Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes

Virgil D. Gligor* Pompiliu Donescu

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

October 27, 2000

Abstract

We present the eXtended Ciphertext Block Chaining (XCBC) 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., bitwise 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 encryption function. We also present the XECB message authentication modes. These modes 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 [9, 34, 12, 15]. 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 [14].

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

In this paper, we define the eXtended Ciphertext Block Chaining (XCBC) modes that can be used with an exclusive-or function to provide the authentication of encrypted messages in a single pass over the data. 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, and can be executed in a parallel or pipelined manner. The performance and security of these modes scales directly with the performance and security of the underlying block encryption function since separate cryptographic primitives, such as hash functions, are unnecessary. We present some preliminary performance measurements of one of these modes via-a-vis CBC-MD5, CBC-HMAC-SHA1, and CBC-UMAC-STD30.

We also present the XECB modes for message authentication (i.e., XECB-MAC modes) and their salient properties. These message authentication 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 [3], and have better performance. That is, the XECB modes use only about half the number of block encryption 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 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) whenever the 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., [2, 3, 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, pseudorandom permutations (PRPs) and super-pseudorandom permutations (SPRPs) ([2], [21]). Let $R^{l,L}$ the set of all functions $\{0,1\}^l \leftarrow \{0,1\}^L$. We will use F to denote either a family of pseudorandom functions or a family of pseudorandom permutations, as appropriate (e.g., for the encryption schemes, F will be a family of pseudorandom permutations, while for our MAC schemes, F can be a family of pseudorandom functions). A finite family of functions, F, consists of a set of functions and a set of strings (i.e., the set of keys), each string identifying a member function, f. Each function f maps $\{0,1\}^l$ to $\{0,1\}^L$, where l/L denotes the input/output length, and hence we say that F has input/output length l/L. The finite family F is *pseudorandom* if the input-output behavior of a function $f = F_K$, which is identified by key K drawn uniformly at random from the set of keys, "looks random" to someone who does not know K [2]. This means that someone's advantage in distinguishing F from R, which is the set of all functions that map $\{0,1\}^l$ to $\{0,1\}^L$, using q queries of f in time t, is a negligible value, ϵ.

A natural way to model a block cipher is using a family of SPRPs. Let $F : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a pseudorandom permutation family and $f = F_K$ be a permutation randomly chosen by key K (i.e., $K \overset{\mathcal{R}}{\leftarrow} \{0,1\}^k$) and $f^{-1} = F_K^{-1}$ its inverse. Let P^l denote all the permutations on $\{0,1\}^l$, and A be a two-oracle adversary. F is a SPRP if the advantage of function F, $Adv_F^{sprp}(t,q,\mu)$, is

$$Adv_F^{sprp}(t,q,\mu) = \max_A\{Adv_F^{sprp}(A)\} \leq \epsilon,$$

where the maximum is taken over all the adversaries A issuing q enciphering or deciphering queries totalling $\mu = ql$ bits and time t, ϵ is a negligible quantity, and where the advantage of an adversary A is

$$Adv_F^{sprp}(A) = |Pr[A=1 : f, f^{-1} \overset{R}{\leftarrow} F] \Leftrightarrow Pr[A=1 : f, f^{-1} \overset{R}{\leftarrow} P^l]|.$$

Given encryption scheme $\Pi = (E, D, KG)$ that is implemented with SPRP F, we denote the use of the key $K \overset{R}{\leftarrow} KG$ 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 [24]) to a plaintext message and concatenates the result with the plaintext before encryption with $E^{F_K}(x)$. A message thus encrypted can be decrypted and accepted as valid only after the integrity check is passed; i.e., after decryption with $D^{F_K}(y)$, 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 [9, 34, 12, 24]. 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 [28, 30] and DCE [10, 30], 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.

Forgeries can be created in many ways, for example (1) by modifying the ciphertexts of legitimate messages whose plaintext may be known by the forgerer, (2) by including arbitrary, never-seen-before, strings into existing ciphertexts, or (3) by combinations of the two. Ciphertexts of legitimate message encryptions can be obtained as a result of different attack scenarios, such as chosen-plaintext attacks (CPA) or ciphertext-only attacks (CoA). Hence, message integrity attacks can be defined as a combination of attack goals (e.g., EF) and attack scenarios (e.g., CPA), as suggested by Naor [25].

Message Integrity Attack: the EF-CPA Combination. The attack is defined by a protocol between an adversary A and an oracle O 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.

[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 [24, 4]. These methods require two passes over the message data, require more power, 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. A sends encryption queries (i.e., plaintext messages to be encrypted) x^p, $p = 1, \cdot\ \ \cdot, q_e$, to the encryption function of O. O responds to A by returning $y^p = (E^{F_K}\text{-}g)(x^p)$, $p = 1, \cdot\ \ \cdot, q_e$, 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 $y'^i, 1 \quad i \quad q_v$ where $y'^i \neq y^p, \forall p = 1, \cdot\ \ \cdot, q_e$, and sends each decryption query y'^i to the decryption function of O. O returns a success or failure indicator to A, depending on whether of $(D^{F_K}\text{-}g)(y'^i) \neq Null$.

Adversary A is successful if at least one $(D^{F_K}\text{-}g)(y'^i) \neq Null$ for $1 \quad i \quad q_v$; 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 [25].)

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 defining the chosen-plaintext security of Π = (E,D,KG) in a left-or-right sense [1], for instance, and a constant c' defining the speed of the function g. (Briefly, the notion of security in the left-or-right sense allows adversary A to query the encryption function of oracle Q with q' queries of the form (x_l, x_r), where x_l and x_r are equal-length plaintext messages. O flips a coin and decides to encrypt the left or right messages of the q_e queries depending on the outcome of the coin flip. The scheme is considered to be secure if, after receiving the q' encryption queries totaling μ' bits, and taking time t', adversary A cannot obtain a non-negligible advantage (i.e., greater than ϵ') in distinguishing which side of the queries was chosen for encryption by the oracle. Note that this notion of security implies the more intuitive one whereby an adversary is allowed q' chosen-plaintext queries, is given the encryption of a secret plaintext, and is supposed to find that plaintext.) In proving the security of scheme Π in a left-or-right sense, parameters $(q', \mu', t', \epsilon')$ are expressed in terms of the given parameters (t, q, ϵ) 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, μ) of the super-pseudorandom permutation F, namely $\mu_e + \mu_v \quad ql$, and $t_e + t_v \quad t$. (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 [3], 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 [4, 16, 17].

3 Definition of the XCBC and XCBC-XOR Modes

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), $x = x_1 \quad x_n$. 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,
$\mathcal{E}\Leftrightarrow XCBC\$^{F_K}(x)$ and $\mathcal{D}\Leftrightarrow XCBC\$^{F_K}(y)$, are defined as follows.

```
**function** $\mathcal{E}\Leftrightarrow\text{XCBC\$}^f(x)$
$r_0 \leftarrow \{0,1\}^l$
$y_0 = f(r_0);\ z_0 = f'(r_0)$
**for** $i = 1, \quad , n$ **do** {
$z_i = f(x_i \oplus z_{i-1})$
$y_i = z_i + i \quad r_0$ }
**return** $y = y_0||y_1y_2 \quad y_n$
```

```
**function** $\mathcal{D}\Leftrightarrow\text{XCBC\$}^f(y)$
Parse $y$ as $y_0||y_1 \quad y_n$
$r_0 = f^{-1}(y_0);\ z_0 = f'(r_0)$
**for** $i = 1, \quad , n$ **do** {
$z_i = y_i \Leftrightarrow i \quad r_0$
$x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
**return** $x = x_1x_2 \quad x_n$
```

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

```
**function** $\mathcal{E}\Leftrightarrow\text{XCBC}^f(x, ctr)$
$r_0 = f(ctr);\ z_0 = f'(r_0)$
**for** $i = 1, \quad , n$ **do** {
$z_i = f(x_i \oplus z_{i-1})$
$y_i = z_i + i \quad r_0$ }
$ctr' \leftarrow ctr + 1$
$y = ctr||y_1y_2 \quad y_n$
**return** $y$
```

```
**function** $\mathcal{D}\Leftrightarrow\text{XCBC}^f(y)$
Parse $y$ as $ctr||y_1 \quad y_n$
$r_0 = f(ctr);\ z_0 = f'(r_0)$
**for** $i = 1, \quad , n$ **do** {
$z_i = y_i \Leftrightarrow i \quad r_0$
$x_i = f^{-1}(z_i) \oplus z_{i-1}$ }
**return** $x = x_1x_2 \quad x_n$
```

Note that in the XCBC mode the counter *ctr* can be initialized to a known constant such as $\Leftrightarrow 1$ by the sender. ctr' represents the updated *ctr* value.

The encryption modes defined above use the same *block chaining sequence* as 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 $i, i = 1, \quad , n$. 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 mode is defined using extra randomization, namely $y_i = z_i + i \quad r_0$, where $i \quad r_0$ is the *modulo* 2^l *addition* of the random, uniformly distributed, variable r_0, i times to itself; i.e., $i \quad r_0 \overset{def}{=} \underbrace{r_0 + \quad + r_0}_{i\ times}$. (In systems where the modular multiplication with a constant is fast, $i \quad r_0$ can be implemented as a per-block multiplication.) It should be noted that other functions, or combinations of functions, not just the incremental addition modulo 2^l of r_0, could be used to define the ciphertext block sequence y_i; e.g., subtraction modulo 2^l (viz., also *Support for Multiple Encryption Modes* in the next section). Note that these functions may allow the low-order bits of some z_i's to become known. Furthermore, it should be noted that the ciphertext blocks of other block-chaining sequences, such as

Campbell's "infinite garble extension mode," [9] can be modified by $y_i = z_i + i \quad r_0$, not just the CBC ciphertext blocks.

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* implementations, 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 both stateless and stateful implementations, the initialization vector z_0 is set to $f'(r_0)$, which is independent of r_0 and, just as r_0, remains secret. Alternate stateful implementations are possible whereby the counter *ctr* and the secret r_0 are shared by both the sender and receiver. As a consequence, the sender need not compute y_0 and send its value to the receiver. We also note that other functions, not just $f' = F_{K'}$, can be used for generating the secret initialization vector z_0. For instance, $z_0 = f(r_0 + 1)$, in which case only a single key, K, is used. It is important that the encryption of these functions of r_0 produce a pseudorandom value for z_0 that is independent of r_0, and remains secret.

XCBC-XOR Modes. To illustrate the properties of the XCBC modes in integrity attacks, we choose $g(x) = z_0 \oplus x_1 \oplus \quad \oplus x_n$ for plaintext $x = x_1 \quad x_n$, where z_0 is internally defined by both the XCBC$ and XCBC modes. In this example, block $g(x)$ is appended to the *end* of a n-block message plaintext x, and hence block $x_{n+1} = z_0 \oplus x_1 \oplus \quad \oplus x_n$. For this choice of $g(x)$, the integrity check performed at decryption becomes $z_0 \oplus x_1 \oplus \quad \oplus x_n = f^{-1}(z_{n+1}) \oplus z_n$, where $z_{n+1} = y_{n+1} \Leftrightarrow (n+1) \quad r_0$, and $z_n = y_n \Leftrightarrow n \quad r_0$. An adversary is successful if the forged ciphertext produced in the attack defined above passes this check for at least one of the q_v verification queries. Hence, an upper bound for the probability of adversary's success represents a quantitative measure of the integrity properties of the XCBC modes with respect to the choice of function $g(x) = z_0 \oplus x_1 \oplus \quad \oplus x_n$.

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

Examples of Other Encryption Modes that Preserve Message Integrity. Few modes of encryption Π-g, where g is a simple, non-cryptographic MDC function, are known that are EF-CPA secure. The performance characteristics of most of these modes do not satisfy all our goals, however. For example, when implemented with the CBC mode and used to encrypt messages consisting of an integer number of l-bit blocks (possibly after padding), the Variable Input Length (VIL) cipher of Bellare and Rogaway [5, 6] can be shown to be EF-CPA secure when using simple non-cryptographic MDC functions g,[2] such as those for the bitwise exclusive-or, CRCs, addition *modulo* $2^l \Leftrightarrow 1$, the selection of a single constant-filled block or just block x_1 of every message, whose output is appended *to the end* of the message before encryption. However, the VIL cipher uses two sequential passes over its input and, thus, its performance is lower than those of single-pass schemes using hash functions or separate-key MACs.

Katz and Yung [16] 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.[3] However, it wastes a substantial amount of throughput since

[2]This is neither the reason VIL was introduced nor its intended use.

[3]RPC preserves the block ordering in the same way as the XOR-MAC [3]; i.e., it reserves part of every plaintext block for the block sequence number. It also shares the same operational advantages and disadvantages as the XOR-MAC.

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.

More recently, C.S. Jutla [20] 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 S_i of pairwise independent elements. The complexity of this mode is superior to that of both VIL and RPC; i.e., this mode exceeds the complexity of single-pass schemes only by $\Omega(\log n)$ block encryptions, where n is the number of plaintext blocks of a message. This is shown to be a lower bound for a model where the only operations allowed in addition to block encryptions are linear operations over $(GF2)^l$ (i.e., bitwise exclusive-or operations on l-bit blocks). Jutla also proposes a slightly different model that, just as the XCBC modes, also allows modular additions. In this model, $S_i = (i \quad r_0 + r_1) mod\ p$, where r_0, r_1 are random values and p is prime, and the complexity $n+3$. In contrast with Jutla's scheme, the elements of the XCBC sequence, $S_i = (i \quad r_0) mod\ 2^l$, are not pairwise independent, and the complexity is $n + 2$. Also, the performance of the required modular 2^l additions is somewhat better than that of *mod p* additions, where p is prime. However, the pairwise independence of Jutla's S_i sequence should yield a somewhat tighter bound on the probability of successful forgery (i.e., tighter by a fraction of a log_2 factor depending on the value of p), illustrating, yet again, a fundamental tradeoff between performance and security.

4 Properties of the XCBC Modes

The XCBC modes have notable secrecy and integrity properties in several areas.

1. Support for Message Integrity. The XCBC modes require only a single cryptographic primitive, namely the block cipher that is necessary for encryption, to maintain integrity. Further, other functions (i.e., not just the $g(x)$ function defined above), such as the CRCs and modular addition checksums, can also be used with the XCBC modes for protection against message integrity attacks (unlike the original CBC and PCBC modes).

2. Support for Real-Time Message Authentication.

Both the stateless and stateful XCBC modes can be used with $g(x) = z_0 \oplus x_1 \oplus \quad \oplus x_n$ for real-time message sources in which (1) the message length remains unknown until the message ends, (2) the beginning of message authentication cannot be deferred until the end of message receipt, and (3) only small, fix-sized, buffers for authentication processing are available, as would be the case with most low-cost, low-power, hardware implementations. Also the XCBC modes can produce good Message Authentication Codes (MACs). For example, a Double MAC approach [26] can be used for both the XCBC$ (XCBC) modes to obtain good MACs.

3. Support for Multiple Encryption Modes. The definition of the ciphertext generation y_i from the hidden ciphertext block z_i, (i.e., the output of the internal CBC encryption mode), can be changed to obtain other modes of encryption that may be faster or have better security bounds. For example,

- $y_i = z_i \oplus (i \quad r_0)$ in which one of the additions *mod* 2^l per block is replaced by an exclusive-or;

- $y_i = z_i + r_i$, where $r_i = a^i \quad r_0$ is a linear congruence sequence with multiplier a. The multiplier 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 [18]. This mode may have a better upper bound for the probability of breaking the integrity condition.

We also note that the traditional PCBC modes can also be used to generate an XPCBC mode in the same way as the XCBC mode was generated based on the traditional CBC mode above. The conventional initialization-vector attacks defined by Voydock and Kent [33] are also countered by the use of z_0 as the initialization vector.

The XCBC modes capture the history of the message encryption only from the previous block, just as the CBC modes. However, in contrast to the original CBC modes, the XCBC modes add an extra randomization step which is the key ingredient that assures that the integrity check can pass only with low probability.

4. Support for Interleaved-Parallel or Pipelined Encryption. The choice of $g(x) = z_0 \oplus x_1 \oplus \quad \oplus x_n$, allows the interleaved-parallel or pipelined implementation of the XCBC modes. Other non-cryptographic MDC functions $g(x)$ would also allow such implementation, since they be executed in a parallel or a pipelined manner (by definition). This mode is useful when the number of processors available for encryption and decryption in parallel is a priori known or negotiated. For example, for interleaved-parallel execution using $g(x)$, each plaintext message x is partitioned into L segments, $x^{(1)} \quad x^{(L)}$ each of length n_s, $s = 1, \quad , L$, after customary block-level padding (n.b., this L should not be confused with the output length of a PRF, which is typically denoted by L, also). Each segment, $x^{(s)}, s = 1, \quad , L$, consists of one or more l-bit blocks, and if $g(x^{(s)}) = z_0^{(s)} \oplus x_1^{(s)} \oplus \quad \oplus x_{n_s}^{(s)}$ is used, then an additional l-bit block is included in each segment. Each segment is encrypted/decrypted in parallel on a separate processor.

In interleaved-parallel or pipelined implementations of the XCBC modes, the initialization and computation of the block chaining sequence is performed on a per-segment basis starting with a common value of r_0, which is a random, uniformly distributed, l-bit value for every message. Also, the per-message value y_0 is computed as $y_0 \leftarrow f(r_0)$ in stateless implementations. The initialization of the block chaining sequence for message segment s can be $r_0^{(s)} = r_0 + s$, $z_0^{(s)} = f'(r_0^{(s)})$, and the encryption sequence can be $z_i^{(s)} = f(x_i^{(s)} \oplus z_{i-1}^{(s)}), y_i^{(s)} = z_i^{(s)} + i \quad r_0^{(s)}$. In stateful implementations *ctr* is updated to *ctr* $+ L$ after the encryption of each message. (Other functions, not just addition modulo 2^l, can be used for the computation of the per-segment, block chaining sequence, and initialization sequence can be used for $r_0^{(s)}$ and $z_0^{(s)}$.)

The encrypted segments of a message are assembled to form the message ciphertext. Segment assembly encodes the number of segments L, the length of each segment n_s and, implicitly, the segment sequence in the message (e.g., all can be found in the ASN.1 encoding). If the segments of a message have different lengths, segment assembly is also synchronized with the end of each segment encryption or decryption within a message.

At decryption, the parsing of the message ciphertext yields the message length, L, segment sequence number, s, and the length of each segment, n_s. Message integrity is maintained both on a per segment and per message basis by performing the per-segment integrity check; if $g(x) = z_0 \oplus x_1 \oplus \quad \oplus x_n$, the per-segment check is $z_0^{(s)} \oplus x_1^{(s)} \oplus \quad \oplus x_{n_s}^{(s)} = f^{-1}(z_{n_s+1}^{(s)}) \oplus z_{n_s}^{(s)}$ where $z_{n_s+1}^{(s)} = y_{n_s+1}^{(s)} \Leftrightarrow (n_s + 1) \quad r_0^{(s)}$ and $z_{n_s}^{(s)} = y_{n_s}^{(s)} \Leftrightarrow n_s \quad r_0^{(s)}$. Failure of any per-segment integrity check, which also detects out-of-sequence segments and message-length modifications, signals a message integrity violation.

We illustrate an interleaved- parallel implementation of the XCBC modes below.

