The low-call diet: Authenticated Encryption for call counting HSM users

M. Bond[1], G. French[2], N.P. Smart[3], and G.J. Watson[3]

[1] Cryptomathic A/S, Cambridge, UK.
[2] Barclays Bank Plc, London, UK.
[3] University of Bristol, UK.

.

Abstract. We present a new mode of operation for obtaining authenticated encryption suited for use in environments, e.g. banking and government, where cryptographic services are only available via a Hardware Security Module (HSM) which protects the keys but offers a limited API. The practical problem is that despite the existence of better modes of operation, modern HSMs still provide nothing but a basic (unauthenticated) CBC mode of encryption, and since they mediate all access to the key, solutions must work around this. Our mode of operation makes only a single call to the HSM, yet provides a secure authenticated encryption scheme; authentication is obtained by manipulation of the plaintext being passed to the HSM via a call to an unkeyed hash function. The scheme offers a considerable performance improvement compared to more traditional authenticated encryption techniques which must be implemented using multiple calls to the HSM. Our new mode of operation is provided with a proof of security, on the assumption that the underlying block cipher used in the CBC mode is a strong pseudorandom permutation, and that the hash function is modelled as a random oracle.

1 Introduction

Authenticated symmetric encryption, namely an encryption scheme which is both IND-CPA and INT-CTXT secure [3], is regarded as *the* goal for symmetric encryption. There is no shortage of constructions in the literature for such authenticated encryption (AE) schemes. The most famous of these is the Encrypt-Then-MAC construction, which first encrypts the message using an IND-CPA encryption scheme and then appends a secure MAC to the ciphertext. Over the last decade various special modes of operation have been defined which implement authenticated encryption such as OCB [17], CCM [19], EAX [5] and GCM [12]. However, while these modes have been optimised for modern use and parallel services, there are some situations in which generic constructions (like Encrypt-Then-MAC) or special modes cannot be used.

In the financial and government sectors, a common industrial deployment of cryptography is for keys to reside within a special piece of hardware known as a Hardware Security Module or HSM. Such HSMs store the keys and manipulate sensitive data on behalf of applications, and offer a measure of assurance that neither a lone corrupt insider manipulating his own systems nor an external hacker will be able to steal or abuse cryptographic keys. Another benefit of utilizing HSMs is that they define the point at which any abuse of the key material may occur; so whilst a given HSM may not be truly secure (e.g. it may be susceptible to API attacks [6, 8]), nor may they require authentication of the caller, they do from a pragmatic point of view provide the location where any abuse by an attacker will occur.

This article is the final version (bar fomatting) submitted by the author(s) to Springer-Verlag in Nov 2012. The version published by Springer-Verlag is available at http://dx.doi.org/10.1007/978-3-642-36095-4_23

In practice the dominant characteristic of HSMs is their compliance and certification against security standards such as NIST FIPS 140-2, PCI HSM, and in certain business areas common criteria evaluation (e.g. against the Secure Signature Creation Device (SSCD) profile). These standards have varying levels of practical applicability – for instance FIPS 140 has relatively little to say about software API security but is widely depended upon as a measure of all round security. The standards are completely entrenched and not just adherence but certification is required in order to do business. Certification against such standards is expensive and applies to each version of HSM firmware and hardware released. Consequently the focus on security compliance and certification has slowed the rate of software change on HSMs, and it is rarely cost effective for a supplier to speculatively add new modes of operation in the hope they will be adopted.

To compound the matter, in a chicken-and-egg situation, the banking industry has been reluctant to specify authenticated encryption as part of standards for core HSM activities such as manipulation of customer PINs in banking networks and EMV[4], since in making this demand they will be requiring implementers to change their HSM firmware or radically increase their HSM load by making multiple calls per encrypt/decrypt.

Finally, for software vendors implementing software which use HSMs for crypto, different HSMs are dominant and are demanded in different regional markets[5]. Supporting a variety of vendors is both required by the market and is wise to enable price competition, hence the algorithm chosen for crypto must be the lowest common denominator across all these platforms.

Thus creation of more complex cryptographic construction is usually done via multiple API calls to the HSM. For example an HSM may provide an API call to perform CBC Mode with AES. To call this operation the user passes the plaintext to the HSM and specifies the key to use. The HSM recovers the key (usually looking it up in an internal store by name, or possibly by unwrapping a supplied encrypted key via some other internal key), applies CBC Mode to the message and then passes the ciphertext back to the caller. If Encrypt-then-MAC is required the whole process is then repeated again to obtain the MAC on the ciphertext. Thus traditional ways of obtaining authenticated encryption are computationally expensive.

The whole process is also very expensive in terms of latency; nowadays HSMs are network attached remote computers and thus are often orders of magnitude slower than using local crypto in the CPU of the calling machine. In addition HSMs are also rate-limited and price differentiated by supported number of transactions per second. Thus minimizing calls to an HSM is a major application requirement.

Note that considering the lowest common denominator across HSMs, CBC Mode may only be available with the all-zero IV, thus the HSM does not even provide an IND-CPA encryption scheme. This can be fixed by using the HSMs ability to generate random numbers, or using some other random number source, to produce a random IV and then xor-ing the first block of the plaintext with the IV before calling the HSM, but with a naive implementation this introduces an extra call to acquire randomness - yet again incurring a performance penalty.

Fundamentally HSM APIs were designed before the need for authenticated encryption was properly understood. Yet the cryptographic community has developed new modes for authenticated encryption which focus on squeezing all data fields including randomisation into a single blockcipher

[4] EMV is the major international smartcard based payment scheme named after its founders Europay Mastercard and Visa.

[5] Since the USA reclassification of crypto hardware in 2000 as no longer a munition the market is consolidating, but this takes time.

block; as well as obtaining authentication with only a single pass of the data. In particular, the specific new modes of operation for authenticated encryption (such as Galois Counter Mode) are not supported by such HSMs; and are unlikely to be so universally in the near future.

