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

John Black[1] and Phillip Rogaway[2]

[1] Dept. of Computer Science, University of Nevada, Reno NV 89557, USA,
blackj@cs.ucdavis.edu
[2] Dept. of Computer Science, University of California at Davis, Davis, CA 95616,
USA, rogaway@cs.ucdavis.edu, http://www.cs.ucdavis.edu/∼rogaway

Abstract. We suggest some simple variants of the CBC MAC that let you efficiently MAC messages of arbitrary lengths. 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^i padding ($i \geq 0$), 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 her inability to distinguish the block cipher from a random permutation. Our analysis exploits new ideas which simplify proofs compared to prior work.

1 Introduction

This paper describes some simple variants of CBC MAC. These 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 [6,8] 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 $E : \mathsf{Key} \times \Sigma^n \to \Sigma^n$ be a block cipher: it uses a key $K \in \mathsf{Key}$ to encipher an n-bit block X into an n-bit ciphertext $Y = E_K(X)$. The message space for the CBC MAC is $(\Sigma^n)^+$, meaning binary strings whose lengths are a positive multiple of n. So let $M = M_1 \cdots M_m$ be a string that we want to MAC, where $|M_1| = \cdots = |M_m| = n$. Then $\mathrm{CBC}_{E_K}(M)$, the CBC MAC of M under key K, is defined as 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 [2]. But their proof depends on the assumption that it is only messages of one fixed length, mn bits, that are being

M. Bellare (Ed.): CRYPTO 2000, LNCS 1880, pp. 197–215, 2000.

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 $X \,\|\, (X \oplus T)$, 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.

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 [3]. It has been analyzed by Petrank and Rackoff [10] who show, roughly said, that an adversary who obtains the MACs for messages which 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. 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).

In Section 2 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. We expect that in most settings the added overhead to create and manage the longer key is minimal, and so XCBC may be preferred.

For each of the new schemes we give a proof of security. Rather than adapt the rather complex proof of [10], or the even more complicated one of [2], we follow a new tack, viewing EMAC as an instance of the Carter-Wegman paradigm [5,12]: with EMAC one is enciphering the output of a universal-2 hash function. This universal-2 hash function is 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 EMAC, and ECBC as well. 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

Construct	Domain	#E Appls	#E 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	$k+2n$

Fig. 1. *The CBC MAC and five variants. Here M is the message to MAC and $E : \Sigma^k \times \Sigma^n \to \Sigma^n$ is a block cipher. The third column gives the number of applications of E, assuming $|M| > 0$. The fourth column is the number of different 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.*

a general lemma (Lemma 4) 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.

New standards. This work was largely motivated by the emergence of the Advanced Encryption Standard (AES). With the AES should come a next-generation standard for using it to MAC. Current CBC MAC standards handle this, for example, in an open-ended informational appendix [8]. We suggest that the case of variable message lengths is the *usual* case in applications of MACs, and that a modern MAC standard should specify only algorithms which will correctly MAC any sequence of bit strings. The methods here are simple, efficient, provably sound, timely, and patent-free—all the right features for a contemporary standard.

2 Schemes ECBC, FCBC, and XCBC

Arbitrary-length messages without obligatory padding: ecbc. We have described the algorithm $\mathrm{EMAC}_{E_{K1},E_{K2}}(M) = E_{K2}(\mathrm{CBC}_{E_{K1}}(M))$. One problem with EMAC is that its domain is limited to $(\Sigma^n)^+$. What if we want to MAC messages whose lengths are *not* a multiple of n?

The simplest approach is to use obligatory 10^i 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. Then apply EMAC. We call this method EMAC*. Formally, $\mathrm{EMAC}^*_{E_{K1},E_{K2}}(M) = \mathrm{EMAC}_{E_{K1},E_{K2}}(M \,\|\, 10^{n-1-|M| \bmod n})$. 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 multiple of n then we are appending an entire extra block of padding, and seemingly "wasting" an application of E. People have worked hard to optimize new block ciphers—it seems a shame to squander some of this efficiency with an unnecessary application of E. Moreover, in practical settings we often wish to MAC very short

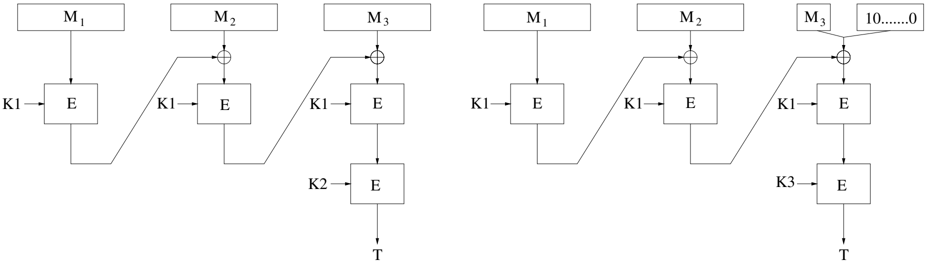

Fig. 2. *The* ECBC *construction using a block cipher* $E : \mathsf{Key} \times \Sigma^n \to \Sigma^n$*. The construction uses three keys,* $K1, K2, K3 \in \mathsf{Key}$*. On the left is the case where* $|M|$ *is a positive multiple of* n*, while on the right is the case where it isn't.*

messages, where saving one invocation of the block cipher 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 $\text{EMAC}_{E_{K1},E_{K2}}(M)$. In the second case we append minimal 10^i padding ($i \geq 0$) to make a padded message P whose length is divisible by n, and then we compute $\text{EMAC}_{E_{K1},E_{K3}}(P)$. 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 $\text{ECBC}_{E_{K1},E_{K2},E_{K3}}(M)$
if $M \in (\Sigma^n)^+$
    then return $E_{K2}(\text{CBC}_{E_{K1}}(M))$
    else return $E_{K3}(\text{CBC}_{E_{K1}}(M \,\|\, 10^i))$, where $i = n - 1 - |M| \bmod n$
```

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 Rackoff's result [10]. 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. With ECBC we are using $\lceil |M|/n \rceil + 1$ applications of the underlying block cipher. We can get rid of the $+1$ (except when $|M| = 0$). 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 follows from the result on the security of ECBC.

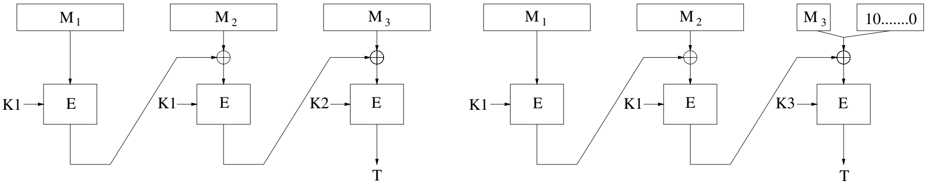

Fig. 3. *The* FCBC *construction with a block cipher $E : \mathsf{Key} \times \Sigma^n \to \Sigma^n$. The construction uses three keys, $K1, K2, K3 \in \mathsf{Key}$. On the left is the case where $|M|$ is a positive multiple of n, while on the right is the case where $|M|$ is not a positive multiple of n.*

```
**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 $P \leftarrow M \,\|\, 10^i$, where $i \leftarrow n - 1 - |M| \bmod n$
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 cost, when the key is turned into subkeys. The subkeys are often 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 actually not so desirable. It would be better to use the same key for all of the block-cipher invocations. The algorithm XCBC does this.

```
**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 $P \leftarrow M \,\|\, 10^i$, where $i \leftarrow n - 1 - |M| \bmod n$
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** $E_{K1}(P_m \oplus C_{m-1} \oplus K)$
```

We again make two cases. If $M \in (\Sigma^n)^+$ we 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^i padding ($i \geq 0$) and CBC as usual, except that we XOR in a different n-bit key, $K3$, before enciphering the last block. Here, in full, is the algorithm. Also see Figure 4. The proof of security can be found in Section 6.