Stateless Parallel XCBC Mode (ipXCBC$)

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

```
**function** $\mathcal{E}\Leftrightarrow$ipXCBC$\$^f(x)$
partition $x$ into $L$ segments $x^{(s)}$
each of length $n_s$;
$r_0 \leftarrow \{0,1\}^l$; $y_0 = f(r_0)$ ;
**for segment** $s, s = 1, \quad , L$, **do** {
$r_0^{(s)} = r_0 + s,\ z_0^{(s)} = f'(r_0^{(s)})$
**for** $i = 1, \quad , n_s$ **do** {
$z_i^{(s)} = f(x_i^{(s)} \oplus z_{i-1}^{(s)})$
$y_i^{(s)} = z_i^{(s)} + i \quad r_0^{(s)}$ }
$y^{(s)} = y_1^{(s)} \quad y_{n_s}^{(s)}$ }
assemble $y = y_0 || y^{(1)} \quad y^{(L)}$;
**return** $y$.
```

```
**function** $\mathcal{D}\Leftrightarrow$ipXCBC$\$^f(y)$
parse $y$ into $y_0$ and $L$ segments $y^{(s)}$
each of length $n_s$;
$r_0 = f^{-1}(y_0)$
**for segment** $s, s = 1, \quad , L$ **do** {
Parse $y^{(s)}$ as $y_1^{(s)} \quad y_{n_s}^{(s)}$
$r_0^{(s)} = r_0 + s;\ z_0^{(s)} = f'(r_0^{(s)})$
**for** $i = 1, \quad , n_s$ **do** {
$z_i^{(s)} = y_i^{(s)} \Leftrightarrow i \quad r_0^{(s)}$
$x_i^{(s)} = f^{-1}(z_i^{(s)}) \oplus z_{i-1}^{(s)}$ }
$x^{(s)} = x_1^{(s)} \quad x_{n_s}^{(s)}$ }
assemble $x = x^{(1)} \quad x^{(L)}$;
**return** $x$.
```

Stateful Parallel XCBC Mode (ipXCBC)

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

```
**function** $\mathcal{E}\Leftrightarrow$ipXCBC$\$^f(x, ctr)$
partition $x$ into $L$ segments $x^{(s)}$
each of length $n_s$;
$r_0 = f(ctr)$;
**for segment** $s, s = 1, \quad , L$, **do** {
$r_0^{(s)} = r_0 + s;\ z_0^{(s)} = f'(r_0^{(s)})$
**for** $i = 1, \quad , n_s$ **do** {
$z_i^{(s)} = f(x_i^{(s)} \oplus z_{i-1}^{(s)})$
$y_i^{(s)} = z_i^{(s)} + i \quad r_0^{(s)}$ }
$y^{(s)} = y_1^{(s)} \quad y_{n_s}^{(s)}$ }
assemble $y = ctr || y^{(1)} \quad y^{(L)}$;
$ctr' \leftarrow ctr + L$;
**return** $y$.
```

```
**function** $\mathcal{D}\Leftrightarrow$ipXCBC$\$^f(y)$
parse $y$ into $ctr$ and $L$ segments $y^{(s)}$
each of length $n_s$;
$r_0 = f(ctr)$
**for segment** $s, s = 1, \quad , L$ **do** {
Parse $y^{(s)}$ as $y_1^{(s)} \quad y_{n_s}^{(s)}$
$r_0^{(s)} = r_0 + s;\ z_0^{(s)} = f'(r_0^{(s)})$
**for** $i = 1, \quad , n_s$ **do** {
$z_i^{(s)} = y_i^{(s)} \Leftrightarrow i \quad r_0^{(s)}$
$x_i^{(s)} = f^{-1}(z_i^{(s)}) \oplus z_{i-1}^{(s)}$ }
$x^{(s)} = x_1^{(s)} \quad x_{n_s}^{(s)}$ }
assemble $x = x^{(1)} \quad x^{(L)}$;
**return** $x$.
```

Note that in the XCBC mode the counter *ctr* can be initialized to a known constant such as $\Leftrightarrow 1$ by the sender. *ctr′* represents the updated *ctr* value.

7. Incremental Updates of Encrypted Data. The segmentation of a message used for parallel and pipelined implementation of the XCBC modes can also be used in sequential encryption of data structures (e.g., a file, a message) whenever incremental updates of data structures are anticipated. Such segmentation enables the localization of the decryption, plaintext update, and encryption to single segments saving the decryption and encryption of other segments unaffected by the updates. Note that message integrity is retained after such incremental updates.

6. Support for Architecture-Independent Parallel Encryption. In Jutla's recent scheme [20], a parallel mode is proposed whereby both the input and output to the pseudo-random function are "whitened" using a collection of *pairwise independent random numbers.* The fully parallel XCBC modes achieves the same effect *without* pairwise independent random numbers. It is sufficient to randomize the input of f using the same type of sequence as that used for the randomization of its output to obtain a low probability of input or output collisions, which would be necessary to break integrity (as illustrated in the next section, for the XECB MAC). More formally, for an arbitrary index $i, 1 \quad i \quad n+1, n = |x|$, the ciphertext block y_i is obtained through the formula:

$$y_i = f(x_i + i \quad z_1) + i \quad r_0,$$

where z_1 is random, uniformly distributed and independent of r_0 and z_0. In this mode, there is no ciphertext chaining, and no a priori knowledge of the number of processors is necessary. The same function $g(x)$ can be used here as in the sequential XCBC modes.

7. Resistance to Key Attacks. Resistance to exhaustive key-guessing attacks can be implemented in a similar manner as that of DESX [29], if deemed necessary, in all of the above modes. However, adoption of modern block ciphers should reduce the need for this.

5 Definition of the XECB Authentication Modes

In this section, we introduce new Message Authentication Modes (MACs) that counter adaptive chosen-message attacks [3]. We call these MACs the eXtended Electronic Code Book MACs, or XECB-MACs. The XECB-MAC modes have all the properties of the XOR MACs [3] plus they do not waste half of the block size for recording the block position. First we define these MACs, and then we present their properties. Several variants of such MACs can be derived, and here we present a stateless version of XECB-MAC, namely the XECB\$-MAC, and a stateful version, namely the XECB-MAC.

A stateless implementation of the XECB\$-MAC uses as initialization sequence $r_0 \leftarrow \{0,1\}^l, y_0 = f(r_0)$ and $z_0 = f'(r_0)$, 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 encrypting a function of r_0. Use of K' is made here exclusively to simplify the proof.) Then, each block of message x, namely $x_i, 1 \quad i \quad n, n = |x|$ is randomized as $x_i + i \quad y_0$, and the result is input to function f; i.e., $y_i = f(x_i + i \quad y_0)$. We also let $x_{n+1} = z_0$ and compute $y_{n+1} = f(z_0 + (n+1) \quad y_0)$. These values, $y_1, \quad , y_n, y_{n+1}$, and z_0 are exclusive-OR-ed to generate the authentication tag:

$$w = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$$

The algorithm outputs the pair (r_0, w). For verification, the attacker submits a forgery $x = x_1 \quad x_n$ and a forged pair (r_0, w).[4] The algorithm proceeds with computing $y_0 = f(r_0)$, then $z_0 = f'(r_0)$, then computes $y_i = f(x_i + i \quad y_0), \forall i, 1 \quad i \quad n$, $y_{n+1} = f(z_0 + (n+1) \quad y_0)$, and the authentication tag $w' = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$. The algorithm outputs a bit that is either 1, if the forged authentication tag is correct, namely $w = w'$, or 0, otherwise.

In the stateful mode, the signer maintains state across consecutive signing requests in the form of a counter (ctr). Hence, the initialization phase is defined as $r_0 = f(ctr), z_0 = f'(r_0)$, where $f' = F_{K'}$ is a PRF selected

[4]The forgery (x, r_0, w) is not a previously signed query. Note also that the length n of the forged message needs not be equal to the length of any signed message.

with the second key K'. Then, for each message block, $y_i = f(x_i + i \quad r_0), \forall i, 1 \quad i \quad n, n = |x|$, and $y_{n+1} = f(z_0 + (n+1) \quad y_0)$. The authentication tag is then defined as

$$w = y_1 \oplus \qquad \oplus\, y_n \oplus y_{n+1}.$$

The algorithm outputs the pair (ctr, w). For verification, the attacker submits a forgery $x = x_1 \quad x_n$ and a forged pair (ctr, w). The algorithm proceeds with computing $r_0 = f(ctr), z_0 = f'(r_0)$, then computes $y_i = f(x_i + i \quad r_0), \forall i, 1 \quad i \quad n$, $y_{n+1} = f(z_0 + (n+1) \quad y_0)$, and the tag $w' = y_1 \oplus \qquad \oplus\, y_n \oplus y_{n+1}$. The algorithm outputs a bit that is either 1, if the forged authentication tag is correct, namely $w = w'$, or 0, otherwise.

The concrete implementation for the signing and verifying algorithms for the stateless and stateful XECB-MAC modes is defined as follows.

Stateless XECB-MAC Mode (XECB\$-MAC)

```
function Sign-XECB$-MAC^f(x)
$r_0 \leftarrow \{0,1\}^l$
$y_0 = f(r_0), z_0 = f'(r_0)$
$x_{n+1} = z_0$
for $i = 1, \quad , n+1$ do {
$y_i = f(x_i + i \quad y_0)$ }
$w = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
return $(r_0, w)$
```

```
function Verify-XECB$-MAC^f(x, r_0, w)
$y_0 = f(r_0), z_0 = f'(r_0)$
$x_{n+1} = z_0$
for $i = 1, \quad , n+1$ do {
$y_i = f(x_i + i \quad y_0)$ }
$w' = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
if $w = w'$ then return 1
else return 0.
```

Stateful XECB-MAC Mode (XECB-MAC)

```
function Sign-XECB-MAC^f(ctr, x)
$r_0 = f(ctr), z_0 = f'(r_0)$
$x_{n+1} = z_0$
for $i = 1, \quad , n+1$ do {
$y_i = f(x_i + i \quad r_0)$ }
$w = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
$ctr' \leftarrow ctr + 1$
return $(ctr, w)$
```

```
function Verify-XECB-MAC^f(x, ctr, w)
$r_0 = f(ctr), z_0 = f'(r_0)$
$x_{n+1} = z_0$
for $i = 1, \quad , n+1$ do {
$y_i = f(x_i + i \quad r_0)$ }
$w' = y_1 \oplus \quad \oplus y_n \oplus y_{n+1}$
if $w = w'$ then return 1
else return 0.
```

Note that in the XECB mode the counter *ctr* can be initialized to a known constant such as $\Leftrightarrow 1$ by the sender. ctr' represents the updated *ctr* value. The stateless XECB-MAC\$ and stateful XECB-MAC modes can be implemented using PRFs.

It should be noted that the implementation of the XECB-MAC modes can be performed in software, hardware, or software with hardware support. Implementations can be in general-purpose computers or in dedicated hardware devices and software. We now present the properties of the stateless and stateful XECB-MAC modes.

6 Properties of the XECB Authentication Modes

1. Security. The XECB authentication modes are intended to be secure against adaptive chosen-message

(q_s, q_v)-attacks [3]. These attacks are similar to the message integrity attack defined in this paper. The only difference is that instead of q_e (encryption) queries totaling μ_e bits and taking time t_e, this attack uses q_s (signature) queries, totaling μ_s bits and taking time t_s. Theorem 3 below shows the security bounds for these modes against adaptive chosen-message attacks. The XECB modes have higher upper bounds on the adversary's success in producing a forgery than those of the XOR-MAC modes.

2. Parallel and Pipelined Operation. Function f (e.g., DES, RC6) 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. This property is important for high speed networks and in both hardware and software implementations.

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 stateless mode. Let the two messages have the same random block r_0; hence, the authentication tag of the new message, w', is obtained from the authentication tag of the previous message, w, by the following formula: $w' = w \oplus f(x_i + i \quad r_0) \oplus f(x'_i + i \quad r_0)$. 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. Block Encryption Computations. In contrast to the XOR-MAC modes, where the number of block encryption computations is twice the number of block encryption computations for CBC-MAC [3], the number of block encryption computations in the XECB-MAC modes is the same as the number of block encryption computations for the CBC-MAC. While in sequential implementations the performance of XECB-MACs is expected to be just slightly lower than the performance of the CBC-MAC (because of the two modular additions per block vs. one exclusive-OR for CBC-MAC), the XECB mode can take advantage of parallelism or pipelining in an architecture-independent manner; i.e., the number of processors available need not be known apriori – a significant feature available only in few modes such as the XOR-MAC. This property, the out-of-order and incremental computation are especially important in hardware implementations, particularly in high-speed networks and for the internet. For this reason, the use of the XECB-MAC modes appears to be more appropriate than that of the XOR-MAC for the integrity protection of messages encrypted under fully parallel or pipelined encryption schemes such as the XORrC [13], which is a random-counter variant of the counter-based XOR encryption mode [1].

7 Security Considerations

In this section, we provide evidence for the security of the XCBC modes against both adaptive chosen-plaintext and message-integrity attacks. We also present the security of the XECB modes in adaptive chosen-message attacks.

We first address the security (i.e., secrecy) of the XCBC$ mode against adaptive chosen-plaintext attacks. The theorems and proofs that demonstrate that the stateful mode (XCBC) and the two-key variations are secure in a left-or-right sense [1] are similar to that for the XCBC$ mode and, therefore, will be omitted.

The Lemma and Theorem below, which establish the security (i.e., secrecy) of the XCBC$ mode are restatements of Lemma 16 and Theorem 17 respectively, which are presented for the CBC mode in the full version of the Bellare *et al.* paper ([1]). The proof of the Lemma and Theorem are similar to those for the CBC mode and hence are omitted.

Lemma 1 [Upper bound on the security of the XCBC$ mode in random function model] Let $XCBC\$^R$ be the implementation of the XCBC$ mode with the family of random functions $R(l, l)$. Let A be any adversary attacking $XCBC\$^R$ in the left-or-right sense, making at most q' queries, totaling at most μ' bits. Then, the adversary's advantage is

$$Adv_A^{lr} \quad \delta_{XCBC\$} \overset{def}{=} \left(\frac{\mu'^2}{l^2} \Leftrightarrow \frac{\mu'}{l}\right) \frac{1}{2^l}.$$

The following theorem defines the security of the XCBC$ mode against an adaptive chosen-plaintext attack when the XCBC$ mode is implemented with a (q, t, ϵ)-pseudorandom function family F. F is (q, t, ϵ)-pseudorandom, or (q, t, ϵ)-secure, if an adversary (1) spends time t to evaluate $f = F_K$ at q input points via adaptively chosen queries, and (2) has a negligible advantage bounded by ϵ over simple guessing in distinguishing the output of f from that of a function chosen at random from R.

Theorem 1 [Security of XCBC$ against Adaptive Chosen-Plaintext Attacks] Suppose F is a (t, q, ϵ)-secure PRF family with block length l. There is a constant $c > 0$ such that for any number of queries q_e totaling μ' bits of memory and taking time t', the $XCBC\$(F)$ is $(t', q', \mu', \epsilon')$-secure in the left-or-right sense, for $\mu' = q'l$, $t' = t \Leftrightarrow c\mu'$, and $\epsilon' = 2\epsilon + \delta_{XCBC\$}$ where $\delta_{XCBC\$} \overset{def}{=} \left(\frac{\mu'^2}{l^2} \Leftrightarrow \frac{\mu'}{l}\right) \frac{1}{2^l}$.

The XCBC$ and XCBC modes should really be analyzed assuming F is a SPRP family (not a PRF family), and hence one needs to apply the results of Proposition 8 of Bellare *et al.* [1] to the results of Theorem 1 (and also of Theorems 2-3 below). A similar lemma and theorem hold for chosen-plaintext attacks in a real-or-random sense, as defined by Bellare *et al.* [1].

In establishing the security of the XCBC$ mode against the message-integrity attack, let the parameters used in the attack be bound as follows: $q_e \quad q'$, since the XCBC$ scheme is also chosen-plaintext secure, $t_e + t_v \quad t$, and $\mu'' = \mu_e + \mu_v \quad ql$. 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 2 [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 $\mu_e + \mu_v \quad ql$ bits, and taking at most $t_e + t_v \quad t$ time; i.e., the probability of adversary's success is

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \qquad \epsilon + \frac{\mu_v(\mu_v \Leftrightarrow l)}{l^2 2^{l+1}} + \frac{q_e(q_e \Leftrightarrow 1)}{2^{l+1}} + \frac{(q_e + 2)\mu_v}{l 2^l} + \frac{q_v}{2^l}\left(\frac{\mu_e}{l}\log_2\frac{\mu_e}{l} + \frac{3\mu_e}{l}\right);$$

and, if $m = \max(n_p + 1)$, where n_p, $1 \quad p \quad q_e$, is the number of blocks encrypted in the message p-th message,

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \qquad \epsilon + \frac{\mu_v(\mu_v \Leftrightarrow l)}{l^2 2^{l+1}} + \frac{q_e(q_e \Leftrightarrow 1)}{2^{l+1}} + \frac{(q_e + 2)\mu_v}{l 2^l} + \frac{q_v}{2^l}\left(\frac{\mu_e}{l}\log_2 m + \frac{3\mu_e}{l}\right).$$

(The proof of Theorem 2 can be found in Appendix A.) Note that parameters q_e, μ_e, t_e can be easily stated in terms of parameters $(t', q', \mu', \epsilon')$ of Theorem 1 above by introducing a constant c' defining the speed of the XOR function.

Theorem 2 above allows us to estimate the complexity of a message-integrity attack. In a successful attack, $Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \in (\text{negligible}, 1]$. To estimate complexity, we set the probability of success when $f \overset{\mathcal{R}}{\leftarrow} P^l$ 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, $q_e, q_v \quad \frac{2^{\alpha l}}{3}\rfloor$. $\leq\lfloor$

In this case, by setting

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

we obtain the equation $2^{2\alpha l}\lfloor\frac{6\alpha l+34}{9}\rfloor + 2^{\alpha l}\frac{8}{3} = 2^l$, 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). If the maximum length m of the encrypted messages is known, the attack is even closer to a square-root attack.

A variant of Theorem 2 can be proved for the stateful mode. In this case, it can be shown that the probability of successful forgery when q_v verification queries are allowed, totaling at most μ_v bits and using at most time t_v after q_e encryption queries totaling μ_e bits and taking time t_e, is

$$Pr_{f \overset{R}{\leftarrow} F}[\text{Succ}] \qquad \epsilon + \frac{\mu_v(\mu_v \Leftrightarrow l)}{l^2 2^{l+1}} + \frac{(q_e+2)\mu_v}{l2^l} + \frac{q_v}{2^l} \quad \frac{\mu_e}{l}\log_2\frac{\mu_e}{l} + \frac{3\mu_e}{l} \quad .$$

Furthermore, similar theorems hold for other stateless modes where $z_0 = f(r_0 + 1)$. 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.

The XECB-MAC mode is intended to be secure against an adaptive chosen-message attack [3] 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 XECB-MAC mode is established by the following theorem.

Theorem 3 [Security of XEBC-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-XECB^f, Verify-XECB^f, KG) is secure against adaptive chosen-message (q_s, q_v) attacks consisting of $q_s + q_v$ queries totaling $\mu_s + \mu_v \quad ql$ bits and taking at most $t_s + t_v \quad t$ time; i.e., the probability of adversary's success is

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

The proof of this theorem is similar to that of Theorem 2 and is presented in Appendix B. A similar theorem can be provided for the stateless message authentication mode. The complexity of an attack against the XECB MAC can be derived in a similar manner to that of the attack against the XCBC$-XOR mode.

8 Performance Considerations for the XCBC Modes

The performance of the XCBC modes in software implementations is (1) minimally degraded in comparison to that of the original CBC mode [11, 1], and (2) superior to that of the original CBC modes, and most other similar modes, when message integrity is desired.

The XCBC$ modes add the overhead of two block encryption per message (i.e., for generating z_0 and y_0), and two *mod* 2^l additions per message block to the traditional CBC mode with random initialization vectors (or, equivalently, with initialization vectors set to zero and a random number in the first plaintext block [28])[5]. However, the execution of both modulo 2^l additions for the current ciphertext block can always be overlapped with the block encryption of the next block and, hence, at peak speeds there is little perceptible overhead over that of the traditional CBC mode. This compares favorably with the overhead added at peak speeds by an original CBC mode that uses any of the hash functions known to date to provide message integrity. In any case, the dominant performance factor is the throughput achieved by block encryption. An important advantage of the new modes is that their performance scales up nearly identically with that of block encryption; furthermore, hardware implementations of the new modes can make the added overhead imperceptible.

To illustrate the performance characteristics of the XCBC$ scheme in software, we used the *SSLeay* library [31], and conducted some preliminary measurements on a Sun SPARC Ultra 10 IIi processor running the SunOS 5.6 operating system. The processor has a 333 MHz clock, 2 MB of external (off-chip) cache, and 16/16 KB of internal (on-chip) instruction/data cache. We used the version 4.2 of the native C compiler with the -xO2 optimization option. We used a lightly loaded machine for our measurements, and the throughput for each of the eight message sizes was generated by averaging the results of fifty runs.

Our implementation of the addition *mod* 2^l operations was also influenced by the SSLeay implementation of the CBC scheme on 32-bit processors. However, we were able to use 64-bit additions for the XCBC$ scheme. The addition uses the *unsigned long long* type (64 bits) for $i \quad r_0 \overset{def}{=} \underbrace{r_0 + \quad + r_0}_{i\ times}$ and $y = z + i \quad r_0$ operations. The hidden ciphertext blocks z_i that result from the DES encryption (which operates on two 32-bit unsigned longs) are packed into the *unsigned long long* z_i for the subsequent modular 64-bit additions. Each packing operation, which would be avoided in a 64-bit implementation, requires a bitwise *or* and a shift.

