insecure networks. We show that any secure channels protocol designed to work with any combination of secure encryption (against chosen plaintext attacks) and secure MAC must use the encrypt-then-authenticate method. We demonstrate this by showing that the other common methods of composing encryption and authentication, including the authenticate-then-encrypt method used in SSL, are not generically secure. We show an example of an encryption function that provides (Shannon’s) perfect secrecy but when combined with any MAC function under the authenticate-then-encrypt method yields a totally insecure protocol (for example, finding passwords or credit card numbers transmitted under the protection of such protocol becomes an easy task for an active attacker). The same applies to the encrypt-and-authenticate method used in SSH.

On the positive side we show that the authenticate-then-encrypt method is secure if the encryption method in use is either CBC mode (with an underlying secure block cipher) or a stream cipher (that xor the data with a random or pseudorandom pad). Thus, while we show the generic security of SSL to be broken, the current practical implementations of the protocol that use the above modes of encryption are safe.

## 1 Introduction

The most widespread application of cryptography in the Internet these days is for implementing a *secure channel* between two end points and then exchanging information over that channel. Typical implementations first call a key-exchange protocol for establishing a shared key between the parties, and then use this key to authenticate and encrypt the transmitted information using (efficient) symmetric-key algorithms. The three most popular protocols that follow this approach are SSL [11] (or TLS [9]), IPsec [18, 19] and SSH [27]. In particular, SSL is used to protect a myriad of passwords, credit card numbers, and other sensitive data transmitted between Web clients and servers, and is used to secure many other applications. IPsec is the standard for establishing a secure channel between any two IP entities for protecting information at the network layer.

\* A full version of this paper can be found in [21].

\*\* EE Department, Technion, Haifa, Israel. Email: hugo@ee.technion.ac.il

As said, all these protocols apply both symmetric authentication (MAC) and encryption to the transmitted data. Interestingly, each of these three popular protocols have chosen a *different* way to combine authentication and encryption. We describe these three methods (here  $x$  is a message;  $\mathcal{Enc}(\cdot)$  is a symmetric encryption function;  $\mathcal{Auth}(\cdot)$  is a message authentication code; and ‘,’ denotes concatenation — in this notation the secret keys to the algorithms are implicit):

SSL:  $a = \mathcal{Auth}(x)$ ,  $C = \mathcal{Enc}(x, a)$ , transmit  $C$

IPSec:  $C = \mathcal{Enc}(x)$ ,  $a = \mathcal{Auth}(C)$ , transmit  $(C, a)$

SSH:  $C = \mathcal{Enc}(x)$ ,  $a = \mathcal{Auth}(x)$ , transmit  $(C, a)$ .

We refer to these three methods as *authenticate-then-encrypt* (abbreviated *AtE*), *encrypt-then-authenticate* (*EtA*), and *encrypt-and-authenticate* (*E&A*), respectively.

This disparity of choices reflects lack of consensus in the cryptography and security communities as for the right way to apply these functions. But is there a “right way”, or are all equally secure? Clearly, the answer to this question depends on the assumptions one makes on the encryption and authentication functions. However, since protocols like the above are usually built using cryptographic functions as replaceable modules, the most useful form of this question is obtained by considering both functionalities, encryption and authentication, as *generic cryptographic primitives* with well defined (and independent from each other) properties. Moreover, we want these properties to be commonly achieved by the known efficient methods of symmetric encryption and authentication, and expected to exist in future practical realizations of these functions as well.

Specifically, we consider generic MAC functions secure against *chosen-message attacks* and generic symmetric encryption functions secure against *chosen-plaintext attacks*. These security properties are the most common notions used to model the security of these cryptographic primitives. In particular, chosen-message security of the authentication function allows to use the MAC in the above protocols independently of the encryption in cases where only integrity protection is required but not secrecy. As for encryption, chosen-plaintext security is the most common property under which encryption modes are designed and analyzed. We note that a stronger property of encryption is resistance to chosen-ciphertext attacks; while this property is important against active attacks it is NOT present in the prevalent modes of symmetric encryption (such as in stream ciphers or CBC mode even when the underlying block cipher is chosen-ciphertext secure) and therefore assuming this strong property as the basic secrecy requirement of the encryption function would exclude the use of such standard efficient mechanisms.

Rather than just studying the above ways of composing encryption and authentication as a stand-alone composed primitive, our focus is on the more comprehensive question of whether these methods provide for truly secure communications (i.e., secrecy and integrity) when embedded in a protocol that runs in a real adversarial network setting (where links are controlled by the attacker,

where some of the parties running the protocol may be corrupted, where multiple security sessions are run simultaneously and maliciously interleaved, etc.).

**Recent results.** In a recent work, Canetti and Krawczyk [8] describe a model of secure channels that encompasses both the initial exchange of a key between pairs of communicating parties and the use of the resultant shared key for the application of symmetric encryption and authentication on the transmitted data. The requirements made from secure channels in this model include protecting the data's integrity (in the sense of simulating ideally authenticated channels) and secrecy (in the sense of plaintext indistinguishability) in the presence of a network attacker with powerful and realistic abilities of the type mentioned above. A main result in [8] is that if the key is shared securely then applying to the data the encrypt-then-authenticate method achieves secure channels provided that the encryption function is semantically secure (or plaintext-indistinguishable) under a chosen-plaintext attack and the authentication function is a MAC that resists chosen message attacks. This provides one important answer to the questions raised above: *it proves that encrypt-then-authenticate is a generically secure method for implementing secure channels.*

**Our results.** In this paper we complement the above result on the encrypt-then-authenticate method with contrasting results on the other two methods.

**THE GENERIC INSECURITY OF *AtE*.** We show that the authenticate-then-encrypt method (as in SSL) *is not generically secure* under the sole assumption that the encryption function is secure against chosen plaintext attacks and the MAC secure against chosen message attacks. We show an example of a simple encryption function that enjoys perfect (in the sense of Shannon) secrecy against chosen plaintext attacks and when combined under the *AtE* method with any MAC (even a perfect one) results in a *totally breakable implementation of secure channels*. To illustrate the insecurity of the resultant scheme we show how passwords (and credit card numbers, etc) transmitted under such a method can be easily discovered by an active attacker that modifies some of the information on the links. A major issue to highlight here is that the attack is not against the authenticity of information but against its secrecy! This result is particularly unfortunate in the case of SSL where protection of this form of sensitive information is one of the most common uses of the protocol.

**THE GENERIC INSECURITY OF *E&A*.** The above example is used also to demonstrate the insecurity of the encrypt-and-authenticate method (as in SSH) where the same attack (and consequences) is possible. It is worth noting that the *E&A* is obviously insecure if one uses a MAC function that leaks information on the data. However, what our attack shows is that the method is not generically secure even if one assumes a stronger MAC function with secrecy properties as commonly used in practice (e.g. a MAC realized via a pseudorandom family or if the MAC's tag itself is encrypted).

**THE SECURITY OF *AtE* WITH SPECIFIC ENCRYPTION MODES.** This paper does not bring just bad news. We also show that the authenticate-then-encrypt method *is secure* under two very common forms of encryption: CBC mode (with an underlying secure block cipher) and stream ciphers (that xor the data with

a random or pseudorandom pad). We provide a (near optimal) quantified security analysis of these methods. While these positive results do not resolve the “generic weakness” of the authenticate-then-encrypt method (and of SSL), they do show that the common implementations currently in use do result in a secure channels protocol.