As the financial industry and local and national government come under greater pressure to protect data such as Personally Identifying Information (PII), there is a problem using existing HSM approaches, since this new data is much larger than the size of a single blockcipher block (unlike bank card PINs or cryptographic keys). The generic construction such as Encrypt-Then-MAC, utilizing two keys within the HSM (one for the encryption algorithm and one for the MAC), are not suitable for the reasons already described, so an efficient mode is required which operates on larger data, but within the constraints of lowest-common-denominator HSM support.

It is to solve this real world cryptographic issue that we turn our attention. The summary requirement is therefore to design a mode of operation which provides a service of authenticated encryption which has the following properties:

- All secret keys used by the mode should reside on the HSM.
- Only one call to the HSM is allowed, i.e. only one application of key material is allowed.
- The only modes of operation available are ECB or CBC-Encrypt with zero IV.

The mode of operation described in this paper has been deployed in several scenarios;

- By major European banks to protect sensitive customer data in transit and storage that does not fall under the existing frameworks for banking PINs or crypto keys.
- By vendors of customised HSM firmware that need a way of offloading cryptographic keys under a master key, held together with access control and usage information. Code which runs internally within an HSM is often subject to the same limitations of crypto primitive availability as that which make external calls.

Yet this paper represents the first time it has been described publicly and has been provided with a full security analysis:

The main idea is to "authenticate" the message using a "MAC-like" function and then encrypt the result; i.e. to apply a MAC-Then-Encrypt methodology. The encryption is performed using CBC Mode with a zero IV, but a random first block (to achieve probabilistic encryption). This is equivalent to CBC Mode with a random IV given by the first block of ciphertext. However, our MAC-like function must be key-less (to avoid another call to the HSM). So we use a hash function on the message sent to the CBC Mode encryption, this is like applying the hash function as a "MAC" with key given by the initial random plaintext block we added before encryption. However, such a MAC on its own is not secure as it suffers from length extension attacks. Amazingly, despite the use of MAC-Then-Encrypt with an insecure keyless MAC and an insecure encryption scheme (zero-IV CBC mode is not even IND-CPA) the whole construction can be proved secure, assuming the hash function is modelled as a random oracle.

2 Preliminaries

Notation We let $x\|y$ denote the concatenation of two strings x and y. $|x|$ denotes the length of the string x. We use $x \xleftarrow{r} \mathcal{X}$ to denote the random selection of an element x from the set $\mathcal{X}$. We denote the addition of x to the set $\mathcal{X}$ by $\mathcal{X} \xleftarrow{\cup} x$.

Pseudorandom Functions and Permutations We define pseudorandom functions and permutations as follows. We also provide a definition for strong PRPs.

Definition 1. ***[Pseudorandom Function/Permutation (PRF/PRP)]*** *Let $F = \{F_K : K \in \{0,1\}^k\}$ where $F_K : \{0,1\}^l \to \{0,1\}^{l'}$ for each $K \in \{0,1\}^k$. Let $P = \{P_K : K \in \{0,1\}^k\}$ where for each $K \in \{0,1\}^k$, $P_K : \{0,1\}^l \to \{0,1\}^l$ is a permutation. Let* Rand *be the set of all functions mapping l-bit strings to l'-bit strings. Let* Perm *be the set of all permutations mapping l-bit strings to l-bit strings. The prf and prp advantage of an adversary $\mathcal{A}$ are defined as:*

$$\mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{A}) = \Pr[K \xleftarrow{r} \{0,1\}^k : \mathcal{A}^{F_K} \Rightarrow 1] - \Pr[f \xleftarrow{r} \mathrm{Rand} : \mathcal{A}^{f} \Rightarrow 1]$$

$$\mathbf{Adv}^{\mathrm{prp}}_P(\mathcal{A}) = \Pr[K \xleftarrow{r} \{0,1\}^k : \mathcal{A}^{P_K} \Rightarrow 1] - \Pr[\pi \xleftarrow{r} \mathrm{Perm} : \mathcal{A}^{\pi} \Rightarrow 1]$$

A function family F (permutation family P resp.) is said a prf (prp resp.) if $\mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{A})$ ($\mathbf{Adv}^{\mathrm{prp}}_F(\mathcal{A})$ resp.) is "small" for all adversaries $\mathcal{A}$ running in time t making at most q_f queries.

Definition 2. ***[Strong Pseudorandom Permutation (SPRP)]*** *Let $P = \{P_K : K \in \{0,1\}^k\}$ where for each $K \in \{0,1\}^k$, $P_K : \{0,1\}^l \to \{0,1\}^l$ is a permutation with corresponding inverse permutation $P_K^{-1} : \{0,1\}^l \to \{0,1\}^l$. Let* Perm *be the set of all permutations mapping l-bit strings to l-bit strings. The sprp advantage of an adversary $\mathcal{A}$ is defined as:*

$$\mathbf{Adv}^{\mathrm{sprp}}_P(\mathcal{A}) = \Pr[K \xleftarrow{r} \{0,1\}^k : \mathcal{A}^{P_K, P_K^{-1}} \Rightarrow 1] - \Pr[\pi \xleftarrow{r} \mathrm{Perm} : \mathcal{A}^{\pi,\pi^{-1}} \Rightarrow 1]$$

A permutation family P is said to be an sprp if $\mathbf{Adv}^{\mathrm{sprp}}_F(\mathcal{A})$ is "small" for all adversaries $\mathcal{A}$ running in time t making at most q_f queries to both oracles.

In our analysis we will use PRP/PRFs and SPRPs to represent the block cipher used in our scheme. We will also make use the following lemma to relate PRF to PRP.

Lemma 1. [PRF→ PRP, [2, Proposition 8]] *For any permutation family $P = \{P_K : K \in \{0,1\}^k\}$ over l-bit strings.*

$$\mathbf{Adv}^{\mathrm{prf}}_P(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prp}}_P(\mathcal{B}) + \frac{q_f^2}{2^{l+1}}$$

Hash function Let $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^l$ be a hash function with outputs truncated to l-bits (where l is the blocksize of the block cipher used in the scheme). When we denote the hash function with input as $\mathsf{hash}(X, Y, Z)$ we simply compute the hash on the concatenation of the input i.e. $\mathsf{hash}(X \| Y \| Z)$.