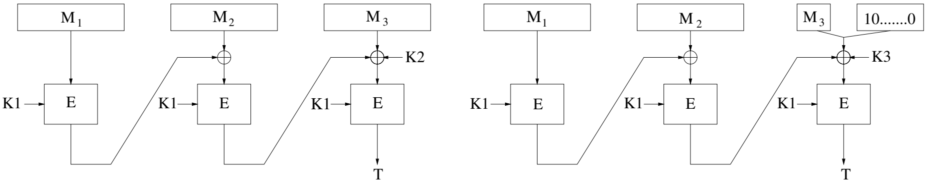

Fig. 4. *The* XCBC *construction with a block cipher* $E : \mathsf{Key} \times \Sigma^n \to \Sigma^n$*. We use keys* $K1 \in \mathsf{Key}$ *and* $K2, K3 \in \Sigma^n$*. On the left is the case where* $|M|$ *is a positive multiple of* n*; on the right is the case where* $|M|$ *is not a positive multiple of* n*.*

Summary. We have now defined $\mathrm{CBC}_{\rho_1}(\cdot)$, $\mathrm{EMAC}_{\rho_1,\,\rho_2}(\cdot)$, $\mathrm{ECBC}_{\rho_1,\,\rho_2,\,\rho_3}(\cdot)$, $\mathrm{FCBC}_{\rho_1,\,\rho_2,\,\rho_3}(\cdot)$, and $\mathrm{XCBC}_{\rho_1,\,k_2,\,k_3}(\cdot)$ where $\rho_1, \rho_2, \rho_3 \colon \Sigma^n \to \Sigma^n$ and $k_2, k_3 \in \Sigma^n$. We emphasize that the definitions make sense for any $\rho_1,\, \rho_2,\, \rho_3 \colon \Sigma^n \to \Sigma^n$; in particular, we don't require ρ_1, ρ_2, ρ_3 be permutations. Notice that we interchangeably use notation such as ρ_1 and E_{K1}; the key $K1$ is simply naming a function $\rho_1 = E_{K1}$.

3 Preliminaries

Notation. If A and B are sets then $\mathrm{Rand}(A, B)$ is the set of all functions from A to B. If A or B is a positive number, n, then the corresponding set is Σ^n. Let $\mathrm{Perm}(n)$ be the set of all permutations from Σ^n to Σ^n. By $x \overset{R}{\leftarrow} A$ we denote the experiment of choosing a random element from A.

A function family is a multiset $F = \{f \colon A \to B\}$, where $A, B \subseteq \Sigma^*$. Each element $f \in F$ has a name K, where $K \in \mathsf{Key}$. So, equivalently, a function family F is a function $F \colon\ \mathsf{Key} \times A \to B$. We call A the domain of F and B the range of F. The first argument to F will be written as a subscript. A block cipher is a function family $F \colon \mathsf{Key} \times \Sigma^n \to \Sigma^n$ where $F_K(\cdot)$ is always a permutation.

An adversary is an algorithm with an oracle. The oracle computes some function. Adversaries are assumed to never ask a query outside the domain of the oracle, and to never repeat a query.

Let $F \colon \mathsf{Key} \times A \to B$ be a function family and let $\mathcal{A}$ be an adversary. We say that $\mathcal{A}^f$ forges if $\mathcal{A}$ outputs $(x, f(x))$ where $x \in A$ and $\mathcal{A}$ never queried its oracle f at x. We let

$$\mathbf{Adv}_F^{\mathrm{mac}}(\mathcal{A}) \overset{\mathrm{def}}{=} \Pr[f \overset{R}{\leftarrow} F :\ \mathcal{A}^{f(\cdot)} \text{ forges }]$$

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) \overset{\mathrm{def}}{=} \Pr[f \overset{R}{\leftarrow} F :\ \mathcal{A}^{f(\cdot)} = 1] - \Pr[R \overset{R}{\leftarrow} \mathrm{Rand}(A, n) :\ \mathcal{A}^{R(\cdot)} = 1]\ ,$$

and when $A = \Sigma^n$

$$\mathbf{Adv}_F^{\mathrm{prp}}(\mathcal{A}) \overset{\mathrm{def}}{=} \Pr[f \overset{R}{\leftarrow} F :\ \mathcal{A}^{f(\cdot)} = 1] - \Pr[\pi \overset{R}{\leftarrow} \mathrm{Perm}(n) :\ \mathcal{A}^{\pi(\cdot)} = 1]\ .$$

We overload this notation and write $\mathbf{Adv}_F^{\mathrm{xxx}}(R)$ (where $\mathrm{xxx} \in \{\mathrm{mac}, \mathrm{prf}, \mathrm{prp}\}$) for the maximal value of $\mathbf{Adv}_F^{\mathrm{xxx}}(\mathcal{A})$ among adversaries who use resources R.

The resources we will deal with are: t, the running time of the adversary; q, the number of queries the adversary makes; and μ, the maximal bit length of each query. Omitted arguments are unbounded or irrelevant. We fix a few conventions. To measure time, t, one assumes some fixed model of computation. This model is assumed to support unit-time operations for computing $f \overset{R}{\leftarrow} \mathsf{Key}$ and $F_K(P)$. Time is understood to include the description size of the adversary $\mathcal{A}$. In the case of $\mathbf{Adv}_F^{\mathrm{mac}}$ the number of queries, q, includes one "invisible" query to test if the adversary's output is a valid forgery.

Basic facts. 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 [2].

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

$$\left|\Pr[\pi \overset{R}{\leftarrow} \mathrm{Perm}(n) : \mathcal{A}^{\pi(\cdot)} = 1] - \Pr[\rho \overset{R}{\leftarrow} \mathrm{Rand}(n,n) : \mathcal{A}^{\rho(\cdot)} = 1]\right| \leq \frac{p(p-1)}{2^{n+1}} \quad \blacksquare$$

As is customary, we will show the security of our MACs by showing that their information-theoretic versions approximate random functions. As is standard, this will be enough to pass to the complexity-theoretic scenario. Part of the proof is Proposition 2.7 of [2].

