Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes

Virgil D. Gligor* and Pompiliu Donescu

VDG Inc., 6009 Brookside Drive, Chevy Chase, MD 20815
{gligor,pompiliu}@eng.umd.edu

Abstract. We present the eXtended Ciphertext Block Chaining (XCBC) and the eXtended Electronic Codebook (XECB) encryption schemes or modes of encryption that can detect encrypted-message forgeries with high probability even when used with typical non-cryptographic Manipulation Detection Code (MDC) functions (e.g., bit-wise exclusive-or and cyclic redundancy code (CRC) functions). These modes detect encrypted-message forgeries at low cost in performance, power, and implementation, and preserve both message secrecy and integrity in a single pass over the message data. Their performance and security scale directly with those of the underlying block cipher function. We also present the XECB message authentication (XECB-MAC) modes that have all the operational properties of the XOR-MAC modes (e.g., fully parallel and pipelined operation, incremental updates, and out-of-order verification), and have better performance. They are intended for use either stand-alone or with encryption modes that have similar properties (e.g., counter-based XOR encryption). However, the XECB-MAC modes have higher upper bounds on the probability of adversary's success in producing a forgery than the XOR-MAC modes.

1 Introduction

No one said this was an easy game !
Paul van Oorschot, March 1999.

A long-standing goal in the design of block encryption modes has been the ability to provide message-integrity protection with simple Manipulation Detection Code (MDC) functions, such as the exclusive-or, cyclic redundancy code (CRC), or even constant functions [5,7,9]. Most attempts to achieve this goal in the face of chosen-plaintext attacks focused on different variations of the Cipher Block Chaining (CBC) mode of encryption, which is the most common block-encryption mode in use. To date, most attempts, including one of our own, failed [8].

* This work was performed while this author was on sabbatical leave from the University of Maryland, Department of Electrical and Computer Engineering, College Park, Maryland 20742.

M. Matsui (Ed.): FSE 2001, LNCS 2355, pp. 92–108, 2002.

In this paper, we define the eXtended Ciphertext Block Chaining (XCBC) modes and the eXtended Electronic Codebook (XECB) encryption modes that can be used with an exclusive-or function to provide the authentication of encrypted messages in a single pass over the data with a single cryptographic primitive (i.e., the block cipher). These modes detect integrity violations at a low cost in performance, power, and implementation, and can be executed in a parallel or pipelined manner. They provide authentication of encrypted messages in real-time, without the need for an additional processing path over the input data. The performance and security of these modes scale directly with the performance and security of the underlying block cipher function since separate cryptographic primitives, such as hash functions, are unnecessary.

We also present the XECB message authentication (i.e., XECB-MAC) modes and their salient properties. The XECB-MAC modes have all the operational properties of the XOR message authentication (XOR-MAC) modes (e.g., they can operate in a fully parallel and pipelined manner, and support incremental updates and out-of-order verification [2]), and have better performance; i.e., they use only about half the number of block-cipher invocations required by the XOR-MAC modes. However, the XECB-MAC modes have higher bounds on the adversary's success of producing a forgery than those of the XOR-MAC modes. The XECB-MAC modes are intended for use either stand-alone to protect the integrity of plaintext messages, or with encryption modes that have similar properties (e.g., counter-based XOR encryption [1] a.k.a "counter mode") whenever it is desired that separate keys be used for secrecy and integrity modes.

2 An Integrity Mode for Encryption

Preliminaries and Notation. In defining the encryption modes we adopt the approach of Bellare *et al.* (viz., [1]), who show that an encryption mode can be viewed as the triple (E, D, KG), where E is the encryption function, D is the decryption function, and KG is the probabilistic key-generation algorithm. (Similarly, a message authentication (MAC) mode can be viewed as the triple (S, V, KG), where S is the message signing function, V is the message verification function, and KG is the probabilistic key-generation algorithm.) Our encryption and authentication modes are implemented with block ciphers, which are modeled with finite families of pseudorandom functions (PRFs) or pseudorandom permutations (PRPs).

In this context, we use the concepts of pseudorandom functions (PRFs), pseudorandom permutations (PRPs), and super-pseudorandom permutations (SPRPs) ([1], [15]). Let $R^{l,L}$ the set of all functions $\{0,1\}^l \to \{0,1\}^L$. We use F to denote either a family of pseudorandom functions or a family of super-pseudorandom permutations, as appropriate (e.g., for the encryption schemes, F will be a family of super-pseudorandom permutations, while for our MAC schemes, F can be a family of pseudorandom functions).

Given encryption scheme $\Pi = (E, D, KG)$ that is implemented with SPRP F, we denote the use of the key $K \xleftarrow{R} KG$ in the encryption of a plaintext string

x by $E^{F_K}(x)$, and in the decryption of ciphertext string y by $D^{F_K}(y)$. The most common method used to detect modifications of encrypted messages applies a MDC function g (e.g., a non-keyed hash, cyclic redundancy code (CRC), bitwise exclusive-or function [16]) to a plaintext message and concatenates the result with the plaintext before encryption with E^{F_K}. A message thus encrypted can be decrypted and accepted as valid only after the integrity check is passed; i.e., after decryption with D^{F_K}, the concatenated value of function g is removed from the plaintext, and the check passes only if this value matches that obtained by applying the MDC function to the remaining plaintext [5,7,16]. If the integrity check is not passed, a special failure indicator, denoted by $Null$ herein, is returned. This method[1] has been used in commercial systems such as Kerberos V5 [18,22] and DCE [6,22], among others. The encryption scheme obtained by using this method is denoted by Π-g = (E-g,D-g,KG), where Π is said to be *composed* with MDC function g. In this mode, we denote the use of the key K in the encryption of a plaintext string x by (E^{F_K}-g)(x), and in the decryption of ciphertext string y by (D^{F_K}-g)(y).

A design goal for Π-g = (E-g, D-g, KG) modes is to find the simplest encryption mode Π = (E,D,KG) (e.g., comparable to the CBC modes) such that, when this mode is composed with a simple, non-cryptographic MDC function g (e.g., as simple as a bitwise exclusive-or function), message encryption is protected against *existential forgeries*. For any key K, a forgery is any ciphertext message that is not the output of E^{F_K}-g. An existential forgery (EF) is a forgery that passes the integrity check of D^{F_K}-g upon decryption; i.e., for forgery y', (D^{F_K}-g)(y') $\neq Null$, where $Null$ is a failure indicator. Note that the plaintext outcome of an existential forgery need not be known to the forgerer. It is sufficient that the receiver of a forged ciphertext decrypt the forgery correctly.

