CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions

J. Black * P. Rogaway †

December 3, 2003

Abstract

We suggest some simple variants of the CBC MAC that enable the efficient authentication of arbitrary-length messages. Our constructions use three keys, $K1$, $K2$, $K3$, to avoid unnecessary padding and MAC any message $M \in \{0,1\}^*$ using $\max\{1, \lceil |M|/n \rceil\}$ applications of the underlying n-bit block cipher. Our favorite construction, XCBC, works like this: if $|M|$ is a positive multiple of n then XOR the n-bit key $K2$ with the last block of M and compute the CBC MAC keyed with $K1$; otherwise, extend M's length to the next multiple of n by appending minimal 10^ℓ padding ($\ell \geq 0$), XOR the n-bit key $K3$ with the last block of the padded message, and compute the CBC MAC keyed with $K1$. We prove the security of this and other constructions, giving concrete bounds on an adversary's inability to forge in terms of his inability to distinguish the block cipher from a random permutation. Our analysis exploits new ideas which simplify proofs compared to prior work.

Keywords: CBC MAC, message authentication codes, modes of operation, provable security, standards.

1 Introduction

This paper describes some simple variants of CBC MAC. Unlike the basic CBC MAC, our algorithms correctly and efficiently handle messages of any bit length. In addition to our schemes, we introduce new techniques to prove them secure. Our proofs are much simpler than prior work. We begin with some background.

The cbc mac. The CBC MAC [7, 9] is the simplest and most well-known way to make a message authentication code (MAC) out of a block cipher. Let's recall how it works. Let $\Sigma = \{0,1\}$ and let $E\colon \mathcal{K} \times \Sigma^n \to \Sigma^n$ be a block cipher. The key space for the CBC MAC is the key space $\mathcal{K}$ for E and the message space for the CBC MAC is the set $(\Sigma^n)^+$ of all binary strings whose length is a positive multiple of n. So let $M = M_1 \cdots M_m$ be a string that we want to MAC, where $|M_1| = \cdots = |M_m| = n$, and let $K \in \mathcal{K}$ be the key we want to use to MAC this string. Then $\mathrm{CBC}_{E_K}(M)$, the CBC MAC of M under key K, is the value C_m where $C_i = E_K(M_i \oplus C_{i-1})$ for $i = 1, \ldots, m$ and $C_0 = 0^n$.

Bellare, Kilian, and Rogaway proved the security of the CBC MAC in the sense of reduction-based cryptography [1]. Their proof depends on the assumption that it is only messages of one fixed length, mn bits, that are being MACed. Indeed when message lengths can vary the CBC MAC is *not* secure. This fact is well-known. As a simple example, notice that given the CBC MAC of a one-block message X, say $T = \mathrm{CBC}_{E_K}(X)$, the adversary immediately knows the CBC MAC for the two-block message $X \parallel (X \oplus T)$ since this is once again T.

Thus the CBC MAC (in the "raw" form that we have described) has two problems: it can't be used to MAC messages outside of $(\Sigma^n)^+$ and all messages must have the same fixed length.

* Department of Computer Science, 430 UCB, Boulder, Colorado 80309 USA. E-mail: jrblack@cs.colorado.edu WWW: www.cs.colorado.edu/∼jrblack/

† Department of Computer Science, University of California, Davis, California, 95616, USA; and Department of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand. E-mail: rogaway@cs.ucdavis.edu WWW: www.cs.ucdavis.edu/∼rogaway/

Construction	Domain	Block cipher calls	Block cipher keys	Key length
CBC	Σ^{nm}	$\lvert M\rvert/n$	1	k
EMAC	$(\Sigma^n)^+$	$1+\lvert M\rvert/n$	2	$2k$
EMAC*	Σ^*	$1+\lceil(\lvert M\rvert+1)/n\rceil$	2	$2k$
ECBC	Σ^*	$1+\lceil\lvert M\rvert/n\rceil$	3	$3k$
FCBC	Σ^*	$\lceil\lvert M\rvert/n\rceil$	3	$3k$
XCBC	Σ^*	$\lceil\lvert M\rvert/n\rceil$	1	$k+2n$

Figure 1: The CBC MAC and variants. Here M is the message to MAC and $E\colon \Sigma^k \times \Sigma^n \to \Sigma^n$ is a block cipher. The third column gives the number of applications of E, assuming $|M| > 0$. The fourth column is the number of keys used to key E. For CBC the domain is actually $(\Sigma^n)^+$ but the scheme is secure only on messages of some fixed length nm.

Dealing with variable message lengths: emac. When message lengths vary, the CBC MAC must be embellished. There have been several suggestions for doing this. The most elegant one we have seen is to encipher $\mathrm{CBC}_{E_{K1}}(M)$ using a new key, $K2$. That is, the domain is still $(\Sigma^n)^+$ but one defines EMAC (for Encrypted MAC) by $\mathrm{EMAC}_{E_{K1}\,E_{K2}}(M) = E_{K2}(\mathrm{CBC}_{E_{K1}}(M))$. This algorithm was developed for the RACE project [2]. It was analyzed by Petrank and Rackoff [13] who show, roughly said, that an adversary who obtains the MACs for messages that total σ blocks cannot forge with probability better than $2\sigma^2/2^n$.

Among the nice features of EMAC is that one need not know $|M|$ prior to processing the message M; the method is said to be *on-line*. All of our suggestions will retain this feature.

Our contributions. EMAC has a domain limited to $(\Sigma^n)^+$ and uses $1+|M|/n$ applications of the block cipher E. In this paper we refine EMAC in three ways: (1) we extend the domain to Σ^*; (2) we shave off one application of E; and (3) we avoid keying E by multiple keys. Of course we insist on retaining provable security (across all messages lengths).

We introduce three refinements to EMAC, which we call ECBC, FCBC, and XCBC. These algorithms are natural extensions of the CBC MAC. We would like to think that this is an asset. The point here is to strive for economy, in terms of both simplicity and efficiency.

Figure 1 summarizes the characteristics of the CBC MAC variants mentioned in this paper. The top three rows give known constructions (two of which we have now defined). The next three rows are our new constructions. Note that our last construction, XCBC, retains essentially all the efficiency characteristics of the CBC MAC, but extends the domain of correct operation to all of Σ^*. The cost to save one invocation of the block cipher and extend our domain to Σ^* is a slightly longer key.

For each of the new schemes we give a proof of security. Rather than adapt the rather complex proof of Petrank and Rackoff [13], or the even more complicated one of Bellare, Kilian, and Rogaway [1], we follow a new tack, viewing EMAC as an instance of the Carter-Wegman paradigm [5, 14]: with EMAC one is enciphering the output of an almost-universal hash-function family, this almost-universal hash-function family being the CBC MAC itself. Since it is not too hard to upper bound the collision probability of the CBC MAC (see Lemma 3), this approach leads to a simple proof for ECBC. We then use the security of ECBC to prove security for FCBC, and then we use the security of FCBC to prove security for XCBC. In passing from FCBC to XCBC we use a general lemma (Lemma 6) which says, in effect, that you can always replace a pair of random independent permutations $\pi_1(\cdot), \pi_2(\cdot)$ by a pair of functions $\pi(\cdot), \pi(\cdot \oplus K)$, where π is a random permutation and K is a random constant.

Subsequent work. Iwata and Kurosawa present a variant of XCBC that makes due with a single block-cipher key [10]. Their OMAC algorithm is identical to XCBC except that one selects $(K1, K2, K3) = (K, 2E_K(0^n), 4E_K(0^n))$, say, with the indicated multiplication being carried out in the finite field $\mathrm{GF}(2^n)$. We like this refinement, and see no significant drawback to it.

Standards. This work was motivated by the emergence of the Advanced Encryption Standard (AES) and the anticipated interest in updating old modes of operation that we hoped AES would engender. Our belief is that a modern MAC standard needs to clearly specify an algorithm that can correctly MAC any sequence

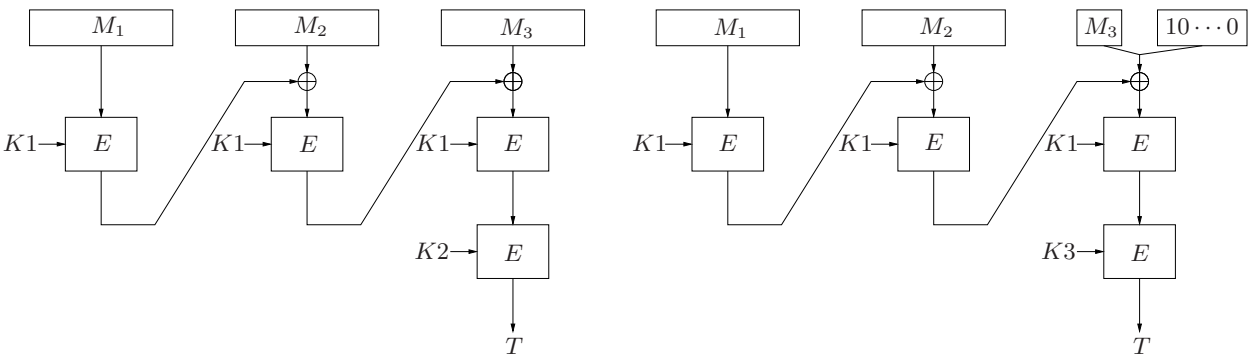

Figure 2: The ECBC construction using a block cipher $E\colon \mathcal{K} \times \Sigma^n \to \Sigma^n$. Three keys are used, $K1, K2, K3 \in \mathcal{K}$. On the left is the case where $|M|$ is a positive multiple of n, while on the right is the case where it isn't.

of bit strings. The methods here are simple, efficient, provably sound, and patent-free—good characteristics for a modern cryptographic standard.