Lemma 2. [Inf. Th. PRF $\Rightarrow$ Comp. Th. PRF] *Fix $n \geq 1$. Let* CONS *be a construction such that* $\mathrm{CONS}_{\rho_1,\rho_2,\rho_3}(\cdot) \colon \Sigma^* \to \Sigma^n$ *for any* $\rho_1, \rho_2, \rho_3 \in \mathrm{Rand}(n,n)$. *Suppose that if* $|M| \leq \mu$ *then* $\mathrm{CONS}_{\rho_1,\rho_2,\rho_3}(M)$ *depends on the values of* ρ_i *on at most* p *points (for* $1 \leq i \leq 3$*). Let* $E \colon \mathsf{Key} \times \Sigma^n \to \Sigma^n$ *be a family of functions. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CONS}[E]}(t,q,\mu) \leq \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CONS}[\mathrm{Perm}(n)]}(q,\mu) + 3 \cdot \mathbf{Adv}^{\mathrm{prp}}_{E}(t',p), \text{ and}$$

$$\mathbf{Adv}^{\mathrm{mac}}_{\mathrm{CONS}[E]}(t,q,\mu) \leq \mathbf{Adv}^{\mathrm{prf}}_{\mathrm{CONS}[\mathrm{Perm}(n)]}(q,\mu) + 3 \cdot \mathbf{Adv}^{\mathrm{prp}}_{E}(t',p) + \frac{1}{2^n}\ ,$$

where $t' = t + O(pn)$. $\blacksquare$

4 Security of ECBC

In this section we prove the security of the ECBC construction. See Section 2 for a definition of $\mathrm{ECBC}_{\rho_1,\rho_2,\rho_3}(M)$, where $\rho_1, \rho_2, \rho_3 \in \mathrm{Rand}(n,n)$.

Our approach is as follows: we view the CBC MAC as an almost universal-2 family of hash functions and prove a bound on its collision probability. Then we create a PRF by applying one of two random functions to the output of CBC MAC and claim that this construction is itself a good PRF. Applying a PRF to a universal-2 hash function is a well-known approach for creating a PRF or MAC [5,12,4]. The novelty here is the extension to three keys and, more significantly, the treatment of the CBC MAC as an almost universal-2 family of hash functions. The latter might be against one's instincts because the CBC MAC is a much stronger object than a universal hash-function family. Here we ignore that extra strength and study only the collision probability.

We wish to show that if $M, M' \in (\Sigma^n)^+$ are distinct then $\Pr_\pi[\mathrm{CBC}_\pi(M) = \mathrm{CBC}_\pi(M')]$ is small. By "small" we mean a slowly growing function of $m = |M|/n$ and $m' = |M'|/n$. Formally, for $n, m, m' \geq 1$, let the *collision probability* of the CBC MAC be

$$V_n(m, m') \stackrel{\mathrm{def}}{=}$$

$$\max_{M\in\Sigma^{nm},\ M'\in\Sigma^{nm'},\ M\neq M'} \{\ \Pr[\pi \stackrel{R}{\leftarrow} \mathrm{Perm}(n) :\ \mathrm{CBC}_\pi(M) = \mathrm{CBC}_\pi(M')]\ \}\ .$$

(The shape of the character "V" is meant to suggest collisions.) It is not hard to infer from [10] a collision bound of $V_n(m, m') \leq 2.5\,(m + m')^2/2^n$ by using their bound on EMAC and realizing that collisions in the underlying CBC MAC must show through to the EMAC output. A direct analysis easily yields a slightly better bound.

Lemma 3. [CBC Collision Bound] *Fix $n \geq 1$ and let $N = 2^n$. Let $M, M' \in (\Sigma^n)^+$ be distinct strings having $m = |M|/n$ and $m' = |M'|/n$ blocks. Assume that $m, m' \leq N/4$. Then*

$$V_n(m, m') \ \leq\ \frac{(m + m')^2}{2^n}$$

Proof. Although M and M' are distinct, they may share some common prefix. Let k be the index of the last block in which M and M' agree. (If M and M' have unequal first blocks then $k = 0$.)

Each particular permutation π is equally likely among all permutations from Σ^n to Σ^n. In our analysis, we will view the selection of π as an incremental procedure. This will be equivalent to selecting π uniformly at random.

In particular, we view the computation of $\mathrm{CBC}_\pi(M)$ and $\mathrm{CBC}_\pi(M')$ as playing the game given in Figure 5. Here the notation M_i indicates the ith block of M. We initially set each range point of π as `undefined`; the notation $\mathrm{Domain}(\pi)$ represents the set of points x where $\pi(x)$ is no longer `undefined`. We use $\mathrm{Range}(\pi)$ to denote the set of points $\pi(x)$ which are no longer `undefined`; we use $\overline{\mathrm{Range}}(\pi)$ to denote $\Sigma^n - \mathrm{Range}(\pi)$.

During the game, the X_i are those values produced after XORing with the current message block, M_i, and the Y_i values are $\pi(X_i)$. See Figure 6.

We examine the probability that π will cause $\mathrm{CBC}_\pi(M) = \mathrm{CBC}_\pi(M')$, which will occur in our game iff $Y_m = Y'_{m'}$. Since π is invertible, this occurs iff $X_m = X'_{m'}$. As we shall see, this condition will cause *bad* = `true` in our game. However, we actually set *bad* to `true` in many other cases in order to simplify the analysis.

The idea behind the variable *bad* is as follows: throughout the program (lines 5, 12, and 17) we randomly choose a range value for π at some `undefined` domain point. Since π has not yet been determined at this point, the selection of our range value will be an independent uniform selection: there is no dependence

```
1:  $bad \leftarrow \texttt{false}$;   for all $x \in \Sigma^n$ do $\pi(x) \leftarrow \texttt{undefined}$
2:  $X_1 \leftarrow M_1$;   $X'_1 \leftarrow M'_1$;  $BAD \leftarrow \{X_1, X'_1\}$

3:  for $i \leftarrow 1$ to $k$ do
4:    if $X_i \in \text{Domain}(\pi)$ then $Y_i \leftarrow Y'_i \leftarrow \pi(X_i)$ else
5:      $Y_i \leftarrow Y'_i \xleftarrow{R} \overline{\text{Range}}(\pi)$;  $\pi(X_i) \leftarrow Y_i$
6:      if $i < m$ then begin $X_{i+1} \leftarrow Y_i \oplus M_{i+1}$
7:        if $X_{i+1} \in BAD$ then $bad \leftarrow \texttt{true}$ else $BAD \leftarrow BAD \cup \{X_{i+1}\}$ end
8:      if $i < m'$ then begin $X'_{i+1} \leftarrow Y'_i \oplus M'_{i+1}$
9:        if $X'_{i+1} \in BAD$ then $bad \leftarrow \texttt{true}$ else $BAD \leftarrow BAD \cup \{X'_{i+1}\}$ end

10: for $i \leftarrow k+1$ to $m$ do
11:   if $X_i \in \text{Domain}(\pi)$ then $Y_i \leftarrow \pi(X_i)$ else
12:     $Y_i \xleftarrow{R} \overline{\text{Range}}(\pi)$;  $\pi(X_i) \leftarrow Y_i$
13:     if $i < m$ then begin $X_{i+1} \leftarrow Y_i \oplus M_{i+1}$
14:       if $X_{i+1} \in BAD$ then $bad \leftarrow \texttt{true}$ else $BAD \leftarrow BAD \cup \{X_{i+1}\}$ end

15: for $i \leftarrow k+1$ to $m'$ do
16:   if $X'_i \in \text{Domain}(\pi)$ then $Y'_i \leftarrow \pi(X'_i)$ else
17:     $Y'_i \xleftarrow{R} \overline{\text{Range}}(\pi)$;  $\pi(X'_i) \leftarrow Y'_i$
18:     if $i < m$ then begin $X'_{i+1} \leftarrow Y'_i \oplus M'_{i+1}$
19:       if $X'_{i+1} \in BAD$ then $bad \leftarrow \texttt{true}$ else $BAD \leftarrow BAD \cup \{X'_{i+1}\}$ end
```