Message Integrity Attack: Existential Forgery in a Chosen-Plaintext Attack. The attack is defined by a protocol between an adversary A and an oracle O[2] as follows.

1. A and O select encryption mode Π-g = (E-g,D-g,KG), and O selects, uniformly at random, a key K of KG.
2. A sends encryption queries (i.e., plaintext messages to be encrypted) x^p, $p = 1, \cdots, q_e$, to the encryption function of O. Oracle O responds to A by returning $y^p = (E^{F_K}\text{-}g)(x^p)$, $p = 1, \cdots, q_e$, where x^p are A's chosen plaintext messages. A records both its encryption queries and O's responses to them.
3. After receiving O's encryption responses, A forges a collection of ciphertexts $y'^i, 1 \leq i \leq q_v$ where $y'^i \neq y^p, \forall p = 1, \cdots, q_e$, and sends each decryption

[1] Note that other methods for protecting the integrity of encrypted messages exist; e.g., encrypting the message with a secret key and then taking the separately keyed MAC of the ciphertext [16,3]. These methods require two passes over the message data, require more power, and are more complex to implement than the modes we envision for most common use. Nevertheless these methods are useful whenever key separation is desired for secrecy and integrity.

[2] O can be viewed as two oracles, the first for the encryption function of O and the second for the decryption function of O.

query y'^i to the decryption function of O. O returns a success or failure indicator to A, depending on whether $(D^{F_K}\text{-}g)(y'^i) \neq Null$.

Adversary A is successful if there is at least one decryption query y'^i such that $(D^{F_K}\text{-}g)(y'^i) \neq Null$ for $1 \leq i \leq q_v$; i.e., y'^i is an existential forgery. The mode Π-g = (E-g,D-g,KG) is said to be secure against a message-integrity attack if the probability of an existential forgery in a chosen-plaintext attack is negligible. (We use the notion of negligible probability in the same sense as that of Naor and Reingold [17].)

Attack Parameters. A is allowed q_e encryption queries (i.e., queries to E^{F_K}-g), and q_v decryption queries (i.e., queries to D^{F_K}-g) totaling $\mu_e + \mu_v$ bits, and taking time $t_e + t_v$.

Parameters q_e, μ_e, t_e are bound by the parameters $(q', \mu', t', \epsilon')$ which define the chosen-plaintext security of Π = (E,D,KG) in a secrecy attack (e.g., in the left-or-right sense [1], for instance), and a constant c' determined by the speed of the function g. Since parameters $(q', \mu', t', \epsilon')$ are expressed in terms of the given parameters (t, q, ϵ) of the SPRP family F, the attack parameters can be related directly to those of the SPRP family F.

Parameters $q_e, \mu_e, t_e, q_v, \mu_v, t_v$ are also bound by the parameters (t, q, ϵ) of the SPRP family F, namely $\mu_e + \mu_v \leq ql$, and $t_e + t_v \leq t$. (The parameters q_e, q_v are determined by μ_e, μ_v.) These parameters can be set to specific values determined by the desired probability of adversary's success. Note that $q_v > 0$ since A must be allowed verification queries. Otherwise, A cannot test whether his forgeries are correct, since A does not know key K.

The message-integrity attack defined above is not weaker than an adaptive one in the sense that the success probability of adversary A bounds from above the success probability of another adversary A' that intersperses the q_e encryption and q_v verification queries; i.e., the adversary is allowed to make his choice of forgery after seeing the result of legitimate encryptions and other forgeries. (This has been shown for chosen-message attacks against MAC functions [2], but the same argument holds here.) To date, this is the strongest of the known goal-attack combinations against the integrity (authentication) of encrypted messages [3,10].

3 Definition of the XCBC and XCBC-XOR Modes

We present three XCBC modes, namely (1) stateless, (2) stateful-sender, and (3) stateful modes, and some implementation options. In general, the fewer state variables the more robust the mode is in the face of failures (or disconnections) and intrusion. This might suggest that, in practice, stateless modes are preferable. However, this may not always be the case because (1) the cost of invoking a good source of randomness for each message can be high (e.g., substantially higher than a single AES block encryption), (2) the random number used in each message encryption by the sender must be securely transmitted to the receiver, which usually costs an additional block-cipher invocation, and (3) the source of

randomness may be hard to protect. The stateful-sender mode (e.g., a counter-based mode) eliminates the need for a good source of randomness but does not always eliminate the extra block-cipher invocation and the need to protect the extra sender state variables; e.g., the source of randomness is replaced by the enciphering of a message counter but the counter must be maintained and its integrity must be protected by the sender across multiple message authentications. Any erroneous or faulty modification of the message counter must trigger re-keying. (The other advantage of counter-based modes, namely the ability to go beyond the "birthday barrier" when used with pseudo-random functions, does not materialize in the context of the AES since AES is modeled as a family of pseudo-random permutations.)

Maintaining secret shared-state variables, as opposed to just sender-state, helps eliminate the extra block-cipher invocations. Extending the shared keying state with extra, per-key, random variables shared by senders and receivers is a fairly straight-forward matter; e.g., these shared variables can be generated and distributed in the same way as the shared secret key, or can be generated using the shared key (at some marginal extra cost per message) by encrypting constants with the shared key. However, maintaining the shared state in the face of failures (or disconnections), and intrusion presents an extra challenge for the mode user; e.g., enlarging the shared state beyond that of a shared secret key may increase the exposure of the mode to physical attacks. The above discussion suggests that none of the three types of operational modes is superior to the others in all environments, and hence all of them should be supported in a general mode definition.