We have been told that NIST intends to release a recommendation specifying the OMAC variant of XCBC [6]. The algorithm will be renamed CMAC.

Publication history. An earlier version of this paper appeared in CRYPTO '00 [4].

2 Schemes ECBC, FCBC, and XCBC

Arbitrary-length messages without obligatory padding: ecbc. We have defined $\mathrm{CBC}_{E_{K1}}(M)$, which is not secure across strings of varying lengths, and $\mathrm{EMAC}_{E_{K1}\,E_{K2}}(M) = E_{K2}(\mathrm{CBC}_{E_{K1}}(M))$, which is. But the domain of EMAC remains limited to $(\Sigma^n)^+$. What if we want to MAC a message whose length is *not* a multiple of n?

The simplest approach is to use obligatory 10^ℓ padding: always append a "1" bit and then the minimum number of "0" bits so as to make the length of the padded message a multiple of n, and then apply EMAC. We call this method EMAC*. Formally, $\mathrm{EMAC}^*_{E_{K1}\,E_{K2}}(M) = \mathrm{EMAC}_{E_{K1},E_{K2}}(M \parallel 10^{n-1-|M| \bmod n})$. This construction works fine. In fact, it is easy to see that this form of padding *always* works to extend the domain of a MAC from $(\Sigma^n)^+$ to Σ^*.

One unfortunate feature of EMAC* is this: if $|M|$ is already a positive multiple of n then we are appending an entire extra block of padding, seemingly "wasting" an application of E. People have worked hard to optimize new block ciphers, and it seems a shame to squander some of this efficiency with an unnecessary block-cipher call. Furthermore, there are settings where one needs to MAC short messages that are always or often a multiple of the block size. In such a case saving one block-cipher call can be a significant performance gain.

Our first new scheme lets us avoid padding when $|M|$ is a nonzero multiple of n. We simply make two cases: one for when $|M|$ is a positive multiple of n and one for when it isn't. In the first case we compute $\mathrm{EMAC}_{E_{K1}\,E_{K2}}(M)$. In the second case we append minimal 10^ℓ padding ($\ell \geq 0$) to make a padded message P whose length is divisible by n, and then we compute $\mathrm{EMAC}_{E_{K1}\,E_{K3}}(P)$. Notice the different second key—$K3$ instead of $K2$—in the case where we've added padding. Here, in full, is the algorithm. It is also shown in Figure 2.

Algorithm $\mathrm{ECBC}_{E_{K1}\,E_{K2}\,E_{K3}}(M)$
if $M \in (\Sigma^n)^+$
  then return $E_{K2}(\mathrm{CBC}_{E_{K1}}(M))$
  else return $E_{K3}(\mathrm{CBC}_{E_{K1}}(M \parallel 10^\ell))$ where $\ell = n - 1 - |M| \bmod n$

In Section 4 we prove that ECBC is secure. We actually show that it is a good pseudorandom function (PRF), not just a good MAC. The security of ECBC does not seem to directly follow from Petrank and

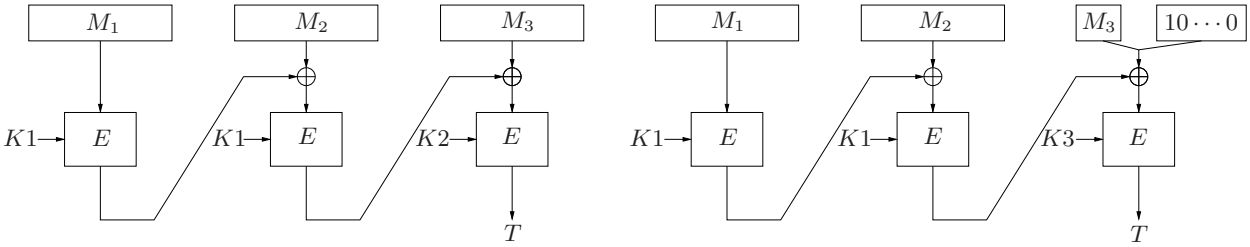

Figure 3: The FCBC construction with a block cipher $E\colon \mathcal{K} \times \Sigma^n \to \Sigma^n$. Three keys are used, $K1, K2, K3 \in \mathcal{K}$. On the left is the case where $|M|$ is a positive multiple of n, while on the right is the case where it isn't.

Rackoff's result [13]. At issue is the fact that there is a relationship between the key $(K1, K2)$ used to MAC messages in $(\Sigma^n)^+$ and the key $(K1, K3)$ used to MAC other messages.

Improving efficiency: fcbc. Ignoring the case of M being the empty string, with ECBC we are using $\lceil |M|/n \rceil + 1$ applications of the underlying block cipher. We now show how to get rid of the $+1$. We start off, as before, by padding M when it is outside $(\Sigma^n)^+$. Next we compute the CBC MAC using key $K1$ for all but the final block, and then use either key $K2$ or $K3$ for the final block. Which key we use depends on whether or not we added padding. The algorithm follows, and is also shown in Figure 3. In Section 5 we prove the security of this construction. Correctness will follow from the correctness of ECBC.

Algorithm $\mathrm{FCBC}_{E_{K1}\, E_{K2}\, E_{K3}}(M)$
if $M \in (\Sigma^n)^+$
  then $K \leftarrow K2$, and $P \leftarrow M$
  else $K \leftarrow K3$, and $P \leftarrow M \parallel 10^\ell$ where $\ell \leftarrow n - 1 - |M| \bmod n$
Let $P = P_1 \cdots P_m$ where $|P_1| = \cdots = |P_m| = n$
$C_0 \leftarrow 0^n$
for $i \leftarrow 1$ **to** $m - 1$ **do**
  $C_i \leftarrow E_{K1}(P_i \oplus C_{i-1})$
return $E_K(P_m \oplus C_{m-1})$

Avoiding multiple encryption keys: xcbc. Most block ciphers have a key-setup step, when the key is turned into subkeys. Taken together, the subkeys are usually larger than the original key, and computing them may be expensive. So keying the underlying block cipher with multiple keys, as is done in EMAC, ECBC, and FCBC, is not so desirable. It would be better to use the same key for all of the block-cipher invocations. The algorithm XCBC does this.

We again make two cases. If $M \in (\Sigma^n)^+$ we invoke CBC as usual, except that we XOR in an n-bit key, $K2$, before enciphering the last block. If $M \notin (\Sigma^n)^+$ then append minimal 10^ℓ padding ($\ell \geq 0$) and invoke CBC as usual, except we XOR in a different n-bit key, $K3$, before enciphering the last block. The algorithm follows; also see Figure 4. The proof of security can be found in Section 6.

Algorithm $\mathrm{XCBC}_{E_{K1}\, K2\, K3}(M)$
if $M \in (\Sigma^n)^+$
  then $K \leftarrow K2$, and $P \leftarrow M$
  else $K \leftarrow K3$, and $P \leftarrow M \parallel 10^\ell$ where $\ell \leftarrow n - 1 - |M| \bmod n$
Let $P = P_1 \cdots P_m$ where $|P_1| = \cdots = |P_m| = n$
$C_0 \leftarrow 0^n$
for $i \leftarrow 1$ **to** $m - 1$ **do**
  $C_i \leftarrow E_{K1}(P_i \oplus C_{i-1})$
return $E_{K1}(P_m \oplus C_{m-1} \oplus K)$

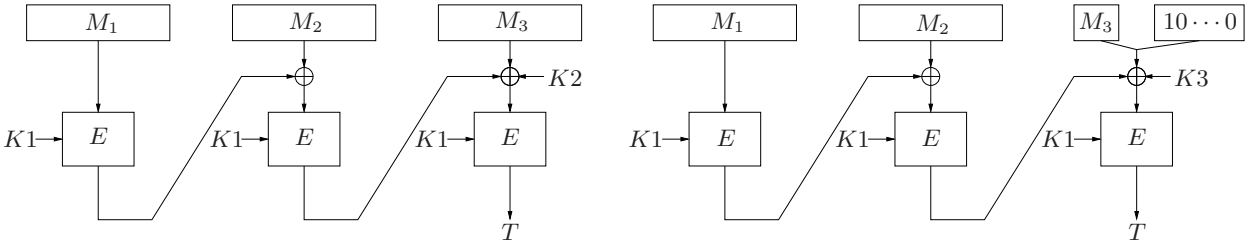

Figure 4: The XCBC construction with a block cipher $E\colon \mathcal{K}\times\Sigma^n \to \Sigma^n$. Here $K1 \in \mathcal{K}$ and $K2, K3 \in \Sigma^n$. On the left is the case where $|M|$ is a positive multiple of n, while on the right is the case where it isn't.