The throughput of the CBC, XCBC$, CBC-MD5, CBC-UMAC-STD30,
CBC-HMAC-SHA1, and XCBC$-XOR modes implemented with DES is shown in Figure 1 for samples of both large and small messages.[6] The percentage gain in the throughput performance of the XCBC$-XOR mode over that of the CBC-MD5, CBC-UMAC-STD30, and CBC-HMAC-SHA1 modes for these message samples is shown in Figure 2.

The results shown in these figures indicate that, in unoptimized software implementations,

- a substantial overall throughput improvement can be expected. For small messages (i.e., between 1 Byte and 1 KB length), we can expect about 15⇔65% improvement for XCBC-XOR vs. CBC-MD5, about 78⇔113% for XCBC-XOR vs. CBC-UMAC-STD30, and about 44⇔99% for XCBC-XOR vs. CBC-HMAC-SHA1. For large messages (i.e., between 10 KB and 1MB length), we can expect about 15⇔20% improvement for XCBC-XOR vs. CBC-MD5, about 15⇔25% for XCBC-XOR vs. CBC-UMAC-STD30, and about 23⇔29% for XCBC-XOR vs. CBC-HMAC-SHA1. In general, we expect higher performance improvement for small messages than for large ones because, for small messages, the performance of the MD5 hash function (and that of most hash functions), of UMAC-STD30

[5]Note that traditional CBC modes require the use of secret initialization vectors that are protected from arbitrary modification, and the most common way of satisfying both requirements is to use (pseudo) random initialization vectors. For this reason, the generation of the initial per-message random number is considered to be a common overhead to both the new and the traditional CBC modes. As shown above, the stateful XCBC mode, however, requires only that a separate random number be generated per key, not per message, thereby eliminating much of this common overhead of stateless modes. The alternative of generating and cacheing values of r_0 as the system runs and ahead of their actual use may also help decrease the overhead of the stateless XCBC mode.

[6]Note that the UMAC-STD30 is the 1999 version. The latest version, not used here, is reported to perform much better for small message sizes.

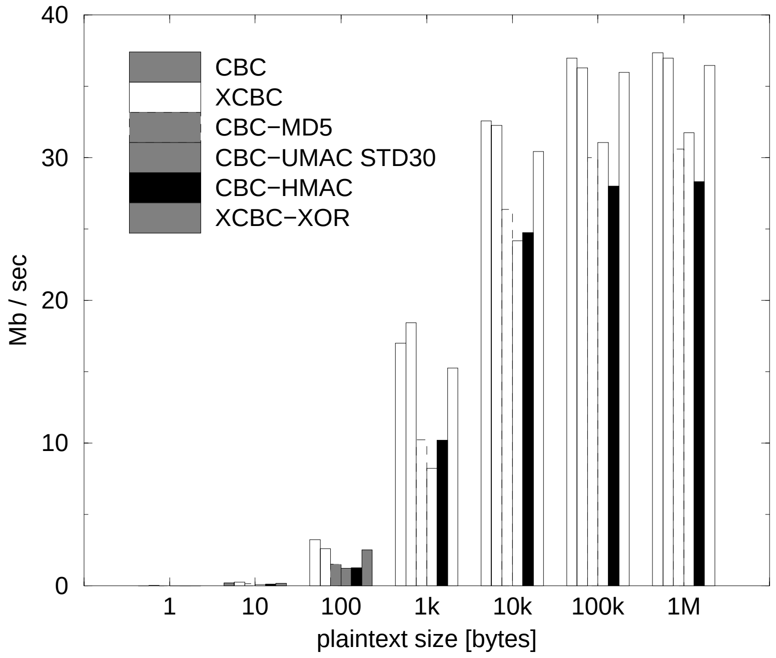

Figure 1: Throughput of the CBC, XCBC\$, CBC-MD5, CBC-UMAC-STD30, CBC-HMAC-SHA1, and XCBC\$-XOR encryption modes implemented with DES for message sizes of 1 B 1 MB.

and HMAC-SHA1 is closer to that of DES than for large messages. Hence, our use of the function $g(x) = z_0 \oplus x_1 \oplus \quad \oplus x_n$ improves a much larger fraction of the overall throughput of encrypted messages.

- throughput measurements for small-size messages shown in Figure 1 are susceptible to a significant margin of error caused by the inability to offset operating system effects over a fairly short run-time for each test for such messages. For example, for 1 KB messages, the throughput of XCBC\$ appears to be higher than that of CBC by about 8%. Nevertheless, the performance illustrated in Figure 1 appears to be consistent with individual MD5 and UMAC-STD30 measurements. For example, measurements reported for UMAC-STD30 [8] show that it reaches peak speeds for messages between 80 KB and 128 KB, whereas Figure 1 indicates that the UMAC-STD30 reaches close-to-peak performance at 100 KB. Also, Figure 1 shows that, for 10 KB messages, UMAC-STD30 is within 22.2% of the speed measured at 100 KB, which appears to be consistent with the measurements reported for UMAC-STD30.

- the clear performance bottleneck is that of the DES-CBC and underlying DES block encryption. Figure 1 shows that the performance differences between the DES-CBC, DES-XCBC\$, and DES-XCBC\$-XOR are almost imperceptible for mid-size and large messages. Given the advantage of using the function $g(x) = z_0 \oplus x_1 \oplus \quad \oplus x_n$ over that of using $g(x) =$ MD5 or any other hash function or MAC in XCBC encryption, we expect that the gain in the performance of the XCBC\$-XOR over that of CBC-MD5 (or any other CBC-hash-function mode), CBC-UMAC-STD30, or CBC-HMAC-SHA1 to be even more pronounced for fast block encryption functions where the UMAC and MD5 (or any other hash function) would represent a higher fraction of the CBC-HMAC-SHA1, CBC-UMAC-STD30, and CBC-MD5 (or CBC-any-hash-function) cost.

We also expect that further improvements can be derived from an assembly language implementation where

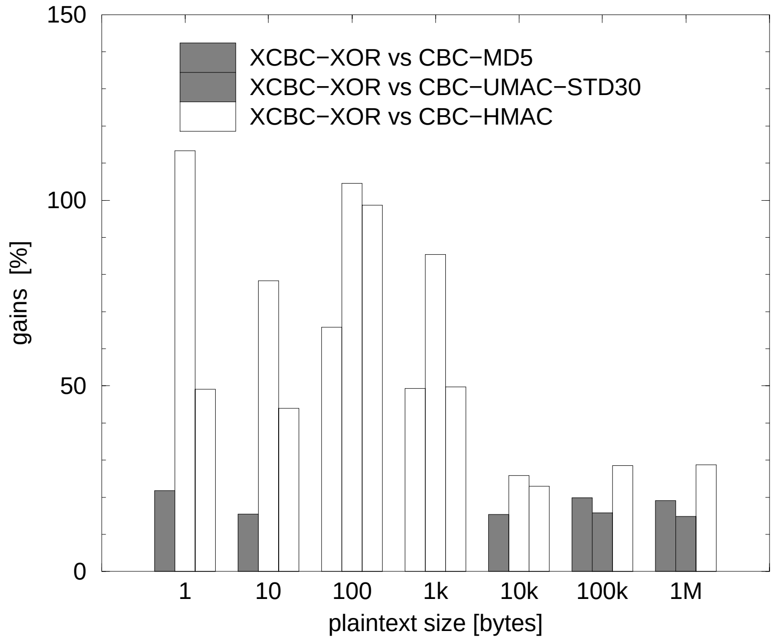

Figure 2: Percentage gains of the DES-XCBC$-XOR mode over the CBC-MD5, CBC-UMAC-STD30, and CBC-HMAC-SHA1 modes for message sizes of 1 B   1 MB.

optimal register allocation can be performed for both the block encryption functions and the XCBC modes.

It should be noted that the implementation of the XCBC modes can be performed in software, hardware, or software with hardware support. Implementations can be in general-purpose computers or in dedicated hardware devices and software. The simplicity of the XCBC modes suggests that substantial cost-performance improvement can be expected when they are implemented in hardware. For example, DES hardware implementation reached 1.6 Gbp whereas HMAC-SHA-1 hardware implementation has reached only about half that speed. (This seems to confirm early predictions that the speed of hash functions and MACs based on them does not scale in hardware implementations as well as that of block encryption functions [32, 7]). Thus, we can expect performance speedups of about 100⇔200% over current hardware implementation modes for message encryption and authentication. Of particular interest in this area are implementations of the XCBC modes on low-power and/or low-cost devices.

Acknowledgments

We thank David Wagner for pointing out an oversight in an earlier version of Theorem 2, 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, J. Killian, and P. Rogaway, "The security of cipher block chaining", *Advances in Cryptology-CRYPTO '94* (LNCS 839), 341-358, 1995.

[3] 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.)

[4] 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.*

[5] M. Bellare and P. Rogaway, "Block Cipher Mode of Operation for Secure, Length-Preserving Encryption," *U.S Patent No. 5,673,319,* September, 1997.

[6] M. Bellare and P. Rogaway, "On the construction of variable-input-length ciphers," Proceedings of the 6th Workshop on Fast Software Encryption, L. Knudsen (Ed), Springer-Verlag, 1999.

[7] S.M. Bellovin, "Cryptography and the Internet," *Advances in Cryptology - CRYPTO '98* (LNCS 1462), 46-55, 1998.

[8] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast Message Authentication via Optimized Universal Hash Functions," *Advances in Cryptology - CRYPTO '99* (LNCS 1666), 216-233, 1999.

[9] 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.

[10] Open Software Foundation, "OSF - Distributed Computing Environment (DCE), Remote Procedure Call Mechanisms," Code Snapshot 3, Release, 1.0, March 17, 1991.

[11] FIPS 81, "DES modes of operation", Federal Information Processing Standards Publication 81, U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1980.

[12] 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.)

[13] V.D. Gligor, "Symmetric Encryption with Random Counters," University of Maryland, Computer Science Technical Report, CS-TR-3968, December 1998.

[14] 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.

[15] 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.

[16] 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.

[17] 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).

[18] D.E. Knuth, “The Art of Computer Programming - Volume 2: Seminumerical Algorithms,” Addison-Wesley, 1981 (second edition), Chapter 3.

[19] J. T. Kohl, “The use of encryption in Kerberos for network authentication”, *Advances in Cryptology-CRYPTO '89* (LNCS 435), 35-43, 1990.

[20] 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.*

[21] M Luby and C. Rackoff, “How to construct pseudorandom permutations from pseudorandom functions”, *SIAM J. Computing*, Vol. 17, No. 2, April 1988.

[22] C. H. Meyer and S. M. Matyas, *Cryptography; A New Dimension in Computer Data Security*, John Wiley & Sons, New York, 1982 (second printing).

[23] C. H. Meyer and S. M. Matyas, *Cryptography; A New Dimension in Computer Data Security*, John Wiley & Sons, New York, 1982 (third printing).

[24] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1997.

[25] 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.

[26] E. Petrank and C. Rackoff, “CBC MAC for Real-Time Data Sources,” manuscript available at *http://philby.ucsd.edu/cryptolib.html*, 1997.

[27] RFC 1321, “The MD5 message-digest algorithm”, Internet Request for Comments 1321, R. L. Rivest, April 1992 (presented at Rump Session of Crypto '91).

[28] RFC 1510, “The Kerberos network authentication service (V5)”, Internet Request for Comments 1510, J. Kohl and B.C. Neuman, September 1993.

[29] P. Rogaway, “The Security of DESX,” RSA Laboratories *Cryptobytes*, Vol. 2, No. 2, Summer 1996.

[30] 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.

[31] SSLeay, available at *ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL*

[32] J. D. Touch, “Performance Analysis of MD5,” Proceedings of ACM, SIGCOMM '95, 77-86, 1996.

[33] V.L. Voydock and S.T. Kent, "Security Mechanisms in high-level network protocols,” *Computing Surveys*, 15(1983), 135-171.

[34] C. Weissman, “Question and Answer Session,” 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. 121.

Appendix A - Proof [Security of the XCBC\$-XOR in a Message-Integrity Attack]

Notation: Throughout this proof, the superscripts of variables x^p, z^p, y^p, and r_0^p denote the plaintext, hidden ciphertext, ciphertext, and initial random value of a queried message $p, 1 \quad p \quad q_e$, whereas the (primed) variables x'^i, z'^i, y'^i, and $r_0'^i$ denote the plaintext, hidden ciphertext, ciphertext, and the initial random value of the i-th forged (i.e., unqueried) message, $1 \quad i \quad q_v$. The length of the plaintext of message p is denoted by $n_p = |x^p|$ and that of forgery y'^i by $n'^i = |x'^i|$ blocks. (These lengths do not include the last plaintext block that holds the value of the XOR function.)

To find an upper bound on the probability of an adversary's success we (1) define four types of events on which we condition the adversary's success, (2) express the upper bound in terms of the conditional probabilities obtained, and (3) compute upper bounds on these probabilities. Our choice and number of conditioning events is motivated exclusively by the need to obtain a (good) upper bound for the probability of the adversary's success. Undoubtedly, other events could be used for deriving alternate upper bounds.

To provide some intuition for the choice of conditioning events defined, we give examples of events that cause an adversary's success. (The reader can skip these examples without loss of continuity.)

Examples of Adversary's Success. A way for the adversary to find a forgery y' that passes the integrity check $g(x') = x'_{n+1}$, is to look for collisions in the input of f^{-1}, namely collisions of the (1) hidden ciphertext blocks generated during the decryption of a forgery, $z'_s, 1 \quad s \quad n+1$, and (2) initialization block y'_0 (i.e., block 0 of the forged ciphertext). These blocks could collide either with blocks $y_0^p, z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_i+1$ obtained at encryption or among themselves. The following four examples illustrate why such collisions cause an adversary's success. Other such examples, and other ways to find forgeries, exist.

Example 1 – Collisions between blocks z'_s and z_k^p

Suppose that all hidden ciphertext blocks z'_s obtained during the decryption of forgery y' collide with some hidden ciphertext blocks z_k^p obtained at encryption. If this event occurs during forgery decryption, we declare pessimistically that the adversary is successful. Why is the adversary successful? Among the forgeries that make this event true, some will decrypt correctly with probability one. For example, if any two of the hidden ciphertext blocks between position 1 and n_p of a queried message p are swapped, the decryption of the resulting hidden ciphertext will pass the integrity check $g(x') = x'_{n+1}$ with probability one (viz., [24], Example 9.89, pp. 367-368, for a similar example). Thus, any forgery that generates such hidden ciphertext at decryption will pass this integrity check with probability one.

Why is our criterion for adversary's success based on such a collision event pessimistic? Among the forgeries that make this event true, some will decrypt correctly with negligible probability. These forgeries include truncations of the ciphertext of already queried messages.[7] For truncations, the integrity check cannot pass with probability greater than $1/2^l$ (and for this reason we can focus on other types of forgeries for the rest of this proof).

[7]Let the forged ciphertext y' be a truncation of ciphertext y^p obtained at encryption; i.e., $y'_s = y_s^p, \forall s, 0 \leq s \leq n'+1, |y'| = n'+1$ and $n' < n_p$, i.e., $n'+1 \leq n_p$. The condition $n'+1 \leq n_p$ (due to truncation) implies that all the plaintext blocks $x_1^p, \cdot \quad \cdot, x_{n'+1}^p$ are constants. In this case, $z'_s = z_s^p, \forall s, 0 \leq s \leq n'+1$ and thus $x'_s = x_s^p, \forall s, 0 \leq s \leq n'+1$. The integrity check, $z_0^p \oplus x_1^p \oplus \cdot \quad \cdot \oplus x_{n'}^p \oplus x_{n'+1}^p = 0$, is the exclusive-or of a random and uniformly distributed variable $z'_0 = f'(r'_0) = f'(r_0^p) = z_0^p$, where $f' \overset{\mathcal{R}}{\leftarrow} R^{l,l}$, and constant plaintexts $x_1^p, \cdot \quad \cdot, x_{n'+1}^p$. Hence, $Pr[z_0^p \oplus x_1^p \oplus \cdot \quad \cdot \oplus x_{n'}^p \oplus x_{n'+1}^p = 0] = \frac{1}{2^l}$.

Example 2   Collisions among the z'_s blocks

Suppose that two hidden ciphertext blocks z'_s and z'_t obtained during forgery decryption do not collide with any hidden ciphertext blocks obtained during encryption, but collide with each other. If this event occurs during forgery decryption, we declare pessimistically that the adversary is successful. Why is the adversary successful? Among the forgeries that make event true, some will decrypt correctly with probability one. For example, if any two identical blocks never seen among the hidden ciphertext blocks obtained at encryption are inserted into two adjacent positions between 1 and n_p of the hidden ciphertext of message p (i.e., $z'_s = z'_{s+1}, 1 \quad s< \; n_p \Leftrightarrow 1$), the decryption of the resulting hidden ciphertext will pass the integrity check $g(x') = x'_{n+1}$ with probability one (viz., [24], Example 9.89, pp. 367-368, for a similar example). Thus, any forgery that generates such hidden ciphertext blocks at decryption will pass this integrity check with probability one.

Why is our criterion for adversary's success based such a collision event pessimistic? Among the forgeries that make this event true, some will decrypt correctly with negligible probability. For example, consider forgeries that cause an odd number of identical hidden ciphertext blocks to be generated during decryption. Suppose these blocks have the following properties: (1) they do not collide with any hidden blocks obtained at encryption, (2) they do not collide with any initialization blocks $y_0^i, 1 \quad i \quad q_e$, obtained at encryption, (3) they do not collide with the initialization block y'_0 of the forgery, and (4) they appear between positions 1 and $n_p + 1$ of the hidden ciphertext of queried message p obtained at encryption. Forgeries that produce such blocks during decryption cannot pass the integrity check with probability greater than $1/2^l$. This is the case because the decryption of these identical hidden blocks produces random, uniformly distributed plaintext blocks that are independent of any other plaintext blocks in $g(x') = x'_{n+1}$ and can only cancel each other out in pairs under the exclusive-or operation.

The next two examples refer to collision events of the initialization block y'_0. These can lead to forgeries that satisfy the conditions of the events defined in Examples 1 and 2 above, and hence such collisions contribute to an adversary's success.

Example 3   Collisions between blocks y'_0 and z^p_{k+1}

Suppose that, during the decryption of forgery y', block y'_0 collides with some hidden ciphertext block obtained during encryption. Let $y'_0 = z^p_{k+1}, 1 \quad p \quad q_e, 1 \quad k \quad n_p$. This means that the lower order bits of $r'_0 = f^{-1}(y'_0) = x^p_{k+1} \oplus z^p_k$ can be predicted (at least) to the same extent as those of z^p_k, since x^p_{k+1} is chosen. In (pessimistic) case the entire r'_0 is predicted, the adversary's forgeries can satisfy the collision events of Examples 1 and 2 above.

Example 4   Collisions between blocks y_0^i and y_0^p

Suppose that an adversary finds a collision between the initialization blocks of two ciphertext messages i and p obtained at encryption, namely y_0^i and y^p, and chooses the initialization block of the forgery y' to be $y'_0 = y_0^i$. If the adversary can find such a collision event at encryption, the adversary can also find forgeries that satisfy the collision events of Example 1 at decryption. For example, the adversary can create a ciphertext message that has not been seen before (i.e., a forgery) by mixing the blocks of two ciphertext messages obtained at encryption whose initial ciphertext blocks collide; e.g., ciphertext block y_k^i of messages i replaces ciphertext $y_k^p \neq y_k^i$ of message p, where $y'_0 = y_0^i = y_0^p, i \neq p, n_i \quad n_p, 1 \quad i, p \quad q_e$, $1 \quad k \quad n_i$.

Conditioning Events. To compute an upper bound on the probability of successful forgery, we choose four conditioning events based on collisions in the input of f^{-1}. Intuition for the choice of events is provided by Examples 1   4 above.

For each verification query (or forgery) y'^i, $1 \quad i \quad q_v$, we define two types of collision events, C_i and D_i, that refer to the hidden ciphertext blocks z'^i_s obtained during forgery decryption.