In conjunction, these results show a quite complete picture of the security (and lack of security) of these methods. They point to the important conclusion that any secure channels protocol designed to work with any combination of secure encryption (against chosen plaintext attacks) and secure MAC must use the encrypt-then-authenticate method. On the other hand, protocols that use the authenticate-then-encrypt method with encryption in either stream cipher or CBC modes are safe. However, we note the fragility of this last statement: very simple (seemingly innocuous) changes to the encryption function, including changes that do not influence the secrecy protection provided by the encryption when considered as a stand-alone primitive, can be fatal for the security of the implemented channels. This is illustrated by our example of a perfect cipher where the sole use of a simple encoding before encryption compromises the security of the transmitted data, or by the case of CBC encryption where the joint encryption of message and MAC results in a secure protocol but separate encryption of these elements is insecure. Thus, when using a non-generically secure method one has to be very careful with *any* changes to existing functions or with the introduction of new encryption mechanisms (even if these mechanisms are secure as stand-alone functions).

**Open question.** Our results demonstrate that chosen-plaintext security is not a sufficient condition for an encryption scheme to guarantee a secure authenticate-then-encrypt composition even if the MAC is secure. An interesting open question is to find a stronger property that is enjoyed by common modes of encryption but at the same time is sufficient to ensure the security of the authenticate-then-encrypt method when combined with a secure MAC. Note that we are looking for a property that is significantly weaker than chosen-ciphertext security since the latter is *not* achieved by most symmetric encryption modes, but also because our results show that this condition is not really necessary.

**Related work.** While the interaction between symmetric encryption and authentication is a fundamental issue in the design of cryptographic protocols, this question seems to have received surprisingly little explicit attention in the cryptographic literature until very recently. In contrast, in the last year we have seen a significant amount of work dealing with this and related questions.

We already mentioned the work by Canetti and Krawczyk [8] that establishes the security of the encrypt-then-authenticate method for building secure channels. Here, we use this result (and some extensions of it) as a basis to derive some of our positive results. In particular, we borrow from that paper the formalization of the notion of secure channels; a short outline of this model is presented in Section 2.3 but the reader is referred directly to [8] for the (many missing) details.

A recent, independent, work that deals directly with the ordering of generic encryption and authentication is Bellare and Namprempre [5]. They study the same three forms of composition as in this paper but focus on the properties of the composed function as a stand-alone composed primitive rather than in the context of its application to secure channels as we do. The main contribution of [5] is in providing careful quantitative relations and reductions between different methods and security notions related to these forms of composition. These results, however, are insufficient in general for claiming the security, or demonstrating the insecurity, of channels that use these methods for protecting data. For example, while [5] show that authenticate-then-encrypt is not necessarily CCA-secure, it turns out (by results in [8] and here) that the lack of this property is no reason to consider insecure the channels that use such a method (moreover, even the specific non-CCA example in [5] does provide secure channels). This demonstrates that the consideration of secure channels requires a finer treatment of the question of encryption/authentication composition (see discussion at the beginning of Section 4.2). In particular, none of our results is claimed or implied by [5].

A related subject that received much attention recently is the construction of encryption modes that provide integrity in addition to secrecy. Katz and Yung [16] suggest a mode of operation for block ciphers that provides such functional combination; for their analysis (and for its independent interest) they introduce the notion of “unforgeable encryption”. A very similar notion is also introduced in [5] and called there “integrity of ciphertexts” (INT-CTXT). We use this notion in our work too (see Section 3) as a tool in some of our proofs. In another recent work, An and Bellare [1] study the use of redundancy functions (with and without secret keys) as a method for adding authentication to encryption functions. They show several positive and negative results about the type of redundancy functions that are required in combination with different forms of encryption and security notions. Our results concerning the authenticate-then-encrypt method with stream ciphers and CBC modes contribute also to this research direction since these results provide sufficient and necessary conditions on the redundancy functions (viewed as MAC functions) required for providing integrity to these important modes of encryption. Of particular interest is our proof that a secure *AtE* composition that uses CBC encryption requires a strong underlying MAC; this contradicts a common intuition that (since the message and MAC are encrypted) weaker “redundancy functions” could replace the full-fledge MAC.

Recently, Jutla [15] devised an elegant CBC-like scheme that provides integrity at little cost beyond the traditional CBC method, as well as a parallel mode of encryption with integrity guarantee (a related scheme is presented in [26]). We note that while schemes such as [15] can be used to efficiently implement secure channels that provide secrecy and authenticity, generic schemes like encrypt-then-authenticate have several design and analysis advantages due to their modularity and the fact that the encryption and authentication components can be designed, analyzed and replaced independently of each other. In

particular, generic schemes can allow for faster implementations than the specific ones; even today the combination of fast stream ciphers with a fast MAC function such as UMAC [6] under the encrypt-then-authenticate method would result in a faster mechanism than the one proposed in [15] which requires the use of block ciphers. Also, having a separate MAC from encryption allows for much more efficient authentication in the cases where secrecy is not required.

## 2 Preliminaries

We informally outline some well-known notions of security for MAC and encryption functions as used throughout the paper, and introduce some notation. References are given below for formal treatment of these notions. We also sketch the model of “secure channels” from [8].

### 2.1 Secure message authentication.

Functions that provide a way to verify the integrity of information (for example, against unauthorized changes over a communications network) and which use a shared secret key are called *MAC* (*message authentication codes*). The notion of a MAC and its security definition is well understood [4]. Here we outline the main ingredients of this definition as used later in the paper.

A MAC scheme is described as a family of (deterministic) functions over a given domain and range. (We will usually assume the domain to be  $\{0, 1\}^*$  and the range  $\{0, 1\}^n$  for fixed size  $n$ .) The key shared by the parties that use the MAC scheme determines a specific function from this family. This specific function is used to compute an *authentication tag* on each transmitted message and the tag is appended to the message. A recipient of the information that knows the MAC key can re-compute the tag on the received message and compare to the received tag. Security of a MAC scheme is defined through the inability of an attacker to produce a *forgery*, namely, to generate a message, not transmitted between the legitimate parties, with its valid authentication tag. The formal definition of security provides the attacker with access to a *MAC oracle*  $\mathcal{O}_{\text{MAC}}$  that on input a message  $x$  outputs the authentication tag corresponding to that message. The oracle uses for its responses a key that is generated according to the probability distribution of keys defined by the MAC scheme. The attacker succeeds if after this interaction with the oracle it is able to find a forgery (for a message not previously queried). To quantify security we say that a *MAC scheme has security*  $\mathcal{E}_M(q, Q, T)$  if any attacker that works time  $T$  and asks  $q$  queries from  $\mathcal{O}_{\text{MAC}}$  involving a total of  $Q$  bits has probability at most  $\mathcal{E}_M(q, Q, T)$  to produce a forgery.

**Remark.** In the case of MAC functions (e.g., randomized ones) where there may be multi-valued valid tags for the same message, we extend the definition of security as follows. If the messages queried to  $\mathcal{O}_{\text{MAC}}$  are  $x_1, x_2, \dots, x_q$  and the responses from  $\mathcal{O}_{\text{MAC}}$  are  $t_1, t_2, \dots, t_q$  then a forgery  $(x, t)$  output by the attacker is considered valid if  $(x, t) \neq (x_i, t_i)$  for all  $i = 1, \dots, q$ . (Namely, we consider the attacker successful even in case its forgery includes a queried message as long

as the tag  $t$  was not generated by the oracle for that message.) This technical strengthening of the definition is used in some of our results. This notion appears (due to similar reasons) also in [5].

## 2.2 Secure symmetric encryption

We do not develop a formal definition of encryption security here as the subject is well established and treated extensively in the literature. Yet, we summarize informally the main aspects of the security notions of symmetric encryption that are relevant to our work and establish some notation. For formal and precise definitions see the references mentioned below.