Comment. Note that the XCBC-variant that XORs the second key just *after* applying the final enciphering does not work. That is, insisting that $|M|$ is a nonzero multiple of the blocksize, we'd have that $\mathrm{MAC}_{\pi,K}(M) = \mathrm{CBC}_\pi(M) \oplus K$. For an attack, let the adversary ask for the MACs of three messages: the message $\mathbf{0} = 0^n$, the message $\mathbf{1} = 1^n$, and the message $\mathbf{1} \parallel \mathbf{0}$. As a result of these three queries the adversary gets tag $T_0 = \pi(\mathbf{0}) \oplus K$, tag $T_1 = \pi(\mathbf{1}) \oplus K$, and tag $T_2 = \pi(\pi(\mathbf{1})) \oplus K$. But now the adversary knows the correct tag for the (unqueried) message $\mathbf{0} \parallel (T_0 \oplus T_1)$, since this is just T_2: namely, $\mathrm{MAC}_{\pi,K}(\mathbf{0} \parallel (T_0 \oplus T_1)) = \pi(\pi(\mathbf{0}) \ \oplus\ (\pi(\mathbf{0}) \oplus K) \oplus (\pi(\mathbf{1}) \oplus K)) \ \oplus K = \pi(\pi(\mathbf{1})) \oplus K = T_2$. Thanks to Mihir Bellare for pointing out this attack.

Alternative notation. The subscripts to CBC, ECBC, FCBC, and XCBC have been written as E_K (for CBC) and $E_{K1}\, E_{K2}\, E_{K3}$ (for ECBC and FCBC) and $E_{K1}\, K2\, K3$ (for XCBC). It is important for the remainder of this paper to realize that, used as subscripts above, E_K and E_{K1} and E_{K2} and E_{K3} may be considered as *names* of functions—we could just as well have written chosen ρ and ρ_1 and ρ_2 and ρ_3, for example. Even though we had called E a block cipher, it was never necessary for E_K, E_{K1}, E_{K2}, or E_{K3} to be permutations—any function from Σ^n to Σ^n would be fine. So at this point, for any $\rho, \rho_1, \rho_2, \rho_3\colon \Sigma^n \to \Sigma^n$ and any $K2, K3 \in \Sigma^n$, we have defined the map $\mathrm{CBC}_\rho\colon (\Sigma^n)^+ \to \Sigma^n$ and maps from Σ^* to Σ^n of $\mathrm{ECBC}_{\rho_1\,\rho_2\,\rho_3}$ and $\mathrm{FCBC}_{\rho_1\,\rho_2\,\rho_3}$ and $\mathrm{XCBC}_{\rho_1\,K2\,K3}$. We will henceforth be using this less cumbersome notation.

3 Preliminaries

Basic notation. All strings are understood to be over the alphabet $\Sigma = \{0,1\}$. If X is a string then $\|X\|_n = \max\{\lceil |X|/n \rceil,\ 1\}$ is its length in n-bit blocks. By $X \xleftarrow{\$} \mathcal{X}$ we denote the experiment of choosing a random element from the set $\mathcal{X}$ and assigning it to X. By $\Pr[\text{Experiment}: \text{Event}]$ we denote the probability of the specified event after performing the specified experiment.

Function families. A *function family* is a collection of functions $F = \{f\colon \mathcal{M} \to \mathcal{N}\}$ where $\mathcal{M}, \mathcal{N} \subseteq \Sigma^*$ are sets of strings, along with an associated probability distribution. We speak of choosing a random element of F by writing $f \xleftarrow{\$} F$.

Let $\mathrm{Rand}(n)$ be the set of all functions from Σ^n to Σ^n, let $\mathrm{Perm}(n)$ be the set of all permutations on Σ^n, and let $\mathrm{Rand}(\mathcal{M}, n)$ be the set of all functions from $\mathcal{M}$ to Σ^n, where $\mathcal{M}$ is a set of strings. $\mathrm{Rand}(n)$, $\mathrm{Perm}(n)$, and $\mathrm{Rand}(\mathcal{M}, n)$ are function families, endowed with the uniform distribution in the natural way.

It is often the case that each function in a function family is named by a key K from a set of possible keys $\mathcal{K}$. The function family is then viewed as a map $F\colon \mathcal{K}\times\mathcal{M} \to \mathcal{N}$. We write $F_K(X)$ instead of $F(K, X)$. It is $\mathcal{K}$ that must now have an understood distribution, so that one can speak of choosing a random element K from $\mathcal{K}$. The two views of a function family are equivalent.

A block cipher is a function family $E\colon \mathcal{K}\times\Sigma^n \to \Sigma^n$. One normally requires that $E_K(\cdot)$ be a permutation for all $K \in \mathcal{K}$, but this restriction is never actually needed in this paper.

Collision probability. Fix $n > 0$ and let $H = \{h\colon \mathcal{M} \to \mathcal{N}\}$ be a function family. The *collision*

probability of H is the function

$$\mathbf{Coll}_H(m,\, m') \quad = \quad \max_{M,M' \in \mathcal{M},\, \|M\|_n = m,\, \|M'\|_n = m',\, M \neq M'} \Big\{ \Pr[h \xleftarrow{\$} H :\ h(M) = h(M')] \Big\} \ .$$

We define $\mathbf{Coll}_H(m,\, m')$ as 0 if there are no distinct M and M' in $\mathcal{M}$ where $\|M\|_n = m$ and $\|M'\|_n = m'$. The collision probability of a function family is a formalization and generalization of the idea of a hash-function family being almost-universal [5].

Making function families from CBC, ECBC, FCBC, XCBC. We defined CBC, ECBC, FCBC, and XCBC as maps that depend on a function ρ (for CBC), on a triple of functions ρ_1, ρ_2, ρ_3 (for ECBC and FCBC), or on a function ρ_1 and strings $K2, K3$ (for XCBC). To speak of security, we will need to think of CBC, ECBC, FCBC, and XCBC as naming function families. Namely, one specifies the function family from which ρ is drawn (for CBC_ρ), or the function families from which ρ_1, ρ_2, and ρ_3 are drawn (for ECBC and FCBC), or the function family from which ρ_1 is drawn (for XCBC, where $K2$ and $K3$ are assumed to be chosen uniformly at random from Σ^n). We write the needed information in brackets, following the mode name, as in $F = \mathrm{ECBC}[\mathrm{Rand}(n) \times \mathrm{Rand}(n) \times \mathrm{Rand}(n)]$, meaning that a random element f is determined by choosing $\rho_1, \rho_2, \rho_3 \xleftarrow{\$} \mathrm{Rand}(n)$ and setting $f = \mathrm{ECBC}_{\rho_1\ \rho_2\ \rho_3}$.

Adversaries. An adversary A is an algorithm with an oracle. In this paper the oracle will compute some deterministic function. Adversaries are assumed to never ask an oracle query outside of the domain of the oracle and to never repeat a query. We write $A \Rightarrow 1$ for the event that A outputs the bit 1.

Security notions. Let $\mathcal{M}$ be a nonempty set, let $F = \{\mathcal{M} \to \Sigma^n\}$ be a function family and let A be an adversary with an oracle $f : \mathcal{M} \to \Sigma^n$. We say that the adversary *forges* if it outputs a pair $(M, f(M))$ where $M \in \mathcal{M}$ and the adversary never queried its oracle f at M. We let

$$\begin{aligned}
\mathbf{Adv}_F^{\mathrm{mac}}(A) &= \Pr[f \xleftarrow{\$} F :\ A^f \text{ forges }] \\
\mathbf{Adv}_F^{\mathrm{prf}}(A) &= \Pr[f \xleftarrow{\$} F :\ A^f \Rightarrow 1] - \Pr[\rho \xleftarrow{\$} \mathrm{Rand}(\mathcal{M}, n) :\ A^\rho \Rightarrow 1] \\
\mathbf{Adv}_F^{\mathrm{prp}}(A) &= \Pr[f \xleftarrow{\$} F :\ A^f \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) :\ A^\pi \Rightarrow 1] \ \text{ for the case of } \mathcal{M} = \Sigma^n
\end{aligned}$$

We overload the notation above and write $\mathbf{Adv}_F^{\mathrm{xxx}}(\mathcal{R})$ (where $\mathrm{xxx} \in \{\mathrm{mac}, \mathrm{prf}, \mathrm{prp}\}$) for the maximal value of $\mathbf{Adv}_F^{\mathrm{xxx}}(A)$ among adversaries that use resources at most $\mathcal{R}$. Resources of interest are: t, the running time of the adversary; q, the number of queries the adversary makes; and σ (where $n \geq 1$), the communications complexity of the adversary, as measured in n-bit blocks. Time is understood to include the description size of the adversary A. Communications complexity is the sum of the block lengths of each oracle query.

Switching prp/prf. It is often convenient to replace random permutations with random functions, or vice versa. The following proposition lets us easily do this. For a proof see Proposition 2.5 in [1].

Lemma 1 [PRF/PRP Switching] *Fix $n \geq 1$. Let A be an adversary that asks at most q queries. Then*

$$\Big| \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\pi(\cdot)} \Rightarrow 1] - \Pr[\rho \xleftarrow{\$} \mathrm{Rand}(n) :\ A^{\rho(\cdot)} \Rightarrow 1] \Big| \ \leq \ \frac{q(q-1)}{2^{n+1}} \qquad \square$$

4 Security of ECBC

We prove the security of ECBC, showing that $\mathrm{ECBC}[\mathrm{Perm}(n) \times \mathrm{Perm}(n) \times \mathrm{Perm}(n)]$ resembles $\mathrm{Rand}(\Sigma^*, n)$ provided the adversary asks no more than some specified maximum number of oracle queries. The proof views ECBC as the CBC MAC followed by an enciphering step. The CBC MAC is regarded as an almost-universal hash-function family, and we bound its collision probability. The enciphering step is done by applying one of two functions, ρ_2 or ρ_3, depending on the padding that was initially performed.

