The Security of the Extended Codebook (XCB) Mode of Operation

David A. McGrew and Scott R. Fluhrer

Cisco Systems, Inc.
170 West Tasman Drive, San Jose, CA 95134
{mcgrew,sfluhrer}@cisco.com

Abstract. The XCB mode of operation was outlined in 2004 as a contribution to the IEEE Security in Storage effort, but no security analysis was provided. In this paper, we provide a proof of security for XCB, and show that it is a secure tweakable (super) pseudorandom permutation. Our analysis makes several new contributions: it uses an algebraic property of XCB's internal universal hash function to simplify the proof, and it defines a *nonce mode* in which XCB can be securely used even when the plaintext is shorter than twice the width of the underlying block cipher. We also show minor modifications that improve the performance of XCB and make it easier to analyze. XCB is interesting because it is highly efficient in both hardware and software, it has no alignment restrictions on input lengths, it can be used in nonce mode, and it uses the internal functions of the Galois/Counter Mode (GCM) of operation, which facilitates design re-use and admits multi-purpose implementations.

1 Introduction

There are several scenarios in which *length-preserving, deterministic encryption* is useful. An encryption method is length-preserving if the ciphertext has exactly the same number of bits as does the plaintext. Such a method must be deterministic, since it is impossible to accommodate random data (such as an initialization vector) within the ciphertext. In some cases, deterministic length-preserving encryption exactly matches the requirements. For example, in some encrypted database applications, determinism is essential in order to ensure a direct correspondence between plaintext values being looked up and previously stored ciphertext values.

In some other cases, there is a length-preservation requirement that makes it impossible to provide all of the security services that are desired. Length-preserving algorithms cannot provide message authentication, since there is no room for a message authentication code, and they cannot meet some strong definitions of confidentiality [1]. Essentially, these algorithms implement a codebook; repeated encryptions of the same plaintext value with the same key result it identical ciphertext values. An adversary gains knowledge about the plaintext by seeing which ciphertext values match, and which do not match. Despite these

C. Adams, A. Miri, and M. Wiener (Eds.): SAC 2007, LNCS 4876, pp. 311–327, 2007.

limitations, in many scenarios it may be desirable to use length-preserving encryption because other methods are unworkable. Length-preservation may allow encryption to be introduced into data processing systems that have already been implemented and deployed. Many network protocols have fixed-width fields, and many network systems have hard limits on the amount of data expansion that is possible. One important example is that of disk-block encryption, which is currently being addressed in the IEEE Security in Storage Working Group [2].

Our goal is to provide the best security possible, given the length-preservation limitation. We require our cipher to be a *pseudorandom permutation*; it is indistinguishable from a uniformly chosen random permutation on the set of messages to a computationally bounded adversary. Because we want our cipher to handle plaintexts whose size may vary, we require the cipher to be a pseudorandom *arbitrary length* permutation: for each of the possible plaintext lengths, the cipher acts as a pseudorandom permutation. To provide as much flexibility as possible, we allow the plaintext lengths to vary even for a single fixed key.

In some cases, some additional data can be associated with the plaintext. By using this data as an input, we can provide better security, by letting each distinct associated data value act as an index into a set of pseudorandom permutations. That is, we require the cipher to be a *pseudorandom arbitrary-length permutation with associated data*: for each plaintext length and each value of the associated data, the cipher acts as a pseudorandom permutation. For maximum flexibility, we allow the length of the associated data field to vary even for a single fixed key. In the disk block example, we can use the block number as the associated data value. This will prevent some attacks which rely on the codebook property, since identical plaintext values encrypted with distinct associated data values give unrelated ciphertext values.

The use of an associated data input to a pseudorandom permutation first appeared in the innovative Hasty Pudding Cipher of Schroeppel [3], where it was called a 'spice', and was given a rigorous mathematical treatment by Liskov, Rivest, and Wagner [4], who called it a 'tweak'. Our security goal follows that of the latter work, with the distinction that we allow the associated data to have an arbitrary length.

1.1 Comparison to Existing Work

Naor and Reingold [5] outlined an mode of operation implementing arbitrary length permutation which used a hash-ECB-hash method in which the hash stages are invertible. Other work used a Feistel approach [6]. These early designs do not include a provision for associated data. More recently, block cipher modes of operation that implement pseudorandom arbitrary-length permutations with associated data have been defined. The first such algorithms to be proven secure were the CMC [7] and EME [8] modes of Halevi and Rogaway. CMC cannot be efficiently pipelined; EME can be pipelined, but lacks flexibility. EME* [9] was designed to address that issue. All of those algorithms use an encrypt-mix-encrypt approach, and do not use universal hashing. ABL4 [10] uses a four-round unbalanced Feistel network. The original XCB version [11], HCTR [12], and

HCH [13], use a hash-CTR-hash approach. PEP [14] and TET [15] make use of the hash-ECB-hash approach. Some of the modes have restrictions on the lengths of their inputs (CMC, EME, PEP), or require length-specific precomputation (TET). A detailed comparison of these modes is beyond the scope of this paper, but we highlight some differences below.

XCB encrypts nw bits of data with only $n + 1$ block cipher invocations and $2n + 6$ multiplications in $GF(2^w)$, where w is the number of bits in the block cipher inputs and outputs. The relative efficiency of these modes of operation depends on the relative efficiency of a block cipher invocation and a multiplication in $GF(2^w)$. In software, XCB is faster than EME* if the time taken by that multiply is less than half of the time taken by the block cipher; otherwise, EME* is faster. Multiplication can be done efficiently using precomputed tables, so that XCB outperforms EME*, but in practice the size of such tables could be undesirable, and the performance of the two modes should be considered roughly equivalent. HCTR is faster than XCB by a single block cipher invocation. ABL4 and PEP are considerably less efficient. XCB uses only a single hash key, which is a significant advantage for software implementations, because it reduces the amount of memory needed to store and encryption or decryption context.

In hardware, XCB has the lowest latency of any of these modes; in this context, latency measures the time between when encryption starts and when the first bit of the ciphertext leaves the circuit. XCB also has the merit that a single circuit can implement both encryption and decryption; the algorithms are equivalent up to a reversal of their subkeys.

XCB is unique in that it has been shown to be secure in nonce mode, and can securely accept plaintexts with lengths between w and $2w$ bits when the associated data contains a nonce. This property allows XCB to protect short plaintexts. An example of an application where that feature is useful is the use of Secure RTP [16] to protect voice over IP with the widely used G.729 voice codec, in which case the plaintext is 20 octets long.