An encryption scheme is a triple of (probabilistic) algorithms ( $KEYGEN$ ,  $ENC$ ,  $DEC$ ) where  $KEYGEN$  defines the process (and resultant probability distribution) by which keys are generated, while  $ENC$  and  $DEC$  are the encryption and decryption operations with the usual inverse properties. To simplify notation we use  $ENC$  to denote the encryption operation itself but also as representing the whole scheme (i.e., a triple as above). The main notion behind the common definitions of security of encryption is *semantic security* [13], or its (usually) equivalent formulation via *plaintext indistinguishability*. In this formulation an attacker against a scheme  $ENC$  is given a *target ciphertext*  $y$  and two candidate plaintexts  $x_1, x_2$  such that  $y = ENC(x_i)$ ,  $i \in_R \{0, 1\}$ .<sup>1</sup> The encryption scheme has the indistinguishability property if the attacker cannot guess the right value of  $i$  with probability significantly better than  $1/2$ . The security of the scheme is quantified via the time invested by the attacker and the probability beyond  $1/2$  to guess correctly.

The above describes the goal of the attacker but not the ways of attack it is allowed to use. Two common models of attack are CPA (chosen plaintext attack) and CCA (chosen ciphertext attack). In the first the attacker has access to an *encryption oracle*  $\mathcal{O}_{ENC}$  to which it can present plaintexts and receive the ciphertexts resulting from the encryption of these plaintexts. In the second model the attacker can, in addition to the above queries to the encryption oracle, also ask for decryptions of arbitrary ciphertexts (except for the target ciphertext  $y$ ) from a *decryption oracle*  $\mathcal{O}_{DEC}$ . We note that both  $\mathcal{O}_{ENC}$  and  $\mathcal{O}_{DEC}$  use the same key for their responses which is also the key under which the target ciphertext  $y$ , as described above, is produced. In both cases the queries to the oracles can be generated adaptively by the attacker, i.e. as a function of previous responses from the oracles and of the target ciphertext  $y$  (actually, also the candidate plaintexts  $x_1, x_2$  on which the target ciphertext  $y$  is computed can be chosen by the attacker). Under these formulations two new parameters enter the quantification of security: the number of queries to  $\mathcal{O}_{ENC}$  and the number of queries to  $\mathcal{O}_{DEC}$  (the latter is 0 in the case of CPA). A finer quantification would also consider the total number of bits in these queries.

As it is customary we denote the above two notions of encryption security as IND-CPA and IND-CCA. Extensive treatment of these notions can be found

<sup>1</sup> We use the notation  $a \in_R A$  to denote that the element  $a$  is chosen with uniform probability from the set  $A$ .

among other works in [13, 12, 2] and [24, 3, 17], respectively. A notion strongly related to IND-CCA is non-malleability of ciphertexts [10] which we do not use directly here; a weaker notion of CCA security was introduced earlier in [23]. We also note that we are only concerned with symmetric encryption; asymmetric encryption shares many of the same aspects but there are some important differences as well (in particular, in the asymmetric case encryption oracles are meaningless since everyone can encrypt at will any plaintext).

### 2.3 Secure Channels

In order to claim our positive results, i.e. that a certain combination of encryption and authentication provides secure communications, we need to define what is meant by such “secure communications”. For this we use the model of **secure channels** introduced by Canetti and Krawczyk [8] and which is intended to capture the standard network-security practice in which communications over public networks are protected through “sessions” between pairs of communicating parties, and where each session consists of two stages. First, the two parties run a key-exchange protocol that establishes an authenticated and secret session key shared between the parties. Then, in the second stage, this session key is used, together with symmetric-key cryptographic functions, to protect the integrity and/or secrecy of the transmitted data. The formalism of [8] involves the definition of a key-exchange protocol for implementation of the session and key establishment stage, as well as of two functions, `snd` and `rcv`, that define the actions applied to transmitted data for protection over otherwise insecure links. A protocol that follows this formalism is called in [8] a “network channels protocol”, and its security is defined in terms of authentication and secrecy.

These notions are defined in [8] in the context of communications controlled by an attacker with full control of the information sent over the links and with the capability of corrupting sessions and parties. We refer to the full version of [8] for a full description of the adversarial model and security definitions. Here we only mention briefly the main elements in this definition concerning the functions `snd` and `rcv`. The function `snd` represents the operations and transformations applied to a message by its sender in order to protect it from adversarial action over the communication links. Namely, when a message  $m$  is to be transmitted from party  $P$  to party  $Q$  under a session  $s$  established between these parties, the function `snd` is applied to  $m$  and, possibly, to additional information such as a message identifier. The definition of `snd` typically consists of the application of some combination of a MAC and symmetric encryption keyed via the session key. The function `rcv` describes the action at the receiving end for “decoding” and verifying incoming messages, and it typically involves the verification of a MAC and/or the decryption of an incoming ciphertext.

Roughly speaking, [8] define that authentication is achieved by the protocol if any message decoded and accepted as valid by the receiving party to a session was indeed sent by the partner to that session. (That is, any modification of messages produced by the attacker over the communications links, including the injection or replay of messages, should be detected and rejected by the recipient; in [8] this is formalized as the “emulation” of an ideally-authenticated channel.) Secrecy



is formalized in the tradition of semantic security: among the many messages exchanged in a session the attacker chooses a pair of “test messages” of which only one is sent; the attacker’s goal is to guess which one was sent. Security is obtained if the attacker cannot guess correctly with probability significantly greater than  $1/2$ . A network channels protocol is called a **secure channels protocol** if it achieves both authentication and secrecy in the sense outlined above.

In this paper we focus on the way the functions `snd` and `rcv` are to be defined to achieve secure channels, i.e. to provide both authentication and secrecy in the presence of an attacker as above. We say that any of the combinations *EtA*, *AtE*, *E&A* implements secure channels if when used as the specification of the `snd` and `rcv` functions the resultant protocol is a “secure channels protocol”. Note that we are not concerned here with a specific key-exchange mechanism, but rather assume a secure key-exchange protocol [8], and may even assume an “ideally shared” session key.

### 3 CUF-CPA: Ciphertext Unforgeability

In addition to the traditional notions of security for an encryption scheme outlined in Section 2.2 we use the following notion of security that we call **ciphertext unforgeability**. A similar notion has been recently (and independently) used in [16, 5] where it is called “existential unforgeability of encryption” and “integrity of ciphertexts (INT-CTXT)”, respectively.

Let *ENC* be a symmetric encryption scheme, and *k* be a key for *ENC*. Let  $P(k)$  be the set of plaintexts on which  $ENC_k$  is defined, and  $C(k)$  be the set of ciphertexts  $\{y : \exists x \in P(k) \text{ s.t. } y = ENC_k(x)\}$  (note that if *ENC* is not deterministic then by  $y = ENC_k(x)$  we mean that there is a run of *ENC* on *x* that outputs *y*). We call  $C(k)$  the set of **valid ciphertexts** under key *k*. For example, under a block cipher only strings of the block length are valid ciphertexts while in the basic CBC mode only strings that are multiples of the block length can be valid ciphertexts. We assume that the decryption oracle  $\mathcal{O}_{DEC}$  outputs a special “invalidity symbol”  $\perp$  when queried with an invalid ciphertext (and otherwise outputs the unique decrypted plaintext *x*).