Fig. 5. *Game used in the proof of Lemma 3. The algorithm gives one way to compute the CBC MAC of distinct messages $M = M_1 \cdots M_m$ and $M' = M'_1 \cdots M'_{m'}$. These messages are identical up to block k. The computed MACs are Y_m and $Y_{m'}$, respectively.*

on any prior choice. If the range value for π were already determined by some earlier choice, the analysis would become more involved. We avoid the latter condition by setting *bad* to `true` whenever such interdependencies are detected. The detection mechanism works as follows: throughout the processing of M and M' we will require π be evaluated at $m + m'$ domain points $X_1, \cdots, X_m$ and $X'_1, \cdots, X'_{m'}$. If all of these domain points are distinct (ignoring duplications due to any common prefix of M and M'), we can rest assured that we are free to assign their corresponding range points without constraint. We maintain a set *BAD* to track which domain points have already been determined; initially X_1 and X'_1 are the only such points, since future values will depend on random choices not yet made. Of course if $k > 0$ then $X_1 = X'_1$ and *BAD* contains only one value. Next we begin randomly choosing range points; if ever any such choice leads to a value already contained in the *BAD* set, we set the flag *bad* to `true`.

We now bound the probability of the event that $bad = \texttt{true}$ by analyzing our game. The variable *bad* can be set `true` in lines 7, 9, 14, and 19. In each case it is required that some Y_i was selected such that $Y_i \oplus M_{i+1} \in BAD$ (or possibly that some Y'_i was selected such that $Y'_i \oplus M'_{i+1} \in BAD$). The set *BAD* begins with at most 2 elements and then grows by 1 with each random choice of Y_i or Y'_i. We know that on the ith random choice in the game the *BAD* set will contain at most $i + 1$ elements. And so each random choice of Y_i (resp. Y'_i)

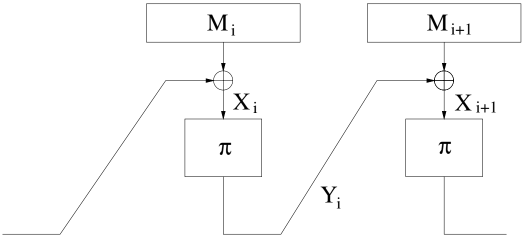

Fig. 6. *The labeling convention used in the proof of Lemma 3.*

from the co-range of π will cause $Y_i \oplus M_{i+1}$ (resp. $Y'_i \oplus M'_{i+1}$) to be in *BAD* with probability at most $(i+1)/(N-i+1)$. We have already argued that in the absence of *bad* = `true` each of the random choices we make are independent. We make $m-1$ choices of Y_i to produce X_2 through X_m and $m'-1$ choices of Y'_i to determine X'_2 through $X'_{m'}$ and so we can compute

$$\Pr[\mathit{bad} = \texttt{true}] \ \le \ \sum_{i=1}^{m-1+m'-1} \frac{i+1}{N-i+1}.$$

Using the fact that $m, m' \le N/4$, we can bound the above by

$$\sum_{i=1}^{m+m'-2} \frac{i+1}{N-i} \ \le \ \frac{2}{N} \sum_{i=1}^{m+m'-2} i+1 \ \le \ \frac{(m+m')^2}{N} .$$

This completes the proof. ▌

Fix $n \ge 1$ and let $F\colon \mathsf{Key} \times \Sigma^n \to \Sigma^n$ be a family of functions. Let ECBC$[F]$ be the family of functions $\mathrm{ECBC}_{f1,f2,f3}(\cdot)$ indexed by $\mathsf{Key} \times \mathsf{Key} \times \mathsf{Key}$. We now use the above to show that ECBC[Perm(n)] is close to being a random function.

Theorem 1. [ECBC ≈ Rand] *Fix $n \ge 1$ and let $N = 2^n$. Let $\mathcal{A}$ be an adversary which asks at most q queries each of which is at most mn-bits. Assume $m \le N/4$. Then*

$$\Pr[\pi_1, \pi_2, \pi_3 \overset{R}{\leftarrow} \mathrm{Perm}(n) :\ \mathcal{A}^{\mathrm{ECBC}_{\pi_1,\pi_2,\pi_3}(\cdot)} = 1] \ - \\ \Pr[R \overset{R}{\leftarrow} \mathrm{Rand}(\Sigma^*, n) :\ \mathcal{A}^{R(\cdot)} = 1] \ \le \ \frac{q^2}{2} V_n(m,m) + \frac{q^2}{2N} \ \le \ \frac{(2m^2+1)q^2}{N}$$

Proof. We first compute a related probability where the final permutation is a random function; this will simplify the analysis. So we are interested in the quantity

$$\Pr[\pi_1 \overset{R}{\leftarrow} \mathrm{Perm}(n);\ \rho_2, \rho_3 \overset{R}{\leftarrow} \mathrm{Rand}(n,n) :\ \mathcal{A}^{\mathrm{ECBC}_{\pi_1,\rho_2,\rho_3}(\cdot)} = 1] \\ - \Pr[R \overset{R}{\leftarrow} \mathrm{Rand}(\Sigma^*, n) :\ \mathcal{A}^{R(\cdot)} = 1] .$$

For economy of notation we encapsulate the initial parts of our experiments into the probability symbols $\Pr_1$ and $\Pr_2$, rewriting the above as

$$\Pr_1[\mathcal{A}^{\mathrm{ECBC}_{\pi_1,\rho_2,\rho_3}(\cdot)} = 1] - \Pr_2[\mathcal{A}^{R(\cdot)} = 1].$$