Event C_i includes all the instances when the hidden blocks z'^i_s of forgery y'^i collide either with initialization blocks y^p_0 or with some hidden ciphertext blocks z^p_k generated during encryption, where $1 \quad p \quad q_e, 1 \quad k \quad n_p + 1$. To define event C_i formally, let S be the the union of all the y^p_0 blocks and all the hidden ciphertext blocks z^p_k produced at encryption:

$$S = \{y^p_0, 1 \quad p \quad q_e\} \cup \{z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}.$$

Also let Z_i be the collection of hidden ciphertext blocks z'^i generated during the decryption of the arbitrary forgery $y'^i, 1 \quad i \quad q_v$, that do not collide with blocks of S:

$$Z_i = \{z'^i_s, 1 \quad s \quad n'_i + 1, z'^i_s \notin S\}.$$

Hence, event C_i (*Collision*) is defined by:

$$C_i : Z_i = \emptyset;$$

i.e., Z_i is empty; or, equivalently, $C_i : Z_i \subseteq S$.

The second type of collision event defined for the arbitrary forgery $y'^i, 1 \quad i \quad q_v$, refers to collisions among blocks $y'^i_0, z'^i_s, 1 \quad s \quad n'_i + 1$ where $z'^i_s \in Z_i$, and is denoted by $\overline{D_i}$ (*not distinct*) below. This event is defined in terms of its complementary event D_i (*distinct*), which states that there is at least a hidden block $z'^i_s \in Z_i$ that does not collide with any other hidden block $z'^i_t \in Z_i$ or with y'^i_0.[8] It is clear that this definition makes sense only when $Z_i \neq \emptyset$. Formally, if $Z_i \neq \emptyset$,

$$D_i : \exists z'^i_s \in Z_i, 1 \quad s \quad n'_i + 1 : z'^i_s \neq z'^i_t, \forall z'^i_t \in Z_i, t \neq s, 1 \quad t \quad n'_i + 1 \text{ and } z'^i_s \neq y'^i_0.$$

The third type of collision event for the arbitrary forgery $y'^i, 1 \quad i \quad q_v$, which is denoted by I_i below, includes all the instances when the initialization block y'^i_0 collides with some hidden ciphertext blocks generated during encryption (i.e., $z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_i + 1$). Formally, event I_i is defined by:

$$I_i : y'^i_0 \in S \Leftrightarrow \{y^p_0, 1 \quad p \quad q_e\},$$

or, equivalently,

$$I_i : y'^i_0 \in \{z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\},$$

The fourth type of collision event, denoted by E below, defines collisions among the initialization blocks (i.e., block 0 of the ciphertext) generated at encryption. (Hence, this collision event is independent of the forgery y'^i.) Formally, this event is defined as

$$E : y^i_0 = y^p_0,$$

where $i \neq p, 1 \quad i, p \quad q_e$.

Note 1: Other events than the four defined above could cause an adversary's forgery y'^i to pass the integrity check $g(x'^i) = x'^i_{n_i+1}$. However, Claim 1 below makes it clear that the success of such a forgery could only

[8]Recall that hidden ciphertext blocks $z'^i_s, z'^i_t \in Z_i$ do not collide with any z^p_k or with any y^p_0 obtained during encryption, where $1 \leq p \leq q_e, 1 \leq k \leq n_p + 1$.

occur with probability no greater than $1/2^l$.

Note 2: Another collision event in the input of f^{-1}, $y'^{i}_0 = y^p_0, 1 \quad i \quad q_v, 1 \quad p \quad q_e$, can be caused simply be the adversary's choice of the initial forgery block. Unlike the four events defined above (and illustrated by Examples 1 4), the occurrence of this collision event cannot cause an adversary's success in the absence of other collision events. Nevertheless, the occurrence of this event is accounted for in the proof; viz., Proof of Claim 3 below.

Upper bound on the Probability of Successful Forgery. Let F be a SPRP family, P^l be the set of all permutations on $\{0,1\}^l$, and $f \overset{\mathcal{R}}{\leftarrow} P^l$ denote the random selection of f and f^{-1} from P^l. Let $S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$ represent all the ciphertext blocks produced at the encryption of the q_e queries (viz., the definition of S used for collision events above) when the XCBC\$-XOR scheme is implemented with $f \overset{\mathcal{R}}{\leftarrow} P^l$; i.e.,

$$S_{f \overset{\mathcal{R}}{\leftarrow} P^l} = \{f(r^p_0), 1 \quad p \quad q_e\} \cup \{f(x^p_k \oplus z^p_{k-1}), 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}.$$

For any $f \overset{\mathcal{R}}{\leftarrow} P^l$ and $S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$, we define the finite family of random functions $G_S : \{0,1\}^k \quad 0,1\}^l \rightarrow \{0,1\}^l$ whose members are $f, \overline{f}$, with $\overline{f}$ defined as:

$$\overline{f} = \begin{cases} f^{-1}(t), & t \in S_{f \overset{\mathcal{R}}{\leftarrow} P^l} \\ v(t), & t \in \{0,1\}^l \Leftrightarrow S_{f \overset{\mathcal{R}}{\leftarrow} P^l}, v \overset{\mathcal{R}}{\leftarrow} R^{l,l} \end{cases},$$

where $R^{l,l}$ is the set of all functions from $\{0,1\}^l$ to $\{0,1\}^l$. We denote by $f \overset{\mathcal{R}}{\leftarrow} G_S$ the random selection of f and $\overline{f}$ from G_S.

The family of functions G_S behaves exactly like P^l when the plaintext blocks input to f and ciphertext blocks input to f^{-1} are those generated during the encryption of any adversary's q_e chosen-plaintext queries, and behaves exactly like $R^{l,l}$ during the decryption of any ciphertext block *not* in $S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$.

Note that the family G_S is well-defined for any message-integrity attack because, by definition of such an attack (viz., Section 2), all q_e encryption queries preceed all q_v forgery verification queries. Thus $S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$ and $\overline{f}$ are completely determined before any of the q_v forgery verification queries are possible, whose processing would require block decryption with $\overline{f}$. (Also note that we allow $q_e = 0$ and, in this case, $S_{f \overset{\mathcal{R}}{\leftarrow} P^l} = \emptyset$ and $\overline{f} = v$.)

For the balance of this proof, we use the result of Fact 1 below (whose proof can be found at the end of this appendix) that provides the reduction from $f \overset{\mathcal{R}}{\leftarrow} F$ to $f \overset{\mathcal{R}}{\leftarrow} G_S$.

Fact 1
(a)

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \quad \epsilon + Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}].$$

(b)

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}] \quad Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}] + \frac{\mu_v(\mu_v \Leftrightarrow l)}{l^2 2^{l+1}}.$$

Fact 1 reduces the problem to finding an upper bound for $Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}]$. Unless we state otherwise, assume that $f \overset{\mathcal{R}}{\leftarrow} G_S$ (and drop this subscript from $Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}]$.)

To compute an upper bound for the probability of successful forgery, $Pr[\text{Succ}]$, we condition on event E first, since this event does not depend on the forgery y'^i. Using standard conditioning, we obtain

$$Pr[\text{Succ}] \quad Pr[E] + Pr[\text{Succ} \mid \overline{E}].$$

Since event E is equivalent to the event that at least a collision happens when q_e balls are thrown at random in 2^l buckets [3],

$$Pr[E] \quad \frac{q_e(q_e \Leftrightarrow 1)}{2^{l+1}}.$$

To find an upper bound for $Pr[\text{Succ} \mid \overline{E}]$, we use the definition of adversary's success (viz., the attack definition), which states that at least one forgery (and verification query) y'^i succeeds; i.e., there exists an index $i, 1 \quad i \quad q_v$ such that $g(x'^i) = x'^i_{n'_i+1}$. Hence, by union bound,

$$Pr[\text{Succ} \mid \overline{E}] \quad \sum_{i=1}^{q_v} Pr[g(x'^i) = x'^i_{n'_i+1} \mid \overline{E}].$$

To find an upper bound for the probability of decrypting a single, arbitrary (non-truncation) forgery y'^i correctly given $\overline{E}$, namely for $Pr[g(x'^i) = x'^i_{n'_i+1} \mid \overline{E}]$, we condition on event (C_i or $\overline{D_i}$). Using the total probability formula we obtain:

$$\begin{aligned} Pr[g(x'^i) = x'^i_{n'_i+1} \mid \overline{E}] \quad = \quad & Pr[g(x'^i) = x'^i_{n'_i+1} \mid \overline{E} \text{ and } (C_i \text{ or } \overline{D_i})]Pr[C_i \text{ or } \overline{D_i} \mid \overline{E}] + \\ & Pr[g(x'^i) = x'^i_{n'^i+1} \mid \overline{E} \text{ and } (\overline{C_i} \text{ and } D_i)]Pr[\overline{C_i} \text{ and } D_i \mid \overline{E}]. \end{aligned}$$

Hence,[9]

$$Pr[g(x'^i) = x'^i_{n'_i+1} \mid \overline{E}] \quad Pr[C_i \text{ or } \overline{D_i} \mid \overline{E}] + Pr[g(x'^i) = x'^i_{n'^i+1} \mid \overline{E} \text{ and } \overline{C_i} \text{ and } D_i].$$

However, both event C_i and event $\overline{D_i}$ depend on the event I_i (viz., Example 3 above). Hence, to compute $Pr[C_i \text{ or } \overline{D_i} \mid \overline{E}]$ we condition on event I_i and, using the total probability formula, we obtain:

$$\begin{aligned} Pr[C_i \text{ or } \overline{D_i} \mid \overline{E}] \quad = \quad & Pr[C_i \text{ or } \overline{D_i} \mid \overline{E} \text{ and } I_i]Pr[I_i \mid \overline{E}] + Pr[C_i \text{ or } \overline{D_i} \mid \overline{E} \text{ and } \overline{I_i}]Pr[\overline{I_i} \mid \overline{E}] \\ & Pr[I_i \mid \overline{E}] + Pr[C_i \text{ or } \overline{D_i} \mid \overline{E} \text{ and } \overline{I_i}]. \end{aligned}$$

Furthermore,

$$\begin{aligned} Pr[C_i \text{ or } \overline{D_i} \mid \overline{E} \text{ and } \overline{I_i}] \quad = \quad & Pr[C_i \text{ or } \overline{D_i} \mid \overline{C_i} \text{ and } \overline{E} \text{ and } \overline{I_i}]Pr[\overline{C_i} \mid \overline{E} \text{ and } \overline{I_i}] \\ & + Pr[C_i \text{ or } \overline{D_i} \mid C_i \text{ and } \overline{E} \text{ and } \overline{I_i}]Pr[C_i \mid \overline{E} \text{ and } \overline{I_i}] \\ & Pr[C_i \text{ or } \overline{D_i} \mid \overline{C_i} \text{ and } \overline{E} \text{ and } \overline{I_i}] + Pr[C_i \mid \overline{E} \text{ and } \overline{I_i}] \\ & = Pr[C_i \mid \overline{E} \text{ and } \overline{I_i}] + Pr[\overline{D_i} \mid \overline{C_i} \text{ and } \overline{E} \text{ and } \overline{I_i}], \end{aligned}$$

since event $[C_i \text{ or } \overline{D_i} \mid \overline{C_i} \text{ and } \overline{E} \text{ and } \overline{I_i}]$ is equivalent to event $[\overline{D_i} \mid \overline{C_i} \text{ and } \overline{E} \text{ and } \overline{I_i}]$.

[9] This also follows from our pessimistic assumption that if event (C_i or $\overline{D_i}$) is true, then the adversary has broken integrity.

Combining the results of the last three inequalities, we obtain:

$$\begin{aligned} Pr[g(x'^{i}) = x'^{i}_{n'_i+1} \mid \overline{E}] \quad & \quad Pr[g(x'^{i}) = x'^{i}_{n'_i+1} \mid \overline{E} \text{ and } \overline{C_i} \text{ and } D_i] + \\ & \quad Pr[I_i \mid \overline{E}] + Pr[C_i | \overline{E} \text{ and } \overline{I_i}]. + Pr[\overline{D_i} \mid \overline{C_i} \text{ and } \overline{E} \text{ and } \overline{I_i}] \end{aligned}$$

The probabilities that appear at the right side of this inequality are bounded as shown in the following four claims whose proofs are included below. (Note again that forgeries based on truncations of ciphertext messages obtained at encryption are *not* included in any of the claims below. All these claims refer to a single, arbitrary (non-truncation) forgery $y'^{i}, 1 \quad i \quad q_v$.)

Claim 1

$$Pr[g(x'^{i}) = x'^{i}_{n'_i+1} \mid \overline{E} \text{ and } \overline{C_i} \text{ and } D_i] \quad \frac{1}{2^l}.$$

Claim 2

$$Pr[I_i \mid \overline{E}] \quad \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2 \frac{\mu_e}{l} + 3 \quad .$$

Claim 3

$$Pr[C_i | \overline{E} \text{ and } \overline{I_i}] \quad \frac{(n'_i+1)q_e}{2^l} + \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2 \frac{\mu_e}{l} + 3 \quad .$$

Claim 4

$$Pr[\overline{D_i} \mid \overline{C_i} \text{ and } \overline{E} \text{ and } \overline{I_i}] \quad \frac{2n'_i+1}{2^l}.$$

Note that if the maximum length m of the encrypted messages is known, the $\log_2 \frac{\mu_e}{l}$ term of Claims 2 and 3 can be replaced with $\log_2 m$.

By Claims 1–4, the probability of success given $\overline{E}$ for a single, arbitrary (non-truncation) forgery is

$$\begin{aligned} Pr[g(x'^{i}) = x'^{i}_{n'_i+1} \mid \overline{E}] \quad & \quad \frac{1}{2^l} + \frac{(n'_i+1)q_e}{2^l} + \frac{1}{2^l} \quad \frac{\mu_e}{l}\log_2 \frac{\mu_e}{l} + \frac{3\mu_e}{l} \quad + \frac{2n'_i+1}{2^l} \\ & = \frac{(n'_i+1)(q_e+2)}{2^l} + \frac{1}{2^l} \quad \frac{\mu_e}{l}\log_2 \frac{\mu_e}{l} + \frac{3\mu_e}{l} \quad . \end{aligned}$$

Hence, the probability of adversary's success when he has up to q_v verification queries totaling at most μ_v bits and using up to t_v time is bounded by

$$\begin{aligned} Pr[\text{Succ}] \quad & \quad Pr[E] + \sum_{i=1}^{q_v} Pr[g(x'^{i}) = x'^{i}_{n'_i+1} \mid \overline{E}] \\ & \quad \frac{q_e(q_e \Leftrightarrow 1)}{2^{l+1}} + \sum_{i=1}^{q_v} \quad \frac{(n'_i+1)(q_e+2)}{2^l} + \frac{1}{2^l} \quad \frac{\mu_e}{l}\log_2 \frac{\mu_e}{l} + \frac{3\mu_e}{l} \\ & \quad \frac{q_e(q_e \Leftrightarrow 1)}{2^{l+1}} + \frac{\mu_v(q_e+2)}{l2^l} + \frac{q_v}{2^l} \quad \frac{\mu_e}{l}\log_2 \frac{\mu_e}{l} + \frac{3\mu_e}{l} \end{aligned}$$

because $\sum_{i=1}^{q_v}(n'_i+1) \quad \frac{\mu_v}{l}$.

Furthermore, by using Fact 1, the probability of adversary's success when $f \overset{\mathcal{R}}{\leftarrow} F$ is bounded by:

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

Also, if the maximum length m of the encrypted messages is known, the last term of the above bounds can be replaced with $\frac{q_v}{2^l} \quad \frac{\mu_e}{l} \log_2 m + \frac{3\mu_e}{l}$.

The parameters of the attack are bounded as follows: $q_e \quad q'$, since the scheme is also supposed to be chosen-plaintext secure, $t_e + t_v \quad t$, and $\mu'' = \mu_e + \mu_v \quad ql$. The forgery verification parameters q_v, μ_v, t_v can be chosen within the constraints of these bounds and the desired $Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[Succ]$. □

Proofs of Claims 1–4

Notation: Recall that Claims 1 4 above refer to a *single*, arbitrary (non-truncation) forgery $y'^i, 1 \quad i \quad q_v$. Hence, to simplify notation in the proof of these claims, we drop the forgery index i from the events D_i, C_i, I_i, and simply use D, C, I for these events. We also drop the forgery index i from the collection Z_i and use Z instead. Furthermore, we drop the prime and forgery index i from the ciphertext y'^i, hidden ciphertext, z'^i, plaintext x'^i, $r_0'^i$, and the length n_i'. Hence, when we refer to the (single) forgery, we use the variables y, for forgery ciphertext, x for forgery plaintext, z for the hidden blocks of forgery y, y_0 for the initialization block of forgery y (and r_0 for the decryption of the initialization block y_0), and n for the length of x. Superscripts continue to identify encryption queries. In the proof of Claims 1 4, we use the notation $Pr_A[\ .\ ] = Pr[\ .\ |A]$, where A is an arbitrary event.

Proof of Claim 1

If $\overline{C}$ is true, then Z is not empty. For any $z_s \in Z$,

$$x_s = \overline{f}(z_s) \oplus z_{s-1}$$

Since z_s does not collide with any hidden blocks obtained at encryption, and event ($\overline{C}$ and D) is true (i.e., there is at least one hidden block $z_s \in Z$ by event $\overline{C}$ that does not collide with another hidden ciphertext block $z_t \in Z, s \neq t$ or with y_0 by event D), then $\overline{f}(z_s) = v(z_s)$ is uniformly distributed and independent of anything else (since $v \overset{\mathcal{R}}{\leftarrow} R^{l,l}$); i.e., independent of any other $\overline{f}(z_k), z_k \in Z$, $k \neq s$, and independent of any $z_k, 0 \quad k \quad n+1$. Hence, the corresponding plaintext block x_s is uniformly distributed and independent of anything else. Thus,

$$g(x) \oplus x_{n+1} = z_0 \oplus x_1 \oplus \quad \oplus x_n \oplus x_{n+1}$$

is random and uniformly distributed, and hence:

$$Pr[g(x) \oplus x_{n+1} = 0 \mid \overline{E} \text{ and } \overline{C} \text{ and } D] = Pr[g(x) = x_{n+1} \mid \overline{E} \text{ and } \overline{C} \text{ and } D] \quad \frac{1}{2^l}.$$

□

In the proofs of Claims 2 4, we use the following three facts, whose proofs can be found at the end of this appendix.

Fact 2
For any $1 \quad i \quad 2^l \Leftrightarrow 1$, let m be defined by $i = d \quad 2^m$, where d is odd. If r_0 is random and uniformly distributed, then for any constant a,

$$Pr[i \quad r_0 = a] \quad \frac{1}{2^{l-m}}.$$

Fact 3
For any $N > 1$, let m be defined by $a = d \quad 2^m$, where $1 \quad a \quad N \Leftrightarrow 1$ and d is odd. Then

$$\sum_{a=1}^{N-1} 2^m \quad \frac{N \Leftrightarrow 1}{2}(\log_2(N \Leftrightarrow 1) + 3).$$

Fact 4
If for any $p, 1 \quad p \quad q_e, n_p > 0$, and if $\sum_{p=1}^{q_e}(n_p + 1) \quad \frac{\mu_e}{l}$, then,

$$\sum_{p=1}^{q_e}(n_p + 1)\log_2(n_p + 1) \quad \frac{\mu_e}{l}\log_2\frac{\mu_e}{l};$$

and, further, if $m = \max(n_p + 1)$, then

$$\sum_{p=1}^{q_e}(n_p + 1)\log_2(n_p + 1) \quad \frac{\mu_e}{l}\log_2 m.$$

Proof of Claim 2

Event $I : y_0 \in S \Leftrightarrow \{y_0^p, 1 \quad p \quad q_e\} = \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}$ is equivalent to the union of all possible events $y_0 = z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1$. Hence, by union bound,

$$Pr[I \mid \overline{E}] \quad \sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr[y_0 = z_k^p \mid \overline{E}].$$

We determine an upper bound for $Pr[y_0 = z_k^p \mid \overline{E}]$ based on

$$y_0 = z_k^p \;\Leftrightarrow\; y_0 = y_k^p \Leftrightarrow k \quad r_0^p \;\Leftrightarrow\; k \quad r_0^p = y_k^p \Leftrightarrow y_0.$$

In this expression, r_0^p is random and uniformly distributed, and from the definition of event E, if $\overline{E}$ is true, then r_0^p is random and uniformly distributed. Hence, since $y_k^p \Leftrightarrow y_0$ is a known constant, by Fact 2,

$$Pr[y_0 = z_k^p \mid \overline{E}] = Pr[k \quad r_0^p = y_k^p \Leftrightarrow y_0 \mid \overline{E}] \quad \frac{1}{2^{l-m}},$$

where the exponent m is defined by $k = d \quad 2^m$ and d is odd. Hence, for each $p, 1 \quad p \quad q_e$, from this and Fact 3 with $N \Leftrightarrow 1 = n_p + 1$ and $a = k$,

$$\sum_{k=1}^{n_p+1} Pr[y_0 = z_k^p \mid \overline{E}] \quad \frac{1}{2^l}\sum_{k=1}^{n_p+1} 2^m \quad \frac{1}{2^l}\frac{n_p + 1}{2}(\log_2(n_p + 1) + 3).$$

Since $\sum_{p=1}^{q_e}(n_p + 1) \quad \frac{\mu_e}{l}$ by the definition of $n + p$ and of the attack, we obtain

$$Pr[I \mid \overline{E}] \quad \sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr[y_0 = z_k^p \mid \overline{E}] \quad \frac{1}{2^l}\sum_{p=1}^{q_e}\frac{n_p + 1}{2}(\log_2(n_p + 1) + 3) \quad \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2\frac{\mu_e}{l} + 3 \quad ,$$

by Fact 4. Further, if $m = \max(n_p + 1)$, then $Pr[I \mid \overline{E}] \quad \frac{1}{2^l}\frac{\mu_e}{2l}(\log_2 m + 3)$, also by Fact 4. □

Proof of Claim 3

Below we use the notation that $Pr_A[\ .\ ] = Pr[\ .\ |A]$, where A is an arbitrary event.