CBC mode An HSM allows us an API call to CBC mode with a zero IV. We therefore define CBC mode accordingly. Let $\mathsf{E\text{-}CBC}^0[F](K, M)$ denote the CBC encryption of message M (with all zero IV) using the function family F under key K, i.e. $\mathsf{E\text{-}CBC}^0[F] : \{0,1\}^k \times \{0,1\}^{ln} \to \{0,1\}^{ln}$, $(n \in \mathbb{N})$, where for $M = M[1]M[2]...M[n]$ we have that $C[i] = F_K(M[i] \oplus C[i-1])$ and $C[0] = 0^l$. Let $\mathsf{D\text{-}CBC}^0[F](K, C)$ denote the CBC decryption of ciphertext C using the function family F under key K, i.e. $\mathsf{D\text{-}CBC}^0[F] : \{0,1\}^k \times \{0,1\}^{ln} \to \{0,1\}^{ln}$, $(n \in \mathbb{N})$, where for $C = C[1]C[2]...C[n]$ and $C[0] = 0^l$ we have $M[i] = F_K^{-1}(C[i]) \oplus C[i-1]$.

It is very important to note that CBC mode with a zero IV is *not* secure. To achieve even IND-CPA security we need to use a random initialisation vector (IV). CBC mode with random IVs

was proven secure by Bellare et al. [2]. The scheme that we present will prepend a random block to the plaintext before the CBC call to achieve security. This random block will effectively replace the zero IV of the API encrypt call to make a random IV. The choice of this random block is internal to the scheme and so the adversary should/will not have control over it (this fact is crucial to the schemes security). The random block will either be generated by the HSM or will be prepended to (or xored with) the first plaintext block prior to being called to the HSM.

Padding scheme When working with arbitrary length messages we need to pad the message before sending it to the blockcipher/CBC mode. Let $\mathsf{pad} : \{0,1\}^* \to \{0,1\}^{ln}$ be the padding function which pads the message to a multiple of the blocksize. Let $\mathsf{dpad} : \{0,1\}^{ln} \to \{0,1\}^{<ln} \cup \{\perp\}$ be the associated depadding function which depads the message, if the message is invalid it returns the symbol $\perp$. The padding scheme used with the Managed Encryption Format is PKCS#7 padding [11] (add p bytes each of value p). Note that in our analysis we assume that uniform error reported is used so that no padding oracle attacks exist. This is the case in the implementation of the scheme in all known deployed instantiations.

2.1 Security Models

An authenticated encryption scheme is secure if it achieves both the IND-CPA and INT-CTXT notions of security [16, 3]. As padding should only pad to the next block boundary and is not variable in length (i.e. we cannot have multiple blocks of padding) we do not consider length-hiding to be a security goal [14]. In implementations of the scheme we stress that uniform error reporting *must* be used. This will be vital for the scheme's security otherwise a padding oracle attack similar to that against SSL/TLS by Canvel et al. [7] may be possible. As a result our analysis we only considers one error type, $\perp$. Let $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be an encryption scheme and $\mathcal{A}$ be an adversary.

$\underline{\mathbf{IND\text{-}CPA}^{\mathcal{A}}(\Pi)}$	$\underline{\mathbf{Enc}(A, M_0, M_1)}$	
$K \leftarrow \mathsf{KeyGen}$	$C_0 \leftarrow \mathsf{Encrypt}(K, A, M_0)$	
$b \xleftarrow{r} \{0,1\}$	$C_1 \leftarrow \mathsf{Encrypt}(K, A, M_1)$	
$b' \leftarrow \mathcal{A}^{\mathbf{Enc}}$	$\mathcal{C} \xleftarrow{\cup} C_b$	
return $(b' = b)$	**return** C_b	

$\underline{\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi)}$	$\underline{\mathbf{Enc}(A, M)}$	$\underline{\mathbf{Test}(A^*, C^*)}$
$K \leftarrow \mathsf{KeyGen}$	$C \leftarrow \mathsf{Encrypt}(K, A, M)$	$M^* \leftarrow \mathsf{Decrypt}(K, A^*, C^*)$
$\mathsf{win} \leftarrow \mathsf{false}$	$\mathcal{C} \xleftarrow{\cup} (A, C)$	**if** $M^* \neq \perp$ and $(A^*, C^*) \notin \mathcal{C}$ **then**
$(A^*, C^*) \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Test}}$	**return** C	$\mathsf{win} \leftarrow \mathsf{true}$
return win		**return** $(M^* \neq \perp)$

Fig. 1. The IND-CPA and INT-CTXT experiments

The security experiment for IND-CPA is found in Figure 1. Note, that the encryption scheme semantics are defined to encrypt messages M as well as (possibly public) associated data A. We define the *advantage* of an adversary $\mathcal{A}$ against the IND-CPA security of Π as:

$$\mathbf{Adv}^{\mathrm{ind-cpa}}_{\Pi}(\mathcal{A}) = 2\Pr[\mathbf{IND\text{-}CPA}^{\mathcal{A}}(\Pi) \Rightarrow \mathsf{true}] - 1,$$