The basic components of XCB are identical to those of the Galois/Counter Mode (GCM) of operation [17], making XCB easy to implement given an implementation of GCM, and making compact GCM/XCB implementations possible.

2 XCB Definition

This section contains the specification for XCB for use with w-bit block ciphers. A typical value is $w = 128$, as with the Advanced Encryption Standard (AES) [21].

2.1 Interface

The encryption operation takes as input a secret key K, a plaintext P, and associated data Z, and outputs a ciphertext C. This operation is denoted as $C = E(K, Z, P)$. The values K, P, Z, and C are bit strings. The length of C is identical to that of P.

The decryption operation takes as input a secret key K, a ciphertext C, and associated data Z, and outputs a plaintext P. This operation is denoted as $P = D(K, Z, C)$. The identity $D(K, Z, E(K, Z, P)) = P$ holds for all values of K and Z.

There are two distinct ways in which XCB can be used. For any fixed value of the key, if all of the values of the associated data Z in all of the encryption operations are distinct, then the plaintext can have a length between w and 2^{39} bits, inclusive. We call this *nonce mode*. Otherwise, if the associated data values are *not* distinct, then the plaintext must have a length between $2w$ and 2^{39} bits, inclusive. We call this *normal mode*.

2.2 Notation

The two primitive functions used in XCB are block cipher encryption and multiplication over the field $GF(2^w)$. The block cipher encryption of the value X with the key K is denoted as $\mathbf{e}(K, X)$, and the block cipher decryption is denoted as $\mathbf{d}(K, X)$. (Note that we reserve the symbols E and D to denote XCB encryption and decryption, respectively.) The number of bits in the inputs and outputs of the block cipher is denoted as w. The multiplication of two elements $X, Y \in GF(2^w)$ is denoted as $X \cdot Y$, and the addition of X and Y is denoted as $X \oplus Y$. Addition in this field is equivalent to the bitwise exclusive-or operation, and the multiplication operation is as defined in GCM [17]. We denote the number of bits in a bit string X as $\#X$.

The function $\mathrm{len}(S)$ returns a $w/2$-bit string containing the nonnegative integer describing the number of bits in its argument S, with the least significant bit on the right. The expression 0^l denotes a string of l zero bits, and $A \| B$ denotes the concatenation of two bit strings A and B. The function $\mathbf{msb}_t(S)$ returns the initial t bits of the string S. We consider bit strings to be indexed starting on the left, so that bit zero of S is the leftmost bit. When S is a bit string and $0 \le a < b < \#S$, we denote as $S[a; b]$ the length $b - a$ substring of S consisting of bits a through b of S. The symbol $\{\}$ denotes the bit string with zero length.

2.3 Definition

The XCB encryption operation is defined in Algorithms 1; the decryption operation is similar and is left implicit. The values K_e, K_d, and K_c can be stored between evaluations of these algorithms, in order to trade off some storage for a decreased computational load.

The function $\mathbf{c} : \{0,1\}^k \times \{0,1\}^w \to \{0,1\}^l$, where the output length l is is bounded by $0 \le l \le 2^{39}$, generates an arbitrary-length output by running the block cipher $\mathbf{e}$ in counter mode, using its w-bit input as the initial counter value. Its definition is

$$\mathbf{c}(K, W, l) = \mathbf{e}(K, W) \| \mathbf{e}(K, \mathrm{incr}(W) \| \ldots \| \mathbf{msb}_t(\mathbf{e}(K, \mathrm{incr}^{s-1}(W)), \qquad (1)$$

where we make the number of bits l in the output an explicit parameter for clarity; $s = \lceil l/w \rceil$ is the number of w-bit blocks in the output and $t = l \bmod w$

Algorithm 1. The XCB encryption operation. Given a key $K \in \{0,1\}^k$, a plaintext $\mathbf{P} \in \{0,1\}^m$ where $m \in [w, 2^{39}]$, and associated data $Z \in \{0,1\}^n$ where $n \in [0, 2^{39}]$, this operation returns a ciphertext $\mathbf{C} \in \{0,1\}^m$.

$H \leftarrow \mathbf{e}(K, 0^w)$
$K_e \leftarrow \mathbf{msb}_k(\mathbf{e}(K, 0^{w-3}\|001)\|\mathbf{e}(K, 0^{w-3}\|010))$
$K_d \leftarrow \mathbf{msb}_k(\mathbf{e}(K, 0^{w-3}\|011)\|\mathbf{e}(K, 0^{w-3}\|100))$
$K_c \leftarrow \mathbf{msb}_k(\mathbf{e}(K, 0^{w-3}\|101)\|\mathbf{e}(K, 0^{w-3}\|110))$
$A \leftarrow \mathbf{P}[\#\mathbf{P} - w; \#\mathbf{P} - 1]$
$B \leftarrow \mathbf{P}[0; \#\mathbf{P} - w - 1]$
$C \leftarrow \mathbf{e}(K_e, A)$
$D \leftarrow C \oplus \mathbf{h}_1(H, Z, B)$
$E \leftarrow B \oplus \mathbf{c}(K_c, D, \#B)$
$F \leftarrow D \oplus \mathbf{h}_2(H, Z, E)$
$G \leftarrow \mathbf{d}(K_d, F)$
return $E\|G$

is number of bits in the trailing block. Here the function $\mathrm{incr} : \{0,1\}^w \to \{0,1\}^w$ is the increment operation that is used to generate successive counter values. This function treats the rightmost 32 bits of its argument as a nonnegative integer with the least significant bit on the right, increments this value modulo 2^{32}. More formally,

$$\mathrm{incr}(X) = X[0; w-33] \,\|\, (X[w-32; w-1] + 1 \bmod 2^{32}), \tag{2}$$

where we rely on the implicit conversion of bit strings to integers.