C is equivalent to the event that every hidden ciphertext block obtained during decryption is found among the hidden ciphertext blocks obtained during encryption or among the y_0^p blocks obtained at encryption. This implies that *for any* $s, 1 \quad s \quad n+1$: $Pr_{\overline{I} \text{ and } \overline{E}}[C] \quad Pr_{\overline{I} \text{ and } \overline{E}}[z_s \in S]$ by union bound. Since, $S = \{y_0^p, 1 \quad p \quad q_e\} \cup \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}$, it follows that, by union bound,

$$\begin{aligned} Pr_{\overline{I} \text{ and } \overline{E}}[z_s \in S] &\quad Pr_{\overline{I} \text{ and } \overline{E}}[z_s \in \{y_0^p, 1 \quad p \quad q_e\}] \\ &+ \quad Pr_{\overline{I} \text{ and } \overline{E}}[z_s \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}]. \end{aligned}$$

For the first term, for any $s, 1 \quad s \quad n+1$, the event $z_s \in \{y_0^p, 1 \quad p \quad q_e\}$ is the union of all collision events $z_s = y_0^p, 1 \quad p \quad q_e$. Hence,

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_s \in \{y_0^p, 1 \quad p \quad q_e\}] \quad \sum_{p=1}^{q_e} Pr_{\overline{I} \text{ and } \overline{E}}[z_s = y_0^p].$$

But $z_s = y_s \Leftrightarrow s \quad r_0$ by the scheme definition, and hence $s \quad r_0 = y_s \Leftrightarrow y_0^p$. To compute $Pr_{\overline{I} \text{ and } \overline{E}}[s \quad r_0 = y_s \Leftrightarrow y_0^p]$, we use the following claim, whose proof can be found at the end of this appendix:

Claim 3.1
Let $y_0^p y_1^p \quad y_{n_p+1}^p$ be a queried message, and $y = y_0 y_1 \quad y_{n+1}$ be a forged ciphertext. If event $\overline{I}$ is true, then r_0 is random and uniformly distributed. Furthermore, if $y_0 \neq y_0^p$, then r_0 is also independent of r_0^p.

Since event $\overline{I}$ is true, it follows that r_0 is random and uniformly distributed (by Claim 3.1 above). Also, event $\overline{I}$ and $\overline{E}$ implies that r_0 is random and uniformly distributed by the definition of event E. Hence, by Fact 2,

$$Pr_{\overline{I} \text{ and } \overline{E}}[s \quad r_0 = y_s \Leftrightarrow y_0^p] \quad \frac{1}{2^{l-m}},$$

where m is defined by $s = d \quad 2^m$ and d is odd. Furthermore, $m \quad \log_2 s \quad \log_2(n+1)$, since $s \quad n+1$. Hence, $2^m \quad n+1$, and

$$Pr_{\overline{I} \text{ and } \overline{E}}[s \quad r_0 = y_s \Leftrightarrow y_0^p] \quad \frac{n+1}{2^l}.$$

Hence, for any $s, 1 \quad s \quad n+1$:

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_s \in \{y_0^p, 1 \quad p \quad q_e\}] \quad \sum_{p=1}^{q_e} \frac{n+1}{2^l} = \frac{(n+1)q_e}{2^l}.$$

To compute an upper bound for the second term, namely on $Pr_{\overline{I} \text{ and } \overline{E}}[z_s \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}]$, we are free to choose a hidden ciphertext block at index j of forgery y, namely z_j, and then we only need to show that $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}]$, is bounded. (This is the case because the bound must be true *for any* $s, 1 \quad s \quad n+1$.)

Thus, the balance of the proof of Claim 3 consists of two parts. In the first part, we partition the space of forgeries that are not truncations into three complementary types and choose a z_j (and hence, index j) for each type. In the second part, we find an upper bound for the probability $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}]$ for each of the chosen z_j's. Hence, the maximum of these three upper bounds represents the upper bound for $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}]$ for all forgeries that are not truncations.

Part 1. Finding index j depends on the type of forgery. A forgery can be such that a ciphertext obtained at encryption is the prefix of the forgery; we call this the *prefix* case. The complementary case for the prefix case, which we call *non-prefix*, includes two separate subcases, namely when y_0 is different from any y^i_0 of any ciphertext obtained at encryption, or when there is an index i such that $y_0 = y^i_0$. Hence, in the latter case, there must be at least a block in the forged ciphertext y that is different from the corresponding block of the ciphertext of a queried message i, namely y^i. Further, the length of the forged ciphertext y, denoted by n, may be different from the length of the message plaintext defined by n_i.

This partition of forgery types shows that a forged ciphertext $y = y_0 y_1 \quad y_{n+1}$, which is not a truncation, can be in one of the following three complementary types:
(a) $\exists i, 1 \quad i \quad q_e : n > n_i, \forall k, 0 \quad k \quad n_i + 1 : y_k = y^i_k$; i.e., the forged ciphertext is an extension of the ciphertext y^i (the prefix case). The non-prefix case consists of the following two forgery types:
(b1) $y_0 \neq y^i_0, \forall i, 1 \quad i \quad q_e$; i.e., the forged ciphertext and all queried-message ciphertexts differ in the first block.
(b2) $\exists i, 1 \quad i \quad q_e : y_0 = y^i_0, \exists k, 1 \quad k \quad \min(n_i + 1, n + 1) : y_k \neq y^i_k$; i.e., the forged ciphertext is obtained by modifying a queried message ciphertext starting with some block between the second and last block of that queried-message ciphertext. In this case, let j be the smallest index such that $y_j \neq y^i_j$ (i.e., $\forall k, 0 \quad k \quad j \Leftrightarrow 1 : y_k = y^i_k$).

Let us choose index j (and hence z_j) as follows. For forgeries of type (a), $j = n_i + 2$ (or $j > n_i + 1$); for forgeries of type (b1), $j = 1$; and for forgeries of type (b2), j is the smallest index such that $y_j \neq y^i_j, 1 \quad j \quad \min\{n_i + 1, n + 1\}$. In all cases $j \geq 1$, and hence, the chosen ciphertext block z_j is well defined.

Part 2. For the index j chosen in Part 1, we find an upper bound for $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}]$. Event $z_j \in \{z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}$ is the union of all possible events $z_j = z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1$. Hence, union bound leads to:

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z^p_k, 1 \quad p \quad q_e, 1 \quad k \quad n_p + 1\}] \quad \sum_{p=1}^{q_e} \sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^p_k].$$

Now we find an upper bound for $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^p_k]$ for each of the three forgery types. In determining this upper bound, we use the following claim, whose proof can be found at the end of this appendix:

Claim 3.2
Let $z^p_k, 1 \quad p \quad q_e$, be the hidden ciphertext blocks generated at the encryption of a queried message $y^p_0 y^p_1 \quad y^p_{n_p+1}$, and z_j be the chosen hidden ciphertext block generated during the decryption of forgery $y = y_0, y_1, \quad y_{n+1}$. Then $\forall k, 1 \quad k \quad n_p + 1$,

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^p_k] \quad \frac{1}{2^{l-m}},$$

where
(a) if $y_0 \neq y^p_0$, then $m = \min(m_1, m_2)$, with m_1 and m_2 being defined by $j = d_1 \quad 2^{m_1}$, $k = d_2 \quad 2^{m_2}$, where d_1, d_2 are odd; and

(b) if $y_0 = y_0^p$, where m is defined by $k \Leftrightarrow j = d \quad 2^m$ if $k > j$, or by $j \Leftrightarrow k = d \quad 2^m$ if $j < \ k$, and d is odd.

Claim 3.2 provides upper bounds for $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p]$, where p, k are arbitrary values that satisfy the hypotheses of parts (a) or (b) and z_j is the chosen hidden ciphertext block defined in Part 1. These hypotheses are verified for the chosen j of each forgery type as shown below.

Upper bound for forgeries of type (a).
Let the ciphertext of queried message i be the prefix of forgery y. To find the upper bound in this case, we partition the sum $\sum_{p=1}^{q_e} \sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p]$ into two sums, for $p \neq i$ and $p = i$, respectively. For $p \neq i$, we use Claim 3.2(a), and for $p = i$ we use Claim 3.2(b), to find an upper bound for $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p]$. Then we find individual upper bounds for each of these two sums, and add these upper bounds.

$$\sum_{p=1}^{q_e} \sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \ = \ \sum_{p=1, p\neq i}^{q_e} \sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] + \sum_{k=1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^i].$$

For the first sum, note that $p \neq i$, and recall that for forgeries of type (a) $y_0 = y_0^i$. Since $\overline{E}$ is true, $y_0 = y_0^i \neq y_0^p$. Hence, by Claim 3.2(a), $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^{l-m}}$, where $m \quad m_2$ with m_2 being defined by $k = d_2 \quad 2^{m_2}$ and d_2 is odd. Thus,

$$\sum_{p=1, p\neq i}^{q_e} \sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^l} \sum_{p=1, p\neq i}^{q_e} \sum_{k=1}^{n_p+1} 2^{m_2}.$$

But, by Fact 3 with $N \Leftrightarrow 1 = n_p + 1$ and $a = k$,

$$\sum_{k=1}^{n_p+1} 2^{m_2} \quad \frac{n_p+1}{2}(\log_2(n_p+1)+3).$$

Hence,

$$\sum_{p=1, p\neq i}^{q_e} \sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^l} \sum_{p=1, p\neq i}^{q_e} \frac{n_p+1}{2}(\log_2(n_p+1)+3).$$

For the second sum, we note that $p = i$, which means that $y_0 = y_0^i = y_0^p$, and that $j = n_i + 2 > k, \forall k, 1 \quad k \quad n_i + 1$. Hence, by Claim 3.2(b) $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^{l-m}}$, where $j \Leftrightarrow k = d \quad 2^m$ and d is odd. Since $j = n_i + 2$, in follows that $j \Leftrightarrow k = n_i + 1, \quad , 1$, and thus,

$$\sum_{k=1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^i] \quad \sum_{j-k=1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^i] \frac{1}{2^l} \sum_{j-k=1}^{n_i+1} 2^m.$$

But, by Fact 3 with $N \Leftrightarrow 1 = n_i + 1$ and $a = j \Leftrightarrow k$,

$$\sum_{j-k=1}^{n_i+1} 2^m \quad \frac{n_i+1}{2}(\log_2(n_i+1)+3),$$

and hence,

$$\sum_{k=1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^i] \quad \frac{1}{2^l}\frac{n_i+1}{2}(\log_2(n_i+1)+3).$$

Adding the two upper bounds, we obtain

$$\sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \qquad \frac{1}{2^l}\frac{n_i+1}{2}(\log_2(n_i+1)+3) + \frac{1}{2^l}\sum_{p=1,p\neq i}^{q_e}\frac{n_p+1}{2}(\log_2(n_p+1)+3)$$
$$= \frac{1}{2^l}\sum_{p=1}^{q_e}\frac{n_p+1}{2}(\log_2(n_p+1)+3).$$

Since $\sum_{p=1}^{q_e}(n_p+1) \quad \frac{\mu_e}{l}$, by Fact 4, it follows that

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}]$$
$$\sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2\frac{\mu_e}{l} + 3 \quad .$$

Further, if $m = \max(n_p+1)$, then $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}] \quad \frac{1}{2^l}\frac{\mu_e}{2l}(\log_2 m + 3)$, also by Fact 4.

Upper bound for forgeries of type (b1).
For this type of forgery, $y_0 \neq y_0^p, \forall p, 1 \quad p \quad q_e$. Hence, by Claim 3.2(a), $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^{l-m}}$, where $m \quad m_2$ with m_2 being defined by $k = d_2 \quad 2^{m_2}$ and d_2 is odd. By following the same derivation as that for forgeries of type (a), we obtain:

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}] \quad \sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p]$$
$$\frac{1}{2^l}\sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} 2^{m_2} \quad \frac{1}{2^l}\sum_{p=1}^{q_e}\frac{n_p+1}{2}(\log_2(n_p+1)+3) \quad \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2\frac{\mu_e}{l} + 3 \quad .$$

Also, if $m = \max(n_p+1)$, then $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}] \quad \frac{1}{2^l}\frac{\mu_e}{2l}(\log_2 m + 3)$.

Upper bound for forgeries of type (b2).
Let the first $j \Leftrightarrow 1$ ciphertext blocks of queried message i provide the first $j \Leftrightarrow 1$ ciphertext blocks of forgery y. To find the upper bound in this case, we partition the sum $\sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p]$ into four terms, find individual upper bounds for each term, and then add these upper bounds. The first term is a sum taken for $p \neq i$ and in this case we use Claim 3.2(a) to find an upper bound for $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p]$. The last three terms are for the case $p = i$, and two of these terms are sums taken for $k \quad$ and $k > j$, respectively. For these sums, we apply Claim 3.2(b) to find an upper bound for $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p]$. For the remaining term corresponding to $i = p$ and $k = j$, we show that the event $z_j = z_k^p$ is impossible.

$$\sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] = \sum_{p=1,p\neq i}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] + \sum_{k=1}^{j-1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^i] + Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_j^i] +$$
$$\sum_{k=j+1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^i].$$

For the first of the four terms above, we have the same bound as that of the first of the two sums in the case of forgeries of type (a) above, namely,

$$\sum_{p=1,p\neq i}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^l}\sum_{p=1,p\neq i}^{q_e}\frac{n_p+1}{2}(\log_2(n_p+1)+3).$$

For the second term, namely $\sum_{k=1}^{j-1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k]$, we note that $i = p$, which means that $y_0 = y^i_0 = y^p_0$, and $k< \; j$. Hence, by Claim 3.2(b), $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] \quad \frac{1}{2^{l-m}}$, where $j \Leftrightarrow k = d \quad 2^m$ and d is odd. Since $k = 1, \quad , j \Leftrightarrow 1$, it follows that $j \Leftrightarrow k = j \Leftrightarrow 1, \quad , 1$, and by Fact 3 with $N \Leftrightarrow 1 = j \Leftrightarrow 1$ and $a = j \Leftrightarrow k$,

$$\sum_{k=1}^{j-1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] \;=\; \sum_{j-k=1}^{j-1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] \quad \frac{1}{2^l} \sum_{j-k=1}^{j-1} 2^m \quad \frac{1}{2^l}\frac{j \Leftrightarrow 1}{2}(\log_2(j \Leftrightarrow 1) + 3).$$

For the third term, $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_j] = 0$. This is the case because $z_j = z^i_j \;\Leftrightarrow\; y_j \Leftrightarrow j \quad r_0 = y^i_j \Leftrightarrow j \quad r^i_0$ and, since $y_0 = y^i_0 \;\Leftrightarrow\; r_0 = r^i_0$, it follows that $z_j = z^i_j \;\Leftrightarrow\; y_j = y^i_j$, which is impossible by the definition of j. (Recall that for forgeries of type (b2), j is the smallest index such that $y_j \neq y^i_j, 1 \quad j \quad \min\{n_i+1, n+1\}$.)

For the fourth term, namely $\sum_{k=j+1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k]$, we note that $i = p$, which means that $y_0 = y^i_0 = y^p_0$, and $j< \; k$. Hence, by Claim 3.2(b), $Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] \quad \frac{1}{2^{l-m}}$, where $k \Leftrightarrow j = d \quad 2^m$ and d is odd. Since $k = j+1, \quad , n_i + 1$, it follows that $k \Leftrightarrow j = 1, \quad , n_i \Leftrightarrow j + 1$, and by Fact 3 with $N \Leftrightarrow 1 = n_i + 1 \Leftrightarrow j$ and $a = k \Leftrightarrow j$,

$$\sum_{k=j+1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] \;=\; \sum_{k-j=1}^{n_i-j+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] \quad \frac{1}{2^l} \sum_{k-j=1}^{n_i-j+1} 2^m$$
$$\frac{1}{2^l}\frac{n_i \Leftrightarrow j+1}{2}(\log_2(n_i \Leftrightarrow j+1) + 3).$$

Now, we add the bounds of the last three of the individual upper bounds, and then we add the first upper bound to obtain the total upper bound for forgeries of type (b2).

$$\sum_{k=1}^{j-1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] + Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_j] + \sum_{k=j+1}^{n_i+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k]$$
$$\frac{1}{2^l}\frac{j \Leftrightarrow 1}{2}(\log_2(j \Leftrightarrow 1) + 3) + \frac{1}{2^l}\frac{n_i \Leftrightarrow j+1}{2}(\log_2(n_i \Leftrightarrow j+1) + 3).$$

Since for this type of forgeries $1 \quad j \quad n_i + 1$, the terms under $\log_2$ are $j \Leftrightarrow 1 \quad n_i, n_i \Leftrightarrow j + 1 \quad n_i$. Thus, the sum of the last three terms is bounded as follows:

$$\sum_{k=1}^{j-1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k] + Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_j] + \sum_{k=j+1}^{n_i-j+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^i_k]$$
$$\frac{1}{2^l}\frac{j \Leftrightarrow 1}{2}(\log_2 n_i + 3) + \frac{1}{2^l}\frac{n_i \Leftrightarrow j+1}{2}(\log_2 n_i + 3) = \frac{1}{2^l}\frac{n_i}{2}(\log_2 n_i + 3)$$
$$\frac{1}{2^l}\frac{n_i+1}{2}(\log_2(n_i+1) + 3).$$

Hence, by adding the first of the individual upper bounds to this above sum, we obtain:

$$\sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z^p_k] \qquad \frac{1}{2^l}\frac{n_i+1}{2}(\log_2(n_i+1) + 3) +$$
$$\frac{1}{2^l}\sum_{p=1, p\neq i}^{q_e}\frac{n_p+1}{2}(\log_2(n_p+1) + 3)$$
$$= \frac{1}{2^l}\sum_{p=1}^{q_e}\frac{n_p+1}{2}(\log_2(n_p+1) + 3).$$

Since $\sum_{p=1}^{q_e}(n_p+1) \quad \frac{\mu_e}{l}$, by Fact 4, it follows that

$$\begin{aligned} &Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}] \\ &\sum_{p=1}^{q_e}\sum_{k=1}^{n_p+1} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2\frac{\mu_e}{l}+3 \quad . \end{aligned}$$

Further, if $m = \max(n_p+1)$, then $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}] \quad \frac{1}{2^l}\frac{\mu_e}{2l}(\log_2 m+3)$.