We say that an encryption scheme is **ciphertext unforgeable**, and denote it CUF-CPA, if it is infeasible for any attacker  $\mathcal{F}$  (called a “*ciphertext forger*”) that has access to an encryption oracle  $\mathcal{O}_{ENC}$  with key *k* to produce a valid ciphertext under *k* not generated by  $\mathcal{O}_{ENC}$  as response to one of the queries by  $\mathcal{F}$ . More precisely, we quantify ciphertext unforgeability by the function  $\mathcal{E}_U(q, Q, T)$  defined as the maximal probability of success for any ciphertext forger  $\mathcal{F}$  that queries *q* plaintexts totalling *Q* bits and spends time *T* in the attack. We stress that this definition does not involve access to a decryption oracle and thus its name CUF-CPA (this is consistent with other common notations of the form X-Y where X represents the goal of the attacker and Y the assumed abilities of the attacker).

Our main use of the CUF-CPA notion is for proving (see Section 5) that under certain conditions the *AtE* composition is secure, i.e., it implements secure

channels. However, the notion of CUF-CPA while sufficient for our purposes is actually stronger than needed. For example, any scheme  $ENC$  that allows for arbitrary padding of ciphertexts to a length-boundary (e.g., to a multiple of 8-bits) will not be CUF-CPA (since given a ciphertext with padded bits any change to these bits will result in a different yet valid ciphertext). However, such a scheme may be perfectly secure in the context of implementing secure channels (see [8]); moreover, schemes of this type are common in practice. Thus, in order to avoid an artificial limitation of the schemes that we identify as secure for implementing secure channels we present next a relaxation of the CUF-CPA notion that is still sufficient for our purposes (we stress that this is not necessarily the weakest relaxation for this purpose and other weakenings of the CUF-CPA notion are possible).

Let  $\rho$  be a *polynomial-time computable* relation on pairs of ciphertexts computed under the encryption function  $ENC$  with the property that  $\rho(c, c')$  implies that  $c$  and  $c'$  decrypt to the same plaintext. Then we say that the encryption scheme  $ENC$  is CUF $_{\rho}$ -CPA if for any valid ciphertext  $c$  that the attacker can feasibly produce there exists a ciphertext  $c'$  output by the encryption oracle such that  $\rho(c, c')$ . When the relation  $\rho$  is not explicitly described we will refer to this notion as *loose ciphertext unforgeability*.

For instance, in the above example of a scheme that allows for arbitrary padding of ciphertexts, if one defines  $\rho(c, c')$  to hold if  $c$  and  $c'$  differ only on the padding bits, then the scheme can achieve CUF $_{\rho}$ -CPA. We note that while CUF-CPA implies CCA-security, loose CUF-CPA does not (as the above “padding example” shows). Indeed, as we pointed out in the introduction (see also Section 4.2) CCA-security is not a necessary condition for a MAC/encryption combination to implement secure channels.

## 4 Generic composition of encryption and authentication

In this section we study the security of the three methods,  $EtA$ ,  $AtE$ ,  $E\&A$ , under generic symmetric encryption and MAC functions where the only assumption is that the encryption is IND-CPA and the MAC is secure against chosen message attacks. Our focus is on the appropriateness of these methods to provide security to transmitted data in a realistic setting of adversarially-controlled networks. In other words, we are interested in whether each one of these methods when applied to adversarially-controlled communication channels achieve the goals of information secrecy and integrity. As we will see only the encrypt-then-authenticate method is generically secure.

### 4.1 The known security of encrypt-then-authenticate

The results in this subsection are from [8] and we present them briefly for completeness. We refer the reader to that paper for details. In particular, in the statement of the next theorem we use the notion of “secure channels” as introduced in the above paper and sketched in Section 2.3.

**Theorem 1.** [8] *If  $ENC$  is a symmetric encryption scheme secure in the sense of IND-CPA and  $MAC$  is a secure MAC family then method  $EtA(ENC, MAC)$  implements secure channels.*

Following our terminology from Section 2.3, the meaning of the above theorem is that if in the network channels model of [8] one applies to each transmitted message the composed function  $EtA(ENC, MAC)$  (as the  $snd$  function) then the secrecy and authenticity of the resultant network channels is guaranteed. More precisely, in proving the above theorem, [8] specify the  $snd$  function as follows. First, a pair of (computationally independent) keys,  $\kappa_a$  and  $\kappa_e$ , are derived from each session key. Then, for each transmitted message,  $m$ , a *unique* message identifier  $m-id$  is chosen (e.g., a sequence number). Finally, the function  $snd$  produces a triple  $(x, y, z)$  where  $x = m-id$ ,  $y = ENC_{\kappa_e}(m)$ ,  $z = MAC_{\kappa_a}(m-id, y)$ . On an incoming message  $(x', y', z')$  the  $rcv$  function verifies the uniqueness of message identifier  $x'$  and the validity of the  $MAC$  tag  $z$  (computed on  $(x', y')$ ); if the checks succeeds  $y'$  is decrypted under key  $\kappa_e$  and the resultant plaintext accepted as a valid message.<sup>2</sup>

A main contribution of the present paper is in showing (see next subsections) that a *generic* result as in Theorem 1 cannot hold for any of the other two methods,  $AtE$  and  $E&A$  (even if the used keys are shared with perfect security). Therefore, any secure channels protocol designed to work with *any* combination of secure encryption (against chosen plaintext attacks) and secure MAC must use the encrypt-then-authenticate method. However, we note in Section 5 that the above theorem can be extended in the setting of method  $AtE$  if one assumes a stronger property on the encryption function; in particular, we show two important cases that satisfy the added security requirement.

**Remark.** Note that the authentication of the ciphertext provides plaintext integrity as long as the encryption and decryption keys used at the sender and receiver, respectively, are the same. While this key synchrony is implicit in our analytical models [8], a key mismatch can happen in practice. A system concerned with detecting such cases can check the plaintext for redundancy information (such redundancy exists in most applications: e.g., message formats, non-cryptographic checksums, etc.). If the redundancy entropy is significant then a key mismatch will corrupt this redundancy with high probability.

#### 4.2 Authenticate-then-encrypt is not generically secure

Here we show that the authenticate-then-encrypt method  $AtE(ENC, MAC)$  is not guaranteed to be secure for implementing secure channels even if the function  $ENC$  is IND-CPA and  $MAC$  provides message unforgeability against chosen message attacks. First, however, we discuss shortly why this result does not follow from [5] where it is shown that the  $AtE$  composition (viewed as an encryption scheme) does not necessarily provide IND-CCA. The reason is simple: as

<sup>2</sup> Protocols that use a synchronized counter as the message identifier, e.g. SSL, do not need to transmit this value; yet they must include it under the  $MAC$  computation and verification. If transmitted, identifiers are not encrypted under  $ENC_{\kappa_e}$  since they are needed for verifying the  $MAC$  value before the decryption is applied.

demonstrated in [8] IND-CCA *is not a necessary condition for a combination of encryption and MAC functions to implement secure channels*. An example is provided by the main construction of secure channels in [8] (see Theorem 1): if the MAC used in this scheme enjoys regular MAC security, rather than the strengthened notion described in the last remark of Section 2.1, then this construction guarantees secure channels but not necessarily CCA security. (For example, if the MAC function has the property that flipping the last bit of an authentication tag does not change the validity of the tag, then the scheme in [8] is not IND-CCA yet it suffices for implementing secure channels. For a similar example, see remark on “multi-valued MAC” following our Theorem 3.) Moreover, the specific example from [5] of a non-CCA  $AtE(ENC, MAC)$  scheme<sup>3</sup> can by itself be used to show an example of a non-CCA scheme that provably provides secure channels. Therefore, the result in [5] does not say anything about the suitability of  $AtE(ENC, MAC)$  for implementing secure channels; it rather points out to the fact that while CCA security is a useful security notion it is certainly too strong for some (fundamental) applications such as secure channels.