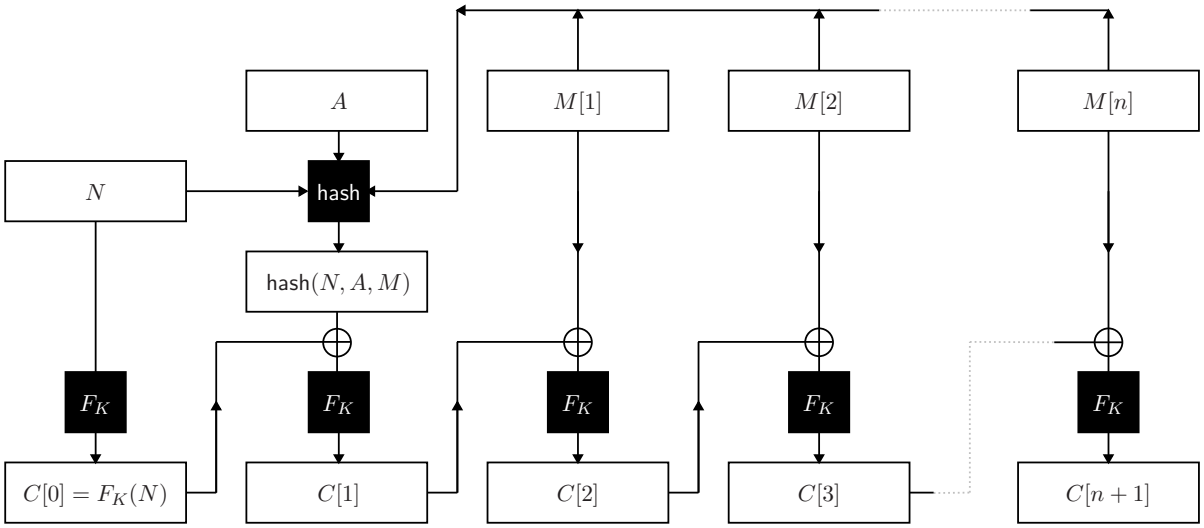
We say that the scheme Π is IND-CPA secure if for all adversaries $\mathcal{A}$ the advantage $\mathbf{Adv}_{\Pi}^{\mathrm{ind-cpa}}(\mathcal{A})$ is "small", where the adversary $\mathcal{A}$ makes q_e queries to the encryption oracle $\mathbf{Enc}(A, M_0, M_1)$, totaling at most μ_e bits in each of the left M_0 and right M_1 inputs.

The security experiment for INT-CTXT, is found in Figure 1. We define the *advantage* of an adversary $\mathcal{A}$ against the INT-CTXT security of Π as:

$$\mathbf{Adv}_{\Pi}^{\mathrm{int-ctxt}}(\mathcal{A}) = \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi) \Rightarrow \mathsf{true}]$$

We say that Π is INT-CTXT secure if for all adversaries $\mathcal{A}$ the advantage $\mathbf{Adv}_{\Pi}^{\mathrm{int-ctxt}}(\mathcal{A})$ is "small", where $\mathcal{A}$ makes q_e queries to the encryption oracle $\mathbf{Enc}(A, M)$, totaling at most μ_e bits in the M input and q_t queries to the test oracle $\mathbf{Test}(A, C)$ totaling at most μ_t of ciphertext bits.

3 Description of the Scheme



KeyGen(k)	Encrypt(K, A, M)	Decrypt(K, A, C)
$K \xleftarrow{r} \{0,1\}^k$	$N \xleftarrow{r} \{0,1\}^l$	$N \Vert h \Vert M' \leftarrow$ D-CBC$^0[F](K, C)$
return K	$h \leftarrow \mathsf{hash}(N, A, M)$	$M \leftarrow \mathsf{dpad}(M')$
	$C \leftarrow$ E-CBC$^0[F](K, N \Vert h \Vert \mathsf{pad}(M))$	**if** $M \neq \perp$ **then**
	return C	$\overline{h} \leftarrow \mathsf{hash}(N, A, M)$
		if $\overline{h} \neq h$ **then** $M = \perp$
		return M

Fig. 2. Managed Encryption Format $\Pi[F] = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

The Managed Encryption Format combines CBC mode encryption and a hash function in a MAC-then-Encrypt style configuration. We shall denote the Manage Encryption Format scheme by $\Pi[F]$. The scheme $\Pi[F]$ consists of three algorithms: key-generation KeyGen, encryption Encrypt and decryption Decrypt. The function F signifies the underlying blockcipher (i.e. the permutation family). We use the following notation: K is a key, N denotes a random IV, A is the header

(associated data which is always the same fixed length), M is a (unpadded) message and C is a ciphertext.

To encrypt a message M with associated data A, a random N is first chosen. Then a hash is calculated over N, A, M. Following this CBC encryption (using the block cipher F_K and zero IV) is performed on the (padded) message M, prepended with N and the hash value. Effectively, N ensures that we will have a random IV despite our API crypto call being to CBC mode with a zero IV. Decryption is the obvious inverse operation of encryption. The diagram and description in Figure 2 are given for aid of analysis.

4 Links With Prior Constructions

4.1 Analysis of the Underlying Message Authentication Code

If we look at the underlying MAC we see that it is of the Wegman-Carter style [18] and is particularly similar to VMAC [9, 10]. The VMAC algorithm uses a keyed hash function hash and a prf F. Tags are constructed on a message M together with a nonce N and keys K_1 and K_2 as follows:

$$\tau = \mathsf{hash}_{K_1}(M) + F_{K_2}(N)$$

Notice that when $K_1 = N$ this is almost exactly the MAC we have in the above mode of operation, i.e.

$$\tau = \mathsf{hash}_N(M) \oplus F_K(N)$$

In VMAC we return the nonce N, message M and tag τ but in our MAC we cannot return N since this would break the schemes security instead we must return $F_K(N)$, message M and tag τ. Effectively this means we simply return the message M and $\mathsf{hash}_N(M)$. We now require unforgeability properties from $\mathsf{hash}_N(M)$ but since key N is simply a secret-prefix of the hash function input this falls to extension attacks when used with an iterated hash function [15]. The underlying MAC is therefore not secure on its own.

4.2 Encryption with Redundancy

Bellare and Rogaway [4] and An and Bellare [1] have previously study the problem of how to achieve a secure AE scheme by appending some redundancy to the data before encryption. We shall review the results of An and Bellare [1] here and discuss how to relate these to our work.

A redundancy function may simple take the form of a hash function, as in our scheme. An and Bellare consider two types of redundancy function; one with a secret key and the other where any keying material is public. Perhaps the most important result that they show from our perspective, is that an IND-CPA scheme when combined with either a secret or a public redundancy function is not secure in general. Further to this, they describe an attack on the generic construction which combines CBC mode with a public redundancy function.