In the encryption modes presented below, the key generation algorithm, KG, outputs a random, uniformly distributed, k-bit string or key K for the underlying $SPRP$ family F, thereby specifying $f = F_K$ and $f^{-1} = F_K^{-1}$ of l-bits to l-bits. If a separate second key is needed in a mode, then a new string or key K' is generated by KG identifying $f' = F_{K'}$ and $f'^{-1} = F_{K'}^{-1}$. The plaintext message to be encrypted is partitioned into a sequence of l-bit blocks (padding is done first, if necessary), $x = x_1 \cdots x_n$. Throughout this paper, $\oplus$ is the *exclusive-or* operator and $+$ represents *modulo* 2^l *addition.*

Stateless XCBC Mode (XCBC$)
The encryption and decryption functions of the stateless mode, $\mathcal{E}{-}XCBC\$^{F_K}(x)$ and $\mathcal{D}{-}XCBC\$^{F_K}(y)$, are defined as follows.

function $\mathcal{E}$−XCBC$$^f(x)$
$r_0 \leftarrow \{0,1\}^l$
$y_0 = f(r_0);\ z_0 = f'(r_0)$
for $i = 1, \cdots, n$ **do** {
$z_i = f(x_i \oplus z_{i-1})$
$y_i = z_i + i \times r_0$ }
return $y = y_0 || y_1 y_2 \cdots y_n$

function $\mathcal{D}$−XCBC$$^f(y)$
Parse y as $y_0 || y_1 \cdots y_n$
$r_0 = f^{-1}(y_0);\ z_0 = f'(r_0)$
for $i = 1, \cdots, n$ **do** {
$z_i = y_i - i \times r_0$
$x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
return $x = x_1 x_2 \cdots x_n$

Stateful-Sender XCBC Mode (XCBCC)

The encryption and decryption functions of the stateful-sender mode, $\mathcal{E}{-}XCBCC^{F_K}(x, ctr)$ and $\mathcal{D}{-}XCBCC^{F_K}(y)$, are defined as follows.

function $\mathcal{E}$−XCBCC$^f(x, ctr)$
$r_0 = f(ctr);\ z_0 = f'(r_0)$
for $i = 1, \cdots, n$ **do** {
$z_i = f(x_i \oplus z_{i-1})$
$y_i = z_i + i \times r_0$ }
$ctr' \leftarrow ctr + 1$
$y = ctr||y_1 y_2 \cdots y_n$
return y

function $\mathcal{D}$−XCBCC$^f(y)$
Parse y as $ctr||y_1 \cdots y_n$
$r_0 = f(ctr);\ z_0 = f'(r_0)$
for $i = 1, \cdots, n$ **do** {
$z_i = y_i - i \times r_0$
$x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
return $x = x_1 x_2 \cdots x_n$

Note that in the XCBCC mode the counter ctr can be initialized to a known constant such as -1 by the sender. ctr' represents the updated ctr value. In both of the above modes the complexity is $n + 2$ block-cipher invocations, where n in the length of input string x in blocks.

Stateful XCBC Mode (XCBCS)

Let IV be a random and uniformly distributed variable that is part of the keying state shared by the sender and receiver.
$\mathcal{E}{-}XCBCS\$^{F_K}(x)$ and $\mathcal{D}{-}XCBCS\$^{F_K}(y)$, are defined as follows.

function $\mathcal{E}$−XCBCS$\$^f(x)$
$r_0 \leftarrow \{0,1\}^l$
$y_0 = f(r_0);\ z_0 = IV + r_0$
for $i = 1, \cdots, n$ **do** {
$z_i = f(x_i \oplus z_{i-1})$
$y_i = z_i + i \times r_0$ }
return $y = y_0||y_1 y_2 \cdots y_n$

function $\mathcal{D}$−XCBCS$\$^f(y)$
Parse y as $y_0||y_1 \cdots y_n$
$r_0 = f^{-1}(y_0);\ z_0 = IV + r_0$
for $i = 1, \cdots, n$ **do** {
$z_i = y_i - i \times r_0$
$x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
return $x = x_1 x_2 \cdots x_n$

Note that in the XCBCS mode the shared IV value can be generated randomly by KG and distributed to the sender and receiver along with key K thereby saving one block cipher invocation, or can be can be generated using key K by standard key-separation techniques thereby requiring an additional block encryption operation per key. In the former case, the complexity of the mode is exactly $n+1$ block-cipher invocations and, in the latter, is *asymptotically* $n + 1$ block-cipher invocations.

Chaining Sequence. The *block chaining sequence* is that used for the traditional CBC mode, namely $z_i = f(x_i \oplus z_{i-1})$, where z_0 is the initialization vector, x_i is the plaintext and z_i is the ciphertext of block $i, i = 1, \cdots, n$. In contrast with the traditional CBC mode, the value of z_i is not revealed outside the encryption modes, and, for this reason, z_i is called a *hidden* ciphertext block. The actual ciphertext output, y_i, of the XCBC modes is defined using extra randomization, namely $y_i = z_i + i \times r_0$, where $i \times r_0$ is the *modulo* 2^l *addition* of the random, uniformly distributed, variable r_0, i times to itself; i.e., $i \times r_0 \stackrel{def}{=} \underbrace{r_0 + \cdots + r_0}_{i\ times}$.

Examples for why the randomization is necessary include those which show that, without randomization, the swapping of two z_i blocks of a ciphertext mes-

sage, or the insertion of two arbitrary but identical blocks into two adjacent positions of a ciphertext message, would cause the decryption of the resulting forgery with probability one whenever an bitwise exclusive-or function is used as the MDC (which is what we intend to use, since these functions are among the fastest known). Correct randomization sequences, such as $i \times r_0$, ensure that, among other things, collisions between any two z_i values is negligible regardless of whether these values are obtained during message encryption, forgery decryption, or both. Note that this probability is negligible even though the randomization sequence $i \times r_0$ allows low-order bits of some z_i's to become known. (A detailed account of why such collisions contribute to an adversary's success in breaking message integrity is provided in the proof of the XCBC$-XOR mode; viz., *http://csrc.nist.gov/encryption/modes/proposedmodes.*) Examples of incorrect randomization sequences can be readily found; e.g., the sequence whereby each element i is computed as an bitwise exclusive-or of i instances of r_0.

Initialization. In *stateless* implementations of the XCBC modes $r_0 \leftarrow \{0,1\}^l$; i.e., r_0 is initialized to a random, uniformly distributed, l-bit value for every message. The value of r_0 is sent by the sender to the receiver as $y_0 = f(r_0)$. In contrast, in *stateful-sender* implementations, which avoid the use of a random number generator, a counter, *ctr*, is initialized to a new l-bit constant (e.g., -1) for every key K, and incremented on every message encryption. In *stateful* implementations, a random initialization-vector value IV that is shared by the sender and receiver is generated for every key K, and used to create a per-message random initialization vector z_0.

In all XCBC modes, the initialization vector z_0 is independent of r_0. While non-independent z_0 and r_0 values might yield secure initialization, simple relationships between these values can lead to the discovery of r_0 with non-negligible probability, and integrity can be easily broken.[3] Since we use z_0 in the definition of function $g(x)$ (see below), z_0 should also be unpredictable so that $g(x)$ has a per-message unpredictable value.