Applying a PRF to the output of an almost-universal hash-function family is a well-known approach for creating a PRF or MAC [3, 5, 14]. The novelty here is our method of dealing with messages that are not a multiple of the block length and, more significantly, the treatment of the CBC MAC as an almost-universal hash-function family. The latter might run against one's instincts because the CBC MAC is a much stronger object than an almost-universal hash-function family. What we are doing is to *ignore* this extra strength and focus just on the object's collision probability.

4.1 Security of the three-key Carter-Wegman construction

We abstract the structure of ECBC, replacing the CBC construction that it uses by an arbitrary hash function h. So, letting $h\colon (\Sigma^n)^+ \to \Sigma^n$ and $\rho_2\colon \Sigma^n \to \Sigma^n$ and $\rho_3\colon \Sigma^n \to \Sigma^n$, define $\mathrm{CW3}_{h\ \rho_2\ \rho_3}(M)$ as $\rho_2(h(M))$ if $|M|$ is a positive multiple of n, and $\rho_3(h(M \parallel 10^\ell))$ otherwise, where $\ell = n - 1 - |M| \bmod n$. Now let $H = \{h\colon (\Sigma^n)^+ \to \Sigma^n\}$ and $F2 = \{\rho_2\colon \Sigma^n \to \Sigma^n\}$ and $F3 = \{\rho_3\colon \Sigma^n \to \Sigma^n\}$ be function families. Then the definition of $\mathrm{CW3}_{h\ \rho_2\ \rho_3}$ lifts to give a function family $F = \mathrm{CW3}[H \times F2 \times F3]$ where a random element f from F is determined by choosing $h \xleftarrow{\$} H$, $\rho_2 \xleftarrow{\$} F2$, $\rho_3 \xleftarrow{\$} F3$ and setting $f(M) = \mathrm{CW3}_{h\ \rho_2\ \rho_3}(M)$. We now prove the security of this function family, assuming ρ_2 and ρ_3 are random functions and H is good as an almost-universal hash-function family.

Lemma 2 [Three-Key CW] *Let $H = \{h\colon (\Sigma^n)^+ \to \Sigma^n\}$ be a function family. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CW3}[H\times\mathrm{Rand}(n)\times\mathrm{Rand}(n)]}(\sigma) \quad \le \quad \max_{\substack{q,m_1,\ldots,m_q \\ m_1+\cdots+m_q=\sigma}} \left\{ \sum_{1\le i<j\le q} \mathbf{Coll}_H(m_i,\, m_j) \right\} \qquad \square$$

Proof: We provide a game-playing argument, as used in works like [11]. Let A be an adversary with access to an oracle. Assume that the queries asked by A to this oracle total at most σ blocks. Without loss of generality, assume that A never repeats an oracle query. We must compare (1) what A does when its oracle is $f = \mathrm{CW3}_{h\ \rho_2\ \rho_3}$, for $h \xleftarrow{\$} H$ and $\rho_2, \rho_3 \xleftarrow{\$} \mathrm{Rand}(n)$, and (2) what A does when its oracle is $\rho \xleftarrow{\$} \mathrm{Rand}(\Sigma^*, n)$. In particular, we are aiming to bound

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CW3}[H\times\mathrm{Rand}(n)\times\mathrm{Rand}(n)]}(A) \quad = \quad \Pr[A^{\mathrm{CW3}_{h\ \rho_2\ \rho_3}(\cdot)} \Rightarrow 1] - \Pr[A^{\rho(\cdot)} \Rightarrow 1]\ .$$

Consider providing an oracle for A in one of two ways, game E1 or game E2, as specified in Figure 5. In the pseudocode there, and in later proofs, we let $\mathrm{Domain}(\rho)$ denote the set of points $X \in \Sigma^*$ such that $\rho(X) \neq \mathsf{undefined}$. Game E1 provides a perfect simulation of $f = \mathrm{CW3}_{h\ \rho_2\ \rho_3}$ for $h \xleftarrow{\$} H$ and $\rho_2, \rho_3 \xleftarrow{\$} \mathrm{Rand}(n)$ and game E2 provides a perfect simulation of $\rho \xleftarrow{\$} \mathrm{Rand}(\Sigma^*, n)$. Thus $\Pr[A^{\mathrm{CW3}_{h\ \rho_2\ \rho_3}(\cdot)} \Rightarrow 1] = \Pr[A^{\mathrm{E1}(\cdot)} \Rightarrow 1]$ and $\Pr[A^{\rho(\cdot)} \Rightarrow 1] = \Pr[A^{\mathrm{E2}(\cdot)} \Rightarrow 1]$ and so $\mathbf{Adv}_{\mathrm{CW3}[H\times\mathrm{Rand}(n)\times\mathrm{Rand}(n)]}{}^{\mathrm{prf}}(A) = \Pr[A^{\mathrm{E1}(\cdot)} \Rightarrow 1] - \Pr[A^{\mathrm{E2}(\cdot)} \Rightarrow 1]$. Since games E1 and E2 are identical until the flag *bad* gets set to true, we can use the usual game-playing paradigm to conclude that $\Pr[A^{\mathrm{E1}(\cdot)} \Rightarrow 1] - \Pr[A^{\mathrm{E2}(\cdot)} \Rightarrow 1] \le \Pr[A^{\mathrm{E2}(\cdot)}$ sets *bad*$]$. We are left with finding a suitable upper bound on $\Pr[A^{\mathrm{E2}(\cdot)}$ sets *bad*$]$.

We note that, in game E2, what is returned to the adversary in response to a query is just n random bits, unrelated to the adversary's query or the function h. Thus the adversary's ability to produce collisions in h—to set the flag *bad* to true—is not made smaller if we insist that the adversary prepare all of its queries $M_1, \ldots, M_q$ in advance (still subject to the constraint that these messages sum to σ blocks). At this point we have eliminated the interaction in game E2. We may also reorder the queries without impacting the probability that *bad* will get set to true, so relabel $M_1, \ldots, M_q$ so that $M_1, \ldots, M_p$ are the queries in $(\Sigma^n)^+$ and let $M_{p+1}, \ldots, M_q$ be the remaining queries. For each $s \in [p+1\,..\,q]$, let $M'_s = M_s 10^{n-1-|M_s| \bmod n}$. Since $M_1, \ldots, M_q$ are distinct, $M_1, \ldots, M_p$ are distinct and $M'_{p+1}, \ldots, M'_q$ are distinct as well. (This follows by noticing that distinct strings $M_r, M_s \notin (\Sigma^n)^+$ will still be distinct after padding.) Let $m_j = \|M_j\|_n$ and note that $m_j = \|M'_j\|_n$ as well, for $j \ge p+1$. By the sum bound, we can now conclude that

$$\begin{aligned} \Pr[A^{\mathrm{E2}(\cdot)} \text{ sets } bad] \quad &\le \quad \Pr[A^{\mathrm{E2}(\cdot)} \text{ sets } bad \text{ at line 23}] + \Pr[A^{\mathrm{E2}(\cdot)} \text{ sets } bad \text{ at line 34}] \\ &\le \quad \sum_{1\le i<j\le p} \mathbf{Coll}_H(m_i,\, m_j) + \sum_{p+1\le i<j\le q} \mathbf{Coll}_H(m_i,\, m_j) \end{aligned}$$

Initialization:

$h \xleftarrow{\$} H$
$bad \leftarrow$ false
for all $X \in \Sigma^n$, let $\rho_2(X) \leftarrow \rho_3(X) \leftarrow$ undefined

Oracle f, when asked $f(M)$:

$Z \xleftarrow{\$} \Sigma^n$
if $M \in (\Sigma^n)^+$ **then**
  $Y \leftarrow h(M)$
  if $Y \in \mathrm{Domain}(\rho_2)$ **then**
    $bad \leftarrow$ true , $Z \leftarrow \rho_2(Y)$
  $\rho_2(Y) \leftarrow Z$
else
  $M' \leftarrow M10^{n-1-|M| \bmod n}$
  $Y \leftarrow h(M')$
  if $Y \in \mathrm{Domain}(\rho_3)$ **then**
    $bad \leftarrow$ true , $Z \leftarrow \rho_3(Y)$
  $\rho_3(Y) \leftarrow Z$
return Z

Figure 5: Games used in the analysis of CW3. Game E1 is the mechanism above, as written, while game E2 omits the shaded statement.

$$\leq \sum_{1 \leq i < j \leq q} \mathbf{Coll}_H(m_i,\, m_j)$$

The result follows. ▌

4.2 The collision probability of the CBC MAC

We show that if $M, M' \in (\Sigma^n)^+$ are distinct then $\Pr[\rho \xleftarrow{\$} \mathrm{Rand}(n) :\ \mathrm{CBC}_\rho(M) = \mathrm{CBC}_\rho(M')]$ is small. By "small" we mean a slowly growing function of $m = \|M\|_n$ and $m' = \|M'\|_n$. One way at getting such a result is to infer it from the result of Petrank and Rackoff [13]. Here we provide a direct analysis, using a game-playing argument. The lemma improves the $(m+m')^2/2^n$ bound we proved in the proceedings version of this paper [4].