The functions $\mathbf{h}_1$ and $\mathbf{h}_2$ are defined in terms of the underlying hash function $\mathbf{h}$ as

$$\begin{aligned} \mathbf{h}_1(H, Z, B) &= \mathbf{h}(H, 0^w\|Z, B\|0^{\#B \bmod w+w}) \\ \mathbf{h}_2(H, Z, B) &= \mathbf{h}(H, Z\|0^w, E\|0^{\#B \bmod w}\|\mathrm{len}(Z\|L)\|\mathrm{len}(B)). \end{aligned} \tag{3}$$

The function $\mathbf{h} : \{0,1\}^w \times \{0,1\}^a \times \{0,1\}^c \to \{0,1\}^w, a \in [w, 2^{39}], c \in [0, 2^{39}]$ is defined by $\mathbf{h}(H, A, C) = X_{m+n+1}$, where the variables $X_i \in \{0,1\}^w$ for $i = 0, \ldots, m+n+1$ are defined as

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A^*_m\|0^{w-v})) \cdot H & \text{for } i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & \text{for } i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C^*_n\|0^{w-u})) \cdot H & \text{for } i = m+n \\ (X_{m+n} \oplus (\mathrm{len}(A)\|\mathrm{len}(C))) \cdot H & \text{for } i = m+n+1. \end{cases} \tag{4}$$

Here we let A_i denote the w-bit substring $A[(i-1)w; iw-1]$, and let C_i denote $C[(i-1)w; iw-i]$. In other words, A_i and C_i are the i^{th} blocks of A and C, respectively, if those bit strings are decomposed into w-bit blocks. Here u and v

denote the number of bits in the trailing blocks of A and C, respectively. This function is identical to GHASH, the universal hash that is used as a component of the AES Galois/Counter Mode (GCM) of Operation [17] when $w = 128$.

2.4 Improvements in Our Version

The initial version of XCB appeared on the IACR eprint website in 2004 [11]. Our new definition of XCB incorporates changes that make its security properties easier to analyze. First, only a single hash key is used, which enables algebraic relations about the hash function to be brought to bear during the analysis. This change also benefits software implementations by relieving them of the need to store precomputed tables for an additional hash key. Second, the inputs to the hash functions are slightly rearranged, in order to make use of the properties of the hash function; this strategy is explained through the lemmas and theorems of Section 3.1. Additionally, the new design reorders the operations in a way that makes XCB more amenable to pipelined implementation, by changing the way that plaintexts are mapped to internal variables.

3 Security Analysis

In this section, we analyze the security of XCB in the concrete security model introduced by Bellare et. al. [18], and show that XCB is a secure pseudorandom arbitrary-length permutation with associated data (ALPA), using only the assumption that $\mathbf{e}$ is a secure w-bit pseudorandom permutation, as follows. We review the properties of $\mathbf{h}$ and how they are used in XCB (Section 3.1), then define our security model and analyze security under the assumption that $\mathbf{e}$ is a random permutation (Section 3.2), then bound the security when $\mathbf{e}$ is a pseudorandom permutation (Section 3.3).

3.1 Properties of h and XCB

In this section we describe several properties of the hash function $\mathbf{h}$, and some properties of XCB that follow from them. Foremost, $\mathbf{h}$ is an ϵ-almost xor universal function; loosely speaking, this means that the exclusive-or of any two hash values has a low probability to take on any particular value. We provide a precise definition below.

Definition 1. *A function* $f : \{0,1\}^k \times \{0,1\}^m \rightarrow \{0,1\}^t$ *is* ϵ*-almost xor universal if*

$$\mathbf{P}[f(K, M) \oplus f(K, M') = a \mid K \xleftarrow{R} \{0,1\}^k] \leq \epsilon \tag{5}$$

for all $M \neq M' \in \{0,1\}^m$ *and all* $a \in \{0,1\}^t$

Here the expression $\mathbf{P}[\mathbb{E} \mid \mathbb{F}]$ denotes the probability of the event $\mathbb{E}$ given that the event $\mathbb{F}$ has occurred, and the expression $K \xleftarrow{R} \{0,1\}^k$ means that K is chosen uniformly at random from the set $\{0,1\}^k$. We diverge slightly from the

usual definition for explicitness[1]. This definition extends naturally to the case in which f has multiple arguments, as is the case below.

Lemma 1 (h is ϵ-AXU). *The function* $\mathbf{h}$ *defined by Equation 4 is* $\epsilon_h(l)$*-almost xor universal where* $\epsilon_h(l) = \lceil l/w + 2 \rceil 2^{-w}$ *whenever the inputs* A *and* C *are restricted to so that the sum of their lengths is* l *or fewer bits.*

The proof of this lemma appears in Appendix A.

Because the increment function incr() defined in Equation 2 that is used to generate successive counters does not commute with addition in $GF(2^w)$, we need to establish another property of $\mathbf{h}$, which is related to but slightly different from the ϵ-almost xor universal property.

Theorem 1 (h is unlikely to collide with $\mathrm{incr}^s(\mathbf{h})$**).** *For any* $A, A', C, C', E,$ E' *where either* $A' \neq A$ *or* $C' \neq C$ *or both inequalities hold, and any index* s*,*

$$\mathbf{P}[\mathbf{h}(H, A, C) \oplus E = \mathrm{incr}^s(\mathbf{h}(H, A', C') \oplus E') \mid H \xleftarrow{R} GF(2^w)] \leq \epsilon_h(l), \qquad (6)$$

whenever the inputs A *and* C *are restricted to so that the sum of their lengths is* l *or fewer bits.*

This theorem is proved in Appendix A.

The function $\mathbf{h}(H, A, C)$ has the property that it is linear in terms of its arguments A and C, and this fact is used in the XCB design. We next establish the linear property in Theorem 2, then we use it to show a useful expression for the XCB variables F and C in Theorem 3.

Theorem 2 (h is linear). *For any* $H \in V^w$ *and any* A, A', C *and* C' *such that* $\#A = \#A'$ *and* $\#C = \#C'$*,*

$$\mathbf{h}(H, A, C) \oplus \mathbf{h}(H, A', C') = \mathbf{h}(H, A \oplus A', C \oplus C') \oplus (len(A) \| len(C))) \cdot H.$$

The proof appears in Appendix A.

A simple relationship between F and C follows from this theorem, which is captured in the next lemma. The proof is simple, so we include it in this section.

Theorem 3 (F and C have a simple relation)

$$F = C \oplus \mathbf{g}(H, V(Z), \mathbf{c}(K_c, D, \#B)), \qquad (7)$$

where $\mathbf{g}(H, A, C) = \mathbf{h}(H, A, C) \cdot H$ *and* $V(Z) = (Z \| 0^w) \oplus (0^w \| Z)$*.*

Proof

$$\begin{aligned} F &= C \oplus \mathbf{h}(H, Z \| 0^w, E \| 0^{\#B \bmod w} \| (\mathrm{len}(Z \| 0^w) \| \mathrm{len}(B))) \\ &\quad \oplus \mathbf{h}(H, 0^w \| Z, B \| 0^{\#B \bmod w + w}) \\ &= C \oplus \mathbf{h}(H, (Z \| 0^w) \oplus (0^w \| Z), \mathbf{c}(K_c, D, \mathrm{len}(D))) \cdot H. \end{aligned} \qquad (8)$$

[1] In the standard definition, g would define a hash function family, and the selection of a key K would choose a particular hash function from that family of functions.