We condition $\Pr_1$ on whether a collision occurred within π_1, and we differentiate between collisions among messages whose lengths are a nonzero multiple of n (which are not padded) and other messages (which are padded). Let $\mathsf{UnpadCol}$ be the event that there is a collision among the unpadded messages and let PadCol be the event that there is a collision among the padded messages. We rewrite the above as

$$\begin{aligned}&\Pr_1[\mathcal{A}^{\mathrm{ECBC}_{\pi_1,\rho_2,\rho_3}(\cdot)} = 1 \mid \mathsf{UnpadCol} \vee \mathsf{PadCol}]\ \Pr_1[\mathsf{UnpadCol} \vee \mathsf{PadCol}]\\ &+ \Pr_1[\mathcal{A}^{\mathrm{ECBC}_{\pi_1,\rho_2,\rho_3}(\cdot)} = 1 \mid \overline{\mathsf{UnpadCol} \vee \mathsf{PadCol}}]\ \Pr_1[\overline{\mathsf{UnpadCol} \vee \mathsf{PadCol}}]\\ &- \Pr_2[\mathcal{A}^{R}(\cdot) = 1].\end{aligned}$$

Observe that in the absence of event ($\mathsf{UnpadCol} \vee \mathsf{PadCol}$), the adversary sees the output of a random function on distinct points; that is, she is presented with random, uncorrelated points over the range Σ^n. Therefore we know

$$\Pr_1[\mathcal{A}^{\mathrm{ECBC}_{\pi_1,\rho_2,\rho_3}(\cdot)} = 1 \mid \overline{\mathsf{UnpadCol} \vee \mathsf{PadCol}}] \;=\; \Pr_2[\mathcal{A}^{R(\cdot)} = 1].$$

Bounding $\Pr_1[\overline{\mathsf{UnpadCol} \vee \mathsf{PadCol}}]$ and $\Pr_1[\mathcal{A}^{\mathrm{ECBC}_{\pi_1,\rho_2,\rho_3}(\cdot)} = 1 \mid \mathsf{UnpadCol} \vee \mathsf{PadCol}]$ by 1, we reduce the bound on the obtainable advantage to $\Pr_1[\mathsf{UnpadCol} \vee \mathsf{PadCol}] \;\le\; \Pr_1[\mathsf{UnpadCol}] + \Pr_1[\mathsf{PadCol}]$. To bound this quantity, we break each event into the disjoint union of several other events; define $\mathsf{UnpadCol}_i$ to be the event that a collision in π_1 occurs for the first time as a result of the ith unpadded query. Let q_u be the number of unpadded queries made by the adversary, and let q_p be the number of padded queries. Then

$$\Pr_1[\mathsf{UnpadCol}] = \sum_{i=1}^{q_u} \Pr_1[\mathsf{UnpadCol}_i]\,.$$

Now we compute each of the probabilities on the righthand side above. If we know no collisions occurred during the first $i-1$ queries we know the adversary has thus far seen only images of distinct inputs under a random function. Any adaptive strategy she adopts in this case could be replaced by a non-adaptive strategy where she pre-computes her queries under the assumption that the first $i-1$ queries produce random points. In other words, any adaptive adversary can be replaced by a non-adaptive adversary which does at least as well. A non-adaptive strategy would consist of generating all q inputs in advance, and from the collision bound on the hash family we know $\Pr_1[\mathsf{UnpadCol}_i] \;\le\; (i-1)V_n(m,m)$. Summing over i we get

$$V_n(m,m)\sum_{i=1}^{q_u}(i-1) \;\le\; \frac{q_u^2}{2}V_n(m,m)\,.$$

Repeating this analysis for the padded queries we obtain an overall bound of

$$\frac{q_u^2}{2} V_n(m,m) + \frac{q_p^2}{2} V_n(m,m) \;\le\; \frac{q^2}{2} V_n(m,m)\,.$$

Finally we must replace the PRFs ρ_2 and ρ_3 with the PRPs π_2 and π_3. Using Lemma 1 this costs an extra $q_u^2/2N + q_p^2/2N \le q^2/2N$, and so the bound becomes

$$\frac{q^2}{2} V_n(m,m) + \frac{q^2}{2N}\,,$$

and if we apply the bound on $V_n(m,m)$ from Lemma 3 we have

$$\frac{q^2}{2}\,\frac{4m^2}{N} + \frac{q^2}{2N} \;\le\; \frac{(2m^2+1)q^2}{N}\,. \quad \blacksquare$$

Tightness, and non-tightness, of our bound. Employing well-known techniques (similar to [11]), it is easy to exhibit a known-message attack which forges with high probability (> 0.3) after seeing $q = 2^{n/2}$ message/tag pairs. In this sense the analysis looks tight. The same statement can be made for the FCBC and XCBC analyses. But if we pay attention not only to the number of messages MACed, but also their lengths, then none of these analyses is tight. That is, because the security bound degrades quadratically with the total number of message blocks, while the the attack efficiency improves quadratically with the total number of messages. Previous analyses for the CBC MAC and its variants all shared these same characteristics.

Complexity-Theoretic Result. In the usual way we can now pass from the information-theoretic result to a complexity-theoretic one. For completeness, we state the result, which follows using Lemma 2.

Corollary 1. [ECBC is a PRF] *Fix $n \ge 1$ and let $N = 2^n$. Let $E\colon \mathsf{Key} \times \Sigma^n \to \Sigma^n$ be a block cipher. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{ECBC}[E]}(t,q,mn) \;\le\; \frac{2m^2q^2+q^2}{N} + 3\cdot\mathbf{Adv}^{\mathrm{prp}}_{E}(t',q'), \quad \textit{and}$$

$$\mathbf{Adv}^{\mathrm{mac}}_{\mathrm{ECBC}[E]}(t,q,mn) \;\le\; \frac{2m^2q^2+q^2+1}{N} + 3\cdot\mathbf{Adv}^{\mathrm{prp}}_{E}(t',q')$$

where $t' = t + O(mq)$ and $q' = mq$. $\blacksquare$

It is worth noting that using a very similar argument to Theorem 1 we can easily obtain the same bound for the EMAC construction [3]. This yields a proof which is quite a bit simpler than that found in [10]. We state the theorem here, but omit the proof which is very similar to the preceding.

Theorem 2. [EMAC ≈ Rand] *Fix $n \ge 1$ and let $N = 2^n$. Let $\mathcal{A}$ be an adversary which asks at most q queries, each of which is at most m n-bit blocks,* *where $m \le N/4$. Then*

$$\Pr[\pi, \sigma \stackrel{R}{\leftarrow} \mathrm{Perm}(n) :\ \mathcal{A}^{\mathrm{EMAC}_{\pi,\sigma}(\cdot)} = 1] - \Pr[\rho \stackrel{R}{\leftarrow} \mathrm{Rand}((\Sigma^n)^+, n) :\ \mathcal{A}^{\rho(\cdot)} = 1]$$
$$\le \frac{q^2}{2} V_n(m,m) + \frac{q^2}{2N} \ \le\ \frac{(2m^2+1)q^2}{N} \quad \blacksquare$$

5 Security of FCBC

In this section we prove the security of FCBC, obtaining the same bound we had for ECBC. See Section 2 for a definition of $\mathrm{FCBC}_{\rho_1,\rho_2,\rho_3}(M)$, where $\rho_1, \rho_2, \rho_3 \in \mathrm{Rand}(n,n)$.