Lemma 3 [CBC Collision Bound] *Fix $n, m, m' \geq 1$ and let $M \in (\Sigma^n)^m$ and $M' \in (\Sigma^n)^{m'}$ be distinct strings. Then*

$$\mathbf{Coll}_{\mathrm{CBC}[\mathrm{Rand}(n)]}(m,\, m') \;\leq\; \frac{mm'}{2^n} + \frac{\max\{m, m'\}}{2^n} \qquad \square$$

Proof: If the lengths of M and M' differ then, without loss of generality, let M name the shorter of the two strings and let M' name the longer. Then write $M = M_1 \cdots M_m$ and $M' = M'_1 \cdots M'_{m'}$ where each M_i and each M'_j is n-bits long and $m' \geq m$. Write M as $M = N \parallel I$ and write M' as $M' = N \parallel I'$ where N is the longest common prefix of M and M' whose length is divisible by n. Let $k = \|N\|_n$. Note that $k = m$ if M is a prefix of M' and otherwise k is the largest nonnegative integer such that $M_1 \cdots M_k = M'_1 \cdots M'_k$ but $M_{k+1} \neq M'_{k+1}$. By definition, if $M_1 \neq M'_1$ then $k = 0$. Further note that $k < m'$.

Consider game C1, as defined in Figure 6. This game realizes one way to compute $Y_m = \mathrm{CBC}_\rho(M)$ and $Y'_{m'} = \mathrm{CBC}_\rho(M')$ for a random $\rho \in \mathrm{Rand}(n)$, and so we would like to bound the probability, in game C1, that $Y_m = Y'_{m'}$. Also depicted in Figure 6 is game C2, obtained by eliminating the shaded statements.

```
01  bad ← false
02  for X ∈ Σ^n do η(X) ← ι(X) ← ι'(X) ← undefined

10  Y_0 ← 0^n
11  for i ← 1 to k do
12      Y_i ←$ Σ^n
13      X_i ← Y_{i-1} ⊕ M_i
14      if X_i ∈ Domain(η) then Y_i ← η(X_i),  bad ← true else η(X_i) ← Y_i

20  for i ← k+1 to m do
21      Y_i ←$ Σ^n
22      X_i ← Y_{i-1} ⊕ M_i
23      if X_i ∈ Domain(ι) then Y_i ← ι(X_i) else ι(X_i) ← Y_i
24      if X_i ∈ Domain(η) then Y_i ← η(X_i),  bad ← true

30  Y'_k ← Y_k
40  for j ← k+1 to m' do
41      Y'_j ←$ Σ^n
42      if Y'_j = Y_m then bad ← true
43      X'_j ← Y'_{j-1} ⊕ M'_j
44      if X'_j ∈ Domain(ι') then Y_j ← ι'(X'_j) else ι'(X'_j) ← Y'_j
45      if X'_j ∈ Domain(ι) then Y'_j ← ι(X'_j),  bad ← true
46      if X'_j ∈ Domain(η) then Y'_j ← η(X'_j),  bad ← true
```

Figure 6: Game C1, as written, and game C2, after eliminating the shaded statements. Game C1 provides a way to compute the CBC MAC of distinct messages $M = M_1 \cdots M_m$ and $M' = M'_1 \cdots M'_{m'}$ that are identical up to block k. The computed MACs are Y_m and $Y'_{m'}$.

Before taking up the analysis of the games in earnest, we give a bit of intuition about what they aim to capture. We are interested in the probability that $Y_m = Y'_{m'}$. This can happen due either to an *internal collision* between X_m and $X'_{m'}$, where Y_m was produced as $\rho(X_m)$ and $Y'_{m'}$ was produced as $\rho(X'_{m'})$, or Y_m might equal $Y'_{m'}$ even in the absence of this collision between X_m and $X'_{m'}$. Intuitively, the latter seems unlikely, and it is. The former is unlikely too, but to prove this we generalize and consider a broader set of internal collisions than just X_m coinciding with $X'_{m'}$. That is, consider the $X_1, \ldots, X_m$ values that get fed to ρ as we process $M = M_1 \cdots M_m$, and consider the $X'_1, \ldots, X'_{m'}$ values that get fed to ρ as we process $M' = M'_1 \cdots M'_{m'}$. Accounting for the fact that $X_i = X'_i$ for all $i \in [1 \,..\, k]$ (that is, the X_i-values that occur as we process the common prefix N of M and M'), we don't really expect to see collisions among $X_1, \ldots, X_m, X'_{i+1}, \ldots, X'_{m'}$. To get a better bound, we will "give up" in the analysis for some of these possibilities, but not all. Specifically, the internal collisions that we focus on are (a) collisions that occur while we process N (meaning collisions among any of $X_1, \ldots, X_k$); (b) collisions that occur between N and I (those between one of $X_1, \ldots, X_k$ and one of $X_{k+1}, \ldots, X_m$); (c) collisions that occur between N and I' (one of $X_1, \ldots, X_k$ coincides with one of $X'_{k+1}, \ldots, X'_{m'}$); and (d) collisions that occur between I' and I (one of $X_{k+1}, \ldots, X_m$ coincides with one of $X'_{k+1}, \ldots, X'_{m'}$). There is nothing "magical" about giving up exactly on these internal collisions—it is simply that the more internal collisions one gives up on the worse the bound but the easier the analysis. The proof we give formalizes the intuition of this paragraph and does all the necessary accounting.

Now look at game C1. It works not by growing a single random function ρ but by growing *three* random functions: η, ι, and ι'. The function η keeps track of the association of points that arise during the processing of N, the function ι keeps track of the association of points that arise during the processing of I, and the function ι' keeps track of the association of points that arise during the processing of I'. If a value X should get placed in the domain of two or more of these three functions then we regard the corresponding range

value as that specified first by η if such a value has been specified, and as the value secondarily by ι otherwise. Game C1 can thus be seen to provide a perfect simulation of the CBC algorithm over a random function $\rho \in \mathrm{Rand}(n)$, and we seek to bound the probability, in game C1, that $Y_m = Y'_{m'}$.

We first claim that, in game C1, any time that $Y_m = Y'_{m'}$ it is also the case that flag *bad* gets set to true. Focus on line 42. Since $m' \geq k+1$ we know that the loop covering lines 40–46 will be executed at least once and $Y'_{m'}$ will take its value as a result of these statements. When a preliminary $Y'_{m'}$ value gets chosen at line 41 for iteration $j = m'$ we see that if $Y'_{m'} = Y_m$ then line 42 will set the flag *bad*. But observe that the initially chosen $Y'_{m'}$ value is not necessarily the final $Y'_{m'}$ value returned by game C1—the $Y'_{m'}$ value chosen at line 42 could get overwritten at any of lines line 44, 45, or 46. If the value gets overwritten at either of line 45 or line 46 then *bad* will get set to true. If $Y'_{m'}$ gets overwritten by Y_m at line 44 then Y_m had earlier been placed in the range of ι' and it could only have gotten there (since ι' grows only by the **else**-clause of line 44) by $Y'_{m'}$ being equal to Y'_j for an already selected $j \in [k+1 \,..\, m'-1]$. In that case the flag *bad* would have already been set to true by an earlier execution of line 42: when the Y'_j value was randomly selected at line 41 and found to coincide with Y_m the flag *bad* is set. We thus have that every execution of game C1 that results in $Y_m = Y'_{m'}$ also results in *bad* being true.

Let $\Pr_1[\cdot]$ denote the probability of an event in game C1 and let $\Pr_2[\cdot]$ denote the probability of an event in game C2. Let B be the event (whether in game C1 or game C2) that the flag *bad* gets set to true. By the contents of the last paragraph, $\Pr_\rho[\mathrm{CBC}_\rho(M) = \mathrm{CBC}_\rho(M')] \leq \Pr_1[\mathsf{B}]$. Also note that games C1 and C2 are identical until the flag *bad* gets set to true (meaning that any execution in which a highlighted statement is executed it is also the case that *bad* is set to true in that execution) and so, in particular, $\Pr_1[\mathsf{B}] = \Pr_2[\mathsf{B}]$. We thus have that $\Pr_\rho[\mathrm{CBC}_\rho(M) = \mathrm{CBC}_\rho(M')] \leq \Pr_2[\mathsf{B}]$. We now proceed to bound $\Pr_2[\mathsf{B}]$, showing that $\Pr_2[\mathsf{B}] \leq mm'/2^n + \max\{m, m'\}/2^n$. This is done by summing the probabilities that *bad* gets set to true at lines 14, 24, 42, 45, and 46.

The probability that *bad* gets set at **line 14** of game C2 is at most $0.5k(k-1)/2^n$. This is because every point placed into the domain of η is of the form $X_i = Y_{i-1} \oplus M_i$ where each Y_{i-1} is randomly selected from Σ^n (at line 12) with the single exception of Y_0, which is a constant. So for any i and j with $1 \leq i < j \leq k$ we have that $\Pr[Y_{i-1} \oplus M_i = Y_{j-1} \oplus M_j] = 2^{-n}$ and there are $0.5\,k(k-1)/2^n$ such (i, j) pairs possible.

The probability that *bad* gets set at **line 24** of game C2 is at most $k(m-k)/2^n$. Each point X_i whose presence in the domain of η we test is of the form $X_i = Y_{i-1} \oplus M_i$, as defined at line 22. Each of these Y_{i-1} values with the exception of Y_k is uniformly selected at line 21, independent of the now-determined domain of η. As for $X_{k+1} = Y_k \oplus M_{k+1}$, the value Y_k was just selected uniformly at random (at the last execution of line 12) and is independent of the domain of η. Thus each time line 24 executes there is at most a $k/2^n$ chance that *bad* will get set, and the line is executed $m-k$ times.

