OCB Mode

Proposal to NIST for a block-cipher mode of operation
which simultaneously provides privacy and authenticity

April 1, 2001

Submitter:

Phillip Rogaway

rogaway@cs.ucdavis.edu
http://www.cs.ucdavis.edu/∼rogaway

Department of Computer Science
University of California at Davis
Engineering II Building
Davis, California 95616 USA

+1 530 752 7583 office
+1 530 752 4767 fax

Department of Computer Science
Faculty of Science
Chiang Mai University
Chiang Mai 50200 Thailand

+66 1 881 8290 cell
+66 53 943 433 fax

Inventor and Owner:
Same as submitter

Auxiliary submitters:

Mihir Bellare
mihir@cs.ucsd.edu

Dept. of Comp. Sci. & Engineering
University of California at San Diego
La Jolla, California 92093 USA

John Black
jrb@cs.unr.edu

Dept. of Computer Science
University of Nevada
Reno, Nevada 89557 USA

Ted Krovetz
tdk@acm.org

Digital Fountain
600 Alabama Street
San Francisco, CA 94110 USA

Contents

1 Introduction

An authenticated-encryption scheme is a shared-key encryption scheme whose goal is to provide *both* privacy *and* authenticity. The encryption algorithm takes a key, a plaintext, and a nonce, and it returns a ciphertext. The decryption algorithm takes a key, a ciphertext, and a nonce, and it returns either a plaintext or a special symbol, INVALID. In addition to the customary privacy goal, an authenticated-encryption scheme aims for authenticity: if an adversary should try to create some new ciphertext, the decryption algorithm will almost certainly regard it as INVALID.

An authenticated-encryption scheme can be constructed by appropriately combining an encryption scheme and a message authentication code (MAC), an approach used pervasively in practice and in standards. (Analyses of these methods are provided in [6]). But an extremely attractive goal is an authenticated-encryption scheme having computational cost significantly lower than the cost to encrypt plus the cost to MAC. The classical approach for trying to do this is to encrypt-with-redundancy, where one appends a noncryptographic checksum to the message before encrypting it, typically with CBC mode. In the past, such schemes have invariably been broken. Recently, however, Jutla has proposed two authenticated-encryption schemes supported by a claim of provable security [16]. Jutla's schemes were the first ones publicly disclosed which meet the stated goals.

This present submission presents a new mode of operation, OCB, which refines one of Jutla's schemes, IAPM. OCB (which stands for "offset codebook") retains the desirable characteristics of IAPM—in particular, OCB is fully parallelizable and adds minor overhead compared to conventional modes. But OCB provides several new features, including that it works for messages of any bit length (returning a ciphertext of minimal length), it uses a single block-cipher key, and it employs an arbitrary nonce (as opposed to a random IV). The number of block-cipher invocations to encrypt+authenticate a message M is reduced to $\lceil |M|/n \rceil + 2$, where n is the block length. The overhead beyond block-cipher calls is likewise reduced. In settings where there is no opportunity for parallelizability, OCB-AES-128 costs about 7% more than CBC-AES-128. In settings where there is adequate opportunity for parallelizability, OCB will be faster than CBC.

OCB comes with a full proof of security. Specifically, we prove indistinguishability under chosen-plaintext attack (IND-CPA) [2, 14] and authenticity of ciphertexts [6, 7, 18]. As shown in [6, 18] this combination implies indistinguishability under the strongest form of chosen-ciphertext attack (CCA) (which is equivalent to non-malleability [9] under CCA [3]). Our proof of privacy assumes that the underlying block cipher is good in the sense of a pseudorandom permutation (PRP) [5, 19], while our proof of authenticity assumes that the block cipher is a strong PRP [19]. The actual results are quantitative; the security analysis is in the concrete-security paradigm.

2 Mathematical Preliminaries

NOTATION. If a and b are integers, $a \leq b$, then $[a..b]$ is the set $\{a, a+1, \ldots, b\}$. If $i \geq 1$ is an integer then $\mathsf{ntz}(i)$ is the number of trailing 0-bits in the binary representation of i (equivalently, $\mathsf{ntz}(i)$ is the largest integer z such that 2^z divides i). So, for example, $\mathsf{ntz}(7) = 0$ and $\mathsf{ntz}(8) = 3$.

A *string* is a finite sequence of symbols, each symbol being 0 or 1. The string of length 0 is called the *empty string* and is denoted ε. Let $\{0,1\}^*$ denote the set of all strings. If $A, B \in \{0,1\}^*$ then $A\,B$, or $A \parallel B$, is their concatenation. If $A \in \{0,1\}^*$ and $A \neq \varepsilon$ then $\mathsf{firstbit}(A)$ is the first bit of A and $\mathsf{lastbit}(A)$ is the last bit of A. Let i, n be nonnegative integers. Then 0^i and 1^i denote the strings of i 0's and 1's, respectively. Let $\{0,1\}^n$ denote the set of all strings of length n. If $A \in \{0,1\}^*$ then $|A|$ denotes the length of A, in bits, while $\|A\|_n = \max\{1,\ \lceil |A|/n \rceil\}$ denotes the length of A in n-bit blocks, where the empty string counts as one block. For $A \in \{0,1\}^*$ and

$|A| \le n$, $\mathsf{zpad}_n(A)$ is the string $A\,0^{n-|A|}$. With n understood we will write $A\,0^*$ for $\mathsf{zpad}_n(A)$. If $A \in \{0,1\}^*$ and $\tau \in [0..|A|]$ then $A\,[\text{first } \tau \text{ bits}]$ and $A[\text{last } \tau \text{ bits}]$ denote the first τ bits of A and the last τ bits of A, respectively. Both of these values are the empty string if $\tau = 0$. If $A, B \in \{0,1\}^*$ then $A \oplus B$ is the bitwise xor of $A\,[\text{first } \ell \text{ bits}]$ and $B\,[\text{first } \ell \text{ bits}]$, where $\ell = \min\{|A|, |B|\}$ (where $\varepsilon \oplus \varepsilon = \varepsilon$). So, for example, $1001 \oplus 11 = 01$. If $A = a_{n-1} \cdots a_1 a_0 \in \{0,1\}^n$ is a string (each $a_i \in \{0,1\}$) then $\mathsf{str2num}(A)$ is the number $\sum_{i=0}^{n-1} 2^i a_i$. If $a \in [0..2^n - 1]$ then $\mathsf{num2str}_n(a)$ is the n-bit string A such that $\mathsf{str2num}(A) = a$. Let $\mathrm{len}_n(A) = \mathsf{num2str}_n(|A|)$. We omit the subscript when n is understood.

If $A = a_{n-1}a_{n-2} \cdots a_1 a_0 \in \{0,1\}^n$ then $A{\ll}1 = a_{n-2}a_{n-3} \cdots a_1 a_0 0$ is the n-bit string which is a left shift of A by 1 bit (the first bit of A disappearing and a zero coming into the last bit), while $A{\gg}1 = 0a_{n-1}a_{n-2} \ldots a_2 a_1$ is the n-bit string which is a right shift of A by one bit (the last bit disappearing and a zero coming into the first bit).

In pseudocode we write "Partition M into $M[1] \cdots M[m]$" as shorthand for "Let $m = \|M\|_n$ and let $M[1], \ldots, M[m]$ be strings such that $M[1] \cdots M[m] = M$ and $|M[i]| = n$ for $1 \le i < m$." We write "Partition $\mathcal{C}$ into $C[1] \cdots C[m]T$" as shorthand for "if $|\mathcal{C}| < \tau$ then return INVALID. Otherwise, let $C = \mathcal{C}\,[\text{first } |\mathcal{C}| - \tau \text{ bits}]$, let $T = \mathcal{C}[\text{last } \tau \text{ bits}]$, let $m = \|C\|_n$, and let $C[1], \ldots, C[m]$ be strings such that $C[1] \cdots C[m] = C$ and $|C[i]| = n$ for $1 \le i < m$. Recall that $\|M\|_n = \max\{1,\ \lceil |M|/n \rceil\}$, so the empty string partitions into $m = 1$ block, that one block being the empty string.

THE FIELD WITH 2^n POINTS. Recall that a finite field is a finite set together with an addition operation and a multiplication operation, each defined to take a pair of points in the field to another point in the field. The operations must obey certain basic axioms. (For example, there must be a point 0 in the field such that $a + 0 = 0 + a = a$ for every a; there must be a point 1 in the field such that $a \cdot 1 = 1 \cdot a = a$ for every a; and for every $a \ne 0$ there must be a point a^{-1} in the field such that $a \cdot a^{-1} = a^{-1} \cdot a = 1$.) If one fixes a positive integer n, then there turns out to be a unique finite field (up to the naming of the points) that has 2^n elements. It is called the Galois field of size 2^n, and it is denoted $\mathrm{GF}(2^n)$.

Example 1 *The field* $\mathrm{GF}(2)$ *has two points,* 0 *and* 1*, and operations* $\oplus$ *(addition) and* $\cdot$ *(multiplication) are defined by* $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$, $0 \cdot 0 = 0$, $0 \cdot 1 = 0$, $1 \cdot 0 = 0$, *and* $1 \cdot 1 = 1$.

We interchangeably think of a point a in $\mathrm{GF}(2^n)$ in any of the following ways: (1) as an abstract point in a field; (2) as an n-bit string $a_{n-1} \ldots a_1 a_0 \in \{0,1\}^n$; (3) as a formal polynomial $a(\mathtt{x}) = a_{n-1}\mathtt{x}^{n-1} + \cdots + a_1\mathtt{x} + a_0$ with binary coefficients; (4) as a nonnegative integer between 0 and $2^n - 1$, where the string $a \in \{0,1\}^n$ corresponds to the number $\mathsf{str2num}(a)$. For example, one can regard the string $a = 0^{125}101$ as a 128-bit string, as the number 5, as the polynomial $\mathtt{x}^2 + 1$, or as a particular point in the finite field $\mathrm{GF}(2^{128})$. We write $a(\mathtt{x})$ instead of a if we wish to emphasize that we are thinking of a as a polynomial.

To add two points in $\mathrm{GF}(2^n)$, take their bitwise xor. We denote this operation by $a \oplus b$.

Before we can say how to multiply two points we must fix some irreducible polynomial $p_n(\mathtt{x})$ having binary coefficients and degree n. (Saying that $p_n(\mathtt{x})$ is irreducible means that if $q(\mathtt{x})$ and $q'(\mathtt{x})$ are polynomials over $\mathrm{GF}(2)$ which multiply to give $p_n(\mathtt{x})$, then one of these polynomials is 1 and the other is $p_n(\mathtt{x})$.) For OCB, choose the lexicographically first polynomial among the irreducible degree n polynomials having a minimum number of coefficients. For $n = 128$, the indicated polynomial is

$$p_{128}(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$$

A few other $p_n(\mathtt{x})$-values are $\mathtt{x}^{64} + \mathtt{x}^4 + \mathtt{x}^3 + \mathtt{x} + 1$ and $\mathtt{x}^{96} + \mathtt{x}^{10} + \mathtt{x}^9 + \mathtt{x}^6 + 1$ and $\mathtt{x}^{160} + \mathtt{x}^5 + \mathtt{x}^3 + \mathtt{x}^2 + 1$ and $\mathtt{x}^{192} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$ and $\mathtt{x}^{224} + \mathtt{x}^9 + \mathtt{x}^8 + \mathtt{x}^3 + 1$ and $\mathtt{x}^{256} + \mathtt{x}^{10} + \mathtt{x}^5 + \mathtt{x}^2 + 1$.