The choice of encrypting r_0 with a second key K' to obtain z_0 (i.e., $z_0 = f'(r_0)$) is made exclusively to simplify both the secrecy [1] and the integrity proofs; e.g., such a z_0 is independent of r_0 and is unpredictable. To eliminate the use of the second key and still satisfy the requirements for z_0 suggested above, we can compute $z_0 = f(r_0 + 1)$ in stateless implementations, whereas in stateful implementations we compute $z_0 = IV + r_0$, where the per-message r_0 can be generated as a random value, or as an encryption of *ctr* in the XORC mode. This eliminates the additional block-cipher invocations necessary in the stateless and stateful-sender modes at the cost of maintaining an extra shared state variable (IV). This choice still satisfies the requirements for z_0.

Generalization. The above method for protecting message integrity against existential forgeries in chosen-plaintext attacks can be generalized as follows. Let the output ciphertext y_i be computed as $y_i = z_i \; op \; E_i$, where *op* is the ran-

[3] As a simple example illustrating why this is the case, let $z_0 = r_0 + 1$, choose x_1 such that $z_0 \oplus x_1 = r_0$ with non-negligible probability, and then compute $y_1 - y_0 = r_0$. With a known r_0, one can cause collisions in the values of z_i and break integrity.

domization operation, E_i are the elements of the randomization sequence, and z_i the hidden ciphertext generated by the encryption mode Π. The encryption mode Π (1) must be secure against adaptive chosen-plaintext attacks with respect to secrecy, and (2) must use the input plaintext blocks x_i to generate the input to f. The PCBC [13,16], and the "infinite garble extension" [5] modes are suitable, but counter-mode/XORC and XOR$ are not (since they fail condition (2)). Operation *op* must be invertible, so $\oplus$, modular 2^l addition and subtraction are appropriate. Elements E_i must be unpredictable such that collisions among z_i's (discussed above) could only occur with negligible probability. Other sequences can be used. For example $E_i = a^i \times r_0$ can be used, where E_i is a linear congruence sequence with multiplier a, where a can be chosen so that the sequence passes spectral tests to whatever degree of accuracy is deemed necessary. (Examples of good multipliers are readily available in the literature [12].)

XCBC-XOR Modes. To illustrate the properties of the XCBC modes in integrity attacks, we choose $g(x) = z_0 \oplus x_1 \oplus \cdots \oplus x_n$ for plaintext $x = x_1 \cdots x_n$, where z_0 is defined as the initialization vector of the mode. In this example, block $g(x)$ is appended to the *end* of a n-block message plaintext x, and hence block $x_{n+1} = z_0 \oplus x_1 \oplus \cdots \oplus x_n$. For this choice of $g(x)$, the integrity check performed at decryption becomes $z_0 \oplus x_1 \oplus \cdots \oplus x_n = f^{-1}(z_{n+1}) \oplus z_n$, where $z_{n+1} = y_{n+1} - (n+1) \times r_0$, and $z_n = y_n - n \times r_0$.

In the XCBC modes padding is done with a standard pattern that always starts with a "1" bit followed by the minimum number of "0" bits necessary to fill the last block of plaintext. If the last block of a message does not need padding, use block $g'(x) = \overline{z_0} \oplus x_1 \oplus \cdots \oplus x_n$ as the x_{n+1} plaintext block, where $\overline{z_0}$ is the bitwise complement of z_0; otherwise, use $g(x) = z_0 \oplus x_1 \oplus \cdots \oplus x_n$. This avoids the extra block encryption which would otherwise be necessary for plaintexts consisting of an integral number of blocks.

The stateless and stateful encryption modes Π-g obtained by the use of schemes Π = XCBC$, Π = XCBCC, or Π = XCBCS with function $g(x) = z_0 \oplus x_1 \oplus \cdots \oplus x_n$ are denoted by XCBC$-$XOR$, XCBCC-$XOR$, and XCBCS-XOR respectively.

Examples of Other Encryption Modes that Preserve Message Integrity.

Katz and Yung [11] proposed an interesting single-pass encryption mode, called the Related Plaintext Chaining (RPC), that is EF-CPA secure when using a non-cryptographic MDC function g consisting only of message start and end tokens. RPC has several important operational advantages, such as full parallelization, incremental updates, out-of-order processing, and low upper bound on the probability of adversary's success in producing a forgery. However, it wastes a substantial amount of throughput since it encrypts the block sequence number and message data in the same block. This may make the selection of modern hash functions as the MDC function g for common encryption modes, such as CBC, a superior performance alternative, at least for sequential implementations. Similarly the use of modern MACs, such as the UMAC, with a separate

key may also produce better overall throughput performance than RPC when used with common encryption modes.

Recently, C.S. Jutla [14] proposed an interesting scheme in which the output blocks z_i of CBC encryption are modified by (i.e., bitwise exclusive-or operations) with a sequence E_i of pairwise independent elements. In this model, $E_i = (i \times IV_1 + IV_2) mod\ p$, where IV_1, IV_2 are random values generated from an initial random value r, and p is prime, and the complexity is $n + 3$, where n is the length of the plaintext input in blocks. In contrast with C.S. Jutla's scheme, the elements of the XCBC sequence, $E_i = (i \times r_0) mod\ 2^l$, are not pairwise independent, and the complexity is $n + 2$ for the stateless and stateful-sender cases, and $n + 1$ for the stateful case. Also, the performance of the required modular 2^l additions is slightly better than that of *mod p* additions, where p is prime. However, the pairwise independence of C.S. Jutla's E_i sequence should yield a slightly tighter bound on the probability of successful forgery illustrating, yet again, a fundamental tradeoff between performance and security. (The bound is tighter by a fraction of a $\log_2$ factor depending on the value of p, which would mean that the attack complexity is within the same order of magnitude of the XCBC bound – viz., Section 5).

More recently, P. Rogaway [20] has proposed other schemes that use interesting variations of non-independent and pairwise-independent elements for the E_i sequence, similar to the sequences presented in this paper and C.S. Jutla's, to achieve $n+1$ complexity. Under the same assumptions regarding stateful and stateless implementations, C.S. Jutla's modes require an extra block enciphering over the XCBC and P. Rogaway's modes. We note that all modes for authenticated encryption include an extra block cipher operation for the enciphering of the exclusive-or MDC; e.g., stateful XCBC and P. Rogaway's OCB mode, which is also stateful, require $n + 2$ block-cipher invocations.