The probability that *bad* gets set at **line 42** of game C2 is at most $(m'-k)/2^n$. This is because the line is executed $m'-k$ times and each time the chance that *bad* gets set is $1/2^n$ since Y'_j was just selected at random from Σ^n.

The probability that *bad* gets set at **line 45** of game C2 is at most $(m-k)(m'-k)$. Each point X'_j whose presence in the domain of ι is being tested is of the form $X'_j = Y'_{j-1} \oplus M'_j$, as defined at line 43. Each of these Y'_{i-1} values with the exception of Y'_k was uniformly selected at line 41, independent of the now-determined domain of ι. As for $X'_{k+1} = Y'_k \oplus M'_{k+1}$, the value Y'_k was just selected uniformly at random back at line 12 and is independent of the domain of ι apart from the domain point $Y_k \oplus M_{k+1}$, which is *guaranteed* to be distinct from $Y'_k \oplus M'_{k+1}$ by our criterion for choosing k. Thus each time line 45 executes there is at most a $(m-k)/2^n$ chance that *bad* will get set, and the line is executed $m'-k$ times.

The probability that *bad* gets set at **line 46** of game C2 is at most $k(m'-k)$ for reasons exactly analogous to that argument used at line 24.

We conclude by the sum bound that the probability that *bad* gets set somewhere in game C2 is at most $(k(k-1)/2 + k(m-k) + (m'-k) + (m-k)(m'-k) + k(m'-k))/2^n = mm' + m' - k(k-3)/2$. Since k can be zero, this value is at most $(mm' + m')/2^n$. Recalling that m' is the block length of the longer of M and M', the proof is complete. ▌

4.3 The security of ECBC

The lemmas given establish the security of ECBC. The result is as follows.

Theorem 4 [Security of ECBC] *Fix $n \geq 1$. Then for any $\sigma \geq 0$,*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{ECBC}[\mathrm{Perm}(n)\times\mathrm{Perm}(n)\times\mathrm{Perm}(n)]}(\sigma) \quad \leq \quad \frac{2.5\,\sigma^2}{2^n} \qquad \square$$

Proof: Combining Lemma 2 and Lemma 3, we have that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{ECBC}[\mathrm{Rand}(n),\mathrm{Rand}(n),\mathrm{Rand}(n)]}(\sigma) &= \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CW3}[\mathrm{CBC}[\mathrm{Rand}(n)],\mathrm{Rand}(n),\mathrm{Rand}(n)]}(\sigma) \\
&\leq \max_{\substack{q,m_1,\ldots,m_q \\ m_1+\cdots+m_q=\sigma}} \left\{ \sum_{1\leq i<j\leq q} \mathbf{Coll}_{\mathrm{CBC}[\mathrm{Rand}(n)]}(m_i,\, m_j) \right\} \\
&\leq \max_{\substack{q,m_1,\ldots,m_q \\ m_1+\cdots+m_q=\sigma}} \left\{ \sum_{1\leq i<j\leq q} \frac{m_i m_j}{2^n} + \frac{\max\{m_i, m_j\}}{2^n} \right\} \qquad (1) \\
&\leq \binom{\sigma}{2}/2^n + \binom{\sigma}{2}/2^n \\
&\leq \sigma^2/2^n
\end{aligned}$$

where the second-to-last statement derives from the fact that equation (1) is maximized for $q = \sigma$ and $m_1 = \cdots = m_q = 1$. To see this, consider any sequence $\{m_1, \ldots, m_q\}$ where $m_i > 1$, for some i. Without loss of generality, assume $i = q$. Then it is easily seen that the new sequence $\{m_1, \ldots, m_{q-1}, m_q - 1, 1\}$ increases the sum (1) by exactly $m_q - 1$ in the first term, and by at least $m_q - 1$ in the second term. So the maximum must be achieved when $q = \sigma$ and $m_1 = \cdots = m_q = 1$. Finally, using Lemma 1, we know that

$$\left|\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CW3}[\mathrm{CBC}[\mathrm{Perm}(n)]\times\mathrm{Perm}(n)\times\mathrm{Perm}(n)]}(q,\sigma) - \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CW3}[\mathrm{CBC}[\mathrm{Rand}(n)]\times\mathrm{Rand}(n)\times\mathrm{Rand}(n)]}(q,\sigma)\right| \quad \leq \quad (0.5\,\sigma^2 + q^2)/2^n$$

from which the result follows. ∎

Comments. We have given an entirely information-theoretic treatment of ECBC. In the standard way one can now pass from the information-theoretic setting to a complexity-theoretic one. This is standard and so we omit doing this. For our final scheme, XCBC, we will include a complexity-theoretic bound.

The security of ECBC as a MAC follows immediately from the security of ECBC as a PRF. This too is standard. See [1] for an exposition.

It is easy to exhibit an attack on FCBC that asks $\sigma < C2^n$ blocks worth of queries and then forges with success probability at least $c\sigma^2/2^n$, where $c, C > 0$ are absolute constants. In this sense our analysis is tight. The same claim of tightness can be made for the FCBC and XCBC analyses. Thus our security bounds say that if the underlying block cipher is a good PRP, then it is *safe* to use ECBC, FCBC, and XCBC on a number of blocks σ well under $2^{n/2}$, while easy attacks make clear that it is *unsafe* to use ECBC, FCBC, and XCBC on a number of blocks σ near or beyond $2^{n/2}$.

Similar arguments to what we have given can be used to prove the security of the EMAC construction [2], yielding a proof much simpler than that found in [13].

5 Security of FCBC

In this section we prove the security of FCBC, obtaining the same bound we had for ECBC.

Theorem 5 [Security of FCBC] *Fix $n \geq 1$. Then for any $\sigma \geq 0$,*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{FCBC}[\mathrm{Perm}(n)\times\mathrm{Perm}(n)\times\mathrm{Perm}(n)]}(\sigma) \quad \leq \quad \frac{2.5\,\sigma^2}{2^n}$$

□

Proof: Let us compare the distribution on functions

$$\{\mathrm{ECBC}_{\pi_1\ \pi_2\ \pi_3}(\cdot) \mid\ \pi_1, \pi_2, \pi_3 \xleftarrow{\$} \mathrm{Perm}(n)\} \quad \text{and}$$
$$\{\mathrm{FCBC}_{\pi_1\ \sigma_2\ \sigma_3}(\cdot) \mid\ \pi_1, \sigma_2, \sigma_3 \xleftarrow{\$} \mathrm{Perm}(n)\}\ .$$

We claim that these are the *same* distribution, so, information theoretically, the adversary has no way to distinguish a random sample drawn from one distribution from a random sample from the other. The reason is simple. In the ECBC construction we compose the permutation π_1 with the random permutation π_2. But the result of such a composition is just a random permutation, σ_2. Elsewhere in the ECBC construction we compose the permutation π_1 with the random permutation π_3. But the result of such a composition is just a random permutation, σ_3. Making these substitutions—σ_2 for $\pi_2 \circ \pi_1$, and σ_3 for $\pi_3 \circ \pi_1$, we recover the definition of ECBC. Changing back to the old variable names we have

$$\Pr[\pi_1, \pi_2, \pi_3 \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\mathrm{FCBC}_{\pi_1\ \pi_2\ \pi_3}(\cdot)} \Rightarrow 1] \quad = \quad \Pr[\pi_1, \pi_2, \pi_3 \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\mathrm{ECBC}_{\pi_1\ \pi_2\ \pi_3}(\cdot)} \Rightarrow 1]$$

so the bound of the present theorem follows immediately from Theorem 4. ▌

6 Security of XCBC

We now prove the security of the XCBC construction. We begin with a lemma that bounds an adversary's ability to distinguish between a pair of random permutations $(\pi_1(\cdot),\ \pi_2(\cdot))$ and the pair of permutations $(\pi(\cdot),\ \pi(K \oplus \cdot))$, where π is a random permutation and K is a random n-bit string. This lemma, and ones like it, may make generally useful tools. Indeed, this technique anticipates the generalization of making *multiple* permutations from one [12].

Lemma 6 [Two permutations from one] *Fix $n \geq 1$. Let A be an adversary with a left oracle and a right oracle, and assume that A asks at most q total queries. Then*

$$\left| \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n);\ K \xleftarrow{\$} \Sigma^n :\ A^{\pi(\cdot),\ \pi(K \oplus \cdot)} \Rightarrow 1] - \Pr[\pi_1, \pi_2 \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\pi_1(\cdot),\ \pi_2(\cdot)} \Rightarrow 1] \right| \quad \leq \quad \frac{1.25\, q^2}{2^n}\ . \ \square$$

Proof: Without loss of generality, assume that adversary A is deterministic and never repeats a left-query and never repeats a right-query. We consider answering the adversary's queries by either of two games, X1 or X2, shown in Figure 7. These games are identical except for game X2 omitting the four shaded statements. As before, $\mathrm{Domain}(\pi)$ is the set of all $X \in \Sigma^n$ for which $\pi(X)$ is not undefined. We write $\mathrm{Range}(\pi)$ for the set of all $Y \in \Sigma^*$ such that $\pi(X) = Y$ for some X. By $\overline{\mathrm{Range}}(\pi)$ we denote $\Sigma^n - \mathrm{Range}(\pi)$. For both games X1 and X2, First we invoke the initialization procedure. Then when the adversary makes a query X to its left oracle we compute $f(X)$, and when the adversary makes a query X to its right oracle we compute $g(X)$.