Despite this, An and Bellare are able to provide a construction for an encryption scheme which when combined with a secret key redundancy function would achieve INT-CTXT [1, Theorem 6.5]. This construction is called Nested CBC or NCBC. The encryption procedure followed for this scheme is to proceed with CBC encryption as normal until the last block. When encrypting the last block a different key shall be used for the block cipher. This means that NCBC requires two

keys and therefore, in the setting of HSMs, would require two key unwrap operations. As a result this construction would not meet the single key requirement of our setting.

The Managed Encryption Format can be viewed within the context of the Encryption with Redundancy paradigm with one main difference. Due to the encryption algorithm's selection of a new N upon each encryption call, this effectively means we are choosing a new random "key" for each hash function call. As a result we can view the Managed Encryption Format as an encryption scheme with secret redundancy, where the redundancy function is chosen anew upon each encryption call. Looking at a general construction of any IND-CPA secure encryption scheme and one time redundancy functions, we would still not necessarily achieve INT-CTXT security. An IND-CPA secure encryption scheme can be constructed, as in the attack presented by An and Bellare in the normal setting [1, Theorem 5.1], that when combined with one time secret redundancy functions, would not achieve INT-CTXT security. Despite this, in the next section we show that when we use a construction based on CBC mode (as in the Managed Encryption Format) we will obtain a secure AE scheme.

5 Security Analysis

In this section we shall prove that the Managed Encryption Format does achieve both IND-CPA security and INT-CTXT security and is therefore a secure authenticated encryption scheme. Note that to simplify our analysis we do not formally discuss the padding which is added to messages. This will not affect our security analysis since uniform-error reporting is used. We will therefore assume that all messages are already padded and omit the padding procedure from our proofs. In practice however the scheme could fall to a padding oracle attack of the style in [7] if uniform-error reporting is not present. Our proof is in the random oracle model (although this is only necessary for the proof of INT-CTXT). We denote by q_h the number of queries the adversary makes to the random oracle (not including those made through encryption, decryption or test queries).

Theorem 1. [IND-CPA] *Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family. Let $\Pi[F]$ be the encryption scheme for the Managed Encryption Format using the permutation family F. Let $\mathcal{A}$ be an adversary against the IND-CPA security which runs in time t; making q_e encryption queries totalling at most μ_e bits. Then there exists an adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\mathrm{ind-cpa}}_{\Pi[F]}(\mathcal{A}) \leq 2\mathbf{Adv}^{\mathrm{prp}}_{F}(\mathcal{B}) + \frac{q_f^2}{2^l} + \frac{1}{2^l}\left(\left(\frac{\mu_e}{l} + 2q_e\right)^2 - \left(\frac{\mu_e}{l} + 2q_e\right)\right)$$

where $\mathcal{B}$ runs in time $t + O(\mu_e)$ asking at most $q_f = \frac{\mu_e}{l} + 2q_e$ queries.

Proof. This can be proven by extension to the existing proof of security for CBC mode by Bellare et al. [2]. Since N is chosen uniformly at random by the encryption algorithm this gives us the necessary randomness for the existing CBC mode proof to still hold. Our proof follows a series of game hops. The different encryption algorithms used in each hop are shown in Figure 3.

Let Game0 be the normal IND-CPA game where $\mathcal{A}$ has access to a left-or-right oracle that uses algorithm Encrypt0(K, A, M) to encrypt M_b.

Let Game1 be the same as Game0 but replace the encryption algorithm Encrypt0(K, A, M) used by the left-or-right oracle with Encrypt1(K, A, M). In Encrypt1(K, A, M) the value of N is now returned with the ciphertext. Knowledge of N allows the adversary to recalculate any hash

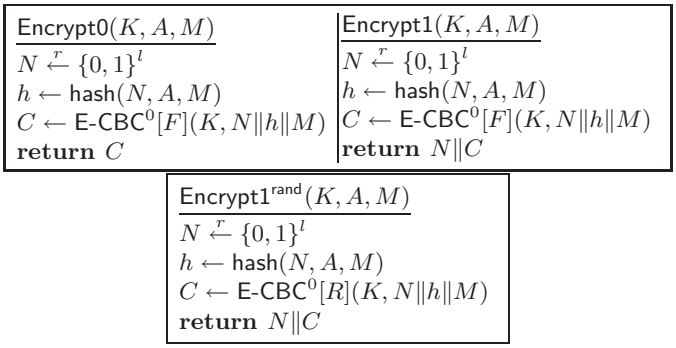


Fig. 3. Encryption Algorithm Hops in Proof of Lemma 1

values $\mathsf{hash}(N, A, M)$. This means that in the left-or-right indistinguishability game the adversary now has retrospective knowledge of his query, i.e. his left-or-right encryption is effectively $(0^l \| \mathsf{hash}(N, A, M_0) \| M_0, 0^l \| \mathsf{hash}(N, A, M_1) \| M_1)$, which will be encrypted by CBC mode with random IV N. Since N is chosen at random for each call, giving the adversary knowledge of a previously used N will not allow the adversary to predict any future N'. We therefore have that

$$\Pr[\mathsf{Game1} \Rightarrow \mathsf{true}] = \Pr[\mathsf{Game0} \Rightarrow \mathsf{true}].$$

Now we effectively have normal CBC encryption with a random IV given by N, plus an encryption query of the form $0^l \| h \| M$ (note that in Figure 3 we have already xored on the IV N to the first block, as the internal algorithm called is CBC mode with zero-IV).