Finally, for any forgery that is not a truncation, $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}]$ is bounded by the maximum of the bounds for the types (a), (b1) and (b2), namely

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}] \quad \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2\frac{\mu_e}{l}+3 \quad ;$$

or, if $m = \max(n_p+1)$, then $Pr_{\overline{I} \text{ and } \overline{E}}[z_j \in \{z_k^p, 1 \quad p \quad q_e, 1 \quad k \quad n_p+1\}] \quad \frac{1}{2^l}\frac{\mu_e}{2l}(\log_2 m+3)$. Hence, returning to the probability of event C conditioned by ($\overline{I}$ and $\overline{E}$),

$$Pr_{\overline{I} \text{ and } \overline{E}}[C] = Pr[C \mid \overline{I} \text{ and } \overline{E}] \quad \frac{(n+1)q_e}{2^l} + \frac{1}{2^l}\frac{\mu_e}{2l} \quad \log_2\frac{\mu_e}{l}+3 \quad .$$

Also, if the maximum length m of the encrypted messages is known, the last term of the above bound can be replaced with $\frac{1}{2^l}\frac{\mu_e}{2l}(\log_2 m+3)$. □

Proof of Claim 4

Event $\overline{C}$ is true implies that there is at least one element $z_s \in Z$. Event $\overline{D}$ states that any hidden ciphertext block $z_s \in Z$ collides with another hidden block $z_t \in Z, t \neq s$, or z_s collides with y_0. Hence, $\overline{D} \Rightarrow (z_s = z_t,$ for some $s, t, 1 \quad s, t \quad n+1, z_s, z_t \in Z, s \neq t$ or $z_s = y_0)$. This implies that

$$Pr[\overline{D} \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \quad Pr[(z_s = z_t, z_s, z_t \in Z, t \neq s) \text{ or } z_s = y_0 \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}]$$

Union bound leads to:

$$\begin{aligned} Pr[\overline{D} \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \quad & Pr[z_s = z_t, z_s, z_t \in Z, t \neq s \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \\ & +Pr[z_s = y_0 \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \end{aligned}$$

To compute the upper bound of the first probability of the sum, $Pr[z_s = z_t, z_s, z_t \in Z, t \neq s \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}]$, recall that Z must have at least one element (since $\overline{C}$ is true). If Z has only one element, then this probability is zero. If Z has at least two elements, z_s, z_t, we use the following claim, whose proof can be found at the end of this Appendix:

Claim 4.1
(a) For any $z_s, z_t \in Z, 1 \quad s \quad t \quad n+1$:

$$Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[z_s = z_t] \quad \frac{1}{2^{l-m}},$$

where the exponent m is defined by $t \Leftrightarrow s = d \quad 2^m$ and d is odd.
(b) For any $z_s \in Z, 1 \quad s \quad n+1$, and for any y_0:

$$Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[z_s = y_0] \quad \frac{1}{2^{l-m}},$$

where the exponent m is defined by $s = d \quad 2^m$ and d is odd.

Then, by Claim 4.1(a)

$$Pr[z_s = z_t, z_s, z_t in Z, t \neq s \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \quad \frac{2^m}{2^l}$$

where $m \quad \log_2(t \Leftrightarrow s)$ if $t > s$, or $m \quad \log_2(s \Leftrightarrow t)$ if $t \quad s$. But, $|t \Leftrightarrow s| \leq n$; hence $m \quad \log_2 n$, and then $2^m \quad n$. Thus,

$$Pr[z_s = z_t, z_s, z_t \in Z, t \neq s \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \quad \frac{n}{2^l}.$$

To compute an upper bound for the second probability of the above sum, namely on $Pr[z_s = y_0 \mid \overline{C}$ and $\overline{E}$ and $\overline{I}]$, we use Claim 4.1(b) and obtain:

$$Pr[z_s = y_0 \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \quad \frac{1}{2^{l-m}},$$

where m is defined by $s = d \quad 2^m$ and d is odd. By definition, $m \quad \log_2 s \quad \log_2(n+1)$, and hence $2^m \quad n+1$. Thus,

$$Pr[z_s = y_0 \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \quad \frac{n+1}{2^l}.$$

By adding the two upper bounds, it follows that

$$Pr[\overline{D} \mid \overline{C} \text{ and } \overline{E} \text{ and } \overline{I}] \quad \frac{n}{2^l} + \frac{n+1}{2^l} = \frac{2n+1}{2^l}.$$

□

Proof of Fact 1

(a) Let A be an adversary attacking the $XCBC\$ \Leftrightarrow XOR$ mode using $q_e + q_v$ queries, $\mu_e + \mu_v$ total memory for these queries, and time $t_e + t_v$. The probability of success is related directly to the security of the underlying encryption mode XCBC\$ and F. To find an upper bound for this probability, we introduce a distinguisher D for F, which is given two oracles f and f^{-1}, where f is a permutation used by the $XCBC\$ \Leftrightarrow XOR$ mode. D runs A, simulates an oracle for $XCBC\$ \Leftrightarrow XOR$ via queries for its own oracles f and f^{-1}, responds to A's q_e encryption queries, and verifies A's choices of ciphertext forgeries $y'^i = y_0'^i y_1'^i \quad y_n'^i, y_{n+1}'^i, 1 \quad i \quad q_v$. D returns the result of each y'^i's verification to A; i.e., D returns either *Success* or *Failure* to A. D outputs 1 if A's forgery decrypts successfully, and 0, otherwise.

Distinguisher D's advantage, $Adv_D(F, P^l) \quad \epsilon$, is defined as:

$$Adv_D^{sprp}(F, P^l) = Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[D^f = 1] \Leftrightarrow Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[D^f = 1].$$

where $f \overset{\mathcal{R}}{\leftarrow} F$ denotes the selection of function f from the SPRP family F by the random key K, and $f \overset{\mathcal{R}}{\leftarrow} P^l$ denotes the random selection of f from the set of all permutations P^l.

By the definition of the distinguisher algorithm:

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[D^f = 1] = Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\mathcal{D} \Leftrightarrow XCBC\$ \Leftrightarrow XOR(y) \neq Null] = Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[Succ]$$

and

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[D^f = 1] = Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\mathcal{D} \Leftrightarrow XCBC\$ \Leftrightarrow XOR(y) \neq Null] = Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}].$$

The above probabilities are over the random choice of r_0, $f \overset{\mathcal{R}}{\leftarrow} F$, $f \overset{\mathcal{R}}{\leftarrow} P^l$, and D's guesses. Hence,

$$\begin{aligned} Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succc}] &= Pr_{f \overset{\mathcal{R}}{\leftarrow} F}[\text{Succ}] \Leftrightarrow Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}] + Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}] \\ &= Adv_D^{sprp}(F, P^l) + Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}] \quad \epsilon + Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}]. \end{aligned}$$

(b) This proof is based on constructing a polynomial-time algorithm D that distinguishes between $f^{-1} \overset{\mathcal{R}}{\leftarrow} P^l$ and $\overline{f} \overset{\mathcal{R}}{\leftarrow} G_S$ using an adversary A for the $XCBC\$ \Leftrightarrow XOR$ mode.

In a similar manner to the one used in part (a) (repeated here for completeness), let A be an adversary attacking the $XCBC\$ \Leftrightarrow XOR$ mode using $q_e + q_v$ queries, $\mu_e + \mu_v$ total memory for these queries, and time $t_e + t_v$. To find an upper bound for $Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[Succ]$, we introduce a distinguisher D for P^l which is given two oracles $\mathcal{O}$, $\mathcal{O}^{-1}$. These oracles simulate the block encryption and decryption operations needed by D to simulate the $XCBC\$ \Leftrightarrow XOR$ mode for adversary A. Oracle $\mathcal{O}$ simply uses $f \overset{\mathcal{R}}{\leftarrow} P^l$ to respond to D's block encryption requests. In contrast, oracle $\mathcal{O}^{-1}$ flips a coin $b \in \{0, 1\}$ and responds to D's block decryption requests by using either $f^{-1} \overset{\mathcal{R}}{\leftarrow} P^l$ or $\overline{f} \overset{\mathcal{R}}{\leftarrow} G_S$. D runs A, responds to A's q_e encryption queries, and then verifies A's choices of ciphertext forgeries $y'^i = y_0'^i y_1'^i \quad y_n'^i, y_{n+1}'^i, 1 \quad i \quad q_v$. [As a consequence, D issues all its requests for block encryption to $\mathcal{O}$, if any, before all the requests for block decryption to $\mathcal{O}^{-1}$.] D returns the result of each y'^i's decryption to A; i.e., D returns either *Success* or *Failure* to A. D outputs 1 if A's forgery decrypts successfully, and 0, otherwise.

Distinguisher D's advantage, $Adv_D(P^l, G_S)$, is defined as:

$$Adv_D(P^l, G_S) = Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[D^f = 1] \Leftrightarrow Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[D^f = 1].$$

where $f \overset{\mathcal{R}}{\leftarrow} P^l$ denotes the selection of function f, and its inverse f^{-1}, from the set of all permutations P^l by the random key K, and $f \overset{\mathcal{R}}{\leftarrow} G_S$ denotes the random selection of f from P^l to encrypt and the associated function $\overline{f} \overset{\mathcal{R}}{\leftarrow} G_S$ to decrypt.

By the definition of the distinguisher algorithm:

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[D^f = 1] = Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\mathcal{D} \Leftrightarrow XCBC\$ \Leftrightarrow XOR(y) \neq Null] = Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[Succ]$$

and

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[D^f = 1] = Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\mathcal{D} \Leftrightarrow XCBC\$ \Leftrightarrow XOR(y) \neq Null] = Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}].$$

The above probabilities are over the random choice of r_0, $f \overset{\mathcal{R}}{\leftarrow} P^l$, $f \overset{\mathcal{R}}{\leftarrow} G_S$, and D's guesses. Hence,

$$\begin{aligned} Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}] &= Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}] \Leftrightarrow Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}] + Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}] \\ &= Adv_D(P^l, G_S) + Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}]. \end{aligned}$$

Now we find an upper bound for D's advantage in distinguishing between P^l and G_S, namely $Adv_D(P^l, G_S)$. By the definition of the two oracles $\mathcal{O}$ and $\mathcal{O}^{-1}$, only oracle $\mathcal{O}^{-1}$ can be used by D to distinguish between P^l and G_S. Furthermore, whenever a block decryption request to oracle $\mathcal{O}^{-1}$ is a ciphertext block that was generated during the encryption of A's q_e queries, the output of oracle $\mathcal{O}^{-1}$ is the same for both $f \overset{\mathcal{R}}{\leftarrow} P^l$ and $f \overset{\mathcal{R}}{\leftarrow} G_S$ (by the definition of $\overline{f}$), and a distinction between P^l and G_S cannot be made. D can make a distinction between P^l and G_S only when the ciphertext blocks of the decryption requests to oracle $\mathcal{O}^{-1}$ (i.e., the inputs to f^{-1} or $\overline{f}$) have never been generated during the encryption of A's q_e queries; i.e., the ciphertext blocks are not in $S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$. To make this distinction, D needs to send only the ciphertext blocks of A's forgeries that are not in $S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$ to oracle $\mathcal{O}^{-1}$, since D already has the plaintext blocks corresponding to all the ciphertext blocks in $S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$. In this case, $\overline{f} = v$, where $v \overset{\mathcal{R}}{\leftarrow} R^{l,l}$, and the advantage of distinguisher D cannot be higher than the advantage of any polynomial-time algorithm D' that distinguishes a random permutation from a random function using the same block decryption requests to oracle $\mathcal{O}^{-1}$ as those made by distinguisher $\mathcal{D}$; i.e., ciphertext blocks from from $\{0,1\}^l \Leftrightarrow S_{f \overset{\mathcal{R}}{\leftarrow} P^l}$. Hence, $Adv_D(P^l, G_S) \quad Adv_{D'}(P^l, R^{l,l})$. However, by the bound of the birthday attack, $Adv_{D'}(P^l, R^{l,l}) \quad \frac{q(q-1)}{2^{l+1}}$ where q is the number of the block decryption requests to oracle $\mathcal{O}^{-1}$; i.e., $q \quad \frac{\mu_v}{l}$ Hence,

$$Adv_D(P^l, G_S) \quad Adv_{D'}(P^l, R^{l,l}) \quad \frac{\mu_v(\mu_v \Leftrightarrow l)}{l^2 2^{l+1}}.$$

Hence,

$$Pr_{f \overset{\mathcal{R}}{\leftarrow} P^l}[\text{Succ}] \quad Pr_{f \overset{\mathcal{R}}{\leftarrow} G_S}[\text{Succ}] + \frac{\mu_v(\mu_v \Leftrightarrow l)}{l^2 2^{l+1}}.$$

□

Proof of Fact 2

If $i = d \quad 2^m$, then $i \quad r_0 = d \quad 2^m \quad r_0$ has (at least) the first (i.e., least significant) m bits zero. Also, since $i \quad 2^l$, it follows that $d \quad 2^{l-m}$. Let $r_{0m} = r_0[1 \quad l \Leftrightarrow m]$ be the least significant $l \Leftrightarrow m$ bits of r_0. (These bits will be shifted in the most significant $l \Leftrightarrow m$ bit positions of a block by multiplication with 2^m.)

First, we note that

$$i \quad r_0 = (dr_{0m})||\underbrace{0 \quad 0}_{m}$$

where $dr_{0m} = \underbrace{r_{0m} + \quad + r_{0m}}_{d\ times}$ mod 2^{l-m} and $||$ is the concatenation operator. To see this:

$$\begin{aligned} i \quad r_0 &= (d \quad 2^m) \quad r_0 = d \quad (r_0 \quad 2^m) = \underbrace{(r_0 \quad 2^m) + \quad + (r_0 \quad 2^m)}_{d\ times} \\ &= \underbrace{(r_{0m}||\underbrace{0 \quad 0}_{m}) + \quad + (r_{0m}||\underbrace{0 \quad 0}_{m})}_{d\ times} = (\underbrace{r_{0m} + \quad + r_{0m}}_{d\ times})||\underbrace{0 \quad 0}_{m} \\ &= (dr_{0m})||\underbrace{0 \quad 0}_{m} \end{aligned}$$

where $dr_{0m} = \underbrace{r_{0m} + \quad + r_{0m}}_{d\ times}$ mod 2^{l-m}.

Second, we divide all values of an arbitrary constant a into two complementary classes based on whether the first (i.e., least significant) m bits of a are all zero, compute $Pr[i \quad r_0 = a]$ for each class separately, and then take the maximum of the two probabilities as the overall bound.

Let $a[1 \quad m] = 0$ denote the values of a for which the first m bits are zero, and $a[1 \quad m] \neq 0$ those for which at least one of the the first m bits is not zero. Since $i \quad r_0 = (dr_{0m})||\underbrace{0 \quad 0}_{m}$, it follows that, if $a[1 \quad m] \neq 0$, $Pr[\ i \quad r_0 = a\ ] = 0$. However, if $a[1 \quad m] = 0$, then $[\ i \quad r_0 = a\ ] \Leftrightarrow [\ dr_{0m} = b\ ]$, where

$b = a[m+1 \quad l]$ represents bits $m+1, \quad l$ of a, i.e., the $l \Leftrightarrow m$ most significant bits of a. Hence, in this case,

$$Pr[\ i \quad r_0 = a\ ] = Pr[\ dr_{0m} = b\ ],$$

where $d, r_{0m}, b \in \{0,1\}^{l-m}$. However, d and 2^{l-m} are relatively prime because d is odd. Hence, d has a left inverse,[10] e, and $dr_{0m} = b \Leftrightarrow edr_{0m} = eb \Leftrightarrow r_{0m} = eb \pmod{2^{l-m}}$, which happens with probability $1/2^{l-m}$ because $r_{0m} = r[1 \quad l \Leftrightarrow m]$ is random and uniformly distributed in $\{0,1\}^{l-m}$. Thus, if $a[1 \quad m] = 0$,

$$Pr[i \quad r_0 = a] = \frac{1}{2^{l-m}}.$$

Hence, for any value of constant a, $Pr[i \quad r_0 = a] \quad \frac{1}{2^{l-m}}$. □

Proof of Fact 3

Since any a can be expressed as $a = d \quad 2^m$, where d is odd, there are multiple values of a that have the same exponent m. (For example, for all odd values of a, $m = 0$, and for all even values of a that are not a multiple of 4, $m = 1$.) Hence, when computing the sum $\sum_{a=1}^{N-1} 2^m$, we can group together the terms 2^m that have the same exponent m (i.e., we group the terms 2^m that are equal).

For a given exponent m, we find the number of distinct values of a that have the same exponent m when represented as $d \quad 2^m$. To find this number, we note that $1 \quad a \quad N \Leftrightarrow 1$ and, hence, $1 \quad d \quad \lfloor \frac{N-1}{2^m} \rfloor$. Hence, the number of distinct values of a that yield the same exponent m is $\lfloor \frac{1}{2} \quad \lfloor \frac{N-1}{2^m} \rfloor + 1 \quad \rfloor$, since this number is bounded by the number of distinct values of d odd.

¿From the definition of exponent m, $2^m \quad N \Leftrightarrow 1$ (i.e., $0 \quad m \quad \log_2(N \Leftrightarrow 1)$). Hence,

$$\begin{aligned}\sum_{a=1}^{N-1} 2^m &= \sum_{m=0}^{\lfloor \log_2(N-1) \rfloor} \lfloor \frac{1}{2} \quad \lfloor \frac{N \Leftrightarrow 1}{2^m} \rfloor + 1 \quad \rfloor 2^m \quad \sum_{m=0}^{\lfloor \log_2(N-1) \rfloor} \frac{N \Leftrightarrow 1}{2} + \frac{2^m}{2} \\ &= \frac{N \Leftrightarrow 1}{2} (\lfloor \log_2(N \Leftrightarrow 1) \rfloor + 1) + \frac{2^{\lfloor \log_2(N-1) \rfloor + 1} \Leftrightarrow 1}{2}\end{aligned}$$

because, for any $M > 0$, $\sum_{m=0}^{M} 2^m = 2^{M+1} \Leftrightarrow 1$. Hence,

$$\sum_{a=1}^{N-1} 2^m \quad \frac{N \Leftrightarrow 1}{2} (\log_2(N \Leftrightarrow 1) + 1) + (N \Leftrightarrow 1) = \frac{N \Leftrightarrow 1}{2} (\log_2(N \Leftrightarrow 1) + 3).$$

□

Proof of Fact 4

Since, by hypothesis, $\sum_{p=1}^{q_e} (n_p + 1) \quad \frac{\mu_e}{l}$, the term under the $\log_2$ is $n_p + 1 \quad \frac{\mu_e}{l}$. Hence, we obtain:

$$\sum_{p=1}^{q_e} (n_p + 1) \log_2(n_p + 1) \quad \log_2 \frac{\mu_e}{l} \sum_{p=1}^{q_e} (n_p + 1),$$

[10] A way to see that d has a left inverse, e, is to label $2^{l-m} = f$, and to note that, if d and f are relatively prime, then, by Euclid's gcd algorithm, there exists e and h such that $ed + hf = 1$; i.e., $ed = 1 - hf$ or $ed = 1 (\mathrm{mod} f)$.

and thus,

$$\sum_{p=1}^{q_e}(n_p+1)\log_2(n_p+1) \quad \frac{\mu_e}{l}\log_2\frac{\mu_e}{l}.$$

Further, if $m = \max(n_p+1)$, then $\log_2(n_p+1) \quad \log_2 m$. Hence,

$$\sum_{p=1}^{q_e}(n_p+1)\log_2(n_p+1) \quad \frac{\mu_e}{l}\log_2 m.$$

□

Proof of Claim 3.1

There are three possible complementary cases to consider:
(1) $y_0 = y_0^i$, for some queried message $i, 1 \quad i \quad q_e$. Then $r_0 = \overline{f} = f^{-1}(y_0^i) = r_0^i$ is random and uniformly distributed, by definition. Furthermore, if $r_0 = r_0^i \neq r_0^p$ (i.e., $y_0 = y_0^i \neq y_0^p$), then $i \neq p$ and r_0 is also independent of r_0^p, by definition.
(2) $y_0 = z_j^i$, for some queried message $i, 1 \quad i \quad q_e, 1 \quad j \quad n_i + 1$; i.e., y_0 collides with some hidden ciphertext block, z_j^i, generated during the encryption of message i. But this is exactly the event prohibited by $\overline{I}$.
(3) $y_0 \neq y_0^i$ and $y_0 \neq z_k^i$, for all queried messages $i, 1 \quad i \quad q_e, k \geq 1$. Then $r_0 = \overline{f}(y_0) = v(y_0) \neq r_0^i, \forall i, 1 \quad i \quad q_e$ is random, uniformly distributed and independent of anything else because $v \overset{R}{\leftarrow} R^{l,l}$ and $\overline{f}$ has never been invoked with argument y_0. Hence, r_0 is random, uniformly distributed and independent of r_0^p. □

Proof of Claim 3.2

The event $z_j = z_k^p$ is equivalent to $y_j \Leftrightarrow j \quad r_0 = y_k^p \Leftrightarrow k \quad r_0^p \Leftrightarrow j \quad r_0 = k \quad r_0^p \Leftrightarrow y_k^p + y_j \Leftrightarrow k \quad r_0^p = j \quad r_0 \Leftrightarrow y_j + y_k^p$.
(a) If $y_0 \neq y_0^p$, and since event $\overline{I}$ is true, it follows that r_0 is random, uniformly distributed, and independent of r_0^p, by Claim 3.1 above. Also, event $\overline{I}$ and $\overline{E}$ implies that r_0 is random, uniformly distributed, and independent of r_0^p by the definition of event E. Thus, $j \quad r_0$ is independent of $k \quad r_0^p \Leftrightarrow y_k^p + y_j$ and $k \quad r_0^p$ is independent of $j \quad r_0 \Leftrightarrow y_j + y_k^p$, since $j, k > 0$, and y_j, y_k^p, j, k are known constants. Furthermore, event $[z_j = z_k^p] \equiv [j \quad r_0 = k \quad r_0^p \Leftrightarrow y_k^p + y_j] \equiv [k \quad r_0^p = j \quad r_0 \Leftrightarrow y_j + y_k^p]$. Hence,

$$\begin{aligned} Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] &= Pr_{\overline{I} \text{ and } \overline{E}}[j \quad r_0 = k \quad r_0^p \Leftrightarrow y_k^p + y_j] \\ &= Pr_{\overline{I} \text{ and } \overline{E}}[k \quad r_0^p = j \quad r_0 \Leftrightarrow y_j + y_k^p]. \end{aligned}$$

However, $Pr_{\overline{I} \text{ and } \overline{E}}[j \quad r_0 = k \quad r_0^p \Leftrightarrow y_k^p + y_j] \quad \frac{1}{2^{l-m_1}}$, where $j = d_1 \quad 2^{m_1}$ and d_1 is odd, by Fact 2. Also, $Pr_{\overline{I} \text{ and } \overline{E}}[k \quad r_0^p = j \quad r_0 \Leftrightarrow y_j + y_k^p] \quad \frac{1}{2^{l-m_2}}$, where $k = d_2 \quad 2^{m_2}$ and d_2 is odd. Hence,

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] \quad \min \quad \frac{1}{2^{l-m_1}}, \frac{1}{2^{l-m_2}} \quad = \frac{1}{2^{l-m}},$$

where $m = \min(m_1, m_2)$.