Thus if we want to establish the *insecurity* of authenticate-then-encrypt channels under generic composition we need to show an explicit example and a successful attack. We provide such example now. In this example the encryption scheme is IND-CPA (actually, it enjoys “perfect secrecy” in the sense of Shannon) but when combined with *any* MAC function under the  $AtE$  method the secrecy of the composed scheme breaks completely under an active attack.

**The encryption function  $ENC^*$ .** We start by defining an encryption scheme  $ENC^*$  that can be based on any stream cipher  $ENC$  (i.e. any encryption function that uses a random or pseudorandom pad to xor with the data). The scheme  $ENC^*$  preserves the IND-CPA security of the underlying scheme  $ENC$ . In particular, if  $ENC$  has perfect secrecy (i.e., uses a perfect one-time pad encryption) so does  $ENC^*$ . Next, we define  $ENC^*$ .

Given an  $n$ -bit plaintext  $x$  (for any  $n$ ),  $ENC^*$  first applies an encoding of  $x$  into a  $2n$ -bit string  $x'$  obtained by representing each bit  $x_i$ ,  $i = 1, \dots, n$ , in  $x$  with two bits in  $x'$  as follows:

1. if bit  $x_i = 0$  then the pair of bits  $(x'_{2i-1}, x'_{2i})$  is set to  $(0, 0)$ ;
2. if bit  $x_i = 1$  then the pair of bits  $(x'_{2i-1}, x'_{2i})$  is set to  $(0, 1)$  or to  $(1, 0)$  (by arbitrary choice of the encrypting party).

The encryption function  $ENC$  is then applied to  $x'$ . For decrypting  $y = ENC^*(x)$  one first applies the decryption function of  $ENC$  to obtain  $x'$  which is then decoded into  $x$  by mapping a pair  $(0, 0)$  into 0 and either pair  $(0, 1)$  or  $(1, 0)$  into 1. If  $x'$  contains a pair  $(x'_{2i-1}, x'_{2i})$  that equals  $(1, 1)$  the decoding outputs the invalidity sign  $\perp$ .

**The attack when only encryption is used.** For the sake of presentation let's first assume that *only*  $ENC^*$  is applied to the transmitted data (we will then treat the  $AtE$  case where a MAC is applied to the data before encryption). In

<sup>3</sup> Just append an arbitrary one-bit pad to the ciphertext and discard the bit before decryption.

this case when an attacker  $\mathcal{A}$  sees a transmitted ciphertext  $y = ENC^*(x)$  it can learn the first bit  $x_1$  of  $x$  as follows. It intercepts  $y$ , flips (from 0 to 1 and from 1 to 0) the first two bits  $(y_1, y_2)$  of  $y$ , and sends the modified ciphertext  $y'$  to its destination. If  $A$  can obtain the information of whether the decryption output a valid or invalid plaintext then  $A$  learns the first bit of  $x$ . This is so since, as it can be easily seen, the modified  $y'$  is valid if and only if  $x_1 = 1$ . (Remember that we are using a stream cipher to encrypt  $x'$ .) Clearly, this breaks the secrecy of the channel (note that the described attack can be applied to any of the bits of the plaintext). One question that arises is whether it is realistic to assume that the attacker learns the validity or invalidity of the ciphertext. The answer is that this is so for many practical applications that will show an observable change of behavior if the ciphertext is invalid (in particular, many applications will return an error message in this case).

To make the point even clearer consider a protocol that transmits passwords and uses  $ENC^*$  to protect passwords over the network (this is, for example, one of the very common uses of SSL). The above attack if applied to one of the bits of the password (we assume that the attacker knows the placement of the password field in the transmitted data) will work as follows. If the attacked bit is 1 then the password authentication will succeed in spite of the change in the ciphertext. If it is 0 the password authentication will fail. In this case success or failure is reported back to the remote machine and then learned by the attacker. In applications where the same password is used multiple times (again, as in many applications protected by SSL) the attacker can learn the password bit-by-bit. The same can be applied to other sensitive information such as to credit card numbers where a mistake in this number will be usually reported back and the validity/invalidity information will be learned by  $A$ .

**The attack against the  $AtE(ENC^*, MAC)$  scheme.** Consider now the case of interest for us in which the encryption is applied not just to the data but also to a MAC function computed on this data. Does the above attack applies? The answer is YES. The MAC is applied to the data before encoding and encryption and therefore if the original bit is 1 the change in ciphertext will result in the same decrypted plaintext and then the MAC check will succeed. Similarly, if the original bit is 0 the decrypted plaintext will have a 1 instead and the MAC will fail. All the attacker needs now is the information of whether the MAC succeeded or not. Note that in a sense *the MAC just makes things worse* since regardless of the semantics of the application a failure of authentication is easier to learn by the attacker: either via returned error messages, or by other effects on the application that can be observed by the attacker.

**Discussion: what have we learned?** The example using  $ENC^*$  is certainly sufficient to show that the method  $AtE$  can be insecure even if the encryption function is IND-CPA secure and the MAC unforgeable (note that this conclusion does not depend on any specific formalization of secure communications; any reasonable definition of security must label the above protocol as insecure). Therefore, if one wants to claim the security of  $AtE(ENC, MAC)$  for particular functions  $ENC$  and  $MAC$  one needs to analyze the combination as a whole or

use stronger or specific properties of the encryption function (see Section 5). An interesting issue here is how plausible it is that people will ever use an encryption scheme such as  $ENC^*$ . We note that although this scheme does not appear to be the most natural encryption mechanism some (equally insecure) variants of it may arise in practice. First the application of an encoding to a plaintext before encryption is used many times for padding and other purposes and is a particularly common practice in public key encryption algorithms. Second, encodings of this type can be motivated by *stronger* security requirements: e.g. to prevent an attacker from learning the exact length of transmitted messages or other traffic analysis information. In this case one could use an encoding similar to  $ENC^*$  but with variable size codes. (Just to make the point: note that a good example of traffic analysis arises in the above examples where the attacker has a lot to learn from error-reporting messages; even in cases where this information is encrypted it can usually be learned through the analysis of packet lengths, etc.) Another setting where plaintext encoding is introduced in order to improve security is for combating timing and power analysis attacks.

The bottom line is that it is highly desirable to have schemes that are robust to generic composition and are not vulnerable when seemingly innocuous changes are made to an algorithm (or when a new more secure or more efficient algorithm or mode is adopted)<sup>4</sup>.

### 4.3 Encrypt-and-authenticate is not generically secure

The first observation to make regarding the encrypt-and-authenticate method is that under the common requirements from a MAC function this method cannot guarantee the protection of secrecy (even against a passive eavesdropper). This is so since a MAC can be secure against forgeries but still leak information on the plaintext. Thus, the really interesting question is whether the method becomes secure if we avoid this obvious weakness via the use of a “secrecy protecting” MAC such as one implemented via a pseudorandom function or when the MAC tag is encrypted (most, if not all, MAC functions used in practice are believed to protect secrecy). Unfortunately, however, the attack from the previous section applies here too, thus showing the (generic) insecurity of the  $E\&A$  method even under the above stronger forms of MAC. (See also last remark in Section 5.2.)

## 5 Authenticate-then-encrypt with CBC and OTP modes

In Section 4.2 we saw that authenticate-then-encrypt cannot guarantee secure channels under the sole assumption that the encryption function is IND-CPA, even if the MAC function is perfectly secure. In this section we prove that for two common modes of encryption, CBC (with a secure underlying block cipher) and OTP (stream ciphers that xor data with a (pseudo) random pad), the  $AtE$  mode does work for implementing secure channels.