It therefore remains to analyse $\mathcal{A}$'s success probability in $\mathsf{Game1}$. This can be proven by extension to the existing proof of security for CBC mode by Bellare et al. [2]. Note that we cannot perform a direct reduction because prior knowledge of N is necessary to calculate the hash. We begin the proof of security by switching to consider F as a random function. Let $\mathsf{Game1}^{\mathsf{rand}}$ be exactly the same as $\mathsf{Game1}$ but we now replace F_K with a function drawn uniformly at random from the set of all functions mapping l-bit strings to $l' = l$-bit strings (i.e. $f \xleftarrow{r} \mathrm{Rand} = R$). We can then construct a distinguisher $\mathcal{B}$ as in [2] such that:

$$\Pr[\mathsf{Game1} \Rightarrow \mathsf{true}] - \Pr[\mathsf{Game1}^{\mathsf{rand}} \Rightarrow \mathsf{true}] \leq 2\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{B})$$

We shall later use Lemma 1 to consider the prp advantage.

We now examine $\Pr[\mathsf{Game1}^{\mathsf{rand}} \Rightarrow \mathsf{true}]$. $\mathsf{Game1}$ proceeds as in the original proof for CBC mode by Bellare et al. [2], the encryption algorithm randomly chooses the IV N, and then proceeds with CBC encryption. Bellare et al.'s proof shall therefore remain almost as is and we just give a brief summary here. Note now that we simply treat the hash h like an additional plaintext block.

We denote the left and right encryption queries as follows:

Let $\tilde{M}^L = 0^l \| \mathsf{hash}(N, A, M_0) \| M_0$ denote the left query and let $\tilde{M}^R = 0^l \| \mathsf{hash}(N, A, M_1) \| M_1$ denote the right query. We also let $C_j[k]$ denote the k-th block of the j-th ciphertext, $\tilde{M}_j^L[k]$ denote the k-th block of the j-th left query and $\tilde{M}_j^R[k]$ denote the k-th block of the j-th right query. For a fixed adversary $\mathcal{A}$ we define the event $\mathsf{D}_{i,u}$, where $i \in [q_e]$ and $u \in [n_i + 2]$, to be when the following two events occur:

$$C_j[k-1] \oplus \tilde{M}_j^L[k] \neq C_{j'}[k'-1] \oplus \tilde{M}_{j'}^L[k']$$
$$C_j[k-1] \oplus \tilde{M}_j^R[k] \neq C_{j'}[k'-1] \oplus \tilde{M}_{j'}^R[k']$$

for all $(j,k),(j',k') \in \{(j,k) : j \in [q_e] \text{ and } k \in [n_j+2]\}$ satisfying $(j',k') \prec (j,k) \preceq (i,u)$, where n_j denotes the number of blocks in the j-th encryption query, note the addition of 2 here due to the extra all-zero block and hash block. Here $\prec$ denotes an ordering on the blocks queried to the encryption oracle. With $(j',k') \prec (j,k)$ implying that the k'-th block of the j'-th ciphertext was queried before the k-th block of the j-th ciphertext.

We now wish to study the probability of the event $\overline{\mathsf{D}} = \overline{\mathsf{D}_{q,n_q}}$, i.e. that a collision occurs at some point in the output ciphertexts. Bellare et al. prove that $\Pr[\overline{\mathsf{D}}|b=0] = \Pr[\overline{\mathsf{D}}|b=1]$, i.e. this probability is independent of whether we are observing left queries or right queries, the analysis is therefore the same for both $b=0$ and $b=1$. This probability is then denoted $p = \Pr[\overline{\mathsf{D}}|b=0] = \Pr[\overline{\mathsf{D}}|b=1]$ and was proved to be bounded as follows.

$$\begin{aligned} p &= \sum_{i=1}^{q_e}\sum_{j=1}^{n_i} \Pr[\overline{\mathsf{D}_{i,u}}|\mathsf{D}_{i,u-1}] \\ &\leq \frac{1}{2^l}\left(\left(\frac{\mu_e}{l}+2q_e\right)^2 - \left(\frac{\mu_e}{l}+2q_e\right)\right) \end{aligned}$$

Note that we adjust slightly from the original bound since we must account for the additional plaintext blocks caused by the hash and the initial all-zero block.

Combining the above and following similar arguments to the original CBC proof by Bellare et al. (along with Lemma 1) we obtain the following:

$$\begin{aligned} \mathbf{Adv}^{\mathrm{ind-cpa}}_{\Pi[F]}(\mathcal{A}) &= \Pr[\mathsf{Game0} \Rightarrow \mathsf{true}] \\ &\leq \Pr[\mathsf{Game1} \Rightarrow \mathsf{true}] \\ &\leq 2\mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{B}) + \Pr[\mathsf{Game1}^{\mathsf{rand}} \Rightarrow \mathsf{true}] \\ &\leq 2\mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{B}) + \frac{1}{2^l}\left(\left(\frac{\mu_e}{l}+2q_e\right)^2 - \left(\frac{\mu_e}{l}+2q_e\right)\right) \\ &\leq 2\mathbf{Adv}^{\mathrm{prp}}_F(\mathcal{B}) + \frac{q_f^2}{2^l} + \frac{1}{2^l}\left(\left(\frac{\mu_e}{l}+2q_e\right)^2 - \left(\frac{\mu_e}{l}+2q_e\right)\right). \end{aligned}$$

Theorem 2. [INT-CTXT] *Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family. Let $\Pi[F]$ be the encryption scheme for the Managed Encryption Format using the permutation family F. Let $\mathcal{A}$ be an adversary against the INT-CTXT security which runs in time t; making q_e encryption queries totalling at most μ_e bits, q_t test queries totalling at most μ_t bits and q_h random oracle queries. Then there exists an adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\mathrm{int-ctxt}}_{\Pi[F]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{sprp}}_F(\mathcal{B}) + \frac{q_t}{2^l} + \frac{q_h\mu_e}{l2^l}$$

where $\mathcal{B}$ makes $q_f = \frac{\mu_e}{l} + 2q_e + \frac{\mu_t}{l}$ queries and runs in time $t + O(\mu_e + \mu_t)$.