Theorem 3. [FCBC ≈ Rand] *Fix $n \ge 1$ and let $N = 2^n$. Let $\mathcal{A}$ be an adversary which asks at most q queries, each of which is at most mn bits. Assume $m \le N/4$. Then*

$$\Pr[\pi_1, \pi_2, \pi_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) :\ \mathcal{A}^{\mathrm{FCBC}_{\pi_1,\pi_2,\pi_3}(\cdot)} = 1] \ -$$
$$\Pr[R \stackrel{R}{\leftarrow} \mathrm{Rand}(\Sigma^*, n) :\ \mathcal{A}^{R(\cdot)} = 1] \ \le\ \frac{q^2}{2} V_n(m,m) + \frac{q^2}{2N} \ \le\ \frac{(2m^2+1)q^2}{N}$$

Proof. Let us compare the distribution on functions

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

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

$$\Pr[\pi_1, \pi_2, \pi_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) :\ \mathcal{A}^{\mathrm{FCBC}_{\pi_1,\pi_2,\pi_3}(\cdot)} = 1]$$
$$= \ \Pr[\pi_1, \pi_2, \pi_3 \stackrel{R}{\leftarrow} \mathrm{Perm}(n) :\ \mathcal{A}^{\mathrm{ECBC}_{\pi_1,\pi_2,\pi_3}(\cdot)} = 1]$$

So the bound of our theorem follows immediately from Theorem 1. ■

Since the bound for FCBC exactly matches the bound for ECBC, Corollary 1 applies to FCBC as well.

6 Security of XCBC

In this section we prove the security of the XCBC construction. See Section 2 for a definition of $\mathrm{XCBC}_{\rho_1,\rho_2,\rho_3}(M)$, where $\rho_1, \rho_2, \rho_3 \in \mathrm{Rand}(n, n)$ and $M \in \Sigma^*$.

We first give a lemma which bounds an adversary's ability to distinguish between a pair of random permutations, $\pi_1(\cdot)$, $\pi_2(\cdot)$, and the pair $\pi(\cdot)$, $\pi(K \oplus \cdot)$, 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.

Lemma 4. [Two permutations from one] *Fix $n \geq 1$ and let $N = 2^n$. Let $\mathcal{A}$ be an adversary which asks at most p queries. Then*

$$\Big| \Pr[\pi \xleftarrow{R} \mathrm{Perm}(n);\, K \xleftarrow{R} \Sigma^n :\ \mathcal{A}^{\pi(\cdot),\, \pi(K \oplus \cdot)} = 1] - \Pr[\pi_1, \pi_2 \xleftarrow{R} \mathrm{Perm}(n) :\ \mathcal{A}^{\pi_1(\cdot),\, \pi_2(\cdot)} = 1] \Big| \leq \frac{p^2}{N} \,.$$

Proof. We use the game shown in Figure 7 to facilitate the analysis. Call the game in that figure Game 1. We play the game as follows: first, the initialization procedure is executed once before we begin. In this procedure we set each range point of π as `undefined`; the notation $\mathrm{Domain}(\pi)$ represents the set of points x where $\pi(x)$ is no longer `undefined`. We use $\mathrm{Range}(\pi)$ to denote the set of points $\pi(x)$ which are no longer `undefined`. We use $\overline{\mathrm{Range}}(\pi)$ to denote $\Sigma^n - \mathrm{Range}(\pi)$. Now, when the adversary makes a query X to her left oracle, we execute the code in procedure $\pi(X)$. When she makes a query X to her right oracle, we execute procedure $\pi(K \oplus X)$. We claim that in this setting we perfectly simulate a pair of oracles where the first is a random permutation $\pi(\cdot)$ and the second is $\pi(K \oplus \cdot)$. To verify this, let us examine each of the procedures in turn.

For procedure $\pi(X)$, we first check to see if X is in the domain of π. We are assuming that the adversary will not repeat a query, but it is possible that X is in the domain of π if the adversary has previously queried the *second* procedure with the string $K \oplus X$. In this case we faithfully return the proper value $\pi(X)$. If her query is not in the domain of π we choose a random element Y from S which we hope to return in response to her query. However, it may be that Y is already in the range of π. Although we always remove from S any value returned from this procedure, it may be that Y was placed in the range of π by the *second* procedure. If this occurs, we choose a random element from the co-range of π and return it. For procedure $\pi(K \oplus \cdot)$ we behave analogously.

Note during the execution of Game 1, we faithfully simulate the pair of functions $\pi(\cdot)$ and $\pi(K \oplus \cdot)$. That is, the view of the adversary would be exactly the same if we had selected a random permutation π and a random n-bit string K and then let her query $\pi(\cdot)$ and $\pi(K \oplus \cdot)$ directly.

Now consider the game we get by removing the shaded statements of Game 1. Call that game Game 2. We claim that Game 2 exactly simulates two random permutations. In other words, the view of the adversary in this game is exactly as if we had given her two independent random permutations π_1 and π_2. Without

```
Initialization:
    $S, T \leftarrow \Sigma^n$;    $K \xleftarrow{R} \Sigma^n$;    for all $X \in \Sigma^n$ do $\pi(X) \leftarrow$ undefined

Procedure $\pi(X)$ :
    if $X \in \text{Domain}(\pi)$ then $bad \leftarrow$ true,  return $\pi(X)$
    $Y \xleftarrow{R} S$
    if $Y \in \text{Range}(\pi)$ then $bad \leftarrow$ true,  $Y \xleftarrow{R} \overline{\text{Range}}(\pi)$
    $\pi(X) \leftarrow Y$;    $S \leftarrow S - \{Y\}$;    return $Y$

Procedure $\pi(K \oplus X)$ :
    if $(K \oplus X) \in \text{Domain}(\pi)$ then $bad \leftarrow$ true,  return $\pi(K \oplus X)$
    $Y \xleftarrow{R} T$
    if $Y \in \text{Range}(\pi)$ then $bad \leftarrow$ true,  $Y \xleftarrow{R} \overline{\text{Range}}(\pi)$
    $\pi(K \oplus X) \leftarrow Y$;    $T \leftarrow T - \{Y\}$;    return $Y$
```

Fig. 7. *Game used in the proof of Lemma 4. With the shaded text in place the game behaves like a pair of functions $\pi(\cdot)$, $\pi(K \oplus \cdot)$. With the shaded text removed the game behaves like a pair of independent random permutations $\pi_1(\cdot)$, $\pi_2(\cdot)$.*

loss of generality, we assume the adversary never repeats a query. Then examining procedure $\pi(X)$, we see each call to procedure $\pi(X)$ returns a random element from Σ^n which has not been previously returned by this procedure. This clearly is a correct simulation of a random permutation $\pi_1(\cdot)$. For procedure $\pi(K \oplus X)$, we offset the query value X with some hidden string K, and then return a random element from Σ^n which has not been previously returned by this procedure. Note in particular that the offset by K has no effect on the behavior of this procedure relative to the previous one. Therefore this procedure perfectly simulates a random independent permutation $\pi_2(\cdot)$.