<sup>4</sup> See the last remark in Section 5.2 for another example where seemingly harmless changes transform a secure protocol into an insecure one.

## 5.1 A sufficient condition for the security of $AtE$

We start by pointing out to the following Theorem that can be proven in the security model of [8] (see Section 2.3).

**Theorem 2.** (derived from [8]) *Let  $ENC$  be an IND-CPA encryption function and  $MAC$  a MAC function. If the composed function  $AtE(ENC, MAC)$ , considered as an encryption scheme, is (loose) CUF-CPA, then  $AtE(ENC, MAC)$  implements secure channels.*

That is, under the assumptions on the  $ENC$  and  $MAC$  functions as stated in the Theorem, applying the function  $AtE(ENC, MAC)$  to information transmitted over adversarially-controlled links protects the secrecy and integrity of this information. More specifically, the Theorem implies the following definition of the function  $snd$  in the network channels model of [8] (see Section 2.3). For each transmitted message  $m$  with unique message identifier  $m-id$  the function  $snd$  produces a pair  $(x, y)$  where  $x = m-id$  and  $y = ENC_{\kappa_e}(m, MAC_{\kappa_a}(m-id, m))$ , where the keys  $\kappa_e$  and  $\kappa_a$  are computationally independent keys derived from the session key. On an incoming message  $(x', y')$  the  $rcv$  function verifies the uniqueness of message identifier  $x'$ , decrypts  $y'$  under key  $\kappa_e$ , verifies the validity of the decrypted  $MAC$  tag, and if all tests succeed the recipient accepts the decrypted message as valid. We note that if the message identifier is maintained in synchrony by sender and receiver (as in SSL) then there is no need to send its value over the network. On the other hand, if sent, the message identifier can be encrypted too. The above Theorem holds in either case.

We stress that the Theorem holds for strict CUF-CPA as well as for the relaxed “loose” version (see Section 3).

Based on this Theorem, and on the fact that OTP and CBC are IND-CPA [2], we can prove the security of  $AtE$  under OTP and CBC by showing that in this case the resultant  $AtE$  scheme is CUF-CPA. The rest of this section is devoted to prove these facts.

## 5.2 $AtE$ with OTP

**The OTP scheme.** Let  $F$  be a family of functions with domain  $\{0, 1\}^\ell$  and range  $\{0, 1\}^{\ell'}$ . We define the encryption scheme  $OTP(F)$  to work on messages of length at most  $\ell'$  as follows. A key in the encryption scheme is a description of a member  $f$  of the family  $F$ . The OTP encryption under  $f$  of plaintext  $x$  is performed by choosing  $r \in_R \{0, 1\}^\ell$  and computing  $c = f(r) \oplus x$  where  $f(r)$  is truncated to the length of  $x$ . The ciphertext is the pair  $(r, c)$ . Decryption works in the obvious way. If  $F$  is the set of *all* functions with the above domain and range and  $f$  is chosen at random from this family we get perfect secrecy against chosen-plaintext attacks as long as there are no repetitions in the values  $r$  chosen by the encryptor (after encrypting  $q$  different messages a repetition happens with probability  $q^2/2^\ell$ ); we denote this scheme by  $OTP_{\S}$ . If  $F$  is a family of pseudorandom functions then the same security is achieved but in a computational sense, i.e., up to the “indistinguishability distance” between the

pseudorandom family and a truly random function. A formal and exact-security treatment of this mode of encryption can be found in [2].

**The  $AtE(OTP_s, MAC)$  composition.** Let  $MAC$  be a MAC family with  $n$ -bit outputs, and  $k$  a key to a member of that family. Let  $f$  be a random function with domain and range as defined above. The  $AtE(OTP_s, MAC)$  function with  $f$  and  $k$  acts as follows: (i) it receives as input a message  $x$  of length at most  $\ell' - n$ , (ii) computes  $t = MAC_k(x)$ , (iii) appends  $t$  to  $x$ , (iv) outputs the OTP encryption under  $f$  of the concatenated message  $(x, t)$ .

The following theorem establishes the CUF-CPA security of  $AtE(OTP_s, MAC)$  as a function of the security  $\mathcal{E}_M(\cdot, \cdot, \cdot)$  of  $MAC$ .

**Theorem 3.** *If  $MAC$  is a MAC family that resists one-query attacks then  $AtE(OTP_s, MAC)$  is CUF-CPA (and then by Theorem 2 it implements secure channels). More precisely, any ciphertext forger  $\mathcal{F}$  against  $AtE(OTP_s, MAC)$  that runs time  $T$  has success probability  $\mathcal{E}_U$  of at most  $q^2/2^\ell + \mathcal{E}_M(1, p, T')$ , where  $\ell$  is a parameter of  $OTP_s$ ,  $q$  is the number of queries  $\mathcal{F}$  makes during the attack,  $p$  is an upper bound on the length of each such query and on the length of the output forgery, and  $T' = T + cq$  for some constant  $c$ .*

For a proof of the Theorem see [21].

Using standard techniques one can show that the theorem holds also for a OTP scheme realized via a family of pseudorandom functions if we add to the above probability bound the distinguishability distance between the pseudorandom family and a truly random function. Also, the  $q^2/2^\ell$  component can be eliminated if one uses non-repeating nonces instead of random  $r$ 's (such as in counter mode or via a stateful pseudorandom generator used to generate a pseudorandom pad).

**Remark (Tightness: one-query resistance is necessary).** Here is an example of a MAC that *does not* resist one-queries and with which valid ciphertext can be forged against  $AtE(OTP_s, MAC)$ . Assume  $MAC$  allows for finding two same-length messages with the same MAC tag. (For example,  $MAC$  first zeros the last bit of the message and then applies a secure MAC function on the resultant message. Thus,  $MAC$  resists zero-queries but fails to one-queries: ask for a MAC on a message, then forge for the message with last bit flipped.) The strategy of the ciphertext forger against  $AtE(OTP_s, MAC)$  is to find such pair of messages  $x_1, x_2$ . Then, it queries the first one and gets the ciphertext  $(r, c)$ . Finally, it outputs the forgery  $(r, c')$  where  $c'$  is obtained from  $c$  by xor-ing  $x_2$  to the first  $|x_2|$  bits of  $c$ . It is easy to see that  $(r, c')$  decrypts to  $(x_2, MAC(x_2))$ .

**Remark (Multi-valued MAC).** In Section 2.1 we strengthened the regular security definition of a MAC function in the case that the function allows for different valid authentication tags for the same message. This extended definition is used (explicitly) in the proof of Theorem 3 and is essential for ensuring the CUF-CPA property of  $AtE(OTP_s, MAC)$ . To see this, let  $MAC$  be a secure single-valued MAC function and define  $MAC'$  to be the same as  $MAC$  except that an additional arbitrary bit is appended to each authentication tag. The verification procedure will just ignore this bit. It is easy to see that in this case



$AtE(OTP_s, MAC')$  will not be CUF-CPA. However, if one examines the proof of Theorem 3 it can be seen that  $AtE(OTP_s, MAC')$  achieves loose CUF-CPA (see Section 3) and then it is sufficient for implementing secure channels (which is what we care about). So can we dispense of the strengthened notion of MAC when multi-valued MACs are used? The answer is no. It is possible to build a multi-valued function  $MAC'$  that satisfies the regular MAC definition, but not the strengthened version, for which  $AtE(OTP_s, MAC')$  is *insecure* for building secure channels (see [21]).