Architecture-Independent Parallel Encryption. C.S. Jutla's recent parallel mode [14] requires that both the input to and output of the block cipher are randomized using a sequence of *pairwise-independent* random blocks. Our fully parallel modes achieve the same effect *without* using a sequence of pairwise-independent random blocks. For these modes, it is sufficient to randomize the input and output blocks of f using the same type of sequence. In this case, the probability of input or output collisions, which would be necessary to break security and integrity respectively, would remain negligible. An example is the *stateful Extended Electronic Codebook-XOR* encryption (XECBS-*XOR*) mode, in which for index $i, 1 \leq i \leq n+1, n = |x|$, the ciphertext block y_i is obtained through the formulae:

$$y_i = f(x_i + ctr \times R + i \times R^*) + ctr \times R + i \times R^*, \quad \forall i, 1 \leq i \leq n, ctr \leq q_e$$
$$y_{n+1} = f(x_{n+1} + ctr \times R) + ctr \times R + (n+1) \times R^*,$$

where R, R^* are two random, uniformly distributed and independent blocks each of l bits in length that are part of the keying state shared by the sender and receiver, and *ctr* is the counter that serves as message identifier. The counter *ctr* is initialized to 1 and increased by 1 on every message encryption up to

q_e, which is the bound of the number of allowable message encryptions (viz., Theorem 4 below). Note that a per-message unpredictable or random nonce is unnecessary, and that the sequence of elements $E_i = ctr \times R + i \times R^*$ can be precomputed for multiple messages, can be computed incrementally [4], and in an out-of-order manner. To provide authentication, the last block is computed using the following formula for the function g:

$$x_{n+1} = g(x) = x_1 \oplus \cdots \oplus x_n.$$

This authenticated encryption mode achieves optimal performance; i.e., $n + 1$ parallel block cipher invocations, and has a throughput close to that of a single block-cipher invocation. The security of the XECBS-XOR mode with respect to confidentiality in an adaptive chosen-plaintext attack can be demonstrated in the same manner as that used for the CBC mode [1].

For the XECBS-XOR encryption scheme proposed above, padding follows the similar conventions as those for the XCBC-XOR modes to distinguish between padded and unpadded messages; i.e., the following formula is used for the enciphering of the last block.

$$y_{n+1} = f(x_{n+1} + ctr \times Z) + ctr \times R + (n + 1) \times R^*,$$

where $Z = \overline{R}$ is the bitwise complement of R and is used for unpadded messages, and $Z = R$ for padded messages.

Stateless architecture-independent parallel modes and stateful-sender architecture-independent parallel modes can be specified in the same manner as those for the XCBC modes. For example, for the stateless mode, an l-bit random number number r_0 is generated and one can replace $ctr \times R$ with r_0 and use $R^* = f(r_0 + 1)$ in the formulae for the stateless encryption mode; r_0 is also enciphered to generate $y_0 = f(r_0)$ which is part of the ciphertext string. In the stateful-sender mode, $r_0 = f(ctr)$, where ctr is an l-bit counter initialized to a constant such as -1; one can replace $ctr \times R$ with r_0 use $R^* = f(r_0)$ in the formulae for the stateful-sender encryption mode. In the modes thus obtained (and other related variants), there would not be any ciphertext chaining, and a priori knowledge of the number of processors would be unnecessary.

As noted earlier, the sequence $E_i = ctr \times R + i \times R^*$ does not completely hide the low order bits of x_i thereby enabling verification of key guesses by an adversary. Resistance to such attacks can be implemented in a similar manner as that of DESX [19], if deemed necessary. However, adoption of modern block ciphers with long keys should reduce the need for this.

4 Definition of the XECB Authentication Modes

In this section, we introduce new Message Authentication Modes (MACs) that counter adaptive chosen-message attacks [2]. We call these MACs the eXtended Electronic Codebook MACs, or XECB-MACs. The XECB-MAC modes have all the properties of the XOR MACs [2], but they do not waste half of the block

size for recording the block identifier thereby avoid doubling the number of block cipher invocations. Many variants of XECB-MACs are possible, and here we present stateless version, XECB$-MAC, a stateful-sender version XECBC-MAC, and a stateful version, the XECBS-MAC.

Message Signing. In both the stateless and stateful-sender implementation, we generate a per-message random value y_0 that is used to randomize each plaintext block of a message x, namely $x_i, 1 \leq i \leq n, n = |x|$, before it is fed to the block cipher function f, where $f = F_K$ is selected from a PRF family F by a key K, which is random and uniform. The result of the randomization is $x_i + i \times y_0$, and the result of block enciphering with f is $y_i = f(x_i + i \times y_0)$. The stateless mode initialization requires a random number generator to create the random block r_0; i.e., $r_0 \leftarrow \{0,1\}^l$. Then $y_0 = f(r_0)$. Stateful-sender implementations avoid the use of the random number generator, and instead, uses a counter ctr, to create y_0 directly, namely $y_0 = f(ctr)$. The counter ctr is initialized by the sender on a per-key basis to a constant, such as -1, and is maintained across consecutive signing requests for the same key K.

For the purposes of simplifying the proofs, we made the following choices for the generation and use of random vector z_0 in both implementations: (1) an additional per-message unpredictable block z_0 is generated and treated as an additional last block of the message plaintext before it is also randomized and enciphered by f, namely $x_{n+1} = z_0$ and $y_{n+1} = f(z_0 + (n+1) \times y_0)$; and (2) we set $z_0 = f'(r_0)$, where $f' = F_{K'}$ is a PRF selected with the second key K'. Clearly, the generation of z_0 can be performed with the same key, K, by block enciphering a simple function of r_0 (e.g., $f(r_0+1)$), and use of K' becomes unnecessary.

The block cipher outputs, $y_1, \cdots, y_n, y_{n+1}$, are exclusive-or-ed to generate the authentication tag $w = y_1 \oplus \cdots \oplus y_n \oplus y_{n+1}$. The algorithm outputs the pair (r_0, w) in the stateless mode, and (ctr, w) in the stateful-sender mode.

We include the stateful-sender version of the XECB MAC modes below. For the (very similar) stateless version, the reader is referred to *http://csrc.nist.gov/encryption/modes/proposedmodes*.