To multiply points $a, b \in \mathrm{GF}(2^n)$, which we denote $a \cdot b$, regard a and b as polynomials $a(\mathtt{x}) = a_{n-1}\mathtt{x}^{n-1} + \cdots + a_1\mathtt{x} + a_0$ and $b(\mathtt{x}) = b_{n-1}\mathtt{x}^{n-1} + \cdots + b_1\mathtt{x} + b_0$, form their product $c(\mathtt{x})$ where one adds and multiplies coefficients in GF(2) (the coefficient of degree j in $c(\mathtt{x})$, where $j \in [0..2n-2]$, is $c_j = \oplus_{i=0}^{j}\,(a_i \cdot b_{j-i})$) and take the remainder one gets when dividing $c(\mathtt{x})$ by the polynomial $p_n(\mathtt{x})$.

By convention, the multiplication operator has higher precedence than addition operator so, for example, $\gamma_1 \cdot L \oplus R$ means $(\gamma_1 \cdot L) \oplus R$.

Example 2 *Assume $n = 128$. Suppose one multiplies $a(\mathtt{x}) = \mathtt{x}^{127} + \mathtt{x} + 1$ by $b(\mathtt{x}) = \mathtt{x} + 1$. The result is $c(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^2 + \mathtt{x} + \mathtt{x}^{127} + \mathtt{x} + 1 = \mathtt{x}^{128} + \mathtt{x}^{127} + \mathtt{x}^2 + 1$. If one divides $c(\mathtt{x})$ by $p(\mathtt{x})$ one gets a quotient of $q(\mathtt{x}) = 1$ and a remainder (which is the answer) of $r(\mathtt{x}) = \mathtt{x}^{127} + \mathtt{x}^7 + \mathtt{x}$. In string notation, $10^{125}11 \cdot 0^{126}11 = 10^{119}10000010$.*

It is particularly easy to multiply a point $a \in \{0,1\}^n$ by $\mathtt{x}$. We illustrate the method for $n = 128$, where $p(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$. Then multiplying $a = a_{n-1} \cdots a_1 a_0$ by $\mathtt{x}$ yields a product $a_{n-1}\mathtt{x}^n + a_{n-2}\mathtt{x}^{n-1} + a_1\mathtt{x}^2 + a_0\mathtt{x}$. Thus, if the first bit of a is 0, then $a \cdot \mathtt{x} = a \ll 1$. If the first bit of a is 1 then we must add $\mathtt{x}^{128}$ to $a \ll 1$. Since $\mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1 = 0$ we know that $\mathtt{x}^{128} = \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$, so adding $\mathtt{x}^{128}$ means to xor by $0^{120}10000111$. In summary, when $n = 128$,

$$a \cdot \mathtt{x} = \begin{cases} a \ll 1 & \text{if } \mathsf{firstbit}(a) = 0 \\ (a \ll 1) \oplus 0^{120}10000111 & \text{if } \mathsf{firstbit}(a) = 1 \end{cases}$$

Example 3 *Let us again compute $10^{125}11 \cdot 0^{126}11$. Since the latter string is $\mathtt{x} + 1$, we should multiply the first string by $\mathtt{x}$ and then add it to (xor it with) the first string. As the first bit of $10^{125}11$ is 1, multiplying this point by $\mathtt{x}$ yields $0^{125}110 \oplus 0^{120}10000111 = 0^{120}10000001$, and xoring this with $10^{125}11$ gives a final answer of $10^{119}10000010$, as before.*

If $L \in \{0,1\}^n$ and $i \geq 0$, we write $L(i)$ as shorthand for $L \cdot \mathtt{x}^i$. We have an easy way to compute $L(1), L(2), \ldots, L(\mu)$-values, where μ is a small number. Namely, set $L(0) = L$ and compute $L(i) = L(i-1) \cdot \mathtt{x}$ for all $i \in [1..\mu]$.

If $a \neq 0$ is a point in $\{0,1\}^n$, we can divide a by $\mathtt{x}$, meaning that one multiplies a by the multiplicative inverse of $\mathtt{x}$ in the field: $a \cdot \mathtt{x}^{-1}$. It is easy to compute $a \cdot \mathtt{x}^{-1}$. To illustrate, again assume that $n = 128$. Then if the last bit of a is 0, then $a \cdot \mathtt{x}^{-1}$ is $a \gg 1$. If the last bit of a is 1, then we must add (xor) to $a \gg 1$ the value $\mathtt{x}^{-1}$. Since $\mathtt{x}^{128} = \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$ we have $\mathtt{x}^{127} = \mathtt{x}^6 + \mathtt{x} + 1 + \mathtt{x}^{-1}$ and so $\mathtt{x}^{-1} = \mathtt{x}^{127} + \mathtt{x}^6 + \mathtt{x} + 1 = 10^{120}1000011$. In summary, for $n = 128$,

$$a \cdot \mathtt{x}^{-1} \quad = \quad \begin{cases} a \gg 1 & \text{if } \mathsf{lastbit}(a) = 0 \\ (a \gg 1) \oplus 10^{120}1000011 & \text{if } \mathsf{lastbit}(a) = 1 \end{cases}$$

We point out that, for any $n = 128$, the value *huge* $= \mathtt{x}^{-1}$ will be an enormous number (when viewed as a number); in particular, *huge* starts with a 1 bit, so $2^{n-1} \leq$ *huge*. For the remainder of this submission, we will use *huge* as a synonym for $\mathtt{x}^{-1}$ whenever this seems to add to clarity. We will later assume that any messages $M = M[1] \cdots M[m]$ to be MACed has block length $m <$ *huge*, for otherwise our theorem statements assert a non-result. Thus for any message $M = M[1] \cdots M[m]$ to be MACed, each of $\gamma_1, \gamma_2, \ldots, \ldots, \gamma_m$ is different from *huge*.

Gray codes. For $\ell \geq 1$, a Gray code is an ordering $\gamma^\ell = \gamma_0^\ell \;\; \gamma_1^\ell \;\; \ldots \;\; \gamma_{2^\ell-1}^\ell$ of $\{0,1\}^\ell$ such that successive points differ (in the Hamming sense) by just one bit. For n a fixed number, OCB makes use of the "canonical" Gray code $\gamma = \gamma^n$ constructed by

$$\gamma^1 = 0 \;\; 1$$

while, for $\ell > 0$,

$$\gamma^{\ell+1} = 0\gamma_0^\ell \quad 0\gamma_1^\ell \quad \cdots \quad 0\gamma_{2^\ell-2}^\ell \quad 0\gamma_{2^\ell-1}^\ell \quad 1\gamma_{2^\ell-1}^\ell \quad 1\gamma_{2^\ell-2}^\ell \quad \cdots \quad 1\gamma_1^\ell \quad 1\gamma_0^\ell$$

It is easy to see that γ is a Gray code. What is more, for $1 \le i \le 2^n - 1$, $\gamma_i = \gamma_{i-1} \oplus (0^{n-1}1 \ll \mathsf{ntz}(i))$. This makes it easy to compute successive points.

Example 4 *The canonical Gray code with 2 points is* $\gamma^1 = \gamma_0^1\gamma_1^1 = 0\ 1$. *The canonical Gray code with 4 points is obtained by writing this once forward, then once backwards, prefixing each string in the first half by 0 and prefixing each string in the second half by 1: that is,* $\gamma^2 = \gamma_0^2\gamma_1^2\gamma_2^2\gamma_3^2 = 00\ 01\ 11\ 10 = 0\ 1\ 3\ 2$. *Repeating the process, the canonical Gray code with 8 points is* $\gamma^3 = \gamma_0^3\gamma_1^3\gamma_2^3\gamma_3^3\gamma_4^3\gamma_5^3\gamma_6^3\gamma_7^3 = 000\ 001\ 011\ 010\ 110\ 111\ 101\ 100 = 0\ 1\ 3\ 2\ 6\ 7\ 5\ 4$. *In OCB we use the Gray code* $\gamma = \gamma^n$ *having* 2^n *points:* $\gamma = \gamma_0\ \gamma_1\ \gamma_2\ \gamma_3\ \cdots \gamma_{2^n-1} = 0\ 1\ 3\ 2\ 6\ 7\ 5\ 4\ 12\ \cdots\ 2^{n-1}$. *To calculate* γ_i *from* γ_{i-1}, *xor* γ_{i-1} *by* $0^{n-1}1 \ll \mathsf{ntz}(i)$. *For example,* $\gamma_8 = 12$ *can be computed from* $\gamma_7 = 4$ *by xoring* 4 *with* $0^{n-1}1 \ll 3$.

We emphasize the following characteristics of the Gray-code values $\gamma_1, \gamma_2, \ldots, \gamma_{2^n-1}$. First, they are distinct and different from 0. Second, that $\gamma_1 = 1$. Third, that $\gamma_i \le 2i$.

Let $L \in \{0,1\}^n$ and consider the problem of successively forming the strings $\gamma_1 \cdot L$, $\gamma_2 \cdot L$, $\gamma_3 \cdot L$, …, $\gamma_m \cdot L$. Of course $\gamma_1 \cdot L = 1 \cdot L = L$. Now, for $i \ge 2$, assume one has already produced $\gamma_{i-1} \cdot L$. Since $\gamma_i = \gamma_{i-1} \oplus (0^{n-1}1 \ll \mathsf{ntz}(i))$ we know that

$$\begin{aligned}
\gamma_i \cdot L &= (\gamma_{i-1} \oplus (0^{n-1}1 \ll \mathsf{ntz}(i))) \cdot L \\
&= (\gamma_{i-1} \cdot L) \oplus (0^{n-1}1 \ll \mathsf{ntz}(i)) \cdot L \\
&= (\gamma_{i-1} \cdot L) \oplus (L \cdot \mathtt{x}^{\mathsf{ntz}(i)}) \\
&= (\gamma_{i-1} \cdot L) \oplus L(\mathsf{ntz}(i))
\end{aligned}$$

That is, the ith word in the sequence $\gamma_1 \cdot L, \gamma_2 \cdot L, \gamma_3 \cdot L, \ldots$ is obtained by xoring the previous word with $L(\mathsf{ntz}(i))$.

Had the sequence we were considering been $\gamma_1 \cdot L \oplus R, \gamma_2 \cdot L \oplus R, \gamma_3 \cdot L \oplus R, \ldots$ the ith word would be formed in the same way for $i \ge 2$, but the first word in the sequence would have been $L \oplus R$ instead of L.

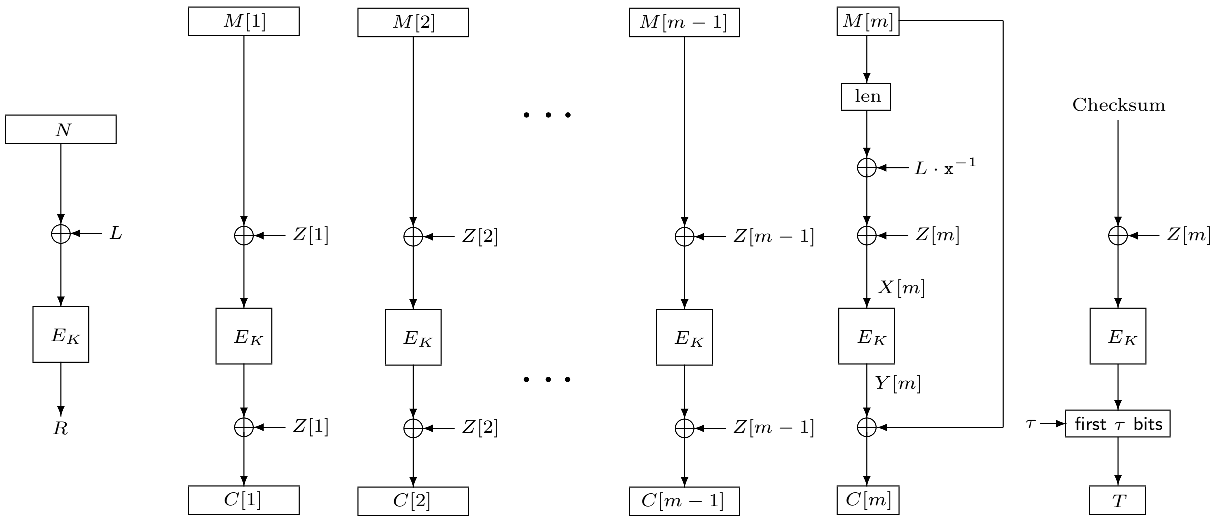
3 Specification

3.1 Definition of the Scheme

Parameters. To use OCB one must specify two parameters: a block cipher and a tag length.