**Remark (Sufficiency of redundancy functions).** In [1] An and Bellare investigate the question of whether simple redundancy functions (such as combinatorial hash functions) applied to a plaintext before encryption suffice for providing ciphertext unforgeability. In the case of  $AtE$  with OTP it seems natural to assume that a simple combinatorial property of the redundancy function such as  $AXU$  [20, 25] should suffice. (In particular, this seems so since such a property is sufficient [20] if one only considers *plaintext integrity* where only the output of the redundancy function is encrypted under an OTP scheme.) However, this turns out not to be true in the case of ciphertext unforgeability. We can show an example of an  $\mathcal{E}$ - $AXU$  (and also  $\mathcal{E}$ -balanced [20])  $MAC$  family for which  $AtE(OTP_s, MAC)$  is not CUF-CPA. It seems plausible, however, that a more involved combinatorial property (involving the length of messages) of the  $MAC$  function could suffice to guarantee ciphertext unforgeability in the case of  $AtE$  with OTP. Actually, it is interesting to note that if the authentication tag is positioned *before* the message, instead of at the end as defined above, the  $AXU$  property is indeed sufficient (assuming fixed-length and single-valued valid authentication tags).

**Remark (Beware of “slight changes”).** To highlight the “fragility” of the result in Theorem 3 we note that the proof of this theorem uses in an essential way the fact that the encryption is applied as a whole on the concatenated message and  $MAC$  tag. If we were to encrypt these two values *separately* (i.e., using separate IVs for the encryption of the message and of the  $MAC$ ) even under a truly random function we would not get CUF or CCA security. More significantly, such separate encryption results in *insecure channels*. Indeed, under this method an active attacker can get to learn whether two transmitted messages, possibly with different message identifiers, are the same, something clearly unwanted in a secure protocol. (This weakness allows for actual attacks on practical applications, in particular several forms of “dictionary attacks”<sup>5</sup>)

In addition, this observation shows another weakness of the encrypt-and-authenticate method (Section 4.3) since it exhibits the insecurity of this method even under the use of a standard stream cipher for encryption and even when the  $MAC$  tag is encrypted.

<sup>5</sup> One such example would be finding passwords sent in the `telnet` protocol even if the protocol is run over a secure channel protected as above; this is particularly facilitated by the fact that in this case *individual password characters* are transmitted separately, and thus a dictionary attack can be mounted on individual characters.

### 5.3 AtE with CBC

**The CBC scheme.** Let  $\ell$  be a positive integer and  $F$  be a family of permutations over  $\{0, 1\}^\ell$ . We define the encryption scheme  $CBC(F)$  to work on messages of length a multiple of  $\ell$ . A key in the encryption scheme is a description of a member  $f$  of the family  $F$ . The CBC encryption under  $f$  of plaintext  $x$  is performed by partitioning  $x$  into blocks  $x[1], \dots, x[p]$  of length  $\ell$  each, then choosing  $r \in_R \{0, 1\}^\ell$  (called the IV) and computing the ciphertext  $c = c[0], c[1], \dots, c[p]$  as  $c[0] = r, c[i] = f(c[i-1] \oplus x[i]), i = 1, \dots, p$ . Decryption works in the obvious inverse way. If  $F$  is the set of *all* permutations over  $\{0, 1\}^\ell$  and  $f$  is chosen at random from  $F$  then we denote the scheme by  $CBC_{\mathfrak{s}}$ . A formal and exact-security treatment of this mode of encryption can be found in [2] who in particular prove it to be IND-CPA also in the case where  $F$  is a pseudorandom family (in this case the security depends on the “indistinguishability distance” between the pseudorandom family and a truly random function).

**The  $AtE(CBC_{\mathfrak{s}}, MAC)$  composition.** Let  $MAC$  be a MAC family with  $\ell$ -bit outputs, and  $k$  a key to a member of that family. Let  $f$  be a random permutation over  $\{0, 1\}^\ell$ . The  $AtE(CBC_{\mathfrak{s}}, MAC)$  function with  $f$  and  $k$  acts as follows: (i) it receives as input a message  $x$  of length multiple of  $\ell$ , (ii) computes  $t = MAC_k(x)$ , (iii) appends  $t$  to  $x$ , (iv) outputs the CBC encryption under  $f$  of the concatenated message  $(x, t)$  (note that the resultant output is two blocks longer than  $x$  due to the added block  $t$  and the prepended IV  $r$ ).

The following theorem establishes the CUF-CPA security of  $AtE(CBC_{\mathfrak{s}}, MAC)$  as a function of the security  $\mathcal{E}_M(\cdot, \cdot, \cdot)$  of  $MAC$ .

**Theorem 4.** *If  $MAC$  is a secure MAC family then  $AtE(CBC_{\mathfrak{s}}, MAC)$  is CUF-CPA (and then by Theorem 2 it implements secure channels). More precisely, any ciphertext forger  $\mathcal{F}$  against  $AtE(CBC_{\mathfrak{s}}, MAC)$  that runs time  $T$  has success probability  $\mathcal{E}_U$  of at most*

$$Q^2/2^\ell + 2q\mathcal{E}_M(0, 0, T') + \mathcal{E}_M(1, p\ell, T') + 2\mathcal{E}_M(q^*, q^*p\ell, T')$$

where  $q$  is the number of plaintexts queried by  $\mathcal{F}$ ,  $p$  is an upper bound on the number of blocks in each of these queries,  $p^*$  is the length in blocks of the forgery  $y^*$  output by  $\mathcal{F}$ ,  $q^* = \min\{q, p^*\}$ ,  $Q$  is the total number of blocks in the responses to  $\mathcal{F}$ 's queries plus  $p^*$ , and  $T' = T + cQ$  for constant  $c$ .

For a proof of the Theorem see [21].

Using standard techniques one can show that the theorem holds also for a CBC scheme realized via a family of pseudorandom permutations if we add to the above probability bound the distinguishability distance between the pseudorandom family and a truly random function. However, we note, that in this case the distinguisher not only gets access to an oracle that computes the function but also to an oracle that computes the inverse function (that is, we need to assume the family of permutations to be “super pseudorandom” [22]).

**Remark (Tightness: the necessity of the bound  $\mathcal{E}_M(q^*)$ ).** The most “expensive” term in MAC security in the expression of the theorem is the value

$\mathcal{E}_M(q^*)$  since other terms only require protection against one-query or zero-query. Since an attacker  $\mathcal{F}$  does not get to see any of the MAC values one could wonder why such a strong security from the MAC is required. We show here that, in contrast to the  $AtE(OTP_{\S}, MAC)$  case, this requirement is unavoidable. Specifically, we present for any  $i = 0, 1, 2, \dots$ , an example of a MAC function  $MAC$  that is secure against  $i$  queries but yields an insecure  $AtE(CBC_{\S}, MAC)$  scheme with  $q = i + 1$  (and  $p^* = 2i + 4$ ). We describe the example for  $i = 1$ , the extension to other values is straightforward.