Figure 1 illustrates these identities. The lowest term of the function **h**, considered as a polynomial in H, cancels with the term $(\text{len}(A)\|\text{len}(C))) \cdot H$. The presence of $\text{len}(Z\|0^w)\|\text{len}(B)$ as the last block of the last argument to **h** makes the coefficient of H^2 in that polynomial match that of the usual coefficient of H in **h**, and the other coefficients are similarly shifted by one. The presence of the term $0^{\#B \bmod w}$ ensures the alignment of the final w bits. □

During the evaluation of an XCB encryption or decryption operation, the first argument of **h**, during both of its evaluations, has the value 0^w prepended or appended to it. If these 0^w terms had not been incorporated into the design, then the second argument to **g** in Theorem 3 would have been $0^{\#Z}$, and the variable F would have no dependancy on Z during an encryption operation. This aspect of the XCB design utilizes the property captured in the following simple lemma.

Lemma 2. *The function $V(Z)$ defined in Theorem 3 has the property that, for any two distinct values Z and Z', the values $V(Z)$ and $V(Z')$ are distinct.*

The validity of this lemma follows from the fact that $V(Z)$ is an invertible transformation of Z; it is easy to compute Z given $V(Z)$ by considering successive w-bit blocks of V.

evaluation	H^7	H^6	H^5	H^4	H^3	H^2	H
first	0^w	Z_1	Z_2	E_1	$E_2^*\|0^{\#B \bmod w}$	0^w	$\text{len}(Z\|0^w)\|\text{len}(B)$
second	Z_1	Z_2	0^w	E_1	$E_2^*\|0^{\#B \bmod w}$	$\text{len}(Z\|0^w)\|\text{len}(B)$	$\text{len}(Z\|0^w)\|\text{len}(B)$
equivalent	Z_1	$Z_1 \oplus Z_2$	Z_2	S_1	$S_2^*\|0^{\#B \bmod w}$	$\text{len}(Z\|0^w)\|\text{len}(B)$	0^w

Fig. 1. An example of the evaluation of **h** during an XCB encryption operation; the table entries are the coefficients of the terms of H in the column headings. The third row shows the equivalent hash operation as in Equation 8. Z_i and E_i denote the i^{th} blocks of Z and E, respectively, and S_i denotes the i^{th} block of $\mathbf{c}(K_c, D, \#B)$.

Lastly, the function **g** as defined in Theorem 3 is almost xor universal as well, as shown by the following lemma; the proof is in Appendix A.

Lemma 3 (g is almost xor universal). *The function* **g** *is $(\epsilon_h(l) + 2^{-w})$-almost xor universal when its second and third inputs are restricted so that their lengths sum to l or fewer bits.*

3.2 Security in the Ideal Model

In this section, we show that XCB is secure against adaptive chosen plaintext/ciphertext attacks, by showing that it is a secure ALFA under the 'ideal' assumption that **e** is a random permutation. More specifically, we model $\mathbf{e}(K_e, X)$, $\mathbf{e}(K_d, X)$, and $\mathbf{e}(K_c, X)$ as independent random permutations. We first establish our security model.

We denote the set of all functions that map $\{0,1\}^m$ to $\{0,1\}^n$ as $\mathbf{F}_{m,n}$. A *random function* is a function chosen uniformly at random from $\mathbf{F}_{m,n}$. We

denote as $\mathbf{I}_n$ the set of all uniquely invertible functions in $\mathbf{F}_{n,n}$, and a *random permutation* is a function chosen uniformly at random from $\mathbf{I}_n$.

A keyed pseudorandom function (PRF) is a subset $\mathbf{PRF}_{m,n} \subseteq \mathbf{F}_{m,n}$ in which the key selects a particular function from $\mathbf{PRF}_{m,n}$. Similarly, a keyed pseudorandom permutation (PRP) is a subset $\mathbf{PRP}_n \subseteq \mathbf{I}_n$ in which the key selects a particular function from $\mathbf{PRP}_n$. In the following, we assume that a key chosen uniformly at random will choose a function from $\mathbf{PRF}_{m,n}$ or $\mathbf{PRP}_n$ uniformly at random.

To measure the 'pseudorandomness' of a particular keyed pseudorandom function $F \subseteq \mathbf{F}_{m,n}$, we use the conventional indistinguishability experiment in which an adversary is challenged to distinguish the PRF from a random function. The adversary is given access to an oracle that provides an interface to a function $f \in \mathbf{F}_{m,n}$. When the adversary provides an input $x \in \{0,1\}^m$ to the oracle, the oracle returns $f(x)$. The adversary is free to choose the inputs adaptively. At the outset of the experiment, the oracle makes a choice to either select f from F or from $\mathbf{F}_{m,n}$. This choice is made uniformly at random and kept hidden from the adversary. At the conclusion of the experiment, the adversary guesses from which set the function has been chosen. We view the adversary as a probabilistic algorithm and consider the probability that it will correctly distinguish a PRF from a random function. We let C_F denote the event that the function f was chosen from the PRF F at the outset of the experiment, and let G_F denote the event that the adversary guesses that the function f was chosen from that PRF at the conclusion of the experiment. An adversary's effectiveness at distinguishing F from a random function is measured by the advantage A_F^{PRF} defined as

$$A_F^{\mathrm{PRF}} = \mathbf{P}[G_F \mid C_F] - \mathbf{P}[G_F \mid C_F^c]. \tag{9}$$

Here $\mathbb{E}^c$ denotes the complement of the event $\mathbb{E}$, that is, the event that $\mathbb{E}$ does not occur. An adversary's advantage in distinguishing a PRP P from a random permutation is defined similarly as

$$A_P^{\mathrm{PRP}} = \mathbf{P}[G_P \mid C_P] - \mathbf{P}[G_P \mid C_P^c] \tag{10}$$

where the events C_P and G_P are the analogues of C_F and G_F. In the PRP experiment, we give the attacker access to two oracles, one for P, and one for the inverse function P^{-1}. In this case, q counts the total number of queries. Definition 2 encapsulates these ideas.

Definition 2. *A PRF F is (q,a)-secure if any adversary making at most q oracle queries has advantage A_F^{PRF} that is less than or equal to a. Similarly, a PRP P is (q,a)-secure if any adversary making at most q oracle queries is has advantage A_P^{PRP} that is less than or equal to a.*

The definition of a secure ALPA is identical, taking into account the fact that the adversary is presented with an ALPA oracle instead of a PRP oracle.