- The **block cipher** E is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, for some number n, where each $E(K, \cdot) = E_K(\cdot)$ is a permutation on $\{0,1\}^n$. Here $\mathcal{K}$ is the set of possible keys and n is the block length. Both are arbitrary, though we insist that $n \ge 64$, and we discourage $n < 128$.
- The **tag length** τ is an integer between 0 and n. By trivial means, the adversary will be able to forge a valid ciphertext with probability $2^{-\tau}$.

The popular block cipher to use with OCB is likely to be AES-128, AES-192, or AES-256, but use of OCB with any other block ciphers is allowed. As for the tag length, a suggested default of $\tau = 64$



```
Algorithm OCB.Enc_K (N, M)

Partition $M$ into $M[1]\cdots M[m]$
$L \leftarrow E_K(0^n)$
$R \leftarrow E_K(N \oplus L)$
for $i \leftarrow 1$ to $m$ do $Z[i] = \gamma_i \cdot L \oplus R$
for $i \leftarrow 1$ to $m-1$ do
    $C[i] \leftarrow E_K(M[i] \oplus Z[i]) \ \oplus\ Z[i]$
$X[m] \leftarrow \text{len}(M[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$C[m] \leftarrow Y[m] \oplus M[m]$
$C \leftarrow C[1]\cdots C[m]$
Checksum $\leftarrow$
    $M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
$T \leftarrow E_K(\text{Checksum} \oplus Z[m])$ [first $\tau$ bits]
return $\mathcal{C} \leftarrow C \parallel T$
```

```
Algorithm OCB.Dec_K (N, 𝒞)

Partition $\mathcal{C}$ into $C[1]\cdots C[m]\,T$
$L \leftarrow E_K(0^n)$
$R \leftarrow E_K(N \oplus L)$
for $i \leftarrow 1$ to $m$ do $Z[i] = \gamma_i \cdot L \oplus R$
for $i \leftarrow 1$ to $m-1$ do
    $M[i] \leftarrow E_K^{-1}(C[i] \oplus Z[i]) \ \oplus\ Z[i]$
$X[m] \leftarrow \text{len}(C[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$M[m] \leftarrow Y[m] \oplus C[m]$
$M \leftarrow M[1]\cdots M[m]$
Checksum $\leftarrow$
    $M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
$T' \leftarrow E_K(\text{Checksum} \oplus Z[m])$ [first $\tau$ bits]
if $T = T'$ then return $M$
             else return INVALID
```

Figure 1: **OCB encryption.** The message to encrypt is M and the key is K. Message M is written as $M = M[1]M[2]\cdots M[m-1]M[m]$, where $m = \max\{1, \lceil |M|/n \rceil\}$ and $|M[1]| = |M[2]| = \cdots = |M[m-1]| = n$. Nonce N is a non-repeating value selected by the party that encrypts. It is sent along with ciphertext $\mathcal{C} = C[1]C[2]C[3]\cdots C[m-1]C[m]\,T$. The Checksum is $M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$. Offset $Z[1] = L \oplus R$ while, for $i \geq 2$, $Z[i] = Z[i-1] \oplus L(\mathsf{ntz}(i))$. String L is defined by applying E_K to a fixed string, 0^n. For $M[m] \oplus Y[m]$ and $C[m] \oplus Y[m]$, truncate $Y[m]$ if it is longer than the other operand. By $C[m]\,0^*$ we mean $C[m]$ padded on the right with 0-bits to get to length n. The function len represents the length of its argument as an n-bit string.

is reasonable. Tags of length $\tau = 32$ bits have been standard for retail banking for many years, while tags of $\tau = 80$ bits are used in IPSec. Using a tag of more than 80 bits adds questionable security benefit, though it does lengthen each ciphertext.

We let $\mathrm{OCB}[E, \tau]$ denote the OCB mode of operation using block cipher E and tag length τ.

Nonces. Encryption under OCB mode requires an n-bit nonce, N. The nonce would typically be a counter (maintained by the sender) or a random value (selected by the sender). Security is maintained even if the adversary can control the nonce, subject to the constraint that no nonce may be repeated within the current session (that is, during the period of use of the current encryption key). The nonce need not be random, unpredictable, or secret.

The nonce N is needed both to encrypt and to decrypt. Typically it would be communicated, in the clear, along with the ciphertext. However, it is out-of-scope how the nonce is communicated to the party who will decrypt. In particular, we do not regard the nonce as part of the ciphertext.

Definition of the mode. See Figure 1 for a definition and illustration of OCB. The figure defines OCB encryption and decryption. The key space for OCB is the key space $\mathcal{K}$ for the underlying block cipher E.

3.2 Conformance Criteria

An implementation of OCB is said to conform to this specification if

- Some specified subset $\mathrm{MsgSpace} \subseteq \{0,1\}^*$ of plaintexts and ciphertexts can be presented for encryption and decryption; and
- The encryption of a string $M \in \mathrm{MsgSpace}$ under key $K \in \mathcal{K}$ and using nonce $N \in \{0,1\}^n$ yields either $\mathrm{OCB.Enc}_K(N, M)$ or else an indication of failure; and
- The decryption of a string $\mathcal{C} \in \mathrm{MsgSpace}$ under key $K \in \mathcal{K}$ and using nonce $N \in \{0,1\}^n$ yields either $\mathrm{OCB.Dec}_K(N, \mathcal{C})$ or else an indication of failure.

For example:

- A conforming implementation might only be able to encrypt and decrypt nonempty byte strings.
- An example of a reason for returning an indication of failure by the encryption process is that the plaintext is too long for the implementation.
- Example reasons for returning an indication of failure by the decryption process (besides OCB.Dec returning Invalid) are: the repetition of a nonce has been detected; or the nonce is outside of some "window" of currently-allowed nonces. These considerations arise when one wants to detect "replay attacks."

3.3 An Equivalent Description

The following description of OCB may help to clarify what a typical implementation might choose to do. (However, it is not the intent of this section to mandate any particular implementation strategy.) In what follows, fix a block length n, block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, and a tag length τ. An OCB implementation may work as follows.

Key generation. Choose a random key $K \overset{R}{\leftarrow} \mathcal{K}$ for the block cipher. The key K is provided to both the entity that encrypts and the entity that decrypts.

Key setup. Once the key K is known, the following may be precomputed.

1. *Setup the block-cipher key.* For the party that encrypts: do any necessary key setup associated to block-cipher encryption. For the party that decrypts: do any necessary key setup associated to block-cipher encryption and do any key setup associated to block-cipher decryption.
2. *Precompute L.* Let $L \leftarrow E_K(0^n)$, where 0^n is the n-bit string specified already.
3. *Precompute $L(i)$-values.* Let m bound the maximum number of n-bit blocks that any message which will be encrypted or decrypted may have. Let $\mu \leftarrow \lceil \log_2 m \rceil$. Let $L(0) \leftarrow L$ and, for $i \in [1..\mu]$, let $L(i) \leftarrow L(i-1) \cdot \mathtt{x}$ using a shift and a conditional xor, as described in Section 2. Compute $L(-1) \leftarrow L \cdot \mathtt{x}^{-1}$ using a shift and a conditional xor, as described in Section 2. Save the values $L(-1), L(0), L(1),\ L(2),\ \ldots,\ L(\mu)$ in a table.

Encryption. To encrypt plaintext $M \in \{0,1\}^*$ using key K nonce $N \in \{0,1\}^n$, obtaining a ciphertext $\mathcal{C}$, do the following steps.

1. *Partition the plaintext.* Let $m \leftarrow \lceil |M|/n \rceil$. If $m = 0$ then let $m \leftarrow 1$. Let $M[1], \ldots, M[m]$ be strings such that $M[1] \cdots M[m] = M$ and $|M[i]| = n$ for $i \in [1..m-1]$.
2. *Initialize variables.* Let *Offset* $\leftarrow E_K(N \oplus L)$. Let Checksum $\leftarrow 0^n$.
3. *Encipher all blocks but the last one.* For $i \leftarrow 1$ to $m-1$, do the following:

```
        Let Checksum ← Checksum $\oplus M[i]$.
        Let $Offset \leftarrow Offset \oplus L(\mathsf{ntz}(i))$.
        Let $C[i] \leftarrow E_K(M[i] \oplus Offset) \oplus Offset$.
```

4. *Mask the final block.*

```
        Let $Offset \leftarrow Offset \oplus L(\mathsf{ntz}(m))$.
        Let $Y[m] \leftarrow E_K(\mathrm{len}(M[m]) \oplus L(-1) \oplus Offset)$.
        Let $C[m] \leftarrow M[m]$ xored with the first $|M[m]|$ bits of $Y[m]$.
        Let Checksum ← Checksum $\oplus Y[m] \oplus C[m]\,0^*$.
```

5. *Form the tag.* Let T be the first τ bits of $E_K(\mathrm{Checksum} \oplus Offset)$.
6. *Return the ciphertext.* The ciphertext is $\mathcal{C} = C[1] \cdots C[m-1] C[m]\, T$. It must be communicated along with the nonce N.

Decryption. To decrypt a ciphertext $\mathcal{C} \in \{0,1\}^*$ using key K and nonce $N \in \{0,1\}^n$, obtaining a plaintext $M \in \{0,1\}^*$ or else an indication Invalid, do the following steps.

1. *Partition the ciphertext.* If $|\mathcal{C}| < \tau$, then return Invalid (the ciphertext has been rejected). Otherwise, let C be the first $|\mathcal{C}| - \tau$ bits of $\mathcal{C}$ and let T be the remaining τ bits. Let $m \leftarrow \lceil |C|/n \rceil$. If $m = 0$ then let $m = 1$. Let $C[1], \ldots, C[m]$ be strings such that $C[1] \cdots C[m] = C$ and $|C[i]| = n$ for $i \in [1..m-1]$.
2. *Initialize variables.* Let *Offset* $\leftarrow E_K(N \oplus L)$. Let Checksum $\leftarrow 0^n$.
3. *Decipher all blocks but the last one.* For $i \leftarrow 1$ to $m-1$, do the following:

```
        Let $Offset \leftarrow Offset \oplus L(\mathsf{ntz}(i))$.
        Let $M[i] \leftarrow E_K^{-1}(C[i] \oplus Offset) \oplus Offset$.
        Let Checksum ← Checksum $\oplus M[i]$.
```

4. *Recover the final block.*

```
        Let $Offset \leftarrow Offset \oplus L(\mathsf{ntz}(m))$.
        Let $Y[m] \leftarrow E_K(\mathrm{len}(C[m]) \oplus L(-1) \oplus Offset)$.
        Let $M[m] \leftarrow C[m]$ xored with the first $|C[m]|$ bits of $Y[m]$.
        Let Checksum ← Checksum $\oplus Y[m] \oplus C[m]\,0^*$.
```

5. *Check the tag.* Let T' be the first τ bits of $E_K(\text{Checksum} \oplus \mathit{Offset})$. If $T \neq T'$ then return Invalid (the ciphertext has been rejected). Otherwise,
6. *Return the plaintext.* The plaintext is $M = M[1] \cdots M[m-1]M[m]$.