Proof. We assume we have an adversary $\mathcal{A}$ against INT-CTXT and we use it to construct an adversary $\mathcal{B}$ against SPRP. This is done as follows:

- When $\mathcal{A}$ makes an encryption query A, M, the algorithm $\mathcal{B}$ chooses N at random and calls the random oracle on N, A, M. Following this $\mathcal{B}$ makes calls to its π oracle for the appropriate CBC encryption (making a total of $\mu_e/l + 2q_e$ queries (where $2q_e$ accounts for the queries $\pi(N)$ and $\pi(h \oplus \pi(N))$). Finally $\mathcal{B}$ returns the ciphertext to $\mathcal{A}$.

– When $\mathcal{A}$ makes a test query, the algorithm $\mathcal{B}$ calls its π^{-1} oracle for the appropriate CBC decryption (μ_t/l queries). Then $\mathcal{B}$ verifies whether M is new or not. Next $\mathcal{B}$ calls the random oracle to verify the hash. The result of the whole verification is returned to $\mathcal{A}$.
– The random oracle maintains a list $\mathcal{H}$ of all queries.
– If $\mathcal{A}$ outputs a successfully forgery then $\mathcal{B}$ guesses it has access to the real permutation.
– If $\mathcal{A}$ is unsuccessful then $\mathcal{B}$ guesses it has access to the random permutation.

The following inequality then holds (where Perm is the set of all l-bit permutations):

$$\begin{aligned}\mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) &\geq \Pr[\mathcal{B}^{\pi,\pi^{-1}} \Rightarrow 1 | \pi \xleftarrow{r} F] - \Pr[\mathcal{B}^{\pi,\pi^{-1}} \Rightarrow 1 | \pi \xleftarrow{r} \mathrm{Perm}].\\ &= \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[F]) \Rightarrow \mathsf{true}]\\ &\qquad - \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}]) \Rightarrow \mathsf{true}]\\ &= \mathbf{Adv}^{\mathrm{int-ctxt}}_{\Pi[F]}(\mathcal{A}) - \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}]) \Rightarrow \mathsf{true}].\end{aligned}$$

If $\mathsf{Decrypt}(K, A^*, C^*) = M^*$ then the probability that this message is a valid forgery is bounded by the probability that the hash verifies on M^*. The decrypted hash value to be verified will be given by $h^* = \pi^{-1}(C^*[1]) \oplus C^*[0] = \pi^{-1}(C^*[1]) \oplus \pi(N^*)$. We therefore obtain the following bound:

$$\Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}]) \Rightarrow \mathsf{true}] \leq \Pr[\mathsf{hash}(N^*, A^*, M^*) = h^* | \pi \xleftarrow{r} \mathrm{Perm}]$$

We shall now consider this probability in two parts. First we study the case where N^*, A^*, M^* was never a random oracle query, i.e. $(N^*, A^*, M^*, h^*) \notin \mathcal{H}$. The actual hash value will now be chosen at random when the random oracle is called upon decryption. The probability that this hash collides with the decrypted value $h^* = \pi^{-1}(C^*[1]) \oplus C^*[0]$ is $\frac{1}{2^l}$ (for a single test query). We therefore obtain the following bound:

$$\Pr[(\mathsf{hash}(N^*, A^*, M^*) = h^*) \wedge ((N^*, A^*, M^*, h^*) \notin \mathcal{H}) | \pi \xleftarrow{r} \mathrm{Perm}] \leq \frac{q_t}{2^l}$$

Next consider the case when N^*, A^*, M^* has been previously called to the random oracle. We shall prove the following bound:

$$\Pr[(\mathsf{hash}(N^*, A^*, M^*) = h^*) \wedge ((N^*, A^*, M^*, h^*) \in \mathcal{H}) | \pi \xleftarrow{r} \mathrm{Perm}] \leq \frac{q_h \mu_e}{l 2^l}.$$

Since $(N^*, A^*, M^*, h^*) \in \mathcal{H}$, the query was already a call to hash but it cannot have been made by a previous encryption query. This is because a hash called previously by $\mathsf{Encrypt}$ on (N^*, A^*, M^*) would imply that C^* was already output by $\mathsf{Encrypt}$, breaking the restrictions of the INT-CTXT game. The query must have therefore been made by a separate call to hash.

Consider the query N^*, A^*, M^* which $\mathcal{A}$ makes to hash (receiving h^*). $\mathcal{A}$ can choose N^* such that it has seen its encryption $\pi(N^*)$ in a previous encryption query, i.e. $N^* = C_i[j] \oplus M_i[j]$ for some $i \in [q_e]$ and $j \in [n_i]$, where q_e is the total number of encryption queries and n_i is the number of blocks in query i. (Note here that it may look odd that $C_i[j]$ and $M_i[j]$ both have the same block index j but we stress this is still a normal CBC encryption step. The apparent difference is due to the hash h shifting the block indices of M_i; to see this more easily we refer the reader to Figure 2.)

To forge a valid ciphertext $\mathcal{A}$ must first ensure that the first ciphertext block is correct, i.e. $C^*[1] = \pi(h^* \oplus \pi(N^*))$. Since π is a random permutation $\mathcal{A}$ will choose $C^*[1]$ correctly only if it has seen $\pi(h^* \oplus \pi(N^*))$ before, i.e. there exists a call to π where $C_i[j] \oplus M_i[j] = h^* \oplus \pi(N^*)$ for some $i \in [q_e]$ and $j \in [n_i]$.

$\underline{\mathsf{KeyGen}^c(k)}$	$\underline{\mathsf{Encrypt}^c(K, A, M)}$	$\underline{\mathsf{Decrypt}^c(K, A, C)}$
$K \xleftarrow{r} \{0,1\}^k$	$\overline{h} \leftarrow \mathsf{hash}(ctr, A, M)$	$ctr \| h \| M' \leftarrow \mathsf{D\text{-}CBC}^0[F](K, C)$
$ctr \xleftarrow{r} \{0,1\}^l$	$C \leftarrow \mathsf{E\text{-}CBC}^0[F](K, ctr \| h \| \mathsf{pad}(M))$	$M \leftarrow \mathsf{dpad}(M')$
return K	$ctr \leftarrow ctr + 1$	**if** $M \neq \perp$ **then**
	return C	$\overline{h} \leftarrow \mathsf{hash}(ctr, A, M)$
		if $\overline{h} \neq h$ **then** $M = \perp$
		return M

Fig. 4. Man Enc Format with Counter, $\Pi^c[F] = (\mathsf{KeyGen}^c, \mathsf{Encrypt}^c, \mathsf{Decrypt}^c)$

If $\mathcal{A}$ makes q_e encryption queries totaling μ_e bits, then the probability that h^* is generated by the random oracle such that $C_i[j] \oplus M_i[j]$ is queried to π for some $i \in [q_e]$ and $j \in [n_i]$, is $\frac{\mu_e/l}{2^l}$. The above probability bound then follows.