Let  $\{g_k\}_k$  be a family of pseudorandom functions from  $(\{0, 1\}^{\ell})^*$  to  $\{0, 1\}^{\ell/2}$ . Define a MAC family  $MAC'$  on the same domain as  $\{g_k\}_k$ , and with  $\ell$ -bit outputs as follows:  $MAC'_{(k_1, k_2)}(x) = (g_{k_1}(x), g_{k_2}(g_{k_1}(x)))$ . Define a second MAC family  $MAC$  that uses the same set of keys as  $MAC'$  and such that on key  $(k_1, k_2)$ :

1. if the input  $x$  contains two  $\ell$ -bit blocks  $b_i$  and  $b_j$ ,  $i < j$ , such that  $b_i \neq b_j$  and both have the property that applying  $g_{k_2}$  to the first half of the block yields the second half of the block then output  $b_i$  as the  $MAC$  value for  $x$ .
2. otherwise, output  $MAC'_{(k_1, k_2)}(x)$

It is easy to see that the so defined  $MAC$  has security of roughly  $2^{\ell/2}$  against single queries (but is totally insecure after two queries since the output of  $MAC$  provides the block format that makes the authentication tag “trivial”). We show that it yields a  $AtE(CBC_{\S}, MAC)$  scheme whose ciphertexts are forgeable after two queries even if the encryption permutation  $f$  is purely random. The ciphertext forger  $\mathcal{F}$  against  $AtE(CBC_{\S}, MAC)$  proceeds as follows:

1. Choose two arbitrary one-block long plaintexts  $x_1, x_2$  as the two queries.
2. Let the responses  $y_1, y_2$  be the triples:  $(r_1, c_1 = f(r_1 \oplus x_1), m_1 = f(c_1 \oplus MAC(x_1)))$  and  $(r_2, c_2 = f(r_2 \oplus x_2), m_2 = f(c_2 \oplus MAC(x_2)))$ .
3. Output forgery  $y^* = (c_1, m_1, c_2, m_2, c_1, m_1)$ .

A simple examination shows that  $y^*$  is a valid ciphertext.

One consequence of the above lower bound on the required security of  $MAC$  is that, somewhat surprisingly, the  $MAC$  function cannot be replaced by a simple combinatorial hash function, such as one enjoying  $AXU$  (see remark on “redundancy functions” in Section 5.2). Indeed, had  $AXU$  been sufficient then one-query resistant  $MAC$ s would suffice too (since one-query resistance implies  $AXU$ ). We note that a modified CBC-like mode for which  $AXU$  is sufficient is presented in [1].

In contrast to the above lower bound, we do not know if the term  $q\mathcal{E}_M(0)$  in the bound of the theorem is necessary or not; we do not have so far an example that shows this term to be unavoidable. Thus, it may well be the case that a more careful analysis could lower the factor  $q$  (actually, even with the current analysis it is possible to replace the factor  $q$  with  $q^*$  by a slightly more involved argument).

**Remark (Non-adaptive security of  $MAC$  suffices).** It is interesting to note that the requirement from the security of the  $MAC$  in Theorem 4 is for *non-*

*adaptive* queries only. This can be seen by inspecting the proof of the theorem, where the MAC forger  $\mathcal{G}$  that we build makes non-adaptive queries only.

**Remark (*Beware of “slight changes”*).** Similarly to the case of  $AtE(OTP_s, MAC)$  the proof of Theorem 4 uses in an essential way the fact that the encryption is done as a whole on the concatenated message and MAC. It is easy to build a ciphertext forgery attack in case the encryption of the plaintext and of the MAC tag are done separately (i.e. with independently chosen IVs).

## Acknowledgment

I would like to thank Yaron Scheffer for motivating conversations on this topic and for “forcing” me to find an explicit counter-example for the  $AtE$  method; Yaron also helped in simplifying a previous example. I also thank Mihir Bellare for interesting conversations and for highlighting some of the subtleties related to the subject of this paper, and to Ran Canetti and Jonathan Katz for valuable comments on earlier drafts of the paper.

This research is supported by an Irwin and Bethea Green & Detroit Chapter Career Development Chair, and by the Fund for the Promotion of Research at the Technion.

## References

1. J. An, M. Bellare, “Does encryption with redundancy provide authenticity?”, *Advances in Cryptology – EUROCRYPT 2001 Proceedings*, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, B. Pfitzmann, ed, 2001.
2. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, “A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation”, *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, “Relations Among Notions of Security for Public-Key Encryption Schemes”, *Advances in Cryptology - CRYPTO’98 Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk, ed., Springer-Verlag, 1998, pp. 26–45.
4. M. Bellare, J. Kilian and P. Rogaway, “The security of cipher block chaining”, *Advances in Cryptology – CRYPTO’94 Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt, ed., Springer-Verlag, 1994. pp. 341-358.
5. M. Bellare and C. Namprempre, “Authenticated encryption: Relations among notions and analysis of the generic composition paradigm”, *Advances in Cryptology - ASIACRYPT’00 Proceedings*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto, ed., Springer-Verlag, 2000.
6. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P., “UMAC: Fast and Secure Message Authentication”, *Advances in Cryptology – CRYPTO’99 Proceedings*, Lecture Notes in Computer Science, Vol. 1666, Springer-Verlag, M. Wiener, ed, 1999, pp. 216–233.
7. Bleichenbacher, D., “Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1”, *Advances in Cryptology - CRYPTO’98 Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk, ed., Springer-Verlag, 1998, pp. 1–12.

8. Canetti, R., and Krawczyk, H., "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", *Advances in Cryptology – EUROCRYPT 2001 Proceedings*, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, B. Pfitzmann, ed, 2001, pp. 453–474. Full version in: *Cryptology ePrint Archive* (<http://eprint.iacr.org/>), Report 2001/040.
9. T. Dierks and C. Allen, "The TLS Protocol – Version 1", *Request for Comments 2246*, 1999.
10. D. Dolev, C. Dwork, and M. Naor. "Non-malleable cryptography". *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 542-552, 1991.
11. A. Frier, P. Karlton, and P. Kocher, "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996. <http://home.netscape.com/eng/ssl3/ssl-toc.html>
12. O. Goldreich, "Foundations of Cryptography (Fragments of a book)", Weizmann Inst. of Science, 1995. <http://www.wisdom.weizmann.ac.il/oded/frag.html>
13. S. Goldwasser, and S. Micali. "Probabilistic Encryption", *Journal of Computer and System Sciences*, Vol. 28, 1984, pp. 270-299.
14. Halevi, S., and Krawczyk H., "Public-Key Cryptography and Password Protocols", *ACM Transactions on Information and System Security*, Vol. 2, No. 3, August 1999, pp. 230–268.
15. C. Jutla, "Encryption Modes with Almost Free Message Integrity", *Advances in Cryptology – EUROCRYPT 2001 Proceedings*, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, B. Pfitzmann, ed, 2001.
16. J. Katz and M. Yung, "Unforgeable encryption and adaptively secure modes of operations", *Fast Software Encryption'00*, 2000.
17. J. Katz and M. Yung, "Complete characterization of security notions for probabilistic private-key encryption", *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 2000.
18. S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", *Request for Comments 2401*, Nov. 1998.
19. S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)", *Request for Comments 2406*, Nov. 1998.
20. H. Krawczyk, "LFSR-based Hashing and Authentication", *Proceedings of CRYPTO '94*, Lecture Notes in Computer Science, vol. 839, Y. Desmedt, ed., Springer-Verlag, 1994, pp. 129-139.
21. H. Krawczyk, "The order of encryption and authentication for protecting communications (Or: how secure is SSL?)". Full version: <http://eprint.iacr.org/2001>.
22. M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions", *SIAM J. on Computing*, Vol 17, Number 2, April 1988, pp. 373–386.
23. M. Naor and M. Yung, "Public key cryptosystems provably secure against chosen ciphertext attacks". *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990.
24. C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack", *Advances in Cryptology - CRYPTO'91 Proceedings*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed, Springer-Verlag.
25. P. Rogaway. "Bucket Hashing and its application to Fast Message Authentication", *Proceedings of CRYPTO '95*, Lecture Notes in Computer Science, vol. 963, D. Coppersmith, ed., Springer-Verlag, 1995, pp. 15-25.
26. P. Rogaway, M. Bellare, J. Black, and T. Krovetz, "OCB Mode", *Cryptology ePrint Archive*, Report 2001/026.
27. T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen, "SSH Transport Layer Protocol", January 2001, [draft-ietf-secsh-transport-09.txt](#).