(b) If $y_0 = y_0^p$, then $r_0 = r_0^p$. Hence,

$$z_j = z_k^p \Leftrightarrow y_j \Leftrightarrow j \quad r_0 = y_k^p \Leftrightarrow k \quad r_0^p \Leftrightarrow (k \Leftrightarrow j) \quad r_0 = y_k^p \Leftrightarrow y_j.$$

Thus,

$$Pr_{\overline{I} \text{ and } \overline{E}}[z_j = z_k^p] = Pr_{\overline{I} \text{ and } \overline{E}}[(k \Leftrightarrow j) \quad r_0 = y_k^p \Leftrightarrow y_j].$$

However, since event $\overline{I}$ is true, it follows that r_0 is random and uniformly distributed, by Claim 3.1 above. Also, event $\overline{I}$ and $\overline{E}$ implies that r_0 is random and uniformly distributed, by the definition of event E. Since $j, k > 0, j \neq k$, and y_j, y_k^p, j, k are known constants, and $k \neq j$, Fact 2 implies that

$$Pr_{\overline{I} \text{ and } \overline{E}}[(k \Leftrightarrow j) \quad r_0 = y_k^p \Leftrightarrow y_j] \quad \frac{1}{2^{l-m}}$$

where m is defined by $k \Leftrightarrow j = d \quad 2^m, k > j$ or $j \Leftrightarrow k = d \quad 2^m, j > k$, and d is odd. □

Proof of Claim 4.1

(a) One can write the event $z_t = z_s \;\Leftrightarrow\; (t \Leftrightarrow s) \quad r_0 = y_t \Leftrightarrow y_s$. Hence,

$$Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[z_s = z_t] = Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[(t \Leftrightarrow s) \quad r_0 = y_t \Leftrightarrow y_s].$$

Since event $\overline{I}$ is true, r_0 is random and uniformly distributed, by Claim 3.1. Furthermore, by the definition of events $\overline{E}$ and $\overline{C}$, event $\overline{C}$ and $\overline{I}$ and $\overline{E}$ implies that r_0 is random and uniformly distributed. Using the definition of m and the facts that (1) r_0 is random and uniformly distributed, (2) y_t, y_s are constants, and (3) $1 \quad t \Leftrightarrow s \quad 2^l \Leftrightarrow 1$, we obtain (by Fact 2) that

$$Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[(t \Leftrightarrow s) \quad r_0 = y_t \Leftrightarrow y_s] \quad \frac{1}{2^{l-m}}$$

where m is defined by $t \Leftrightarrow s = d \quad 2^m$ and d is odd. Hence,

$$Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[z_s = z_t] \quad \frac{1}{2^{l-m}}.$$

(b) The proof of this part is similar to that of part (a) and is included here for completeness.

Note that, since $z_s = y_s \Leftrightarrow s \quad r_0$, event $z_s = y_0 \;\Leftrightarrow\; s \quad r_0 = y_s \Leftrightarrow y_0$, where y_s and y_0 are constants. However, since event $\overline{I}$ is true, r_0 is random and uniformly distributed, by Claim 3.1. Furthermore, event $\overline{C}$ and $\overline{I}$ and $\overline{E}$ implies that r_0 is random and uniformly distributed. Hence, by Fact 2,

$$Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[z_s = y_0] = Pr_{\overline{C} \text{ and } \overline{E} \text{ and } \overline{I}}[s \quad r_0 = y_s \Leftrightarrow y_0] \quad \frac{1}{2^{l-m}}$$

where m is defined by $s = d \quad 2^m$ and d is odd. □

Appendix B – Proof [Security of XEBC-MAC in an Adaptive Chosen-Message Attack]

Throughout this proof, we use the same notation as in the Proof for Security of the XCBC\$-XOR in a Message-Integrity Attack, Appendix A, and the same facts (i.e., Facts 1 4). Unless mentioned otherwise, we focus on the probability for adversary's success when $f \overset{\mathcal{R}}{\leftarrow} R^{l,l}$, and, for simplicity, we will drop the $f \overset{\mathcal{R}}{\leftarrow} R^{l,l}$ subscript from the probability equations.

To find an upper bound on the probability of an adversary's success we use the same proof technique as for the XCBC\$-XOR scheme. That is, we (1) define several types of events on which we condition the adversary's success, (2) express the upper bound in terms of the conditional probabilities obtained, and (3) compute upper bounds on these probabilities. As before, our choice and number of conditioning events is motivated exclusively by the need to obtain a (good) upper bound for the probability of the adversary's success. Undoubtedly, other events could be used for deriving alternate upper bounds.

We provide some intuition for the choice of conditioning events defined, by giving the following examples of events that cause an adversary's success. (The reader can skip these examples without loss of continuity.)

Examples of Adversary's Success. A way for the adversary to find a forgery x' that passes the integrity check $w' = w$, is to look for collisions in the input of f, at forgery verification. The following three examples illustrate why such collisions cause an adversary's success. Other such examples, and other ways to find forgeries, exist.

Example 1 Collisions between inputs of f at forgery verification with those at message signing

Suppose that all inputs to f at forgery verification collide with inputs to f at signing. We pessimistically declare the adversary to be successful. For example, suppose that two of the block inputs to f at the verification of forgery (x', ctr', w') represent two swapped inputs to f at the signing of message x using counter ctr and obtaining the authentication tag w. Also suppose that all other inputs to f at forgery verification are the same as those of message x at signing. Hence, $x' \neq x$. In this case, the authentication check for forgery $(x', ctr' = ctr, w' = w)$ will pass the integrity check.

It should be noted that this criterion for adversary's success is pessimistic because, among the forgeries that make this event true some will decrypt correctly with negligible probability. For instance, if a forgery x' is a truncation of a signed message, the collision of the last forgery block $x'_{n'+1} = z'_0 + (n'+1) \quad r'_0$ with any of the inputs to f or f' at message signing is a negligible-probability event and hence truncation would have a negligible chance of success (viz., Claim 1 below provides some intuition for this statement).

Example 2 Collisions among inputs of f at forgery verification

Suppose that two inputs of f obtained during forgery verification, x'_{n+1} and x'_{n+2}, do not collide with any of the inputs to f obtained during message signing, but collide with each other; $x'_{n+1} = x'_{n+2}$. Also suppose that the adversary's forgery (x', ctr', w') is obtained as follows: $x' = x||x'_{n+1}||x'_{n+2}$, $ctr' = ctr$, and $w' = w$. Clearly, $x' \neq x$ and the forgery (x', ctr', w') passes verification under the pessimistic assumption that $f(z_0 + (n+3) \quad r_0) = f(z_0 + (n+1) \quad r_0)$.

Example 3 Collisions among the inputs of f that cause discovery of r_0

Suppose that the forgery counter ctr'^i collides with an input to f, $x^p_k + k \quad r^p_0, 1 \quad p \quad q_s, 1 \quad k \quad n_p$, obtained during message signing, or with $x'^i_j + j \quad r'^i_0, 1 \quad i \quad q_v, 1 \quad j \quad n'_i$, during the verification of forgery (x', ctr', w'). Suppose that the adversary finds that $x_k^p + k \quad r_0^p = ctr'^i$, for some message p, known plaintext block x_k^p and known counter ctr'^i, $1 \quad i \quad q_v$. Hence, the adversary can determine r_0^p and thus the adversary's forgeries can satisfy collisions of Examples 1 and 2 above. A similar collision event between ctr'^i and an input to f during forgery verification has a similar effect.

Conditioning Events. To compute an upper bound on the probability of successful forgery, we choose three conditioning events based on collisions in the inputs of f. Intuition for the choice of events is provided by Examples 1 3 above. To define the conditioning events, we use the following notation for the last block that is enciphered

$$\begin{aligned} x_{n_p+1}^p &= z_0^p \\ x_{n'_i+1}^{\prime i} &= z_0^{\prime i}. \end{aligned}$$

Next, we introduce the sets:

$$\begin{aligned} I^s &: \{ctr^1, \quad , ctr^{q_s}\} \\ S &: \{x_k^p + k \quad r_0^p, 1 \quad p \quad q_s, 1 \quad k \quad n_p + 1\}, \\ V_i &: \{x_s^{\prime i} + s \quad r_0^{\prime s}, x_s^{\prime i} + s \quad r_0^{\prime s} \notin (I^s \quad S), 1 \quad s \quad n'_i + 1\}, \end{aligned}$$

where I^s is the set of all the counters used at signing, S is the set of all the inputs to function f (aside from the counters) at signing, and V_i is the set of all the inputs to function f (aside from the counters) at verification of query i. Based on sets I^s, S, V_i, we introduce the following collision events that arise at the verification of forgery (x'^i, ctr'^i, w'^i):

$$C^i \ : \ V_i = \emptyset$$

Event C^i includes all instances when inputs of f at forgery verification (aside from counters) collide with either counters or inputs to function f at message signing. Next we define event D^i as follows:

$$\begin{aligned} D^i \ : \ & \exists s, 1 \quad s \quad n'_i + 1 : x_s^{\prime i} + s \quad r_0^{\prime i} \in V_i \\ & \text{and } x_s^{\prime i} + s \quad r_0^{\prime i} \neq x_t^{\prime i} + t \quad r_0^{\prime i}, \forall x_t^{\prime i} + t \quad r_0^{\prime i} \in V_i, t \neq s, 1 \quad t \quad n'_i + 1 \\ & \text{and } x_s^{\prime i} + s \quad r_0^{\prime i} \neq ctr'^i \end{aligned}$$

Event D^i states that there is at least one input block of forgery i that does not collide with any other block and counter of forgery i. It is clear that the definition for D^i makes sense only when event C^i is false.

The rationale for introducing events C^i (or, actually, $\overline{C^i}$) and D^i is similar to the one used in the proof of Theorem 2. That is, we want to find a desirable event which states that there exists a forgery block that does not collide with any other input to f at either message signing or verification of forgery i (as suggested by Examples 1 and 2). Clearly, if this event is true, then the probability of verification passing is $1/2^l$. To find this event, however, we must ensure that all other collisions that that may lead to the discovery of r_0 are also ruled out for this block (as suggested by Example 3). For this reason, we must introduce two events beside $\overline{C^i}$ and D^i, namely events R_i^v and R^s defined below. (Note that these events need not cover the last block or a signed message or of forgery i since such a collision cannot be used to solve for either r'_0 or r_0 since random variables z'_0 and z_0 remain unknown to the adversary.) After we find the desired event for forgery i, we show that the complement of this event has a negligible probability (viz., the section on *Non-truncation Forgeries* below).

$$R_i^v \ : \ ctr'^i \neq x_j^{\prime i} + j \quad r_0^{\prime i}, \forall j, 1 \quad j \quad n'_i$$

Event R_i^v states that all inputs to f during the verification of forgery i (aside from counters and last block) do not collide with forgery counters.

$$R^s \quad : \quad P^s \text{ and } P^v \text{ and } Q^s,$$

where

$$\begin{array}{lcl} P^s & : & ctr^a \neq x_k^p + k \quad r_0^p, \forall a, p, k, 1 \quad a, p \quad q_s, 1 \quad k \quad n_p \\ P^v & : & ctr'^a \neq x_k^p + k \quad r_0^p, \forall a, p, k, 1 \quad a \quad q_v, 1 \quad p \quad q_s, 1 \quad k \quad n_p \\ Q^s & : & x_j^p + j \quad r_0^p \neq x_k^p + k \quad r_0^p, \forall p, j, k, 1 \quad p \quad q_s, 1 \quad j, k \quad n_p, j \neq k \end{array}$$

and j is the index of a block in forgery i; i.e., $x_j'^i$. Event R^s states that all inputs to f at message signing (aside from counters and last block) do not collide with any other such inputs and with any of the counters used at message signing and forgery verification. Note that event R^s is independent of any forgery i.

Upper bound on the Probability of Successful Forgery. By standard conditioning,

$$Pr[\text{Succ}] \quad Pr[\text{Succ} \mid R^s] + Pr[\overline{R^s}] \quad Pr[\text{Succ} \mid R^s] + Pr[\overline{P^s}] + Pr[\overline{P^v}] + Pr[\overline{Q^s}],$$

since $\overline{R^s} = \overline{P^s}$ or $\overline{P^v}$ or $\overline{Q^s}$. The second, third and fourth terms in the sum are bounded as in the following Claim:

Claim 1
(a)

$$Pr[\overline{P^s}] \quad \frac{\mu_s}{l 2^l}.$$

(b)

$$Pr[\overline{P^v}] \quad \frac{\mu_s}{l 2^l}.$$

(c)

$$Pr[\overline{Q^s}] \quad \frac{\mu_s}{l 2^l}.$$

To compute an upper bound for the probability of successful forgery, when event R^s is true, we note that the adversary is successful if one of his q_v forgeries is successful. Let the i-th adversary's forgery be: (ctr'^i, x'^i, w'^i), where $x'^i = x_1'^i \quad x_{n_i'}'^i$. Hence, by union bound, the probability of adversary's success for all q_v verification queries (when $f \overset{\mathcal{R}}{\leftarrow} R^{l,l}$) is:

$$Pr[\text{Succ} \mid R^s] \quad \sum_{i=1}^{q_v} Pr[w'^i = y_1'^i \oplus \quad \oplus y_{n_i'+1}'^i \mid R^s].$$

Hence, we first compute the probability of adversary's success when a single forgery verification is allowed; i.e., we compute $Pr[w'^i = y_1'^i \oplus \quad \oplus y_{n_i'+1}'^i \mid R^s]$. For this computation, we partition the space of all possible forgeries into (1) truncation and (2) non-truncation forgeries.

Truncation Forgeries. For truncation forgeries, we introduce the events:

$$\begin{array}{lcl} Z_{I^s} & : & z_0'^i + (n_i' + 1) \quad r_0'^i \in I^s \\ Z_S & : & z_0'^i + (n_i' + 1) \quad r_0'^i \in S. \end{array}$$

Using these events, we show that the probability of adversary's success in creating a successful forgery i is negligible. If forgery i is a truncation, then there exists $p, 1 \quad p \quad q_s : ctr'^i = ctr^p$ and $x'^i_k = x^p_k, \forall k, 1 \quad k \quad n'_i \quad n_p$, hence $z'^i_0 = z^p_0$. If the input to f at block $n'_i + 1$, namely $z'^i_0 + (n'_i + 1) \quad r'^i_0$, does not collide with any counter (i.e., event $\overline{Z_{I^s}}$ is true) and any input to function f (aside from the counters) at signing (i.e., event $\overline{Z_S}$ is true), then $y'^i_{n'_i+1} = f(z'^i_0 + (n'_i + 1) \quad r'^i_0)$ is random, uniformly distributed and independent of any other block y' in the formula for w'^i. Hence, in this case, the probability of the event that $y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} = w'^i$ during the verification of forgery i is $1/2^l$. Summarizing, by standard conditioning and union bound,

$$\begin{aligned} Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s] \quad & Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid (\overline{Z_{I^s} \text{ or } Z_S}) \text{ and } R^s] + Pr[Z_{I^s} \text{ or } Z_S \mid R^s] \\ & \frac{1}{2^l} + Pr[Z_{I^s} \text{ or } Z_S] \quad \frac{1}{2^l} + Pr[Z_{I^s} \mid R^s] + Pr[Z_S \mid R^s]. \end{aligned}$$

Upper bounds for the probabilities of events $Z_{I^s} \mid R^s$ and $Z_S \mid R^s$ are given by the following Claim:

Claim 2
(a)

$$Pr[Z_{I^s} \mid R^s] \quad \frac{q_s}{2^l}.$$

(b)

$$Pr[Z_S \mid R^s] \quad \frac{\mu_s}{l2^l} + \frac{n_p}{2^l}.$$

Hence, for any truncation forgery,

$$Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s] \quad \frac{1}{2^l} + \frac{q_s}{2^l} + \frac{\mu_s}{l2^l} + \frac{n_p}{2^l} \quad \frac{\mu_s}{l2^l} + \frac{q_s + (n_p + 1)}{2^l}.$$

Non-truncation Forgeries. Now, we find an upper bound for $Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s]$ for non-truncation forgeries. To compute this upper bound, we define an event such that (1) the probability of successful forgery is $1/2^l$ when this event occurs, and (2) the probability of the complement of this event has a negligible upper bound.