4 Discussion

4.1 Properties

OCB has been designed to have a variety of desirable properties. These properties are summarized in Figure 2. We now expand on some of the points referenced in that table.

Security Function. We emphasize that an authenticated-encryption scheme has qualitatively better security guarantees than those provided by standard modes of operation. In particular, non-malleability [9] and indistinguishability under chosen-ciphertext attack are not achieved by CBC, or by any other standard mode, but these properties are achieved by OCB. Achieving these properties is automatic when one achieves indistinguishability under chosen-plaintext attack *and* authenticity-of-ciphertexts [6, 18]. The lack of strong security properties has been a problem for the standard modes of operation, because many users of encryption implicitly assume strong security properties when designing their protocols. For example, it is common to see protocols which use symmetric encryption in order to "bind together" the parts of a plaintext. It is also common to see protocols which encrypt related messages as a way to do a "handshake." Standard modes do not support such practices. This fact has sometimes led practitioners to invent their own peculiar ways to encrypt (as when the Needham-Schroeder 3-party key-distribution protocol, which uses encryption for both binding and a related-message handshake, was modified and implemented within Kerberos). We believe that a mode like OCB is less likely to be misused in applications because the common "abuses" of encryption become correct cryptographic techniques.

By way of comparison, a chosen-ciphertext attack by Bleichenbacher on the public-key encryption scheme of RSA PKCS #1 v.1 motivated the company that controls this de facto standard to upgrade it [8, 20]. In contrast, people seem to accept as a matter of course symmetric-encryption schemes which are not even non-malleable. There would seem to be no technical reason to account for this difference in expectations.

Error-Propagation. We view error-propagation as a largely outmoded idea; in most contexts, it no longer has any significance. But to the extent that "infinite error propagation" is the phrase but "message integrity" is the underlying goal (by message integrity we mean, here, the detection of non-adversarial modifications to a ciphertext), we point out that message integrity is automatic for any authenticated-encryption scheme and, what is more, a scheme like OCB achieves it at a lower added cost, in software, than computing a CRC-32 checksum, for example.

Parallelizability. In settings where there is adequate opportunity for parallelism, OCB encryption will be faster than CBC encryption. We believe that parallelizability is becoming important for obtaining good performance from both high-speed hardware and commodity processors. In the former case, one may want to encrypt-and-authenticate at speeds in excess of 10 Gbits/second—an impossible task for CBC (with today's technology). In the latter case, there is an architectural trend towards highly pipelined machines with multiple instruction pipes and lots of registers. Optimally exploiting such features necessitates algorithms with plenty to do in parallel.

We were pleased that the selected AES algorithm is an algorithm with good parallelizability characteristics. In some ways this lessens the need to achieve parallelizability at the level of the mode of operation. Still, it can only be good that ciphertext blocks can be computed in parallel.

Security Function	**Authenticated encryption**. Provides both privacy and authenticity, eliminating the need to compute a separate MAC. Specifically, the scheme achieves authenticity of ciphertexts [6, 7, 18] and indistinguishability under chosen-plaintext attack [2, 14].
Error Propagation	**Infinite**. If the ciphertext is corrupted in any manner then the received ciphertext will almost certainly (probability $\approx 1 - 2^{-\tau}$) be rejected.
Synchronization	**Optional**. If the nonce N is transmitted along with each ciphertext, there are no synchronization requirements. If it is not sent (to save transmission bits) the receiver must maintain the corresponding value.
Parallelizability	**Fully parallelizable**. Both encryption and decryption are fully parallelizable: all block-cipher invocations (except the first and last) may be computed at the same time.
Keying Material	**One block-cipher key**. One needs a single key, K, which keys all invocations of the underlying block cipher.
Ctr/IV/Nonce Requirements	**Single-use nonce**. The encrypting party must supply a new nonce with each message it encrypts. The nonce need not be unpredictable or secret. The nonce is n bits long (but it would typically be communicated using fewer bits, as determined by the application).
Memory Requirements	**Very modest**. About $6n$ bits beyond the key are sufficient for internal calculations. Implementations may choose whether or not to store $L(i)$-values, allowing some tradeoff between memory and simplicity/speed.
Pre-processing Capability	**Limited.** During key-setup the string L would typically be pre-computed (one block cipher call), as would the first few $L(i)$ values, and maybe $L \cdot \mathtt{x}^{-1}$. The block-cipher key K would be converted into its convenient representation. Unlike counter mode, additional pre-computation prior to knowing the string to encrypt/decrypt is not possible.
Message-Length Requirements	**Any bit string allowed.** Any string $M \in \{0,1\}^*$ may be encrypted, including the empty string and strings which are not an integral number of bytes. The length of the string does need not be known in advance.
Ciphertext Expansion	**Minimal possible** (for a scheme meeting the desired privacy notion). Expansion is 0–n bits for the tag plus 0–n bits for the nonce. The former depends on a user-specified parameter τ, with 32–80 bits being typical. Messages which are not a multiple of the blocksize do not receive additional expansion due to padding.
Other Characteristics	**Efficiency**: Uses $\lceil \lvert M \rvert / n \rceil + 2$ block-cipher calls and very efficient offset-calculations. **Endian neutrality**: Can be implemented equally efficiently on big-endian and little-endian machines. **Provable security**: The mode provably meets its goals, assuming the underlying block cipher meets now-standard cryptographic assumptions.

Figure 2: **Summary properties of OCB.**

Keying Material. Conceptually the key is (K, L), but L is defined from the underlying key K, and then key K is still used. Normally such "lazy key-derivation" would get one in trouble. For OCB we prove that it does not. Avoiding multiple block-cipher keys is important for saving on memory and key-setup time.

Ctr/IV/Nonce Requirements. We believe that modes of operation whose correct operation requires a random IV are error-prone. We have been careful to avoid this.

As an example, consider CBC mode ($C[i] = E_K(M[i] \oplus C[i-1])$ where $C[0] = \mathrm{IV}$). Though this mode has been around for ages, it seems not well known that the IV should be unpredicable: it must not be zero, a counter, or the last block of ciphertext from the previous message. If it is any of these things, one certainly will not achieve any of the standard definitions of security [2, 14]. Implementations and popular books routinely get this wrong. Of course, for a given application, one might be fine, because what is "leaked" by these usage errors will usually be irrelevant. The point is more that we have definitions for encryption-scheme security, and there is no reason to fall short of them—since when one does it is not clear what one is getting. For CBC, the "correction" is simple: define $C[0] = E_K(\mathrm{IV})$ instead of $C[0] = \mathrm{IV}$.

We comment that, as with any nonce-based scheme, it is particularly easy for the receiver to implement replay-detection for OCB, and doing this does not increase the length of ciphertexts.

Message-Length Requirements and Ciphertext Expansion. Any string $M \in \{0,1\}^*$ can be encrypted, and this yields a ciphertext $\mathcal{C}$ of length $|M| + \tau$. (Here we are not including the nonce that may accompany the ciphertext. The encoding of that nonce may occupy up to n additional bits.) This is better (by up to n bits of ciphertext length) than what one would get if one had used conventional padding.

Encryption is "on line," meaning that one does not need to know the length of the message M in advance. Instead, the message can be encrypted as one goes along, continuing until there is an indication that the message is now over. An incremental interface (in the style popular for cryptographic hash functions) would be used to support this functionality.

Efficiency. Shaving off a few block-cipher calls or a few bytes of ciphertext may not seem important. But often one is dealing with short messages. For example, roughly a third of the messages on the Internet backbone are 43 bytes. If one is encrypting messages of such short lengths, one should be very careful about both ciphertext expansion and extra computational work since, by percentage, the inefficiencies can be large.

Endian Neutrality. In contrast to a scheme based on mod p arithmetic or based on mod 2^n arithmetic, there is almost no endian-favoritism implicit in the definition of OCB. (The exception is that the one left shift used for forming $L(i+1)$ from $L(i)$ is more convenient under a big-endian convention, as is the one right shift used for forming $L(-1) = L \cdot \mathtt{x}^{-1}$ from L.)

Provable Security. In recent years provable security has become a popular goal. This is for good reason: provable security it is the best way to gain assurance that a cryptographic scheme does what it is supposed to do. For a scheme which enjoys provable security one does not need to consider attacks, since successful ones imply successful attacks on some simpler object (say the AES algorithm).

When we say that "OCB is provably secure" we are asserting the existence of two theorems. One says that *if* an adversary A could do a good job at forging ciphertexts with $\mathrm{OCB}[E, \tau]$ (the adversary does this much more than a $2^{-\tau}$ fraction of the time) then there would be *another* adversary B that does a good job at distinguishing $(E(K,\cdot), E^{-1}(K,\cdot))$, for a random key K, from $(\pi(\cdot), \pi^{-1}(\cdot))$, for a random permutation $\pi \in \mathrm{Perm}(n)$. The other theorem says that *if* an adversary A could do a good job at distinguishing $\mathrm{OCB}[E,\tau]$-encrypted messages from random strings, then there would be *another* adversary B that does a good job at distinguishing $E(K,\cdot)$, for a random key K, from $\pi(\cdot)$, for a random permutation $\pi \in \mathrm{Perm}(n)$. Theorems of this sort are called *reductions.* In cryptography, provable security normally means giving reductions (along with the associated definitions).

Provable security begins with Goldwasser and Micali [14], though the style of provable security which we use here—where the primitive is a block cipher, the scheme is a usage mode, and the analysis is concrete (no asymptotics)—is the approach of Bellare and Rogaway [2, 4, 5].

It is not enough to know that there is some sort of provable-security result; one should also understand the definitions and the bounds. We have already sketched the definitions. When we speak of the bounds we are addressing "how effective is the adversary B in terms of the efficacy of adversary A" (where A and B are as above). For OCB, the bounds can be summarized as follows. An adversary can always forge with probability $1/2^{\tau}$. Beyond this, the maximal added advantage is about $\sigma^2/2^n$, where σ is the total number of blocks the adversary sees. The privacy bound likewise degrades as $\sigma^2/2^n$. The conclusion is that one is safe using OCB as long as the underlying block cipher is secure and σ is small compared to $2^{n/2}$. This is the same security degradation one observes for CBC encryption and in the bound for the CBC MAC [2, 5]. This kind of security loss was the main motivation for choosing a block length for the AES algorithm of $n = 128$ bits.

Simplicity. Though not a quantifiable property, simplicity has been a central design goal. Among the characteristics which we see as contributing to simplicity:

- Because the offset stream is a key-dependent sequence translated by a nonce-dependent amount, implementations are simplified, since $L(1), L(2), \ldots$ may be pre-computed.
- Short and full final-message-blocks are handled without making a special case: the treatment of messages is uniform, regardless of their length. (Some earlier versions of OCB, including [22], treated full-final-block messages and short-final-block messages differently, using different offsets in these two cases. Correctness became more delicate, showing up in added case analysis and more obscure intuition.)
- Only the simplest form of padding is used: append a minimal number of 0-bits to make a string whose length is a multiple of n. (Some earlier versions used a more complex form of padding: append a 1-bit and then 0-bits when one is not a multiple of the block length; do nothing when one is already a multiple of the block length.) The 0^*-padding method is computationally fastest and helps avoid a proliferation of cases in the analysis.
- Only one algebraic structure is used throughout the algorithm: the finite field $\mathrm{GF}(2^n)$. (Some earlier versions allowed use of the ring of integers modulo n, and there were potential addition/xor interactions to worry about. Earlier schemes also allowed use of the finite field $\mathrm{GF}(p)$, for a prime p, which would be used in addition to $\mathrm{GF}(2^n)$.)
- The first offset is $\gamma_1 = 1$. (Some earlier versions had to "skip" this offset for technical reasons.) Then the offsets are taken monotonically, stopping at γ_m. One never has to go back to some "earlier" offset. (Some earlier versions used one or two extra offsets, or reverted to an early offset at the end.)
- There is still a need for one "peculiar" offset, $\gamma_m \oplus L \cdot \mathtt{x}^{-1}$. Earlier versions used $L\ll 1$ instead of $L \cdot \mathtt{x}^{-1}$. Though this looks simpler, it is fundamentally more complex: $L\ll 1$ can be one of two points in the field, complicating the correctness proof and the intuition, and losing a factor of 2 in the security analysis.
- The base offset, R, is made by a block-cipher call. Though there are alternatives, they seem

to add complexity, at least if one wants a non-cryptographic approach which takes an n-bit nonce and can be executed as fast as a single AES-128 call.