We label the XCB internal variables as $\{A_i, B_i, C_i, D_i, E_i, F_i, G_i\}$ for the i^{th} invocation. If the the i^{th} query is to the XCB encryption oracle, then the adversary determines the values of A_i and B_i, and is given E_i and G_i in return.

If the i^{th} query is to the XCB decryption oracle, then the adversary determines the values of E_i and G_i and is given A_i and B_i in return. We assume without loss of generality that the adversary never repeats a query, and never asks for the decryption of a ciphertext value returned by a previous encryption query, and never asks for the encryption of a plaintext value returned by a previous decryption query.

The basic idea behind our proof is that the each of the ciphertext values $B \oplus \mathbf{c}(K_c, D, \#B))\|\mathbf{d}(K_d, F)$ returned from an encryption query are indistinguishable from random as long as the values D and F do not repeat across different invocations of that function, and the functions $\mathbf{c}$ and $\mathbf{e}$ are indistinguishable from random. Similarly, the plaintext values $E \oplus \mathbf{c}(K_c, D, \#B))\|\mathbf{e}(K_e, C)$ returned from a decryption query are indistinguishable from random as long as the values of D and C do not repeat. We handle our use of the PRP $\mathbf{e}$ as a PRF in the standard way, using the PRP-PRF switching lemma [18].

Lemma 4 (e is a good PRF). *If a function f is a (q, a)-secure w-bit PRP, then it is a $(q, a + q(q-1)2^{-w-1})$-secure PRF.*

We also make use of the fact that the outputs of a PRP are unpredictable to an adversary, as given by the following lemma.

Lemma 5 (e is unpredictable). *If $\mathbf{e}$ is a random permutation, an adversary with oracle access to $\mathbf{e}$ can cause the output $\mathbf{e}(X)$ of the i^{th} query to be equal to a particular value Y with probability no greater than $(2^w - i)^{-1}$, for any fixed value of Y.*

We next define an event Ω whose occurrence ensures the security of XCB in the ideal model, as long as $\mathbf{c}$ and $\mathbf{e}$ are secure PRFs. The event Ω is the conjunction of the events $\Omega_1 \cap \Omega_2 \cap \cdots \cap \Omega_i$, where Ω_i is the event that the following conditions hold during the i^{th} query to the ALPA oracle:

1. $D_i \neq \text{incr}^s(D_j)$ for each integer s such that $-\lceil \#P_i/w \rceil + 1 \leq s \leq \lceil \#P_j/w \rceil - 1$, for all $j < i$, and
2. If the i^{th} query is an encryption query, then $F_i \neq F_j$; if it is a decryption query, then $C_i \neq C_j$, for all $j < i$.

The first condition ensures that, during the invocation of the function $\mathbf{c}$, all of the inputs to $\mathbf{e}$ using the key K_c are distinct from all of the inputs that have previously been made with that key.

We next assume that $\Omega_1, \Omega_2, \ldots, \Omega_{i-1}$ have occurred and show that Ω_i will occur with probability close to one. We bound the total length of the data processed during each query as $l > \#P_i + \#Z_i$.

Lemma 6. *For any of the previous queries $j < i$, and for any single value of s such that $-2^w < s < 2^w$,*

$$\mathbf{P}[D_i = \text{incr}^s(D_j)] \leq (2^w - i)^{-1} + \epsilon_h(l), \text{ and}$$
$$\mathbf{P}[D_i = D_j] \leq \epsilon_h(l),$$

given that the events $\Omega_1, \Omega_2, \ldots, \Omega_{i-1}$ have occured. Here j is either an encryption or decryption query.

Proof. We first assume that the i^{th} query is an encryption query, in which case the condition that $D_i = \mathrm{incr}^s(D_j)$ can be expressed as

$$\mathbf{e}(K_e, A_i) \oplus \mathbf{h}_1(H, Z_i, B_i) = \mathrm{incr}^s(\mathbf{e}(K_e, A_j) \oplus \mathbf{h}_1(H, Z_j, B_j)). \quad (11)$$

We first consider the case that the inputs to the first invocation of $\mathbf{h}_1$ are identical during queries i and j; that is, $Z_i = Z_j$ and $B_i = B_j$. In this case $A_i \neq A_j$ because the queries are required to be distinct. In this case, $D_i \neq D_j$ follows from the invertibility of $\mathbf{e}$. From Lemma 5, the probability that $D_i = \mathrm{incr}^s(D_j)$ will occur for some value $s \neq 0$ is at most $(2^w - i)^{-1}$, for any particular values of j and s. Otherwise, if $A_i = A_j$, then the inputs to $\mathbf{h}$ must be distinct, and $\mathbf{P}[D_i = \mathrm{incr}^s(D_j)] \leq \epsilon_h$ follows from Theorem 1; this bound holds for $0 \leq j < i$.

When the i^{th} query is a decryption query, then similar arguments hold by considering the functions $\mathbf{h}_2$ and $\mathbf{d}$ and the variable G instead of the functions $\mathbf{h}_1$ and $\mathbf{e}$ and the variable A. Thus the probability that $D_i = \mathrm{incr}^s(D_j)$ is no more than $(2^w - i)^{-1} + \epsilon_h(l)$, for any values of j and s. □

Lemma 7. $\mathbf{P}[\Omega_i \mid \Omega_1 \cap \Omega_2 \cap \cdots \cap \Omega_{i-1}] \geq 1 - (i-1)\lceil l/w + 2\rceil 2^{1-w}$.

Proof. Since Lemma 6 holds for any value of s, the probability that it holds for *any* value of s in the range under consideration is no more than $((2^w - i)^{-1} + \epsilon_h(l))\lceil (2l)/w - 2\rceil$.