In both games we sometimes set a variable *bad* to `true` during the execution of the game. In neither case, however, does this have any effect on the values returned by the game.

We name the event that *bad* gets set to `true` as B. This event is well-defined for both Games 1 and 2. Notice that Games 1 and 2 behave identically prior to *bad* becoming `true`. One can imagine having two boxes, one for Game 1 and one for Game 2, each box with a red light that will illuminate if *bad* should get set to `true`. These two boxes are defined as having identical behaviors until the red light comes on. Thus, collapsing the initial parts of the probabilities into the formal symbols $\Pr_1$ and $\Pr_2$, we may now say that $\Pr_1[\mathcal{A}^{\pi(\cdot),\pi(K \oplus \cdot)} = 1 \mid \overline{\mathsf{B}}] = \Pr_2[\mathcal{A}^{\pi_1(\cdot),\pi_2(\cdot)} = 1 \mid \overline{\mathsf{B}}]$. And since Games 1 and 2 behave identically until *bad* becomes `true`, we know $\Pr_1[\mathsf{B}] = \Pr_2[\mathsf{B}]$. We have that

$$\left| \Pr_1[\mathcal{A}^{\pi(\cdot),\pi(K \oplus \cdot)} = 1] - \Pr_2[\mathcal{A}^{\pi_1(\cdot),\pi_2(\cdot)} = 1] \right| =$$

$$\Big|\Pr_1[\mathcal{A}^{\pi(\cdot),\pi(K\oplus\cdot)}=1 \mid \overline{\mathsf{B}}]\cdot\Pr_1[\overline{\mathsf{B}}]+\Pr_1[\mathcal{A}^{\pi(\cdot),\pi(K\oplus\cdot)}=1 \mid \mathsf{B}]\cdot\Pr_1[\mathsf{B}]\ -$$

$$\Pr_2[\mathcal{A}^{\pi_1(\cdot),\pi_2(\cdot)}=1 \mid \overline{\mathsf{B}}]\cdot\Pr_2[\overline{\mathsf{B}}]-\Pr_2[\mathcal{A}^{\pi_1(\cdot),\pi_2(\cdot)}=1 \mid \mathsf{B}]\cdot\Pr_2[\mathsf{B}]\Big|\,.$$

From the previous assertions we know this is equal to

$$\Big|\Pr_1[\mathcal{A}^{\pi(\cdot),\pi(K\oplus\cdot)}=1 \mid \mathsf{B}]\cdot\Pr_1[\mathsf{B}]-\Pr_2[\mathcal{A}^{\pi_1(\cdot),\pi_2(\cdot)}=1 \mid \mathsf{B}]\cdot\Pr_2[\mathsf{B}]\Big| =$$

$$\Big|\Pr_2[\mathsf{B}]\cdot\Big(\Pr_1[\mathcal{A}^{\pi(\cdot),\pi(K\oplus\cdot)}=1 \mid \mathsf{B}]-\Pr_2[\mathcal{A}^{\pi_1(\cdot),\pi_2(\cdot)}=1 \mid \mathsf{B}]\Big)\Big|\ \le\ \Pr_2[\mathsf{B}].$$

Therefore we may bound the adversary's advantage by bounding $\Pr_2[\mathsf{B}]$.

We define p events in Game 2: for $1 \le i \le p$, let B_i be the event that *bad* becomes `true`, for the first time, as a result of the ith query. Thus B is the disjoint union of $\mathsf{B}_1,\ldots,\mathsf{B}_p$ and

$$\Pr_2[\mathsf{B}]=\sum_{i=1}^{p}\Pr_2[\mathsf{B}_i]\,.$$

We now wish to bound $\Pr_2[\mathsf{B}_i]$. To do this, we first claim that adaptivity does not help the adversary to make B happen. In other words, the optimal adaptive strategy for making B happen is no better than the optimal non-adaptive one. Why is this? Adaptivity could help the adversary if Game 2 released information associated to K. But Game 2 has no dependency on K—the variable is not used in computing return values to the adversary. Thus Game 2 never provides the adversary any information that is relevant for creating a good ith query. So let $\mathcal{A}$ be an optimal adaptive adversary for making B happen. By the standard averaging argument there is no loss of generality to assume that $\mathcal{A}$ is deterministic. We can construct an optimal non-adaptive adversary $\mathcal{A}'$ by running $\mathcal{A}$ and simulating two independent permutation oracles. Since Game 2 returns values of the same distribution, for all K, the adversary $\mathcal{A}'$ will do no better or worse in getting B to happen in Game 2 if $\mathcal{A}'$ asks the sequence of questions that $\mathcal{A}$ asked in the simulated run. Adversary $\mathcal{A}'$ can now be made deterministic by the standard averaging argument. The resulting adversary, $\mathcal{A}''$, asks a fixed sequence of queries and yet does just as well as the adaptive adversary $\mathcal{A}$.

Now let us examine the chance that *bad* is set `true` in line 2, assuming the adversary has chosen her queries in advance. As we noted above, this can occur when the adversary asks a query X after having previously issued a query $K\oplus X$ to the *second* procedure. What is the chance that this occurs? We can view the experiment as follows: at most $i-1$ queries were asked of the second procedure; let's name those queries $Q=\{X_1,\cdots,X_{i-1}\}$. We wish to bound the chance that a randomly chosen X will cause $K\oplus X\in Q$. But this is simply asking the chance that $K\in\{X\oplus X_1,\cdots,X\oplus X_{i-1}\}$ which is at most $(i-1)/N$.

What is the chance that *bad* becomes `true` in line 4? This occurs when the randomly chosen Y is already in the range of π. Since Y is removed from S each time the first procedure returns it, this occurs when the *second* procedure returned such a Y and it was then subsequently chosen by the first procedure.

Since at most $i-1$ values were returned by the second procedure, the chance is at most $(i-1)/N$ that this could occur.

The same arguments apply to lines 6 and 8. Therefore we have $\Pr_2[\mathsf{B}_i] \le 2(i-1)/N$, and

$$\Pr_2[\mathsf{B}] \;\le\; \sum_{i=1}^{p} \frac{2(i-1)}{N} \;\le\; \frac{p^2}{N}$$

which completes the proof. ▌

We may now state and prove the theorem for the security of our XCBC construction. The bound follows quickly from the the bound on FCBC and the preceding lemma.