4.2 Design Rationale

Starting point. Jutla specified two modes of operation: a parallelizable mode, IAPM, and a non-parallelizable mode, IACBC. We immediately selected the former as our starting point in the design of OCB. Though the latter mode would be slightly faster for serial execution environments, the overall difference in speed would not be significant, while having a parallelizable scheme is a major win.

Not fixing how the nonce is communicated. We have chosen *not* to specify how the nonce is chosen or how it is communicated. Formally, it is not part of the ciphertext (even though the receiving party needs it to decrypt). This has been done because, in many contexts, there is already a natural value to use as a nonce. For example, there may already be a sequence number present in the flow which includes a ciphertext; or there may be a long random value already present in such a flow; or the sender and receiver may be communicating across a reliable channel, the two maintaining matching sequence numbers. Even when a protocol is designed from scratch, the number of needed bits in order to communicate the nonce will vary. In some applications, 32 or even 8 bits is enough. For example, one might have reason to believe that there are at most 2^{32} messages that will flow during the connection, or one may be communicating only the low bits of a sequence number, counting on the receiver to maintain the high-order bits, assuming a bounded amount of out-of-order message delivery. By not mandating how the nonce is communicated, applications can save on communication bits.

Not fixing the tag length. The number of bits that are necessary for the tag vary according to the application. In a context where the adversary obtains something quite valuable from a successfully forgery, one may wish to choose a tag length of 80 bits or more. In contexts such as authenticating a video stream, where an adversary would have to forge a significant fraction of the frames even to have a noticeable effect on the image, an 8-bit tag may be appropriate. With no universally correct value to choose, it is best to leave this parameter unspecified.

We comment that short tags are more appropriate for OCB than for some other MACs, particularly Carter-Wegman MACs. Many Carter-Wegman MACs have the property that if you can forge one message with probability δ, than you can forge an arbitrary set of (all correct) messages with probability δ. This does not appear to be true for OCB (though we have not investigated formalizing or proving such properties).

Forming R using a block-cipher call (rather than by non-cryptographic means). We discovered during our work that there are methods for authenticated-encryption which encrypt M using $\lceil |M|/n \rceil + 1$ block-cipher calls (as opposed to our $\lceil |M|/n \rceil + 2$ block-cipher calls). Shai Halevi also has made what amounts to this same observation [15]. However, the methods we know which save a block-cipher call either require an unpredicable IV instead of a nonce (which we have already said is an invitation for misuse) or they add conceptual and computation complexity to compute the initial offset R by non-cryptographic means (eg., using a finite-field multiplication). All in all, while methods exist to further reduce the number of block-cipher calls by one, we know of no way to do this that ends up being a net win.

Avoiding mod 2^n addition. Our earlier designs included a scheme based on modular 2^n addition ("addition" for the remainder of this paragraph). Basing an authenticated-encryption scheme on addition is an interesting idea due to Gligor and Donescu [11]. Compared to our GF(2^n)-based approach ("xor" for the remainder of this paragraph), an addition-based scheme is quicker to understand a specification for, and may be easier to implement. But the use of addition (where $n \geq 128$) has significant disadvantages:

- The bit-asymmetry of the addition operator implies that the resulting scheme will have a bias towards big-endian architectures or little-endian architectures; there will be no way to achieve an endian-neutral scheme. The AES algorithm was constructed to be endian-neutral. We did not want to lose this nice attribute with our mode of operation.
- Modular addition of n-bit words is unpleasant when programming in a high-level language, where one does not have access to the underlying add-with-carry instruction.
- Modular addition of n-bit words is not parallelizable. As a consequence, dedicated hardware will perform this operation more slowly than xor, and, correspondingly, modern processors can xor two n-bit quantities faster than they can add them.
- Supporting the last claim, we ran experiments which indicated that, on a Pentium 3 Processor, an assembly-language addition-based OCB used about 50% more overhead than our (xor-based) scheme. An even bigger performance difference was expected for C implementations.
- The concrete security bound is worse with an addition-based scheme: the degradation in the bound appears to be $\Theta(\lg \bar{m})$, where $\bar{m}$ is the maximal message length.
- We had constant difficulties getting our addition-based schemes right. Minimally, the use of addition complicates what is already a complex proof. And unexpected xor/addition interactions would sometimes emerge when working out the proofs.

We eventually came to feel that even the simplicity benefit of addition-based schemes was not quite real: these schemes are harder to understand, prove correct, and implement well. Nonetheless, we choose a design that does have a natural addition-based counterpart.

About the definition of the Checksum. An initially odd-looking aspect of the mode's definition is the definition of $\text{Checksum} = M[1] \oplus \cdots M[m-1] \oplus C[m]\,0^* \oplus Y[m]$. In Jutla's scheme, where one assumes that all messages are a positive multiple of the block length, the checksum is the simpler-looking $\text{Checksum} = M[1] \oplus \cdots M[m-1] \oplus M[m]$. We comment that these two definitions are *identical* in the case that $|M[m]| = n$. What is more, the definition $\text{Checksum} = M[1] \oplus \cdots M[m-1] \oplus M[m]\,0^*$ turns out to be the *wrong* way to generalize the Checksum to allow for short-final-block messages; in particular, the scheme using that checksum is easily attacked.

Avoiding pretag collisions. Many of our earlier schemes, including [22], allowed the adversary to force what we call a "pretag collision." Recall that we compute the tag T by first computing a "pretag" $X[m+1] = \text{Checksum} \oplus \text{SomeOffset}$, and then forming a value $Y[m+1] = E_K(X[m+1])$, and, finally, forming the tag T by doing some further processing to $Y[m+1]$. All schemes that we have considered have taken this form. For a scheme of this form, we say that an adversary can force a pretag collision if there is an N, $\bar{M}$ that can be asked, getting $\bar{C}\,\bar{T}$, and then a forgery attempt N, $C\,T$ can be generated such that, in the forgery attempt, the pretag $X[m+1]$ will coincide with a value $X[i]$ or $\bar{X}[i]$ at which the block cipher E was already evaluated. We have refined OCB so that the adversary can *not* force a pretag collision. We claim this is a good thing:

- An adversary's ability to force a pretag collision complicates any proof and subverts the intuition for correctness, which says that tags are unpredictable because pretag-values rarely repeat. For schemes like IAPM [16] and [22], where the adversary can force pretag collisions, this intuition is simply wrong, and the correct intuition is much more delicate.

- For many schemes, pretag collisions lead to subtle attacks. In [22], the value $Y[m+1]$ for short-final-block messages was offset by $\gamma_{m+1} \cdot L \oplus R$ and then truncated to make T. By forcing a pretag collision the adversary could create a ciphertext which would have a prior-to-truncation tag which differed from a known value by a known multiple of L. But we had also fixed the first two bits of L to be 0, which means that the first bit of L, $2 \cdot L$, and $3 \cdot L$ is known. Since we had allowed arbitrary taglengths, including $\tau = 1$ bit, the scheme did not have the desired security bound. Similarly, the addition-based version of the scheme only avoided trouble because we had taken the first τ bits of the extended tag—taking the last τ bits would have led to a similar attack.

These issues convinced us that we would have a simpler and more robust scheme if we could architect it so as to avoid pretag collisions.

4.3 Limitations

We are aware of the following limitation/drawbacks on the use of OCB.

- OCB decryption uses both the block cipher and its inverse. (OCB encryption uses only the forward direction of the block cipher.) Particularly when the block cipher is the AES algorithm, the effect of this is to have larger tables or different code compared to a mode like CTR, which uses only the forward direction of the block cipher.
- If a nonce N should get reused within the session associated to the underlying key K, the privacy of the messages associated to this nonce is compromised, and future message integrity is lost. Any standard should emphasize that nonces must not be reused within a session. (We comment that if a nonce gets reused, still K and L seem not to be compromised, and the privacy of other messages, past or future, does not seem to be impacted.) (An implementation of OCB encryption may wish to check the current nonce is different from the previous one used within this session. This provides a check against the most flagrant type of nonce-reuse.)
- Unlike CTR mode, there is no useful work that can be done (beyond key-setup) prior to knowing the message to encrypt. (Of course one could pre-compute R if one knows the next nonce, but this is minor.)
- Encryption and decryption are not quite symmetric. In particular, the Checksum is calculated in an inherently asymmetric manner, and the inverse direction of E, not E itself, is used in decryption. (Still, encryption and decryption are "nearly" symmetric.)
- As with all modes of operation, the key K used for OCB should be used only for this one purpose. Standard key-separation techniques should be used to derive a multiplicity of keys when keys are needed for a multiplicity of purposes. The key K itself must not be used to derive additional keys.

4.4 Design History

The submitter began the design of OCB soon after learning of Jutla's work at CRYPTO 2000. It seemed at first a simple project: create cleaner and more efficient version of Jutla's IAPM, and craft a good proof to go with it. The project turned out to be more complex than anticipated. At least 15 unpublished versions of the algorithm were eventually considered, evolving over about five months. The evolution was guided by three factors. First, bugs (sometimes quite subtle) were often found in the schemes. Second, sometimes proofs could not be pushed through even though we knew of no bug. Third, we would abandon a correct scheme every time we found a more elegant alternative.

In advance of the first modes workshop we released our then-current scheme [22]. But we later found that the scheme in [22] fell short of the desired security bound (because of forced pretag collisions sketched in Section 4.2). The scheme continued to evolve—both to address this issue and because we continued to find other, unrelated improvements.

At the time of [22], there were actually three schemes described: an addition based scheme (mod 2^n addition); an xor-based scheme ($\mathrm{GF}(2^n)$ addition); and a mod p scheme (where p is a large prime). But now, at this later point in the design process, it seemed necessary to make a choice. We have already described why we went with the xor-approach.

Our experience on the project has made it clear to us that the authenticated-encryption goal is surprisingly difficult to get right. And we had aimed not only to get it right, and to prove that we had done so, but also to work-in a host of further properties, as documented in Section 4.1. The various "tricks" used to achieve these further properties often wanted to work against one another, and seemed antithetical to getting the mode right.

5 Theorems

This section gives our security results on OCB. The proofs of the lemmas we use are deferred to Appendix A.

5.1 Security Definitions

We begin with the requisite definitions. These are not completely standard because OCB uses a nonce, and we wish to give the adversary every possible advantage (more than is available in real life) by allowing her to choose this nonce (though we forbid the adversary from choosing the same nonce twice).