Stateful-Sender XECB-MAC Mode (XECBC-MAC)

function Sign-XECBC-MAC$^f(ctr, x)$
$y_0 = f(ctr), z_0 = f'(y_0)$
$x_{n+1} = z_0$
for $i = 1, \cdots, n+1$ **do** {
$y_i = f(x_i + i \times y_0)$ }
$w = y_1 \oplus \cdots \oplus y_n \oplus y_{n+1}$
$ctr' \leftarrow ctr + 1$
return (ctr, w)

function Verify-XECBC-MAC$^f(x, ctr, w)$
$y_0 = f(ctr), z_0 = f'(y_0)$
$x_{n+1} = z_0$
for $i = 1, \cdots, n+1$ **do** {
$y_i = f(x_i + i \times y_0)$ }
$w' = y_1 \oplus \cdots \oplus y_n \oplus y_{n+1}$
if $w = w'$ **then return** 1
else return 0.

Note that ctr' represents the updated ctr value.

Message Verification. For verification, an adversary submits a forgery $x = x_1 \cdots x_n$ and a forged pair (r_0, w) or (ctr, w) depending upon the mode.[4] Message x is then signed and an authentication tag $w' = y_1 \oplus \cdots \oplus y_n \oplus y_{n+1}$ is generated. The algorithm outputs a bit that is either 1, if the forged authentication tag is correct, namely $w = w'$, or 0, otherwise.

Block-Cipher Invocations and Mode Throughput. The number of block-cipher invocations in the stateless and stateful-sender XECB modes can be reduced from $n + 3$ to $n + 2$ simply by eliminating the enciphering of block x_{n+1}. For example, the enciphering of the last plaintext block can be changed to $y_n = f(x_n \oplus z_0 + n \times y_0)$ (without affecting the proofs significantly). Furthermore, in a stateful version, a random variable R is maintained on the per-key basis, and $z_0 = R + y_0$, as in the XCBCS encryption mode. This would eliminate the extra block enciphering of the function of r_0 for each message, without affecting the proofs. Hence, in a stateful XECB MAC mode, the number of block-cipher invocations becomes $n + 1$, which is one more than that of PMAC [21], which is also a stateful mode. As is the case with PMAC, the throughput of the XECB modes comes close to that corresponding to two sequential block-cipher invocations (as opposed to that of XOR-MAC, which corresponds to that of a single block-cipher invocation).

The following stateful variant of the XECB modes appears to come close to the optimal performance of any parallel MAC, namely n parallel block-cipher invocations and throughput equivalent of a single block-cipher invocation.

Stateful XECB-MAC Mode (XECBS-MAC)
Let R, R^* be two random, uniformly distributed and independent blocks that are part of the keying state shared by the sender and receiver.

function Sign-XECBS-MAC$^f(ctr, x)$
for $i = 1, \cdots, n$ **do** {
$y_i = f(x_i + ctr \times R + i \times R^*)$ }
$w = y_1 \oplus \cdots \oplus y_n$
$ctr' \leftarrow ctr + 1$
return (ctr, w)

function Verify-XECBS-MAC$^f(x, ctr, w)$
if $ctr > q_s$ **then return** 0
for $i = 1, \cdots, n$ **do** {
$y_i = f(x_i + ctr \times R + i \times R^*)$ }
$w' = y_1 \oplus \cdots \oplus y_n$
if $w = w'$ **then return** 1
else return 0.

Note that ctr is initialized to 1, and ctr' represents the updated ctr value.

For the XECBS-MAC mode proposed above, padding follows the similar conventions as those for the XECBS-XOR mode to distinguish between padded and unpadded messages; i.e., the following formula is used for the enciphering of the last block.

$$y_n = f(x_n + ctr \times Z + n \times R^*),$$

where $Z = \overline{R}$ is the bitwise complement of R and is used for unpadded messages, and $Z = R$ for padded messages. Note that, as in the case of the XECBS-XOR

[4] The forgeries (x, r_0, w) or (x, ctr, w) are not previously signed queries. Note also that the length n of the forged message need not be equal to the length of any signed message.

mode, a per-message unpredictable or random nonce is unnecessary, and that the sequence of elements $E_i = ctr \times R + i \times R^*$ can be precomputed for multiple messages, can be computed incrementally, and in an out-of-order manner.

Properties of the XECB Authentication Modes

1. Security. The XECB authentication modes are intended to be secure against adaptive chosen-message attacks [2]. Theorem 2 below shows the security bounds for the stateful-sender mode. The XECB modes, as well as all the other modes that use similar types of randomization sequences, have higher, but still negligible, upper bounds on the adversary's success in producing a forgery than those of the XOR-MAC modes.

2. Parallel and Pipelined Operation. Block-cipher (e.g., AES) computations on different blocks can be made in a *fully parallel* or *pipelined* manner; i.e., it can exploit any degree of parallelism or pipelining available at the sender or receiver without a priori knowledge of the number of processors available.

3. Incremental Updates. The XECB-MAC modes are incremental with respect to block replacement; e.g., a block x_i of a long message is replaced with a new value x'_i. For instance, let us consider the stateful-sender mode. Let the two messages have the same counter *ctr*; hence, the authentication tag of the new message, w', is obtained from the authentication tag of the previous message, w, by the following formula: $w' = w \oplus f(x_i + i \times y_0) \oplus f(x'_i + i \times y_0)$. The replacement property can be easily extended to insertion and deletion of blocks.

4. Out-of-order Verification. The verification of the authentication tag can proceed even if the blocks of the message arrive out of order as long as each block is accompanied by its index and the first block has been retrieved.

5 Security Considerations

The theorems and proofs that demonstrate that these modes are secure with respect to secrecy (e.g., in a left-or-right sense) are similar to those of the CBC mode [1] and, therefore, are omitted. For an XCBC mode, we can determine the $(t', q', \mu', \epsilon')$ secrecy parameters; i.e., an adversary making at most q' queries, totaling at most μ' bits, and taking time t' has an advantage in breaking the secrecy of that that mode (e.g., in a left-or-right sense) that is bounded by a negligible ϵ'.

In establishing the security of the XCBC$ mode against the message-integrity attack, let the parameters used in the attack be bound as follows: $q_e \leq q'$, since the XCBC$ mode is also chosen-plaintext secure, $t_e + t_v \leq t$, and $\mu'' = \mu_e + \mu_v \leq ql$. Let the forgery verification parameters q_v, μ_v, t_v be chosen within the constraints of these bounds and to obtain the desired $Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[Succ]$.

Theorem 1 [Security of XCBC$-$XOR$ against a Message-Integrity Attack].