We now assume that $D_i \neq D_j$ and consider the probability that $F_i = F_j$ when the i^{th} query is an encryption query. From Theorem 3, that event is equivalent to the condition that

$$\mathbf{g}(H, V(Z_i), \mathbf{c}(K_c, D_i, \#B_i)) \oplus \mathbf{g}(H, V(Z_j), \mathbf{c}(K_c, D_j, \#B_j)) = C_i \oplus C_j. \quad (12)$$

If XCB is being used in normal mode (as defined in Section 2.1), then $\#B_i, \#B_j \geq w$, and the initial w bits of $\mathbf{c}(K_c, D_i, \#B_i)$ and $\mathbf{c}(K_c, D_j, \#B_j)$ must be distinct, because D_i and D_j are distinct and $\mathbf{e}$ is an invertible function. If XCB is being used in nonce mode (as defined in Section 2.1), then $Z_i \neq Z_j$, in which case $V(Z_i) \neq V(Z_j)$. In either mode, the inputs to the invocations of $\mathbf{g}$ are distinct, and $\mathbf{P}[F_i = F_j \mid D_i \neq D_j] \leq \epsilon_h(l) + 2^{-w}$ from Lemma 3. Thus, when the i^{th} query is an encryption query, and the events $\Omega_1, \Omega_2, \ldots, \Omega_{i-1}$ have occurred, then

$$\begin{aligned}\mathbf{P}[F_i \neq F_j \mid D_i \neq D_j]\mathbf{P}[D_i \neq D_j] &\geq (1 - \epsilon_h(l) - 2^{-w})(1 - \epsilon_h(l) \\ &\geq 1 - 2(\epsilon_h(l) + 2^{-w}). \quad (13)\end{aligned}$$

When the i^{th} query is a decryption query, then similar arguments show that $\mathbf{P}[C_i \neq C_j \mid D_i \neq D_j]\mathbf{P}[D_i \neq D_j]$ is bounded by the same value.

Equation 13 holds for each value of j between 1 and $i-1$, inclusive. Thus $\mathbf{P}[\Omega_i \mid \Omega_1 \cap \Omega_2 \ldots \cap \Omega_{i-1}]$ is at least

$$\begin{aligned}&1 - \sum_{j=1,i-1} 2(\epsilon_h(l) + 2^{-w}) + ((2^w - i)^{-1} + \epsilon_h(l))\lceil (2l)/w - 2\rceil \\ &\geq 1 - (i-1)\epsilon_\Omega. \quad (14)\end{aligned}$$

where $\epsilon_\Omega = \lceil l/w + 2\rceil^2 2^{2-w}$. □

We can now bound the probability of the event Ω.

Lemma 8 (Ω is likely). *The probability* $\mathbf{P}[\Omega]$ *is at least* $1 - i^2\lceil l/w + 2\rceil^2 2^{2-w}$.

Proof. For any set of events $A_1, A_2, \ldots, A_n$ such that $\mathbf{P}[A_1 \cap A_2 \cap \ldots \cap A_n] > 0$,

$$\mathbf{P}[A_1 \cap A_2 \cap \ldots \cap A_n] = \mathbf{P}[A_n \mid A_1 \cap A_2 \cap \ldots \cap A_{n-1}] \times \\ \mathbf{P}[A_{n-1} \mid A_1 \cap \ldots \cap A_{n-2}] \times \mathbf{P}[A_{n-2} \mid A_1 \cap \ldots \cap A_{n-3}] \times \cdots \times \mathbf{P}[A_1].$$

The probability $\mathbf{P}[\Omega_1 \cap \Omega_2 \ldots \cap \Omega_i]$ is thus no less than

$$\prod_{j=1,}^{i} (1 - (j-1)\epsilon_\Omega) \geq (1 - i\epsilon_\Omega)^i \geq 1 - 2i^2\epsilon_\Omega. \tag{15}$$

Theorem 4 (XCB is secure in the ideal model). *If* $\mathbf{e}(K_c, *), \mathbf{e}(K_d, *), \mathbf{e}(K_e, *)$ *are independent random permutations and* H *is chosen uniformly at random, then XCB is a* $(q, q^2\lceil l/w + 2\rceil^2 2^{3-w})$*-secure arbitrary length PRP with associated data with input length between* w *and* l *bits in nonce mode, and with input length between* $2w$ *and* l *bits otherwise.*

Proof. If the event Ω occurs, then the adversaries advantage is no greater than that due to our use of the PRP $\mathbf{e}$ as a PRF. No more than $q\lceil l/w\rceil)$ queries are made to $\mathbf{e}$ in which it needs to be considered as a PRF. The result follows directly from Lemmas 4 and 8.

3.3 Security as a Block Cipher Mode

Up to this point, we have assumed that the function $\mathbf{e}$ is a random permutation, while in fact it is a block cipher. Our next step is to assume that the advantage with which any adversary can distinguish that function from a random permutation is low, and then show that this assumption implies that XCB is an ALPA.

Theorem 5. *If* $\mathbf{e}$ *is a* (q, a)*-secure* w*-bit PRP, then XCB is a* $(q, a + q^2\lceil l/w + 2\rceil^2 2^{3-w} + 22 \cdot 2^{-w})$*-secure* l*-bit arbitrary length PRP with associated data.*

The proof is in the Appendix.

4 Conclusions

We have shown that our version of XCB is secure in the concrete reduction-based security model, whenever it is used with a block cipher that can be regarded as a secure PRP in that model. We also introduced the definition of nonce mode for a pseudorandom permutation, and showed that XCB is secure when used in this mode.

Acknowledgments

We thank the anonymous referees, whose careful reading and constructive comments substantially improved this paper.

References

1. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings of the 38th FOCS, IEEE Computer Society Press, Los Alamitos (1997)
2. IEEE Security in Storage Working Group. Web page, `http://siswg.org`
3. Schroeppel, R.: Hasty Pudding Cipher Specification. In: First AES Candidate Workshop (August 1998), available online at `http://www.cs.arizona.edu/people/rcs/hpc/hpc-spec`
4. Liskov, M., Rivest, R., Wagner, D.: Tweakable Block Ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, Springer, Heidelberg (2002)
5. Naor, M., Reingold, O.: A pseudo-random encryption mode. Manuscript (1997), available from `http://www.wisdom.weizmann.ac.il/naor`
6. Anderson, R., Biham, E.: Two Practical and Provably Secure Block Ciphers: BEAR and LION. In: Proceedings of the Third International Workshop on Fast Software Encryption, Cambridge, UK, pp. 113–120 (1996)
7. Halevi, S., Rogaway, P.: A tweakable enciphering mode. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 482–499. Springer, Heidelberg (2003)
8. Halevi, S., Rogaway, P.: A Parallelizable Enciphering Mode. In: 2004 RSA Conference Cryptography Track. LNCS, Springer, Heidelberg (2004)
9. Halevi, S.: EME*: extending EME to handle arbitrary-length messages with associated data. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 315–327. Springer, Heidelberg (2004)
10. McGrew, D., Viega, J.: Arbitrary block length mode. Standards contribution (2004), available on-line from `http://grouper.ieee.org/groups/1619/email/pdf00005.pdf`
11. McGrew, D., Fluhrer, S.: The Extended Codebook (XCB) Mode of Operation, Cryptology ePrint Archive: Report 2004/278 (October 25, 2004) `http://eprint.iacr.org/2004/278`
12. Wang, P., Feng, D., Wu, W.: HCTR: A variable-input-length enciphering mode. In: Feng, D., Lin, D., Yung, M. (eds.) CISC 2005. LNCS, vol. 3822, pp. 175–188. Springer, Heidelberg (2005)
13. Chakraborty, D., Sarkar, P.: HCH: A new tweakable enciphering scheme using the hash-encrypt-hash approach. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 287–302. Springer, Heidelberg (2006)
14. Chakraborty, D., Sarkar, P.: A new mode of encryption providing a tweakable strong pseudo-random permutation. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047. pp. 293–309. Springer, Heidelberg (2006)
15. Halevi, S.: Invertible Universal Hashing and the TET Encryption Mode. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622. Springer, Heidelberg (2007)
16. Baugher, M., McGrew, D., Naslund, M., Carrara, E., Norrman, K.: The Secure Real-time Transport Protocol. IETF RFC 3711 (March 2004)
17. McGrew, D., Viega, J.: The Galois/Counter Mode of Operation (GCM). NIST Modes of Operation Process (submission) (January 2004), available online at `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/`