Syntax. We extend the syntax of an encryption scheme as given in [2]. A (nonce-using, symmetric) encryption scheme Π is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and an associated number n (the nonce length). Here $\mathcal{K}$ is a finite set and $\mathcal{E}$ and $\mathcal{D}$ are deterministic algorithms. Encryption algorithm $\mathcal{E}$ takes strings $K \in \mathcal{K}$, $N \in \{0,1\}^n$, and $M \in \{0,1\}^*$, and returns a string $\mathcal{C} \leftarrow \mathcal{E}_K(N, M)$. Decryption algorithm $\mathcal{D}$ takes strings $K \in \mathcal{K}$, $N \in \{0,1\}^n$, and $\mathcal{C} \in \{0,1\}^*$, and returns $\mathcal{D}_K(N, M)$, which is either a string $M \in \{0,1\}^*$ or the distinguished symbol Invalid. If $\mathcal{C} \leftarrow \mathcal{E}_K(N, M)$ then $\mathcal{D}_K(N, \mathcal{C}) = M$.

Privacy. We modify the notion of real-or-random security from [2] to nonce-using schemes. We also modify the notion to give a particularly strong (and convenient) definition—one asserting indistinguishability from random strings. (The notion is easily seen to imply more standard definitions, and by tight reductions.) Consider an adversary A who has one of two types of oracles: a "real" encryption oracle or a "fake" encryption oracle. A real encryption oracle, $\mathcal{E}_K(\cdot,\cdot)$, takes as input N, M and returns $\mathcal{C} \leftarrow \mathcal{E}_K(N, M)$. It is assumed that $|\mathcal{C}| = \ell(|M|)$ depends only on $|M|$. A fake encryption oracle, $\$(\cdot,\cdot)$, takes as input N, M and returns returns a random string $\mathcal{C} \overset{R}{\leftarrow} \{0,1\}^{\ell(|M|)}$. Given adversary A and encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, define $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A) = \Pr[K \overset{R}{\leftarrow} \mathcal{K} :\ A^{\mathcal{E}_K(\cdot,\cdot)} = 1] - \Pr[K \overset{R}{\leftarrow} \mathcal{K} :\ A^{\$(\cdot,\cdot)} = 1]$. An adversary A is *nonce-respecting* if it never repeats a nonce: if A asks its oracle a query (N, M) it will never subsequently ask its oracle a query (N, M'), regardless of its coins (if any) and regardless of oracle responses. All adversaries are assumed to be nonce-respecting.

Authenticity. We extend the notion of integrity of ciphertexts of [6, 7, 18]. Fix an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and run an adversary A with an oracle $\mathcal{E}_K(\cdot,\cdot)$ for some key K. We say that adversary A *forges* (in this run) if A is nonce-respecting, A outputs $(N, \mathcal{C})$ where $D_K(N, \mathcal{C}) \neq$ Invalid, and A made no earlier query (N, M) which resulted in a response $\mathcal{C}$. Let $\mathbf{Adv}_{\Pi}^{\text{auth}}(A) = \Pr[K \overset{R}{\leftarrow} \mathcal{K} \colon\ A^{\mathcal{E}_K(\cdot,\cdot)} \text{ forges }]$. We stress that the nonce used in the forgery attempt may coincide with a nonce used in one of the adversary's queries.

Block ciphers and PRFs. A *function family* from n-bits to n-bits is a map $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite set of strings. It is a *block cipher* if each $E_K(\cdot) = E(K, \cdot)$ is a permutation. Let $\text{Rand}(n)$ denote the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$ and let $\text{Perm}(n)$ denote the set of all permutations from $\{0,1\}^n$ to $\{0,1\}^n$. These sets can be regarded as a function families by imagining that each member is specified by a string. Define

$$\begin{aligned}
\mathbf{Adv}_E^{\text{prf}}(A) &= \Pr[K \overset{R}{\leftarrow} \mathcal{K} \colon\ A^{E_K(\cdot)} = 1] - \Pr[\rho \overset{R}{\leftarrow} \text{Rand}(n) \colon\ A^{\rho(\cdot)} = 1] \\
\mathbf{Adv}_E^{\text{prp}}(A) &= \Pr[K \overset{R}{\leftarrow} \mathcal{K} \colon\ A^{E_K(\cdot)} = 1] - \Pr[\pi \overset{R}{\leftarrow} \text{Perm}(n) \colon\ A^{\pi(\cdot)} = 1] \\
\mathbf{Adv}_E^{\text{sprp}}(A) &= \Pr[K \overset{R}{\leftarrow} \mathcal{K} \colon\ A^{E_K(\cdot), E_K^{-1}(\cdot)} = 1] - \Pr[\pi \overset{R}{\leftarrow} \text{Perm}(n) \colon\ A^{\pi(\cdot), \pi^{-1}(\cdot)} = 1]
\end{aligned}$$

where $E_K^{-1}(Y)$ is the unique string X such that $E_K(X) = Y$.

5.2 Theorem Statements `[DRAFT]`

We give the following information-theoretic bounds on the security of OCB.

Theorem 1 [Authenticity] *Fix OCB parameters n and τ and let $\Pi = \text{OCB}[\text{Perm}(n), \tau]$. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has m blocks. Let $\bar\sigma = \sigma + 5q + 8m + 16$. Then*

$$\mathbf{Adv}_{\text{OCB}[\text{Perm}(n),\tau]}^{\text{auth}}(A) \ \le\ \frac{2.5\,\bar\sigma^2}{2^n} + \frac{1}{2^\tau}$$

In the theorem statement, and from now on, the aggregate length of queries $M_1, \ldots, M_q$ means the number $\sigma = \sum_{r=1}^{q} \|M_r\|_n$.

From the theorem above it is standard to pass to a complexity-theoretic analog, but in doing this one must be careful: one will need access to an E^{-1} oracle in order to verify a forgery attempt, which translates into needing the strong PRP assumption. One gets the following. Fix OCB parameters n and τ, and a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let $\Pi = \text{OCB}[E, \tau]$. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has m blocks. Let $\bar\sigma = \sigma + 5q + 8m + 16$. Let

$$\delta \ =\ \mathbf{Adv}_{\text{OCB}[\text{Perm}(n),\tau]}^{\text{auth}}(A) - \frac{2.5\,\bar\sigma^2}{2^n} - \frac{1}{2^\tau}$$

Then there is an adversary B for attacking block cipher E that achieves advantage $\mathbf{Adv}_E^{\text{sprp}}(B) \geq \delta$. Adversary B asks at most $q' = \sigma + 2q + m + 3$ oracle queries and has a running time which is equal to A's running time plus the time to compute E or E^{-1} at q' points plus additional time which is $cn\bar\sigma$, where the constant c depends only on details of the model of computation.

We now give the information-theoretic theorem describing the privacy of OCB.

Theorem 2 [Privacy] *Fix OCB parameters n and τ and let $\Pi = \mathrm{OCB}[\mathrm{Perm}(n), \tau]$. Let A be an adversary that asks q queries and then outputs a bit (indicating "real encryption oracle" or "fake encryption oracle"). Suppose the q queries have aggregate block length of σ. Let $\bar\sigma = \sigma + 2q + 1$. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \quad \leq \quad \frac{2.5\,\bar\sigma^2}{2^n}$$

From the theorem above it is standard to pass to a complexity-theoretic analog. One gets the following. Fix OCB parameters n and τ, and a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let $\Pi = \mathrm{OCB}[E, \tau]$. Let A be an adversary that asks q queries and then outputs a bit. Suppose the q queries have aggregate length of σ blocks. Let $\bar\sigma = \sigma + 2q + 1$. Let

$$\delta \quad = \quad \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) - \frac{2.5\,\bar\sigma^2}{2^n}$$

Then there is an adversary B for attacking block cipher E that achieves advantage $\mathbf{Adv}^{\mathrm{prp}}_{E}(B) \geq \delta$. Adversary B asks at most $q' = \sigma + 2q + 1$ oracle queries and has a running time which is equal to A's running time plus the time to compute E at q' points plus additional time which is $cn\bar\sigma$, where the constant c depends only on details of the model of computation.

5.3 Structure of the Proofs `[DRAFT]`

Our proof of Theorem 1 is based on three lemmas. The first, the "structure lemma", relates the authenticity of OCB to two functions: the MM-collision probability, denoted $\mathrm{MMcoll}(\cdot,\cdot)$, and the CM-collision probability, denoted $\mathrm{CMcoll}(\cdot,\cdot)$. We state this lemma and then explain its purpose and the two functions to which it refers.

Lemma 1 [Structure lemma] *Fix n and τ, and let $\Pi = \mathrm{OCB}[\mathrm{Perm}(n), \tau]$. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has m^* blocks. Let $\bar\sigma = \sigma + 2q + 8m^* + 16$. Let* MMcoll *and* CMcoll *be the MM- and CM-collision probabilities for* Π. *Then*

$$\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A) \quad \leq \quad \max_{\substack{m_1,\ldots,m_q \\ \sum m_i = \sigma}} \left\{ \sum_{1 \leq r < s \leq q} \mathrm{MMcoll}(m_r, m_s) + \sum_{1 \leq r \leq q} \mathrm{CMcoll}(m^*, m_r) \right\} + \frac{\bar\sigma^2}{2^{n+1}} + \frac{1}{2^\tau}$$

What this lemma does. The structure lemma provides a recipe for measuring the maximal forging probability of an adversary attacking the authenticity of OCB: compute the MM- and CM-collision probabilities, and then put them together using the formula of the lemma.

Informally, $\mathrm{MMcoll}(m, \bar m)$ measures the probability of running into trouble when the adversary asks some two particular oracle queries of the specified lengths. Trouble means the occurrence of any collision among the input values to the block cipher.

Informally, $\mathrm{CMcoll}(m, \bar m)$ measures the probability of running into trouble when the adversary tries to forge some particular ciphertext of the specified length, there having been an earlier query of some particular message of the specified length, it receiving some particular response. This time trouble basically refers to the final block-cipher input for the forgery attempt, $X[m+1]$, coinciding with some earlier block-cipher input.

The structure lemma simplifies the analysis of OCB in two ways. First, it allows one to excise adaptivity as a concern. Dealing with adaptivity is a major complicating factor in proofs of this type. Second, it allows one to concentrate on what happens to fixed pairs of messages. It is easier to think about what happens with two messages than what is happening with all $q+1$ of them.

The MM-collision probability. We next define the MM-collision probability, and then state our upper bound on it.

Definition 1 **[MM-collision probability]** *Let $M = M[0]\cdots M[m+1]$ be a string of $m+2$ n-bit blocks and let $\bar{M} = \bar{M}[0]\cdots\bar{M}[\bar{m}+1]$ be a string of $\bar{m}+2$ n-bit blocks where $M[0] \neq \bar{M}[0]$. Choose $L, R, \bar{R} \xleftarrow{R} \{0,1\}^n$. If any of the following $m+\bar{m}+5$ values coincide, we say that there is an MM-collision:*

$$\begin{array}{llll}
X[-1] & = 0^n & & \\
 & & & \\
X[0] & = M[0] \oplus L & \bar{X}[0] & = \bar{M}[0] \oplus L \\
X[1] & = M[1] \oplus \gamma_1 \cdot L \oplus R & \bar{X}[1] & = \bar{M}[1] \oplus \gamma_1 \cdot L \oplus \bar{R} \\
X[2] & = M[2] \oplus \gamma_2 \cdot L \oplus R & \bar{X}[2] & = \bar{M}[2] \oplus \gamma_2 \cdot L \oplus \bar{R} \\
\vdots & & \vdots & \\
X[m-1] & = M[m-1] \oplus \gamma_{m-1} \cdot L \oplus R & \bar{X}[\bar{m}-1] & = \bar{M}[\bar{m}-1] \oplus \gamma_{\bar{m}-1} \cdot L \oplus \bar{R} \\
X[m] & = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R & \bar{X}[\bar{m}] & = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R} \\
X[m+1] & = M[m+1] \oplus \gamma_m \cdot L \oplus R & \bar{X}[\bar{m}+1] & = \bar{M}[\bar{m}+1] \oplus \gamma_m \cdot L \oplus \bar{R}
\end{array}$$

The probability of such a collision is written $\mathrm{MMcoll}(M, \bar{M})$. *We let* $\mathrm{MMcoll}(m, \bar{m})$ *be the maximal value of* $\mathrm{MMcoll}(M, \bar{M})$ *when $M = M[0]\cdots M[m+1]$ has $m+2$ blocks and $\bar{M} = \bar{M}[0]\cdots\bar{M}[\bar{m}+1]$ has $\bar{m}+2$ blocks and $M[0] \neq \bar{M}[0]$.* $\diamondsuit$

Think of $M[0]$ as a synonym for the nonce N, think of of $M[m]$ as a generalization of $\mathrm{len}(M[m])$ (where the adversary can effectively control $M[m]$ as opposed to $\mathrm{len}(M[m])$ to influence $X[m]$), and think of $M[m+1]$ as a synonym for Checksum, which one likewise lets the adversary fully control. One similarly understands $\bar{M}[0]$, $\bar{M}[\bar{m}]$, and $\bar{M}[\bar{m}+1]$.