We claim that game X1 perfectly simulates a pair of oracles where the first is a random permutation $\pi(\cdot) \in \mathrm{Perm}(n)$ and the second is $\pi(K \oplus \cdot)$, for a random n-bit string K; while game X2 perfectly simulates a pair of independent random permutations on n-bits, $\pi_1(\cdot)$, $\pi_2(\cdot)$. In particular:

$$\Pr[A^{\mathrm{X1}(\cdot)} \Rightarrow 1] \quad = \quad \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n);\ K \xleftarrow{\$} \Sigma^n :\ A^{\pi(\cdot),\ \pi(K \oplus \cdot)} \Rightarrow 1] \tag{2}$$

$$\Pr[A^{\mathrm{X2}(\cdot)} \Rightarrow 1] \ \text{ and } \quad = \quad \Pr[\pi_1, \pi_2 \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\pi_1(\cdot),\ \pi_2(\cdot)} \Rightarrow 1]\ . \tag{3}$$

To verify (3) note that the externally-visible behavior of game X2 does not depend on lines 10, 12, 20, and 22; these only determine the setting of a flag, *bad*, whose value is never consulted. With those lines expunged the set $\mathcal{S}$ tracks the diminishing set of allowed return-values to f-calls and the set $\mathcal{T}$ tracks the diminishing set

Initialization:

01 $bad \leftarrow$ false; $\mathcal{S}, \mathcal{T} \leftarrow \Sigma^n$; $K \xleftarrow{\$} \Sigma^n$; **for all** $X \in \Sigma^n$ **do** $\pi(X) \leftarrow$ undefined

Oracle f**, when asked** $f(X)$**:**

10 **if** $X \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow$ true , **return** $\pi(X)$

11 $Y \xleftarrow{\$} \mathcal{S}$

12 **if** $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow$ true , $Y \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$

13 $\pi(X) \leftarrow Y$; $\mathcal{S} \leftarrow \mathcal{S} - \{Y\}$; **return** Y

Oracle g**, when asked** $g(X)$**:**

20 **if** $(K \oplus X) \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow$ true , **return** $\pi(K \oplus X)$

21 $Y \xleftarrow{\$} \mathcal{T}$

22 **if** $Y \in \mathrm{Range}(\pi)$ **then** $bad \leftarrow$ true , $Y \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$

23 $\pi(K \oplus X) \leftarrow Y$; $\mathcal{T} \leftarrow \mathcal{T} - \{Y\}$; **return** Y

Figure 7: Game used in the proof of Lemma 6. Game X1 is as written and game X2 has the shaded statements removed. The former behaves like $\pi(\cdot)$, $\pi(K \oplus \cdot)$ and the latter behaves like $\pi_1(\cdot)$, $\pi_2(\cdot)$, where π, π_1, π_2 are random permutations and K is a random n-bit string.

of allowed return-values to g-calls. (Note that in game X2, $\pi(\cdot)$ is updated but never consulted; in game X1, where $\pi(\cdot)$ *is* used, we will require it be a permutation. In game X2 this is not required and indeed $\pi(\cdot)$ will typically be non-injective.) Thus game X2 faithfully simulates a pair of random permutations $\pi_1(\cdot)$, $\pi_2(\cdot)$.

To verify (2) focus on the variable π, which we build up in a manner consistent with answering $(f(\cdot), g(\cdot)) = (\pi(\cdot), \pi(K \oplus \cdot))$ for a random $K \in \Sigma^n$. Lines 10 and 20 ensure that we answer queries consistent with the current assignment to π. When we need to assign a new value to π, selecting a random value Y in $\overline{\mathrm{Range}}(\pi)$, we do this in a somewhat peculiar way, first sampling from a set $\mathcal{S}$ and then, if this fails to give a point in $\overline{\mathrm{Range}}(\pi)$, sampling again, this time from $\overline{\mathrm{Range}}(\pi)$. That this two-step procedure (lines 11–12) gives a random sample from $\overline{\mathrm{Range}}(\pi)$ follows from the invariant that $\mathcal{S} \supseteq \overline{\mathrm{Range}}(\pi)$. Similarly, $\mathcal{T} \supseteq \overline{\mathrm{Range}}(\pi)$ and so lines 21–22 choose a random sample from $\overline{\mathrm{Range}}(\pi)$ and game X1 faithfully simulates a pair of oracles $\pi(\cdot)$, $\pi(K \oplus \cdot)$.

Games X1 and X2 sometimes set a variable *bad* to true during the execution of the game. Notice that the two games behave identically prior to *bad* becoming true, and so, as per the usual game-playing approach, we know that

$$\Pr[A^{\mathrm{X1}} \Rightarrow 1] - \Pr[A^{\mathrm{X2}} \Rightarrow 1] \quad \leq \quad \Pr[A^{\mathrm{X2}} \text{ sets } bad] \ . \tag{4}$$

Putting this together with (2) and (3), the advantage that we aim to bound is $\Pr[A^{\mathrm{X2}} \text{ sets } bad]$.

Note that

$$\Pr[A^{\mathrm{X2}} \text{ sets } bad] \quad \leq \quad \Pr[A^{\mathrm{X2}} \text{ sets } bad \text{ in line 10 or 20}] + \Pr[A^{\mathrm{X2}} \text{ sets } bad \text{ in line 12 or 22}] \ . \tag{5}$$

Let us proceed by bounding the second of these two addends. When the adversary makes its ith oracle call the set $\mathcal{S}$ and the set $\mathcal{T}$ have at least $2^n - i + 1$ points, while $\mathrm{Range}(\pi)$ has at most $i - 1$ points. Thus the chance that we ever choose a random point Y from $\mathcal{S}$ or $\mathcal{T}$ that happens to lie in $\mathrm{Range}(\pi)$ is at most $\sum_{i=1}^{q}(i-1)/(2^n - i + 1) \leq q^2/2^n$, since the denominator is at least 2^{n-1} if $q \leq 2^{n-1}$ and the statement is trivially true otherwise. We have established that

$$\Pr[A^{\mathrm{X2}} \text{ sets } bad \text{ in line 12 or 22}] \quad \leq \quad q^2/2^n \ . \tag{6}$$

We are left with bounding $\Pr[A^{\mathrm{X2}} \text{ sets } bad \text{ in line 10 or 20}]$. To do this, we simplify game X2 in a number of ways that will not impact the probability that *bad* gets set at line 10 or 20. (1) Start off by eliminating

Initialization:

01 $bad \leftarrow \mathsf{true}$; $K \xleftarrow{\$} \Sigma^n$; $\mathcal{X} \leftarrow \varnothing$

Oracle f, when asked $f(X)$:

10 **if** $X \in \mathcal{X}$ **then** $bad \leftarrow \mathsf{true}$

11 $\pi(X) \leftarrow \mathcal{X} \cup \{X\}$

Oracle g, when asked $g(X)$:

20 **if** $(K \oplus X) \in \mathcal{X}$ **then** $bad \leftarrow \mathsf{true}$

21 $\mathcal{X} \leftarrow \mathcal{X} \cup \{K \oplus X\}$

Figure 8: Game X3, which is used in the analysis of game X2.

lines 12 and 22. These lines are not relevant to the event in question. (2) Next, observe that the values assigned to π are never used—all that mattered was whether or not a point was placed in the domain of π. So, letting $\mathcal{X}$ correspond to the domain of π, replace the **for**-statement of line 01 by $\mathcal{X} \leftarrow \varnothing$; replace Domain($\pi$) by $\mathcal{X}$ at line 10; replace the first statement of line 13 by $\mathcal{X} \leftarrow \mathcal{X} \cup \{X\}$; replace Domain($\pi$) by $\mathcal{X}$ at line 20; and replace the first statement of line 23 by $\mathcal{X} \leftarrow \mathcal{X} \cup \{K \oplus X\}$. (3) Finally, observe that the random values Y that are returned to A, and that diminish the sets $\mathcal{S}$ and $\mathcal{T}$ as the game runs, have no other impact on the game. The adversary could simulate this part of the game on its own. That is, imagine replacing the adversary A by an adversary B that also asks q queries but that takes responsibility for maintaining $\mathcal{S}$ and $\mathcal{T}$ and returning correctly-distributed Y-values. Adversary B would initialize $\mathcal{S} \leftarrow \mathcal{T} \leftarrow \varnothing$. Then B would run A. When A makes a query $f(X)$, adversary B would query $f(X)$, sample $Y \xleftarrow{\$} \mathcal{S}$, set $\mathcal{S} \leftarrow \mathcal{S} - \{Y\}$, and return Y to A. When A makes a query $g(X)$, adversary B would query $g(X)$, sample $Y \xleftarrow{\$} \mathcal{T}$, set $\mathcal{T} \leftarrow \mathcal{T} - \{Y\}$, and return Y to A. The probability that *bad* gets set to true at lines 10 or 20 would be unchanged; we have simply moved part of the function of the game into the adversary. The game has at this point been simplified to that shown in Figure 8, and we aim to bound the probability that an adversary B that asks q queries of game X3 manages to set *bad*. Adversary B never repeats an f-query or a g-query, and

$$\Pr[B^{\mathrm{X3}} \text{sets } bad] \quad = \quad \Pr[A^{\mathrm{X2}} \text{sets } bad \text{ at line 10 or 20}] \ . \tag{7}$$