18. Bellare, M., Kilian, J., Rogaway, P.: The Security of the Cipher Block Chaining Message Authentication Code. J. Comput. Syst. Sci. 61(3), 362–399 (2000)
19. McGrew, D., Fluhrer, S.: The Extended Codebook (XCB) Mode of Operation, Version 2, IEEE P1619 (submission)
grouper.ieee.org/groups/1619/email/pdf00019.pdf
20. Krawczyk, H.: LFSR-based hashing and authentication. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, Springer, Heidelberg (2004)
21. U.S. National Institute of Standards and Technology. The Advanced Encryption Standard. Federal Information Processing Standard (FIPS) 197, (2002)
22. Biggs, N.: Discrete Mathematics. Oxford University Press, Oxford (1993) (Revised Edition)

A Proofs

In this appendix we provide proofs for some of the Lemmas and Theorems.

Proof (Lemma 1). We consider two distinct inputs (A, C) and (A', C'), then analyze the probability of the event that

$$\mathbf{h}(H, A, C) \oplus \mathbf{h}(H, A', C') = a, \tag{16}$$

for some fixed value $a \in \{0,1\}^w$. We assume that these inputs are formatted as described in Section 2, in which A, C, A', and C' consist of m, n, m', and n' w-bit blocks, respectively, the final blocks of which have lengths v, u, v', and u', respectively. We assume without essential loss of generality that $m+n \geq m'+n'$, and we define $f = m + n - m' - n'$.

We define the blocks $D_i \in \{0,1\}^w$ for $1 \leq i \leq m + n + 1$ as

$$D_i = \begin{cases} A_i & \text{for } i = 1, \ldots, m-1 \\ A_m^* \| 0^{w-v} & \text{for } i = m \\ C_{i-m} & \text{for } i = m+1, \ldots, m+n-1 \\ C_n^* \| 0^{w-u} & \text{for } i = m+n \\ \text{len}(A) \| \text{len}(C) & \text{for } i = m+n+1 \end{cases} \tag{17}$$

$$D'_i = \begin{cases} 0^w & \text{for } i = 1, \ldots, f \\ A_{i-f} & \text{for } i = f, \ldots, f+m'-1 \\ A_{m'}^* \| 0^{w-v} & \text{for } i = f+m' \\ C_{i-f+m'} & \text{for } i = f+m+1, \ldots, f+m'+n'-1 \\ C_{n'}^* \| 0^{w-u} & \text{for } i = f+m'+n' \\ \text{len}(A) \| \text{len}(C) & \text{for } i = f+m'+n'+1 \end{cases} \tag{18}$$

The condition that Equation 16 holds can be expressed as $R(H) = 0$, where the polynomial R of degree at most $m + n + 1$ over $GF(2^w)$ is defined by

$$R(H) = a \oplus \bigoplus_{i=1,}^{m+n+1} (D_i \oplus D'_i) \cdot H^{m+n-i+2}. \tag{19}$$

The polynomial R must be nonzero, that is, at least one of its coefficients must be nonzero, because the pairs (A, C) and (A', C') are distinct. There are at most $m+n+1$ values of $H \in GF(2^w)$ for which $R(H) = 0$ holds. This follows from the fact that an d^{th} degree polynomial over $GF(2^w)$ has at most d distinct roots (this is the fundamental theorem of algebra over a finite field; see, for example, [22, Theorem 15.8.2]), and the fact that R is nonzero. The probability that $R(H) = 0$ holds, given that H is chosen at random from $GF(2^w)$, is $(m + n + 1)/2^w \leq \lceil l/w + 2 \rceil 2^{-w}$, when the cumulative length of the inputs is restricted to l bits.

For each vector D, there is a unique pair (A, C) where both A and C are bit strings as described in Section 1, and vice-versa. This is because the last element of D unambiguously encodes the lengths of both A and C. Thus, the probability that $R(H) = 0$ holds for any two given messages (A, C) and (A', C'), and a given vector a, is equal to the probability that $\mathbf{h}(H, A, C) \oplus \mathbf{h}(H, A', C') = a$. Equation 16 holds with probability $\lceil l/w + 2 \rceil 2^{-w}$ for any given values of $(A, C), (A', C')$, and a. □

Proof (Theorem 1). We let D_i and D'_i be the coefficients defined as in Equation 17, then we define the polynomials R_1 and R_2 as

$$R_1(H) = E \oplus \bigoplus_{i=1,}^{m+n+1} D_i \cdot H^{m+n-i+2} \tag{20}$$

and

$$R_2(H) = E' \oplus \bigoplus_{i=1,}^{m+n+1} D'_i \cdot H^{m+n-i+2}. \tag{21}$$

The condition $\mathbf{h}(H, A, C) \oplus E = \mathrm{incr}^s(\mathbf{h}(H, A', C') \oplus E')$ can be expressed as

$$R_1(H) = \mathrm{incr}^s(R_2(H)) = T \tag{22}$$

for some value of $T \in \{0, 1\}^w$. For any fixed value of T, there are at most $m+n+1$ values of H such that $R_1(H) = T$, and at most $m + n + 1$ values of H such that $R_2(H) = \mathrm{incr}^{-s}(T)$. Thus the number of values of H that satisfy both equations is at most $m + n + 1$. When H is drawn uniformly at random, the chance of choosing one of these values is at most $(m + n + 1)2^{-w} = \epsilon_h(l)$, where l is an upper bound on the total number of bits in A and C. □