Combining all of the above we obtain our result.

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{int-ctxt}}_{\Pi[F]}(\mathcal{A}) &\leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}) \Rightarrow \mathsf{true}] \\
&\leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \Pr[\mathsf{hash}(N^*, A^*, M^*) = h^* | \pi \xleftarrow{r} \mathrm{Perm}] \\
&\leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \frac{q_t}{2^l} + \frac{q_h \mu_e}{l \cdot 2^l}.
\end{aligned}$$

6 Further Discussions

6.1 Using a Counter

We also introduce a stateful version of the Managed Encryption Format as defined in Figure 4. Here we now replace the value N with a counter ctr. It is possible to prove CBC mode is IND-CPA secure with a ctr in this way [2], when a maximum of 2^l encryptions are permitted. Security is ensured by the fact that a collision $ctr^* = C_i[j-2] \oplus M_i[j]$ which reveals the value of $F_K(ctr^*)$ for some future ctr^*, occurs with small probability. Furthermore, if ctr is initialised as a random string then an adversary must first determine the current version of ctr in order to mount an attack based on the above collision. We omit further details of the proof but it can be seen that we will be able to extend this to prove that the stateful version of the Managed Encryption Format would be a secure AE scheme. The application of such a scheme to the setting of HSMs would of course depend on an HSM's ability to maintain state.

6.2 Parameter Choices

We note that the security bounds in our Theorems come with error terms of the order of

$$\frac{q_f^2}{2^l} \quad \text{and} \quad \frac{q_f \cdot q_h}{2^l}.$$

We recall that l is the block length of the underlying block cipher, q_f is the number of queries to the underlying PRP and q_h is the number of hash function queries. These bounds mean that if we wish to guarantee security against the probability of an adversary breaking the scheme not exceeding 2^{-40} (say), and we assume breaking the underlying PRP F is hard, then the number of queries made to the hash function and managed encryption scheme needs to be bounded.

If using DES as the underlying block cipher, where $l = 64$, this means that we need to ensure that $q_f \ll 2^{12} = 4096$. Thus use with DES can be deemed to be insecure, unless underlying block cipher keys are updated relatively quickly. When used with a block cipher such as AES, where $l = 128$, the number of queries to the underlying block cipher needs to be bounded by 2^{44} if one wishes to make the probability of breaking the scheme be bounded by 2^{-40}; whilst the product of the number of block cipher calls multiplied by the number of hash function calls needs to be bounded by 2^{88} to obtain a similar probability bound. Thus the scheme can be considered secure in practice when instantiated with AES, but needs to be used with care when instantiated with DES.

7 Conclusion

We have presented a new provably secure mode of operation for authenticated encryption. This mode has been designed for use in environments where keys are protect by an HSM but the API offers limited cryptographic functions. The scheme is built around an HSM which provides an API call to CBC mode with zero IV. To minimise expensive HSM calls the scheme uses only one key and hence makes a single call to the HSM.

8 Acknowledgements

The first author thanks his colleagues at Cryptomathic and the Computer Laboratory in Cambridge for useful conversations during the design process. The second author thanks his employers Barclays PLC for their support The third and fourth author were supported by Prof Smart's ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO. The third author was also partially supported by a Royal Society Wolfson Merit Award.

The work in this paper arose from a discussion held during eCrypt-2 sponsored workshop "Is Cryptographic Theory Practically Relevant?" held at the Newton Institute in January 2012. The authors thank eCrypt-2 and the Newton Institute for hosting this workshop. The authors also thank Kenny Paterson and Jean Paul Degabriele for helpful discussions.

References

1. Jee Hea An and Mihir Bellare. Does encryption with redundancy provide authenticity? In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 512–528. Springer, 2001.
2. Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.
3. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Okamoto [13], pages 531–545.
4. Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Okamoto [13], pages 317–330.
5. Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX mode of operation. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 389–407. Springer, 2004.
6. Mike Bond. Attacks on cryptoprocessor transaction sets. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 220–234. Springer, 2001.
7. Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a ssl/tls channel. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 583–599. Springer, 2003.
8. Riccardo Focardi, Flaminia L. Luccio, and Graham Steel. An introduction to security api analysis. In Alessandro Aldini and Roberto Gorrieri, editors, *FOSAD*, volume 6858 of *Lecture Notes in Computer Science*, pages 35–65. Springer, 2011.

9. Ted Krovetz. Message authentication on 64-bit architectures. In Eli Biham and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 327–341. Springer, 2006.
10. Ted Krovetz and Wei Dai. Vmac: Message authentication code using universal hashing. CFRG Working Group INTERNET-DRAFT, April 2007. `http://www.fastcrypto.org/vmac/draft-krovetz-vmac-01.txt`.
11. RSA Laboratories. PKCS #7: Cryptographic message syntax standard, November 1993. Version 1.5.
12. David A. McGrew and John Viega. The security and performance of the galois/counter mode (gcm) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.
13. Tatsuaki Okamoto, editor. *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*. Springer, 2000.
14. Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the tls record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 372–389. Springer, 2011.
15. Bart Preneel and Paul C. van Oorschot. Mdx-mac and building fast macs from hash functions. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 1995.
16. Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 98–107. ACM, 2002.
17. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM Conference on Computer and Communications Security*, pages 196–205. ACM, 2001.
18. Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.
19. D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610 (Informational), September 2003.