Using the events defined above and by standard conditioning, we obtain:

$$\begin{aligned} Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s] \quad & Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid \overline{C^i} \text{ and } D^i \text{ and } R^v_i \text{ and } R^s] + \\ & \quad Pr[C^i \text{ or } \overline{D^i} \text{ or } \overline{R^v_i} \mid R^s] \\ & Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid \overline{C^i} \text{ and } D^i \text{ and } R^v_i \text{ and } R^s] + \\ & \quad Pr[C^i \text{ or } \overline{D^i} \text{ or } \overline{R^v_i} \mid R^v_i \text{ and } R^s] + Pr[\overline{R^v_i} \mid R^s] \\ = \quad & Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid \overline{C^i} \text{ and } D^i \text{ and } R^v_i \text{ and } R^s] \\ & \quad + Pr[C^i \text{ or } \overline{D^i} \mid R^v_i \text{ and } R^s] + Pr[\overline{R^v_i} \mid R^s] \\ & Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid \overline{C^i} \text{ and } D^i \text{ and } R^v_i \text{ and } R^s] + \\ & Pr[C^i \text{ or } \overline{D^i} \mid \overline{C^i} \text{ and } R^v_i \text{ and } R^s] + Pr[C^i \mid R^v_i \text{ and } R^s] + Pr[\overline{R^v_i} \mid R^s] \\ = \quad & Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid \overline{C^i} \text{ and } D^i \text{ and } R^v_i \text{ and } R^s] + \\ & Pr[\overline{D^i} \mid \overline{C^i} \text{ and } R^v_i \text{ and } R^s] + Pr[C^i \mid R^v_i \text{ and } R^s] + Pr[\overline{R^v_i} \mid R^s], \end{aligned}$$

since the following events are equivalent:

$$\begin{aligned}(C^i \text{ or } \overline{D^i} \text{ or } \overline{R_i^v} \mid R_i^v \text{ and } R^s) &\equiv (C^i \text{ or } \overline{D^i} \mid R_i^v \text{ and } R^s)\\ (C^i \text{ or } \overline{D^i} \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s) &\equiv (\overline{D^i} \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s).\end{aligned}$$

Event $(\overline{C^i}$ and D^i and R_i^v and $R^s)$ is the desired event mentioned earlier in this proof. If this event happens, then there must exist an index $j, 1 \quad j \quad n'_i + 1$ such that $x'^i_j + j \quad r'^i_0$ does not collide with any other input to f, at either message signing or verification of forgery i, and hence $y'^i_j = f(x'^i_j + j \quad r'^i_0)$ is random, uniformly distributed and independent of any other terms in the expression $y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1}$. Hence, $y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1}$ is random and uniformly distributed and hence,

$$Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid \overline{C^i} \text{ and } D^i \text{ and } R_i^v \text{ and } R^s] \quad \frac{1}{2^l}.$$

The other probabilities that appear in the expression for the total probability $Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s]$ are bounded as in Claim 3, whose proof can be found below:

Claim 3

(a)

$$Pr[\overline{R_i^v} \mid R^s] \quad \frac{n'_i}{2^l}.$$

(b)

$$Pr[C^i \mid R_i^v \text{ and } R^s] \quad \frac{q_s n'_i}{2^l} + \frac{\mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad .$$

(c)

$$Pr[\overline{D^i} \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s] \quad \frac{n'_i}{2^l}.$$

Based on this claim, for an arbitrary forgery i that is not a truncation, we obtain:

$$\begin{aligned}Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s] \quad & \frac{1}{2^l} + \frac{n'_i}{2^l} + \\ & \frac{q_s n'_i}{2^l} + \frac{\mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad + \frac{n'_i}{2^l} \\ & \frac{2(n'_i + 1)}{2^l} + \frac{q_s n'_i}{2^l} + \frac{\mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad .\end{aligned}$$

For any forgery, the upper bound is the maximum from the upper bounds for truncation and non-truncation forgeries, hence,

$$Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s] \quad \frac{2(n'_i + 1)}{2^l} + \frac{q_s n'_i}{2^l} + \frac{\mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad .$$

Hence, for all q_v verification queries, we obtain by union bound,

$$\begin{aligned}Pr[\text{Succ} \mid R^s] \quad & \sum_{i=1}^{q_v} Pr[w'^i = y'^i_1 \oplus \quad \oplus y'^i_{n'_i+1} \mid R^s] \\ & \sum_{i=1}^{q_v} \quad \frac{2(n'_i + 1)}{2^l} + \frac{q_s n'_i}{2^l} + \frac{\mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \\ = \quad & \frac{2\mu_v}{l 2^l} + \frac{q_s \mu_v}{l 2^l} + \frac{q_v \mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \\ = \quad & \frac{(q_s + 2)\mu_v}{l 2^l} + \frac{q_v \mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad .\end{aligned}$$

Hence, by Claim 1,

$$Pr[\text{Succ}] \quad \frac{(q_s+2)\mu_v}{l2^l} + \frac{q_v\mu_s}{l2^{l+1}} \quad \log_2\frac{\mu_s}{l} + 3 \quad + \frac{3\mu_s}{l2^l}.$$

Finally, when $f \overset{\mathcal{R}}{\leftarrow} F$, the probability for adversary's success is bounded as follows:

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

□

Proofs of Claims 1 – 3

Proof of Claim 1

(a) Event $\overline{P^s}$ deals with collisions between inputs to f at signing, namely $x_k^p + k \quad r_0^p, 1 \quad p \quad q_s, 1 \quad k \quad n_p$ and constant counters at signing, namely $ctr^a, 1 \quad a \quad q_s$. Since $\overline{P^s} \Rightarrow \exists a,p,k, 1 \quad a,p \quad q_s, 1 \quad k \quad n_p :$ $ctr^a = x_k^p + k \quad r_0^p$, it follows that

$$Pr[\overline{P^s}] \quad Pr[ctr^a = x_k^p + k \quad r_0^p]$$

In this event, ctr^a and x_k^p are constants. Since r_0^p is random and uniformly distributed, and the event of interest can be written as $k \quad r_0^p = ctr^a \Leftrightarrow x_k^p$, then, by Fact 2 (Appendix A),

$$Pr[ctr^a = x_k^p + k \quad r_0^p] \quad \frac{2^m}{2^l},$$

where $k = d \quad 2^m$ and d is odd. Here $2^m \quad k \quad n_p \quad \frac{\mu_s}{l}$, hence

$$Pr[\overline{P^s}] \quad Pr[ctr^a = x_k^p + k \quad r_0^p] \quad \frac{\mu_s}{l2^l}.$$

(b) Event $\overline{P^v}$ is very similar with event $\overline{P^s}$, i.e., it deals with collisions between inputs to f at signing, namely $x_k^p + k \quad r_0^p, 1 \quad p \quad q_s, 1 \quad k \quad n_p$ and constant counters at verification, namely $ctr'^a, 1 \quad a \quad q_v$. In a manner similar to the one used in the proof of (a), since ctr'^a are also constants,

$$Pr[\overline{P^v}] \quad \frac{\mu_s}{l2^l}.$$

(c) Event $\overline{Q^s}$, deals with collisions between inputs to f at signing within the same message, namely $x_j^p + j \quad r_0^p \neq x_k^p + k \quad r_0^p$ where $1 \quad p \quad q_s, 1 \quad j,k \quad n_p, j \neq k$. Since $\overline{Q^s} \Rightarrow \exists p,j,k, 1 \quad p \quad q_s, 1 \quad j,k$ $n_p, j \neq k : x_j^p + j \quad r_0^p \neq x_k^p + k \quad r_0^p$ then,

$$Pr[\overline{Q^s}] \quad Pr[x_j^p + j \quad r_0^p = x_k^p + k \quad r_0^p].$$

Without loss of generality, let $k > j$. Event $x_j^p + j \quad r_0^p = x_k^p + k \quad r_0^p$ is equivalent to $(k \Leftrightarrow j) \quad r_0^p = x_j^p \Leftrightarrow x_k^p$. Since r_0^p is random and uniformly distributed, by Fact 2 (Appendix A), this event happens with probability $\frac{2^m}{2^l}$ where $k \Leftrightarrow j = d \quad 2^m$ and d is odd. Here $2^m \quad k \Leftrightarrow j \quad n_p \quad \frac{\mu_s}{l}$, hence,

$$Pr[\overline{Q^s}] \quad \frac{\mu_s}{l2^l}.$$

Proof of Claim 2

(a) Event Z_{I^s} refers to collisions between the last input to f at verification of forgery i, namely $z_0'^i + (n_i' + 1) \quad r_0'^i$, and any counter at signing, namely $ctr^a, 1 \quad a \quad q_s$. By union bound,

$$Pr[Z_{I^s} \mid R^s] \quad \sum_{a=1}^{q_s} Pr[z_0'^i + (n_i' + 1) \quad r_0'^i = ctr^a \mid R^s].$$

$z_0'^i = z_0^p = f'(r_0^p)$ is random, uniformly distributed and independent of r_0^i and of the counter since it is obtained by enciphering with a different key. Hence, since ctr^a is a constant,

$$Pr[z_0'^i + (n_i' + 1) \quad r_0'^i = ctr^a \mid R^s] = \frac{1}{2^l}$$

and

$$Pr[Z_{I^s} \mid R^s] \quad \frac{q_s}{2^l}.$$

□

(b) Event Z_{I^s} refers to collisions between the last input to f at verification of forgery i, namely $z_0'^i + (n_i' + 1) \quad r_0'^i$, and any input to f at signing (other than counters), i.e., $x_b^a + b \quad r_0^a, 1 \quad a \quad q_s, 1 \quad b \quad n_a + 1$. By union bound,

$$Pr[Z_S \mid R^s] \quad \sum_{a=1}^{q_s} \sum_{b=1}^{n_a+1} Pr[z_0'^i + (n_i' + 1) \quad r_0'^i = x_b^a + b \quad r_0^a \mid R^s].$$

If $b \quad n_a$, then x_b^a is a constant in the equation $z_0'^i + (n_i' + 1) \quad r_0'^i = x_b^a + b \quad r_0^a$. Then, since $z_0'^i = z_0^p$ is obtained using a different key, $z_0'^i$ is random, uniformly distributed and independent of $r_0'^i = r_0^p, r_0^a$ and of the constant x_b^a. Hence,

$$Pr[z_0'^i + (n_i' + 1) \quad r_0'^i = x_b^a + b \quad r_0^a \mid R^s] = \frac{1}{2^l}.$$

If $b = n_a + 1$, then $x_b^a = z_0^a$. In this case, if $p \neq a$, then $z_0'^i = z_0^p$ and z_0^a are random, uniformly distributed and independent; they are also independent of $r_0'^i = r_0^p$ and r_0^a. Hence,

$$Pr[z_0'^i + (n_i' + 1) \quad r_0'^i = x_b^a + b \quad r_0^a \mid R^s] = \frac{1}{2^l}.$$

In the complementary case, namely when $b = n_a + 1, p = a$, then $z_0'^i = z_0^p = z_0^a = x_b^a$ and $r_0'^i = r_0^p = r_0^a$. Since, in this case, $b = n_a + 1 = n_p + 1$, it follows that

$$z_0'^i + (n_i' + 1) \quad r_0'^i = x_b^a + b \quad r_0^a \Leftrightarrow (n_p \Leftrightarrow n_i') \quad r_0^p = 0,$$

where, $n_p > n_i'$ (since the forgery is a truncation of message p). Event R^s is true, hence r_0^p is unknown, random and uniformly distributed. Hence, by Fact 2 (Appendix A), the probability of this event is $\frac{2^m}{2^l}$ where $n_p \Leftrightarrow n_i' = d \quad 2^m$ and d is odd. Hence, $2^m \quad n_p \Leftrightarrow n_i' \quad n_p$. Hence,

$$Pr[z_0'^i + (n_i' + 1) \quad r_0'^i = x_b^a + b \quad r_0^a \mid R^s] = Pr[(n_p \Leftrightarrow n_i') \quad r_0^p = 0 \mid R^s] \quad \frac{2^m}{2^l} \quad \frac{n_p}{2^l}.$$

Hence,

$$Pr[Z_S \mid R^s] \qquad \sum_{a=1}^{q_s} \sum_{b=1}^{n_a+1} Pr[z_0'^i + (n_i' + 1) \quad r_0'^i = x_b^a + b \quad r_0^a \mid R^s]$$

$$
\begin{aligned}
= \quad & \sum_{a=1}^{q_s}\sum_{b=1}^{n_a} Pr[z_0'^i + (n_i'+1) \quad r_0'^i = x_b^a + b \quad r_0^a \mid R^s] + \\
& \sum_{a=1, a\neq p}^{q_s} Pr[z_0'^i + (n_i'+1) \quad r_0'^i = z_0^a + (n_a+1) \quad r_0^a \mid R^s] + \\
& Pr[z_0'^i + (n_i'+1) \quad r_0'^i = z_0^p + (n_p+1) \quad r_0^a \mid R^s] \\
& \sum_{a=1}^{q_s}\sum_{b=1}^{n_a}\frac{1}{2^l} + \sum_{a=1,a\neq p}^{q_s}\frac{1}{2^l} + \frac{n_p}{2^l} \quad \sum_{a=1}^{q_s}\sum_{b=1}^{n_a+1}\frac{1}{2^l} + \frac{n_p}{2^l} \quad \frac{\mu_s}{l2^l} + \frac{n_p}{2^l}.
\end{aligned}
$$

□

Proof of Claim 3

(a) Event R_i^v deals with collisions between inputs to f at verification of forgery i and the counter corresponding to forgery i. Hence, in a manner similar to the one used in the Proof of Claim 1(a)

$$Pr[\overline{R_i^v} \mid R^s] \quad \frac{n_i'}{2^l}.$$

□

(b) The proof of this Claim is very similar to the proof of Claim 3 in the Proof of Theorem 2. First, we choose an index j such that for any type of possible non-truncation forgery i, the input to f at the verification of forgery i, namely $x_j'^i + j \quad r_0'^i$, does collide with any input to f during message signing with low probability. Next, we compute an upper bound for these collisions.

All non-truncation forgeries can be partitioned in a similar manner as that used in the proof of Claim 3 of Theorem 2. That is, we define extensions of the plaintext of a signed message, which we call the *prefix* case, and the complementary case, which we call *non-prefix* case. The non-prefix case includes two separate subcases, namely when ctr'^i is different from any ctr^p of any message p obtained at signing (i.e., message (x^p, ctr^p, w^p))), or when there is a signed message p such that $ctr'^i = ctr^p$. Hence, in the latter subcase, there must be at least a block position j in the forged message x'^i that is different from the corresponding block of the signed message p. This partition of all possible forgery types shows that a forged message $x'^i = x_1'^i \quad x_{n_i'}'^i$ which is not a truncation, can be in one of the following three complementary types:
(a) $\exists p, 1 \quad p \quad q_s : n_i' > n_p, ctr'^i = ctr^p$ and $\forall k, 1 \quad k \quad n_p : x_k'^i = x_k^p$; i.e., the forged message is an extension of message x^p (the prefix case). The non-prefix case consists of the following two forgery types:
(b1) $ctr'^i \neq ctr^p, \forall p, 1 \quad p \quad q_s$; and
(b2) $\exists p, 1 \quad p \quad q_s : ctr'^i = ctr^p, \exists k, 1 \quad k \quad \min(n_i', n_p) : x_k'^i \neq x_k^p$; i.e., the forged message is obtained by modifying a queried message starting with some block between the second and last block.

Now we choose index j mentioned above for each type of possible non-truncation forgeries, as follows: for forgeries of type (a), $j = n_p + 1$; for forgeries of type (b1), $j = 1$; and for forgeries of type (b2), j is the smallest index such that $x_j'^i \neq x_j^p, 1 \quad j \quad \min\{n_p, n_i'\}$. In all cases $1 \quad j \quad n_i'$, and hence, the chosen block $x_j'^i$ is well defined.

Event C^i implies that $x_j'^i + j \quad r_0'^i \in I^s$ or $x_j'^i + j \quad r_0'^i \in S$. Hence, by union bound

$$Pr[C^i \mid R_i^v \text{ and } R^s] \quad Pr[x_j'^i + j \quad r_0'^i \in I^s \mid R_i^v \text{ and } R^s] + Pr[x_j'^i + j \quad r_0'^i \in S \mid R_i^v \text{ and } R^s].$$

Let us define the following events:

$$
\begin{aligned}
E_{I^s} &: x_j'^i + j \quad r_0'^i \in I^s \\
E_S &: x_j'^i + j \quad r_0'^i \in S.
\end{aligned}
$$

Hence,

$$Pr[C^i \mid R_i^v \text{ and } R^s] \quad Pr[E_{I^s} \mid R_i^v \text{ and } R^s] + Pr[E_S \mid R_i^v \text{ and } R^s].$$

We determine upper bounds for events $E_{I^s} \mid \overline{R_i^v}, E_S \mid \overline{R_i^v}$ using the following Claim, whose proof is found at the end of this appendix:

Claim 3.1
(a)

$$Pr[E_{I^s} \mid R_i^v \text{ and } R^s] \quad \frac{q_s n_i'}{2^l}.$$

(b)

$$Pr[E_S \mid R_i^v \text{ and } R^s] \quad \frac{\mu_s}{l2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad .$$

Based on Claim 3.1,

$$Pr[C^i \mid R_i^v \text{ and } R^s] \quad Pr[E_{I^s} \mid R_i^v \text{ and } R^s] + Pr[E_S \mid R_i^v \text{ and } R^s] \quad \frac{q_s n_i'}{2^l} + \frac{\mu_s}{l2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad .$$

□

(c) We find an upper bound for $Pr[\overline{D^i} \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s]$ in a manner very similar to the one used in Claim 4 of the Proof of Theorem 2. Event $\overline{D^i}$ is true if and only if there exists an index $s, 1 \quad s \quad n_i' + 1$ such that the block $x_s'^i + s \quad r_0'^i \in V_i$ collides with another block $x_t'^i + t \quad r_0'^i \in V_i, 1 \quad t \quad n_i' + 1, s \neq t$ or with ctr'^i. But the latter collisions, namely $x_s'^i + s \quad r_0'^i = ctr'^i$ where $x_s'^i + s \quad r_0'^i \in V_i$, is already precluded by event R_i^v. Hence,

$$\begin{aligned} &Pr[\overline{D^i} \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s] \\ &\quad Pr[x_s'^i + s \quad r_0'^i = x_t'^i + t \quad r_0'^i, \text{ where } x_s'^i + s \quad r_0'^i, x_t'^i + t \quad r_0'^i \in V_i, s \neq t \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s]. \end{aligned}$$

When event $\overline{C^i}$ is true, there exists at least one block $x_s'^i + s \quad r_0'^i \in V_i$. If V_i has only one element, then

$$Pr[x_s'^i + s \quad r_0'^i = x_t'^i + t \quad r_0'^i, \text{ where } x_s'^i + s \quad r_0'^i, x_t'^i + t \quad r_0'^i \in V_i, s \neq t \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s] = 0.$$

If V_i has at least two elements, then one can choose wlog $s \quad t$ and then

$$x_s'^i + s \quad r_0'^i = x_t'^i + t \quad r_0'^i \Leftrightarrow (t \Leftrightarrow s) \quad r_0'^i = x_s'^i \Leftrightarrow x_t'^i.$$

In this expression, $r_0'^i$ is unknown, random and uniformly distributed since events R_i^v and R^s are true. Furthermore, $x_s'^i, x_t'^i$ are constants, or $x_s'^i$ is a constant and $x_t'^i = z_0'^i$, since $t > s$; if $x_t'^i = z_0'^i$, then it is independent of $r_0'^i$ because $z_0'^i$ was obtained by enciphering with a different key. Hence, by Fact 2 (Appendix A), the probability is at most $\frac{2^m}{2^l}$, where $t \Leftrightarrow s = d \quad 2^m$ and d is odd. ¿From the definition of exponent m, it follows that $2^m \quad t \Leftrightarrow s \quad n_i'$ since $1 \quad s \quad t \quad n_i' + 1$. Hence,

$$Pr[x_s'^i + s \quad r_0'^i = x_t'^i + t \quad r_0'^i, \text{ where } x_s'^i + s \quad r_0'^i, x_t'^i + t \quad r_0'^i \in V_i, s \neq t \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s] \quad \frac{2^m}{2^l} \quad \frac{n_i'}{2^l}.$$

Hence,

$$Pr[\overline{D^i} \mid \overline{C^i} \text{ and } R_i^v \text{ and } R^s] \quad \frac{n_i'}{2^l}.$$

□

Proof of Claim 3.1

(a) Event E_{I^s} refers to collisions between the chosen block $x'^i_j + j \quad r'^i_0$ and counters at signing, namely $ctr^p, 1 \quad p \quad q_s$. Hence, by union bound, and Fact 2 (Appendix A)

$$Pr[E_{I^s} \mid R^v_i \text{ and } R^s] \quad \sum_{p=1}^{q_s} Pr[x'^i_j + j \quad r'^i_0 = ctr^p \mid R^v_i \text{ and } R^s] \quad \sum_{p=1}^{q_s} \frac{2^m}{2^l} = \frac{q_s 2^m}{2^l},$$

where $j = d \quad 2^m$ and d is odd, since, by events R^v_i and R^s r'^i_0 is unknown, random and uniformly distributed, x'^i_j is a constant, and ctr^p is a constant. Furthermore, since $2^m \quad j \quad n'_i$, it follows that

$$Pr[E_{I^s} \mid R^v_i \text{ and } R^s] \quad \frac{q_s j}{2^l} \quad \frac{q_s n'_i}{2^l}.$$

□

(b) Event E_{I^s} refers to collisions between the chosen block $x'^i_j + j \quad r'^i_0$ and inputs to f at signing other than counters, namely blocks $x^p_k + k \quad r^p_0, 1 \quad p \quad q_s, 1 \quad k \quad n_p + 1$. Hence, by union bound,

$$Pr[E_S \mid R^v_i \text{ and } R^s] \quad \sum_{p=1}^{q_s} \sum_{k=1}^{n_p} Pr[x'^i_j + j \quad r'^i_0 = x^p_k + k \quad r^p_0 \mid R^v_i \text{ and } R^s]$$

In a manner similar to the one used for Claim 3 (Part 2) in the proof of Theorem 2, we can show that

$$Pr[E_S \mid R^v_i \text{ and } R^s] \quad \frac{\mu_s}{l 2^{l+1}} \quad \log_2 \frac{\mu_s}{l} + 3 \quad .$$

□