Proof (Theorem 2). We consider the evaluation of $\mathbf{h}(H, A, C)$ and $\mathbf{h}(H, A', C')$, and let X_i be as defined in equation 4, and let X'_i be defined similarly, but with X'_i, A'_i, and C'_i replacing X_i, A_i, and C_i, respectively. We define δX_i to be $X_i \oplus X'_i$, δA_i to be $A_i \oplus A'_i$, and δC_i to be $C_i \oplus C'_i$. Then we note that

$$\delta X_i = \begin{cases} 0 & \text{for } i = 0 \\ (\delta X_{i-1} \oplus \delta A_i) \cdot H & \text{for } i = 1, \ldots, m-1 \\ (\delta X_{m-1} \oplus (\delta A^*_m \| 0^{w-v})) \cdot H & \text{for } i = m \\ (\delta X_{i-1} \oplus \delta C_{i-m}) \cdot H & \text{for } i = m+1, \ldots, m+n-1 \\ (\delta X_{m+n-1} \oplus (\delta C^*_n \| 0^{w-u})) \cdot H & \text{for } i = m+n \\ \delta X_{m+n} \cdot H & \text{for } i = m+n+1. \end{cases} \tag{23}$$

In the case $i^{i=m+n+1}$ of the previous equation, the term $(\text{len}(A) \| \text{len}(C)))$ does not appear due to cancelation. Thus $\mathbf{h}(H, A, C) \oplus \mathbf{h}(H, A', C') = \delta X_{m+n+1} = \mathbf{h}(H, A \oplus A', C \oplus C') \oplus (\text{len}(A) \| \text{len}(C))) \cdot H$. □

Proof (Lemma 3). The condition that $\mathbf{g}(H, A, C) \oplus \mathbf{g}(H, A', C') = a$ holds can be expressed as $S(H) = 0$, where the polynomial S of degree at most $m + n + 2$ over $GF(2^w)$ is defined by

$$S(H) = a \oplus \bigoplus_{i=1,}^{m+n+1} (D_i \oplus D'_i) \cdot H^{m+n-i+3}. \tag{24}$$

Here D_i and D'_i are as defined in Equation 17. The result follows from arguments similar to those made for Lemma 1. □

Proof (Theorem 5). We build an **e**-distinguisher out of an XCB distinguisher by implementing XCB by replacing each invocation of **e** and its inverse by a call to the block cipher oracle, running the XCB distinguisher against that XCB implementation. We denote as C_{XCB} the event that the ALPA oracle is chosen to be XCB. If the XCB-distinguisher indicates that it believes that the inputs were created by XCB (that is, the event G_{XCB} occurs), then our E-distinguisher indicates that the block cipher oracle is E (that is, the event $G_{\mathbf{e}}$ occurs).

We proceed by first considering the case that K_c, K_d, K_e, and H are chosen uniformly at random. We call this algorithm RXCB, and we define the events C_{RXCB} and G_{RXCB} analogous to C_{XCB} and G_{XCB}, respectively. Our analysis uses the following facts.

Fact 1. $\mathbf{P}[G_{\mathbf{e}} \mid C_{\mathbf{e}}] = \mathbf{P}[G_{RXCB} \mid C_{RXCB}]$, *because the events* $C_{\mathbf{e}}$ *and* C_{RXCB} *both provide equivalent inputs to the distinguisher, and the distinguishers are identical.*

Fact 2. *For any three events* A, B *and* C *(with* $\mathbf{P}[B] \neq 0$*),*

$$\mathbf{P}[A \mid B] = \mathbf{P}[A \mid B \cap C]\mathbf{P}[C \mid B] + \mathbf{P}[A \mid B \cap C^c]\mathbf{P}[C^c \mid B].$$

Fact 3. *The events* $B^c_{\mathbf{e}} \cap \Omega$ *and* B^c_{RXCB} *provide equivalent inputs to the distinguishers.*

The advantage with which our distinguisher works against **e** is

$$\begin{aligned}
A_{\mathbf{e}} &= \mathbf{P}[G_{\mathbf{e}} \mid C_{\mathbf{e}}] - \mathbf{P}[G_{\mathbf{e}} \mid B^c_{\mathbf{e}}] \\
&= \mathbf{P}[G_{\text{RXCB}} \mid C_{\text{RXCB}}] - \mathbf{P}[G_{\mathbf{e}} \mid B^c_{\mathbf{e}}] \\
&= \mathbf{P}[G_{\text{RXCB}} \mid C_{\text{RXCB}}] - \mathbf{P}[G_{\mathbf{e}} \mid B^c_{\mathbf{e}} \cap \Omega]\mathbf{P}[\Omega \mid B^c_{\mathbf{e}}] - \mathbf{P}[G_{\mathbf{e}} \mid B^c_{\mathbf{e}} \cap \Omega^c]\mathbf{P}[\Omega^c \mid B^c_{\mathbf{e}}] \\
&\geq \mathbf{P}[G_{\text{RXCB}} \mid C_{\text{RXCB}}] - \mathbf{P}[G_{\text{RXCB}} \mid B^c_{\text{RXCB}}] - \mathbf{P}[G_{\mathbf{e}} \mid B^c_{\mathbf{e}} \cap \Omega^c]\mathbf{P}[\Omega^c \mid B^c_{\mathbf{e}}] \\
&= A_{\text{RXCB}} - \mathbf{P}[G_{\mathbf{e}} \mid B^c_{\mathbf{e}} \cap \Omega^c]\mathbf{P}[\Omega^c \mid B^c_{\mathbf{e}}] \\
&\geq A_{\text{RXCB}} - \mathbf{P}[\Omega^c \mid B^c_{\mathbf{e}}],
\end{aligned} \tag{25}$$

using the facts outlined above. Here A_{RXCB} denotes the adversary's advantage at distinguishing RXCB from a randomly chosen ALPA.

We next consider the case in which XCB is used exactly as defined in Algorithm 1, with K_c, K_d, K_e, and H being derived via seven invocations of $\mathbf{e}$, instead of being set to uniformly random values. Consider the experiment of distinguishing XCB from RXCB; from Lemma 4, we know that

$$A_{\mathrm{XCB}} - A_{\mathrm{RXCB}} = 43 \cdot 2^{-w-1}.$$

Combining this result with Equation 25 gives the theorem. □