Suppose F is a (t, q, ϵ)-secure SPRP family with block length l. The mode XCBC$-$XOR$ is secure against a message-integrity attack consisting of $q_e + q_v$ queries, totaling $\mu_e + \mu_v \leq ql$ bits, and taking at most $t_e + t_v \leq t$ time; i.e., the probability of adversary's success is

$$Pr_{f \stackrel{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \leq \epsilon + \frac{\mu_v(\mu_v - l)}{l^2 2^{l+1}} + \frac{q_e(q_e - 1)}{2^{l+1}} + \frac{(q_e + 1)\mu_v}{l2^l} + \frac{\mu_v}{l2^{l+1}}(\log_2 \frac{\mu_v}{l} + 3) + \frac{q_v \mu_e}{l2^l}(\log_2 \frac{\mu_e}{l} + 3).$$

(The proof of Theorem 1 can be found in the full version of the paper available at *http://csrc.nist.gov/encryption/modes/proposedmodes*.) Note that parameters q_e, μ_e, t_e can be easily stated in terms of secrecy parameters $(t', q', \mu', \epsilon')$ above by introducing a constant c' defining the speed of the XOR function.

Theorem 1 above allows us to estimate the complexity of a message-integrity attack.[5] In a successful attack, $Pr_{f \stackrel{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \in (\text{negligible}, 1]$. To estimate complexity, we set the probability of success when $f \stackrel{\mathcal{R}}{\leftarrow} P^l$ to the customary 1/2, and assume that the attack parameters used in the above bound, namely $\frac{\mu_e}{l}, \frac{\mu_v}{l}$, are of the same order or magnitude, namely $2^{\alpha l}$, where $0 < \alpha < 1$. Also, since the shortest message has at least three blocks, $q_e, q_v \leq \lfloor \frac{2^{\alpha l}}{3} \rfloor$. In this case, by setting

$$\frac{q_e(q_e - 1)}{2^{l+1}} + \frac{\mu_v(\mu_v - l)}{l^2 2^{l+1}} + \frac{(q_e + 1)\mu_v}{l2^l} + \frac{\mu_v}{l2^{l+1}}(\log_2 \frac{\mu_v}{l} + 3) + \frac{q_v \mu_e}{l2^l}(\log_2 \frac{\mu_e}{l} + 3) = 1/2,$$

we obtain (by ignoring the $\lfloor . \rfloor$ function) the equation $2^{2\alpha l} \frac{6\alpha l + 34}{9} + 2^{\alpha l} \frac{3\alpha l + 11}{3} = 2^l$, which allows us to estimate α for different values of l. (In this estimate, we can ignore the term in $2^{\alpha l}$ since it is insignificant compared to the other term of the sum.) For example, for $l = 64, \alpha \approx \frac{29}{64}$, for $l = 128, \alpha \approx \frac{61}{128}$, and for $l = 256, \alpha \approx \frac{124}{256}$. Hence, this attack is very close to a square-root attack (i.e., $\alpha \to \frac{1}{2}$ as l increases), and remains this way even is the secrecy bound of Lemma 1 presented in the full version of the paper available at *http://csrc.nist.gov/encryption/modes/proposedmodes* (adjusted for PRPs) is taken into account for integrity (i.e., half of it is added to the integrity bound). Thus the payoff from improved bounds using families of SPRPs is limited.

A variant of Theorem 1 can be proved for the stateful modes. Furthermore, similar theorems hold for single-key stateless modes. The statement and proof for such theorems are similar to the statement and proof for the integrity theorem for the stateless mode, and hence, are omitted.

[5] Technically, the complexity of a successful integrity attack, and the bound of Theorem 1, should account for the success of a secrecy attack; i.e., half of the secrecy bound shown of Lemma 1 presented in the full version of the paper available at *http://csrc.nist.gov/encryption/modes/proposedmodes* (adjusted for the use of PRPs) should be added to the bound in Theorem 1. This is the case because, in general, in modes using the same key for both secrecy and integrity, a successful secrecy attack can break integrity and, vice-versa, a successful integrity attack can break secrecy. (This can be shown using the secrecy and integrity properties of the *IGE* mode; viz., *http://csrc.nist.gov/encryption/modes/proposedmodes*.) As suggested below, the addition of the secrecy bound would not affect the complexity of a successful integrity attack.

The XECB-MAC modes are intended to be secure against adaptive chosen-message attacks [2] consisting of up to q_s signature queries totaling at most μ_s bits and using time up to t_s, and q_v verification queries totaling at most μ_v bits and using time at most t_v. The security of the XECBC-MAC mode is established by the following theorem.

Theorem 2 [Security of XECBC-MAC in an Adaptive Chosen-Message Attack].

Suppose F is a (t, q, ϵ)-secure PRF family with block length l. The message authentication mode (Sign-XECBC-MAC^f, Verify-XECBC-MAC^f, KG) is secure against adaptive chosen-message (q_s, q_v) attacks consisting of $q_s + q_v$ queries totaling $\mu_s + \mu_v \leq ql$ bits and taking at most $t_s + t_v \leq t$ time; i.e., the probability of adversary's success is

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \leq \epsilon + \frac{\mu_v}{l2^l}(\log_2 \frac{\mu_v}{l} + 3) + \frac{q_s \mu_v}{l2^l} + \left(q_s + 2q_v + \frac{\mu_s}{2l}\right) \frac{\mu_s}{l2^{l+1}}(\log_2 \frac{\mu_s}{l} + 3).$$

The proof of this theorem is similar to that of Theorem 1 and is presented in the full version of the paper available at
http://csrc.nist.gov/encryption/modes/proposedmodes.

A similar theorem can be provided for the stateless message authentication mode. The complexity of an attack against XECBC-MAC can be determined in a similar manner to that of an attack against the XCBC$-XOR mode.

We also present a theorem for the security of the XECBS-MAC mode. (The restatement of this theorem in terms of a family of PRPs, such as AES, and the corresponding proof modifications are standard.)

Theorem 3 [Security of XECBS-MAC in an Adaptive Chosen-Message Attack].