The needed bound is as follows.

Lemma 2 **[Bound on the MM-collision probability]**

$$\mathrm{MMcoll}(m, \bar{m}) \quad \le \quad \frac{(m+\bar{m}+5)^2}{2} \cdot \frac{1}{2^n}$$

The proof is easy, but it is deferred to Appendix A.2.

The CM-collision probability. The CM-collision probability is defined in Figure 3. The following lemma tells us how large it can possibly be.

Lemma 3 **[Bound on the CM-Collision probability]**

$$\mathrm{CMcoll}(m, \bar{m}) \quad \le \quad \frac{2m+3\bar{m}+9}{2^n}$$

```
10   $bad \leftarrow \texttt{false}$;   **for all** $x \in \{0,1\}^n$ **do** $\pi(x) \leftarrow \texttt{undefined}$
11   $L \xleftarrow{R} \{0,1\}^n$;   $\pi(0^n) \leftarrow L$

20   $\bar{X}[0] \leftarrow \bar{N} \oplus L$;   $\bar{Y}[0] \leftarrow \bar{R} \xleftarrow{R} \{0,1\}^n$
21   **for** $i \leftarrow 1$ **to** $\bar{m}$ **do** $\bar{Z}(i) \leftarrow \gamma_i \cdot L \oplus \bar{R}$
22   **for** $i \leftarrow 1$ **to** $\bar{m}-1$ **do** { $\bar{X}[i] \leftarrow \bar{M}[i] \oplus \bar{Z}[i]$;   $\bar{Y}[i] \leftarrow \bar{C}[i] \oplus \bar{Z}[i]$ }
23   $\bar{X}[\bar{m}] \leftarrow \mathrm{len}(\bar{M}[\bar{m}]) \oplus huge \cdot L \oplus \bar{Z}[\bar{m}]$ ;   $\bar{Y}(\bar{m}) \leftarrow \bar{C}[\bar{m}] \oplus \bar{M}[\bar{m}]\,0^*$
24   $\mathrm{Checksum}' \leftarrow \bar{M}[1] \oplus \cdots \oplus \bar{M}[\bar{m}-1] \oplus \bar{C}[\bar{m}]\,0^* \oplus \bar{Y}[\bar{m}]$
25   $\bar{X}[\bar{m}+1] \leftarrow \mathrm{Checksum}' \oplus \bar{Z}[m_r]$
26   **for** $i \leftarrow 0$ **to** $m_r$ **do** $\pi(\bar{X}[i]) \leftarrow \bar{Y}[i]$

30   $X[0] \leftarrow N \oplus L$
31   **if** $N \neq \bar{N}$ **and** $X[0] \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \texttt{true}$
32   **if** $N = \bar{N}$ **then** $R \leftarrow \bar{R}$ **else** $R \xleftarrow{R} \{0,1\}^n$
33   $\pi(X[0]) \leftarrow R$
34   **for** $i \leftarrow 1$ **to** $m$ **do** $Z[i] \leftarrow \gamma_i \cdot L \oplus R$
35   **for** $i \leftarrow 1$ **to** $m-1$ **do** {
36        $Y[i] \leftarrow C[i] \oplus Z[i]$
37        **if** $Y[i] \in \mathrm{Range}(\pi)$ **then** $X[i] \leftarrow \pi^{-1}(Y[i])$ **else** $X[i] \xleftarrow{R} \{0,1\}^n$
38        $\pi(X[i]) \leftarrow Y[i]$;   $M[i] \leftarrow X[i] \oplus Z[i]$ }
39   $X[m] \leftarrow \mathrm{len}(C[m]) \oplus huge \cdot L \oplus Z[m]$
40   **if** $X[m] \in \mathrm{Domain}(\pi)$ **then** $Y[m] \leftarrow \pi(X[m])$ **else** $Y[m] \xleftarrow{R} \{0,1\}^n$
41   $\pi(X[m]) \leftarrow Y[m]$
42   $\mathrm{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
43   $X[m+1] \leftarrow \mathrm{Checksum} \oplus Z[m]$
44   **if** $X[m+1] \in \mathrm{Domain}(\pi)$ **then** $bad \leftarrow \texttt{true}$
```

Figure 3: **Defining the CM-Collision probability.** The function $\mathrm{CMcoll}(\bar{N}, \bar{M}, \bar{C},\ N, C)$ is defined as the probaiblity that *bad* gets set to `true` when executing this game. The value $\mathrm{MMcoll}(m, \bar{m})$ is the maximal value of $\mathrm{CMcoll}(\bar{N}, \bar{M}, \bar{C},\ N, C)$ over all $\bar{m}$-block $\bar{M}$,$\bar{C}$ and all m-block C.

The proof is mostly a case analysis. It is given in Appendix A.3

Concluding the authenticity theorem. To prove Theorem 1, combine lemmas 1, 2, and 3. Given the composite block length σ, one must bound the maximum possible value of

$$\mathbf{Adv}_{\Pi}^{\mathrm{auth}}(A) \quad \leq \quad \sum_{1 \leq r < s \leq q} \frac{(m_r + m_s + 5)^2}{2^n} + \sum_{1 \leq r \leq q} \left(\frac{2m + 3m_r + 9}{2^n} \right) + \frac{(\sigma + 2q + 8m + 16)^2}{2^{n+1}} + \frac{1}{2^\tau}$$

subject to the constraint that $\sum_{i=1}^{q} m_i = \sigma$. One can bound this by letting each m_i be the (possibly non-integral) value σ/q, for which one gets

$$\begin{aligned}
\mathbf{Adv}_{\Pi}^{\mathrm{auth}}(A) \quad &\leq \quad \binom{q}{2} \cdot \frac{(2\sigma/q + 5)^2}{2^n} + q \cdot \frac{2m + 3\sigma/q + 9}{2^n} + \frac{(\sigma + 2q + 8m + 16)^2}{2^{n+1}} + \frac{1}{2^\tau} \\
&\leq \quad \frac{2\sigma^2 + 10\sigma q + 12.5q^2 + 2mq + 3\sigma + 9q}{2^n} + \frac{(\sigma + 2q + 8m + 16)^2}{2^{n+1}} + \frac{1}{2^\tau} \\
&\leq \quad \frac{2(\sigma^2 + 5\sigma q + 6.25q^2 + mq + 1.5\sigma + 4.5q)}{2^n} + \frac{(\sigma + 2q + 8m + 16)^2}{2^{n+1}} + \frac{1}{2^\tau} \\
&\leq \quad \frac{2(\sigma + 2.5q + 2.5q + 0.5m + 0.75 + 2.25)^2}{2^n} + \frac{(\sigma + 2q + 8m + 16)^2}{2^{n+1}} + \frac{1}{2^\tau}
\end{aligned}$$

$$\begin{aligned}
&\leq \frac{2(\sigma+5q+0.5m+3)^2}{2^n} + \frac{(\sigma+2q+8m+16)^2}{2^{n+1}} + \frac{1}{2^\tau} \\
&\leq \frac{2(\sigma+5q+8m+16)^2}{2^n} + \frac{(\sigma+5q+8m+16)^2}{2^{n+1}} + \frac{1}{2^\tau} \\
&\leq \frac{2.5\,\bar{\sigma}^2}{2^n} + \frac{1}{2^\tau}
\end{aligned}$$

where $\bar{\sigma} = \sigma + 5q + 8m + 16$. This completes the theorem. ▮

Privacy. Privacy is obtained rather easily en route to proving authenticity. The is because of the following result, which closely follows the first half of the proof of the structure lemma.

Lemma 4 [Privacy lemma] *Fix OCB parameters n and τ and let* $\Pi = \mathrm{OCB}[\mathrm{Perm}(n), \tau]$. *Let A be an adversary that asks q queries, these having aggregate block length of σ blocks. Let* MMcoll *be the MM-collision probability for* Π. *Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \;\leq\; \frac{(\sigma+2q+1)^2}{2^{n+1}} + \max_{\substack{m_1,\ldots,m_q \\ \sum m_i = \sigma}} \left\{ \sum_{1 \leq r < s \leq q} \mathrm{MMcoll}(m_r, m_s) \right\}$$

The proof is in Appendix A.4.

Combining Lemmas 2 and 4 we conclude Theorem 2. Namely,

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) &\leq \frac{0.5(\sigma+2q+1)^2}{2^n} + \max_{\substack{m_1,\ldots,m_q \\ \sum m_i = \sigma}} \left\{ \sum_{1 \leq r < s \leq q} \mathrm{MMcoll}(m_r, m_s) \right\} \\
&\leq \frac{0.5(\sigma+2q+1)^2}{2^n} + \max_{\substack{m_1,\ldots,m_q \\ \sum m_i = \sigma}} \left\{ \sum_{1 \leq r < s \leq q} \frac{0.5(m_r+m_s+5)^2}{2^n} \right\}
\end{aligned}$$

can be bounded by setting each $m_i = \sigma/q$. This gives

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) &\leq \frac{0.5(\sigma+2q+1)^2 + 0.5q^2(2\sigma/q+5)^2}{2^n} \\
&\leq \frac{0.5(\sigma+2q+1)^2 + 0.5(2\sigma+5q)^2}{2^n} \\
&\leq \frac{0.5(\sigma+2q+1)^2 + 2(\sigma+2.5q)^2}{2^n} \\
&\leq \frac{2.5(\sigma+2q+1)^2}{2^n} \\
&\leq \frac{2.5\,\bar{\sigma}^2}{2^n}
\end{aligned}$$

which completes the proof of the privacy theorem.

6 Performance

Abstract accounting. OCB uses $\lceil |M|/n \rceil + 2$ block-cipher calls for a nonempty message M. (The empty string takes three block-cipher invocations, the same as a one-block message). We compare with CBC encryption and CBC encryption plus a CBC MAC:

Algorithm	16 B	128 B	2 KB
OCB	**56.2**	**23.6**	**19.2**
CBC encrypt	34.9	19.8	17.6
CBC MAC	18.9	17.4	17.1
CBC encrypt + CBC MAC	53.3	36.7	34.3
CBC encrypt + HMAC	144.0	46.7	31.5

Figure 4: **Performance results** (in cycles per byte, on a Pentium 3) across three message lengths, assuming AES-128 as the underlying block cipher.