Theorem 4. [XCBC ≈ Rand] *Fix $n \ge 1$ and let $N = 2^n$. Let $\mathcal{A}$ be an adversary which asks at most q queries, each of which is at most mn bits. Assume $m \le N/4$. Then*

$$\Big| \Pr[\pi_1 \overset{R}{\leftarrow} \mathrm{Perm}(n);\, K2, K3 \overset{R}{\leftarrow} \Sigma^n :\; \mathcal{A}^{\mathrm{XCBC}_{\pi_1,K2,K3}(\cdot)} = 1] - \Pr[R \overset{R}{\leftarrow} \mathrm{Rand}(\Sigma^*, n) :\; \mathcal{A}^{R(\cdot)} = 1] \Big|$$

$$\le\; \frac{q^2}{2} V_n(m,m) + \frac{(2m^2+1)q^2}{N} \;\le\; \frac{(4m^2+1)q^2}{N}$$

Proof. By the triangle inequality, the above difference is at most

$$\Big| \Pr[\pi_1,\pi_2,\pi_3 \overset{R}{\leftarrow} \mathrm{Perm}(n) :\; \mathcal{A}^{\mathrm{FCBC}_{\pi_1,\pi_2,\pi_3}(\cdot)} = 1] - \Pr[R \overset{R}{\leftarrow} \mathrm{Rand}(\Sigma^*, n) :\; \mathcal{A}^{R(\cdot)} = 1] \Big|$$

$$+ \Big| \Pr[\pi_1,\pi_2,\pi_3 \overset{R}{\leftarrow} \mathrm{Perm}(n) :\; \mathcal{A}^{\mathrm{FCBC}_{\pi_1,\pi_2,\pi_3}(\cdot)} = 1] - \Pr[\pi_1 \overset{R}{\leftarrow} \mathrm{Perm}(n);\, K2, K3 \overset{R}{\leftarrow} \Sigma^n :\; \mathcal{A}^{\mathrm{XCBC}_{\pi_1,K2,K3}(\cdot)} = 1] \Big|$$

and Theorem 3 gives us a bound on the first difference above. We now bound the second difference. Clearly this difference is at most

$$\Big| \Pr[\pi_1,\pi_2,\pi_3 \overset{R}{\leftarrow} \mathrm{Perm}(n) :\; \mathcal{A}^{\pi_1(\cdot),\pi_2(\cdot),\pi_3(\cdot)} = 1] - \Pr[\pi_1 \overset{R}{\leftarrow} \mathrm{Perm}(n);\, K2, K3 \overset{R}{\leftarrow} \Sigma^n :\; \mathcal{A}^{\pi_1(\cdot),\pi_1(K2\oplus\cdot),\pi_1(K3\oplus\cdot)} = 1] \Big|$$

since any adversary which does well in the previous setting could be converted to one which does well in this setting. (Here we assume that $\mathcal{A}$ makes at most mq total queries of her oracles). Applying Lemma 4 twice we bound the above by $2m^2q^2/N$. Therefore our overall bound is

$$\frac{q^2}{2} V_n(m,m) + \frac{q^2}{2N} + \frac{2m^2q^2}{N} \;\le\; \frac{q^2}{2} V_n(m,m) + \frac{(2m^2+1)q^2}{N}\,.$$

And if we apply the bound on $V_n(m,m)$ from Lemma 3 we have

$$\frac{q^2}{2}\,\frac{4m^2}{N} + \frac{(2m^2+1)q^2}{N} \;\leq\; \frac{(4m^2+1)q^2}{N}$$

as required. ▮

XOR-AFTER-LAST-ENCIPHERING DOESN'T WORK. The XCBC-variant that XORs the second key just *after* applying the final enciphering does *not* work. That is, when $|M|$ is a nonzero multiple of the blocksize, we'd have $\mathrm{MAC}_{\pi,K}(M) = \mathrm{CBC}_\pi(M) \oplus K$. This is no good. In the attack, the adversary asks 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.

ON KEY-SEARCH ATTACKS. If FCBC or XCBC is used with an underlying block cipher (like DES) which is susceptible to exhaustive key search, then the MACs inherit this vulnerability. (The same can be said of ECBC and EMAC, except that the double encryption which these MACs employ would seem to necessitate a meet-in-the-middle attack.) It was such considerations that led the designers of the retail MAC, ANSI X9.19, to suggest triple encryption for enciphering the last block [1]. It would seem to be possible to gain this same exhaustive-key-search strengthening by modifying XCBC to *again* XOR the second key ($K2$ or $K3$) with the result of the last encipherment. (If one is using DES, this amounts to switching to DESX for the last encipherment [9].) We call this variant XCBCX. Likely one could prove good bounds for it in the Shannon model. However, none of this is necessary or relevant if one simply starts with a strong block cipher.

COMPLEXITY-THEORETIC RESULT. We can again pass from the information-theoretic result to the complexity-theoretic one:

Corollary 2. [XMAC is a PRF] *Fix* $n \geq 1$ *and let* $N = 2^n$. *Let* $E\colon \mathsf{Key} \times \Sigma^n \to \Sigma^n$ *be a block cipher. Then*

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathrm{XCBC}[E]}(t,q,mn) \;\leq\; \frac{4m^2q^2+q^2}{N} + 3\cdot\mathbf{Adv}^{\mathrm{prp}}_{E}(t',q'), \quad \textit{and}$$

$$\mathbf{Adv}^{\mathrm{mac}}_{\mathrm{XCBC}[E]}(t,q,mn) \;\leq\; \frac{4m^2q^2+q^2+1}{N} + 3\cdot\mathbf{Adv}^{\mathrm{prp}}_{E}(t',q')$$

where $t' = t + O(mq)$ *and* $q' = mq$. ▮

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 [4]. Thanks to Shai and Mihir Bellare for their comments on an early draft, and to anonymous Crypto '00 reviewers for their useful comments.

The authors were supported under Rogaway's NSF CAREER Award CCR-962540, and under MICRO grants 98-129 and 99-103, funded by RSA Data Security and ORINCON. This paper was written while Rogaway was on sabbatical at the Department of Computer Science, Faculty of Science, Chiang Mai University. Thanks for their always-kind hospitality.

References

1. ANSI X9.19. American national standard — Financial institution retail message authentication. ASC X9 Secretariat – American Bankers Association, 1986.
2. Bellare, M., Kilian, J., and Rogaway, P. The security of the cipher block chaining message authentication code. See `www.cs.ucdavis.edu/~rogaway`. Older version appears in *Advances in Cryptology – CRYPTO '94* (1994), vol. 839 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 341–358.
3. 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.
4. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and Rogaway, P. UMAC: Fast and secure message authentication. In *Advances in Cryptology – CRYPTO '99* (1999), Lecture Notes in Computer Science, Springer-Verlag.
5. Carter, L., and Wegman, M. Universal hash functions. *J. of Computer and System Sciences*, 18 (1979), 143–154.
6. 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.
7. Goldreich, O., Goldwasser, S., and Micali, S. How to construct random functions. *Journal of the ACM 33*, 4 (1986), 210–217.
8. 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.
9. Kilian, J., and Rogaway, P. How to protect DES against exhaustive key search. In *Advances in Cryptology – CRYPTO '96* (1996), vol. 1109 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 252–267.
10. Petrank, E., and Rackoff, C. CBC MAC for real-time data sources. Manuscript 97-10 in `http://philby.ucsd.edu/cryptolib.html`, 1997.
11. Preneel, B., and van Oorschot, P. On the security of two MAC algorithms. In *Advances in Cryptology — EUROCRYPT '96* (1996), vol. 1070 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 19–32.
12. 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.