To bound $\Pr[B^{\mathrm{X3}}$ sets $bad]$, realize that queries to this game return nothing to the adversary, and so we may assume that the adversary B is noninteractive. Suppose it asks queries $V_1, \ldots, V_p$ of its f-oracle, and queries $V'_1, \ldots, V'_{p'}$ of its g-oracle, where $p + p' = q$. Values $V_1, \ldots, V_p$ are distinct, and values $V'_1, \ldots, V'_{p'}$ are distinct. The points that will be placed into $\mathcal{X}$ are $\{V_1, \ldots, V_p, K \oplus V'_1, \ldots, K \oplus V'_{p'}\}$ and the flag *bad* will be set to true if any two of these coincide. We know that $V_i \neq V_j$ for $i, j \in [1 \,..\, p]$ and $i \neq j$; and we know that $K \oplus V'_i \neq K \oplus V'_j$ for $i, j \in [1 \,..\, p']$ and $i \neq j$. On the other hand, $\Pr[V_i = K \oplus V'_j] = 2^{-n}$ by the randomness of K. There are at most $(q/2)^2$ such pairs of values, and so the probability that *bad* gets set to true in game G3 is at most $0.25\, q^2/2^n$:

$$\Pr[B^{\mathrm{X3}} \text{ sets } bad] \quad \leq \quad 0.25\, q^2/2^n \tag{8}$$

Combining the numbered equations completes the proof. ∎

The security of XCBC follows from the security of FCBC (Theorem 5) and the method embodied by Lemma 6.

Theorem 7 [Security of XCBC] *Fix $n \geq 1$. Then for any $\sigma \geq 0$,*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{XCBC}[\mathrm{Perm}(n)]}(\sigma) \quad \leq \quad \frac{3.75\, \sigma^2}{2^n} \qquad \square$$

Proof: Theorem 5 says that

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{FCBC}[\mathrm{Perm}(n) \times \mathrm{Perm}(n) \times \mathrm{Perm}(n)]}(\sigma) \quad \leq \quad \frac{2.5\, \sigma^2}{2^n}$$

and two applications of Lemma 6 give that

$$\begin{aligned}\left|\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{XCBC}[\mathrm{Perm}(n)]}(q_1,q_2) - \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{FCBC}[\mathrm{Perm}(n)\times\mathrm{Perm}(n)\times\mathrm{Perm}(n)]}(q_1,q_2)\right| &\le \frac{1.25\,(q_1^2+q_2^2)}{2^n}\\ &\le \frac{1.25\,q^2}{2^n}\end{aligned}$$

where resource measure q_1 is the number of oracle queries that are positive multiples of n bits and resource measure q_2 is the number of oracle queries that are not. As $q \le \sigma$, the result follows. ▌

We end this paper by giving the complexity-theoretic analog of of Theorem 7. It follows from the information-theoretic result in the usual way, but, for completeness, we explicitly state the result and provide a proof for it.

Corollary 8 [Security of XCBC, complexity-theoretic version] *Fix $n \ge 1$ and let $E\colon \mathcal{K}\times\Sigma^n \to \Sigma^n$ be a block cipher. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{XCBC}[E]}(t,\sigma) \le \frac{3.75\,\sigma^2}{2^n} + \mathbf{Adv}^{\mathrm{prp}}_{E}(t',\sigma)$$

where $t' = t + Cn\sigma$ for some absolute constant C. □

Proof: Let A be an adversary that runs in time at most t and asks queries totaling at most σ blocks. Making some simplifications to the notation (omitting $K \xleftarrow{\$} \mathcal{K}$ and $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ and $\rho \xleftarrow{\$} \mathrm{Rand}(\Sigma^*,n)$ and $K2, K3 \xleftarrow{\$} \Sigma^n$), we have that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{XCBC}[E]}(A) &= \Pr[A^{\mathrm{XCBC}_{E_K\ K2\ K3}}\Rightarrow 1] - \Pr[A^{\rho}\Rightarrow 1]\\ &= \big(\Pr[A^{\mathrm{XCBC}_{E_K\ K2\ K3}}\Rightarrow 1] - \Pr[A^{\mathrm{XCBC}_{\pi\ K2\ K3}}\Rightarrow 1]\big) + \big(\Pr[A^{\mathrm{XCBC}_{\pi\ K2\ K3}}\Rightarrow 1] - \Pr[A^{\rho}\Rightarrow 1]\big)\\ &\le \mathbf{Adv}^{\mathrm{prp}}_{E}(t',\sigma) + \big(\Pr[A^{\mathrm{XCBC}_{\pi\ K2\ K3}}\Rightarrow 1] - \Pr[A^{\rho}\Rightarrow 1]\big) \qquad (9)\\ &= \mathbf{Adv}^{\mathrm{prp}}_{E}(t',\sigma) + \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{XCBC}[\mathrm{Perm}(n)]}(A)\\ &\le \mathbf{Adv}^{\mathrm{prp}}_{E}(t',\sigma) + \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{XCBC}[\mathrm{Perm}(n)]}(\sigma)\\ &\le \mathbf{Adv}^{\mathrm{prp}}_{E}(t',\sigma) + \frac{3.75\,\sigma^2}{2^n}\,.\end{aligned}$$

Equation (9) is justified as follows. Given adversary A having an oracle $f\colon \Sigma^* \to \Sigma^n$ we can construct an adversary B having an oracle $e\colon \Sigma^n \to \Sigma^n$ as follows. Adversary B begins by choosing $K2, K3 \xleftarrow{\$} \Sigma^n$. Then it runs adversary A. When A makes an oracle query X, adversary B uses its oracle e, along with $K2$ and $K3$, to compute $T \leftarrow \mathrm{XCBC}_{e\ K2\ K3}(X)$. Adversary B returns T to adversary A. When A halts, outputting a bit b, adversary B outputs the same bit b. Then $\mathbf{Adv}^{\mathrm{prp}}_{E}(B) = \Pr[A^{\mathrm{XCBC}_{E_K\ K2\ K3}}\Rightarrow 1] - \Pr[A^{\mathrm{XCBC}_{\pi\ K2\ K3}}\Rightarrow 1]$ and B's running time t' is A's running time t plus overhead of $Cn\sigma$, and B asks σ oracle queries. This establishes the result. ▌

Acknowledgments

Shai Halevi proposed the elegant idea of using three keys to extend the domain of the CBC MAC to Σ^*, nicely simplifying an approach used in an early version of UMAC [3]. Thanks to Shai and Mihir Bellare for their comments on an early draft, and to the referees for their useful comments.

Initial work on this paper was done with the support of Rogaway's NSF CCR-962540 and MICRO grants 98-129 and 99-103. Major revisions to this manuscript were done while John Black was supported by NSF CAREER-0240000 and Phil Rogaway was supported by NSF 0208842 and a gift from Cisco Systems.

References

[1] Bellare, M., Kilian, J., and Rogaway, P. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences (JCSS) 61*, 3 (Dec. 2000), 362–399. Earlier version in CRYPTO '94. See `www.cs.ucdavis.edu/~rogaway`.

[2] Berendschot, A., den Boer, B., Boly, J., Bosselaers, A., Brandt, J., Chaum, D., Damgård, I., Dichtl, M., Fumy, W., van der Ham, M., Jansen, C., Landrock, P., Preneel, B., Roelofsen, G., de Rooij, P., and Vandewalle, J. *Final Report of Race Integrity Primitives*, vol. 1007 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[3] Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P. UMAC: Fast and secure message authentication. In *Advances in Cryptology – CRYPTO '99* (1999), vol. 1666 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 216–233.

[4] Black, J., and Rogaway, P. CBC MACs for arbitrary-length messages: The three-key constructions. In *Advances in Cryptology – CRYPTO '00* (2000), vol. 1800 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 197–215.

[5] Carter, L., and Wegman, M. Universal hash functions. *J. of Computer and System Sciences*, 18 (1979), 143–154.

[6] Dworkin, M., 2003. Personal communication.

[7] FIPS 113. Computer data authentication. Federal Information Processing Standards Publication 113, U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1994.

[8] Goldreich, O., Goldwasser, S., and Micali, S. How to construct random functions. *Journal of the ACM 33*, 4 (1986), 210–217.

[9] Iso/Iec 9797-1. Information technology – security techniques – data integrity mechanism using a cryptographic check function employing a block cipher algorithm. International Organization for Standards, Geneva, Switzerland, 1999. Second edition.

[10] Iwata, T., and Kurosawa, K. OMAC: One-key CBC MAC. In *Fast Software Encryption (FSE 2003)* (2003), Lecture Notes in Computer Science, Springer-Verlag.

[11] Kilian, J., and Rogaway, P. How to protect DES against exhaustive key search (An analysis of DESX). *Journal of Cryptology 14*, 1 (2001), 17–35.

[12] Liskov, M., Rivest, R., and Wagner, D. Tweakable block ciphers. In *Advances in Cryptology – CRYPTO '02* (2002), M. Yung, Ed., vol. 2442 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 31–46.

[13] Petrank, E., and Rackoff, C. CBC MAC for real-time data sources. *Journal of Cryptology 13*, 3 (2000), 315–338.

[14] Wegman, M., and Carter, L. New hash functions and their use in authentication and set equality. In *J. of Comp. and System Sciences* (1981), vol. 22, pp. 265–279.