Suppose F is a (t, q, ϵ)-secure PRF family with block length l. The message authentication mode (Sign-XECBS-MAC^f, Verify-XECBS-MAC^f, KG) is secure against adaptive chosen-message (q_s, q_v) attacks consisting of $q_s + q_v$ queries $(q_v \leq q_s)$ totaling $\mu_s + \mu_v \leq ql$ bits and taking at most $t_s + t_v \leq t$ time; i.e., the probability of adversary's success is

$$\begin{aligned} Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \leq \epsilon + \frac{q_v}{2^l} + \frac{\mu_v}{l2^{l+1}}(\log_2 \frac{\mu_v}{l} + 3) + \left(q_v + \frac{\mu_s}{l}\right) \frac{q_s}{2^{l+1}}(\log_2 q_s + 3) \\ + \left(q_v + \frac{\mu_s}{l}\right) \frac{\mu_s}{l2^{l+1}}(\log_2 \frac{\mu_s}{l} + 3). \end{aligned}$$

The proof of this theorem is similar to that of Theorems 1 and 2 and is presented in the full version of the paper available at
http://csrc.nist.gov/encryption/modes/proposedmodes.

The security of the XECBS-XOR mode in a message-integrity attack is shown by the theorem bellow.

Theorem 4 [Security of XECBS-XOR in a Message-Integrity Attack].

Suppose F is a (t, q, ϵ)-secure SPRP family with block length l. The mode XECBS-XOR is secure against a message-integrity attack consisting of $q_e + q_v$

queries ($q_v \leq q_e$), totaling $\mu_e + \mu_v \leq ql$ bits, and taking at most $t_e + t_v \leq t$ time; i.e., the probability of adversary's success is

$$Pr_{f \xleftarrow{R} F}[\text{Succ}] \leq \epsilon + \frac{\mu_v(\mu_v - l)}{l^2 2^{l+1}} + \frac{q_v}{2^l} + \frac{\mu_v}{l2^{l+1}}(\log_2 \frac{\mu_v}{l} + 3) + \frac{\mu_e(\mu_e - l)}{l^2 2^{l+1}} + \left(q_v + \frac{\mu_e}{l}\right) \frac{q_e}{2^{l+1}}(\log_2 q_e + 3) + \left(q_v + \frac{\mu_e}{l}\right) \frac{\mu_e}{l2^{l+1}}(\log_2 \frac{\mu_e}{l} + 3).$$

(The proof of Theorem 4 can be found in the full version of the paper available at *http://csrc.nist.gov/encryption/modes/proposedmodes*.) Note that maximum allowable values for q_s and q_e in Theorems 3 and 4 can be determined by setting the probability of successful forgery to a desired value.

Acknowledgments. We thank David Wagner for pointing out an oversight in an earlier version of Theorem 1, Tal Malkin for her thoughtful comments and suggestions, Omer Horvitz and Radostina Koleva for their careful reading of this paper.

References

1. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption," Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE, 1997, (394-403). A full version of this paper is available at *http://www-cse.ucsd.edu/users/mihir*.
2. M. Bellare, R. Guerin, and P.Rogaway, "XOR MACs: New methods for message authentication using finite pseudo-random functions", Advances in Cryptology-CRYPTO '95 (LNCS 963), 15-28, 1995. (Also U.S. Patent No. 5,757,913, May 1998, and U.S. Patent No. 5,673,318, Sept. 1997.)
3. M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm," manuscript, May 26, 2000. *http://eprint.iacr.org/2000.025.ps*.
4. E. Buonanno, J. Katz and M. Yung, "Incremental Unforgeable Encryption, " Proc. Fast Software Encryption 2001, M. Matsui (ed.) (to appear in Springer-Verlag, LNCS).
5. C.M. Campbell, "Design and Specification of Cryptographic Capabilities," in *Computer Security and the Data Encryption Standard*, (D.K. Brandstad (ed.)) National Bureau of Standards Special Publications 500-27, U.S. Department of Commerce, February 1978, pp. 54-66.
6. Open Software Foundation, "OSF - Distributed Computing Environment (DCE), Remote Procedure Call Mechanisms," Code Snapshot 3, Release, 1.0, March 17, 1991.
7. V.D. Gligor and B. G. Lindsay,"Object Migration and Authentication," *IEEE-Transactions on Software Engineering*, SE-5 Vol. 6, November 1979. (Also IBM-Research Report RJ 2298 (3l04), August 1978.)
8. V.D. Gligor, and P. Donescu, "Integrity-Aware PCBC Schemes," in Proc. of the 7th Int'l Workshop on *Security Protocols*, (B. Christianson, B.Crispo, and M. Roe (eds.)), Cambridge, U.K., LNCS 1796, April 2000.

9. R.R. Juneman, S.M. Mathias, and C.H. Meyer, "Message Authentication with Manipulation Detection Codes," Proc. of the IEEE Symp. on Security and Privacy, Oakland, CA., April 1983, pp. 33-54.
10. J. Katz and M. Yung, "Complete characterization of security notions for probabilistic private-key encryption," Proc. of the 32nd Annual Symp. on the Theory of Computing, ACM 2000.
11. J. Katz and M. Yung, "Unforgeable Encryption and Adaptively Secure Modes of Operation," Proc. Fast Software Encryption 2000, B. Schneir (ed.) (to appear in Springer-Verlag, LNCS).
12. D.E. Knuth, "The Art of Computer Programming - Volume 2: Seminumerical Algorithms," Addison-Wesley, 1981 (second edition), Chapter 3.
13. J. T. Kohl, "The use of encryption in Kerberos for network authentication", *Advances in Cryptology-CRYPTO '89* (LNCS 435), 35-43, 1990.
14. C.S. Jutla, "Encryption Modes with Almost Free Message Integrity," IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, manuscript, August 1, 2000. *http://eprint.iacr.org/2000/039.*
15. M Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions", *SIAM J. Computing*, Vol. 17, No. 2, April 1988.
16. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1997.
17. M. Naor and O. Reingold, "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs," *Advances in Cryptology - CRYPTO '98* (LNCS 1462), 267-282, 1998.
18. RFC 1510, "The Kerberos network authentication service (V5)", Internet Request for Comments 1510, J. Kohl and B.C. Neuman, September 1993.
19. P. Rogaway, "The Security of DESX," RSA Laboratories *Cryptobytes*, Vol. 2, No. 2, Summer 1996.
20. P. Rogaway, "OCB Mode: Parallelizable Authenticated Encryption", Preliminary Draft, October 16, 2000, available at *http://csrc.nist.gov/encryption/aes/modes/rogaway-ocb1.pdf.*
21. P. Rogaway, "PMAC: A Parallelizable Message Authentication Mode," Preliminary Draft, October 16, 2000, available at *http://csrc.nist.gov/encryption/aes/modes/rogaway-pmac1.pdf.*
22. S. G. Stubblebine and V. D. Gligor, "On message integrity in cryptographic protocols", Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, 85-104, 1992.