- "Basic" CBC encryption, where one assumes a random IV and a message which is a multiple of the block length, uses two fewer block-cipher calls—a total of $|M|/n$.
- A more fair comparison would be to set IV $= E_K(N)$ for CBC encryption, so that both schemes use a (not necessarily random) nonce, and to use obligatory 10^* padding, so that both schemes can handle arbitrary strings. This would bring the total for CBC to $\lceil(|M|+1)/n\rceil + 1$ block-cipher calls, coinciding with OCB in the case that $|M|$ is a multiple of the block length, and using one fewer block-cipher call otherwise.
- If one combines the basic CBC encryption with a MAC, say MACing the ciphertext, then the CBC-encryption will use a number of block-cipher calls as just discussed, while the CBC MAC will use between $\lceil|M|/n\rceil + 1$ and $\lceil(|M|+1)/n\rceil + 3$ block-cipher calls, depending on padding conventions and the optional processing done to the final block in order to ensure security across messages of varying lengths. So the total will be as few as $2\lceil|M|/n\rceil + 1$ or as much as $2\lceil(|M|+1)/n\rceil + 4$ block-cipher calls. Thus OCB saves between $\lceil|M|/n\rceil - 1$ and $\lceil|M|/n\rceil + 3$ block-cipher calls compared to separate CBC encryption and CBC MAC computation

As with any mode, there is further overhead beyond the block-cipher calls. Per block, this overhead is about four n-bit xor operations, plus associated logic. The work for this associated logic will vary according to whether or not one precomputed $L(i)$-values, and many additional details.

Though some of the needed $L(i)$-values are likely to be pre-computed, computing all of them "on the fly" is not inefficient. Starting with 0^n we form successive offsets by xoring the previous offset with L, $2 \cdot L$, L, $4 \cdot L$, L, $2 \cdot L$, L, $8 \cdot L$, and so forth. So half the time we use L itself; a quarter of the time we use $2 \cdot L$; one eighth of the time we use $4 \cdot L$; and so forth. Thus the expected number of times to multiply by $\mathtt{x}$ in order to compute an offset is at most $\sum_{i=1}^{\infty} i/2^{i+1} = 1$. Each $a \cdot \mathtt{x}$ instruction requires an n-bit xor and a conditional 32-bit xor. Said differently, for any $m > 0$, the total number of $a \cdot \mathtt{x}$ operations needed to compute $\gamma_1 \cdot L$, $\gamma_2 \cdot L$, $\ldots$, $\gamma_m \cdot L$ is $\sum_{i=1}^{m} \mathsf{ntz}(i)$, which is less than m. The above assumed that one does not retain or precompute any $L(i)$ value beyond $L = L(0)$. Suppose that one retains three extra $L(i)$ values, precomputing and storing $L(0), L(1), L(2), L(3)$. Computing and storing these three extra values is cheaper than computing $L = L(0)$ itself, which required an application of E_K. But now the desired multiple of L has already been computed $1/2 + 1/4 + 1/8 + 1/16 \approx 94\%$ of the time. When it has not been pre-computed it must be calculated, starting from $L(3)$, so the amortized number of multiplications by $\mathtt{x}$ has thus been reduced from 1 to $\sum_{i=1}^{\infty} = i/2^{i+4} = 0.125$.

Experimental results. We compared OCB-AES-128 performance with four conventional algorithms, as shown in Table 4. By CBC encrypt we mean CBC encryption with an IV that is $E_K(N)$. By CBC MAC we mean the "basic" CBC MAC—nothing extra done to take care of length-variability. By CBC encrypt + CBC MAC we mean to encrypt the message (this time using basic CBC encryption with a fixed IV) and then MAC the ciphertext (including the IV). By HMAC we mean HMAC-SHA1. The CBC variants did not check for the need for padding and did not pad. The code was written in assembly except for HMAC-SHA1, which was written in a combination of C and assembly. The OS was Windows 2000 sp1 and the compiler was Visual C++ 6.0 sp4. All data fit into L1 cache.

Some aspects of the experiments above are unfavorable to OCB, making the performance estimates conservative. In particular, the "raw" CBC MAC (as used above) needs to be modified to correctly handle length-variability, and doing so is normally done in a way that results in additional block-cipher calls. Focusing on the last column, one sees that OCB incurs about 9% overhead compared to CBC encryption, and that the algorithm takes about 56% of the time of a CBC encryption + CBC MAC.

Perhaps a better way to look at the overhead attributable to OCB is to use the identity function as the underlying cipher, which effectively measures the work attributable to the mode itself. For 2KByte messages and a null block cipher the cost was 2.43 cpb. This compares with 1.22 cpb for CBC encryption. Thus the "extra work" for OCB/xor compared to CBC encryption is about 1.21 cpb.

We emphasize that these results are for a serial execution environment with a limited number of registers. In an environment with plenty of registers and multiple instruction pipes, OCB, properly implemented, will of course be faster than CBC.

7 Intellectual Property Statement and Disclosures

Patent applications covering the ideas of this proposal were filed (Rogaway as inventor) on 13 September 2000, 12 October 2000, and 9 February 2001.

The inventor hereby releases IP rights covering OCB for all non-commercial, non-governmental applications. For commercial applications, the inventor will license OCB under a non-exclusive license, on a non-discriminatory basis, based on reasonable terms and conditions.

IBM indicates that they filed a patent application on Jutla's work on 14 April 2000. Virgil Gligor indicates that he filed patent applications on 31 January 2000, 31 March 2000, and 24 August 2000.

The IP status specified in this proposal will be updated when more is known.

Acknowledgments

Virgil Gligor described [11] to Rogaway at CRYPTO '00, and then Charanjit Jutla gave a rump-session talk on [16] at the same conference. These events inspired this work.

The October 2000 modes-of-operation workshop, organized by NIST and run by Elaine Barker, was the motivation for putting together a first version of OCB. The current submission is responsive to a second call by NIST for submissions dealing with block-cipher modes of operation. Elaine Barker, Morris Dworkin, and Jim Foti are among those at NIST involved. We are pleased that NIST has taken the lead in promoting the development of modern modes of operation.

We have received useful feedback from Michael Amling, Johan Håstad, Hugo Krawczyk, David McGrew, Silvio Micali, Alberto Pascual, Tom Shrimpton, and David Wagner.

This submission was prepared while Rogaway was on leave of absence from UC Davis, visiting the Department of Computer Science, Faculty of Science, Chiang Mai University. No external funding has been used for this particular piece of research, but the submitter gratefully acknowledges a recent gift by Cisco Systems. Many thanks to Cisco for their support.

References

[1] J. An and M. Bellare. Does encryption with redundancy provide authenticity? To appear in *Eurocrypt 2001*.

[2] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. *Proceedings of 38th Annual Symposium on Foundations of Computer Science* (FOCS 97), IEEE, 1997. `http://www.cs.ucdavis.edu/~rogaway/`

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – CRYPTO '98.* Lecture Notes in Computer Science, Vol. 1462, H. Krawczyk, ed., Springer-Verlag. `http://www.cs.ucdavis.edu/~rogaway/`

[4] M. Bellare, R. Guérin and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions." *Advances in Cryptology – CRYPTO '95.* Lecture Notes in Computer Science Vol. 963, Springer-Verlag, D. Coppersmith, Ed., pp. 15–28, 1995. `http://www.cs.ucdavis.edu/~rogaway/`

[5] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, vol. 61, no. 3, Dec 2000. (Full version of paper from *Advances in Cryptology – CRYPTO '94.* Lecture Notes in Computer Science, vol. 839, pp. 340–358, 1994.) `http://www.cs.ucdavis.edu/~rogaway/`

[6] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology – ASIACRYPT '00.* Lecture Notes in Computer Science, T. Okamoto., ed., Springer-Verlag, 2000. `http://www-cse.ucsd.edu/users/mihir/`

[7] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient encryption. *Advances in Cryptology – ASIACRYPT '00.* Lecture Notes in Computer Science, T. Okamoto., ed., Springer-Verlag, 2000. `http://www.cs.ucdavis.edu/~rogaway/`

[8] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS #1. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science, vol. 1462, pp. 1–12, 1998. `http://www.bell-labs.com/user/bleichen/`

[9] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. on Computing*, vol. 30, num. 2, 391–437, 2000. Earlier version in *STOC 91.* `http://www.wisdom.weizmann.ac.il/~naor/`

[10] V. Gligor and P. Donescu. Integrity Aware PCBC Encryption. Security Protocols, 7th International Workshop. Lecture notes in Computer Science, vol. 1796, Springer-Verlag, pp. 153–171, 2000.

[11] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Manuscript. August 18, 2000. `http://www.eng.umd.edu/~gligor/`

[12] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Manuscript. October 2000. `http://csrc.nist.gov/encryption/aes/modes/`

[13] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM,* vol. 33, no. 4, pp. 210–217, 1986.

[14] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, vol. 28, April 1984, pp. 270–299.

[15] S. Halevi. An observation regarding Jutla's modes of operation. Manuscript. February 22, 2001. `http://eprint.iacr.org/` (reference number 2001/015).

[16] C. Jutla. Encryption modes with almost free message integrity. Manuscript. August 1, 2000. `http://eprint.iacr.org/` (reference number 2000/039). Proceedings version to appear in Eurocrypt 2001.

[17] C. Jutla. Encryption modes with almost free message integrity. Contribution to NIST based on the above. October 2000. `http://csrc.nist.gov/encryption/modes/workshop1/`

[18] J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption '00.* Lecture Notes in Computer Science, B. Schneier, ed., 2000.

[19] M. Luby and C. Rackoff. How to construct pseudororandom permutations from pseudorandom functions. *SIAM J. Computation*, vol. 17, no. 2, April 1988.

[20] RSA Laboratories. PKCS #1: RSA encryption standard, Version 1.5, November 1993; and PKCS #1: RSA cryptography specifications, Version 2.0, September 1998, B. Kaliski and J. Straddon. `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/`

[21] B. Preneel. Cryptographic primitives for information authentication — State of the art. *State of the Art in Applied Cryptography*, COSIC '97, LNCS 1528, B. Preneel and V. Rijmen, eds., Springer-Verlag, pp. 49–104, 1998.

[22] P. Rogaway. OCB Mode: Parallelizable authenticated encryption. Contribution to NIST. October 16, 2000. Earlier version of the present paper. `http://csrc.nist.gov/encryption/modes/workshop1/`

[23] US National Bureau of Standards. DES modes of operation. Federal Information Processing Standard (FIPS) Publication 81, December 1980. Available as `http://www.itl.nist.gov/fipspubs/fip81.htm`

[24] US National Institute of Standards. Specification for the Advanced Encryption Standard (AES). Draft of a Federal Information Processing Standards (FIPS), February 28, 2001. Based on: J. Daemen and V. Rijmen, AES Proposal: Rijndael. September 3, 1999. `http://www.nist.gov/aes/`

A Proofs `[DRAFT]`

A.1 Proof of the Structure Lemma (Lemma 1)

Let A be a (computationally unbounded) adversary that attempts to violate the authenticity of $\mathrm{OCB}[\mathrm{Perm}(n),\tau]$. Without loss of generality, A is deterministic. The adversary is given an oracle for $\mathrm{OCB.Enc}_\pi(\cdot,\cdot)$. We must bound the probability that A, after adaptively using this oracle q times, on messages with aggregate length σ blocks, produces a properly forged ciphertext having at most m^* blocks. This forgery probability is $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A)$.

Game A. One can conceive of A interacting with $\mathrm{OCB.Enc}_\pi(\cdot,\cdot)$ and then producing a forgery attempt as A playing a certain game, game A, as defined in Figures 5 and 6. Rather than choose $\pi \overset{R}{\leftarrow} \mathrm{Perm}(n)$ all at once, this game defines the values of $\pi(x)$ point-by-point, as needed. We use the notation $\mathrm{Domain}(\pi)$ for the set of values $x \in \{0,1\}^n$ such that $\pi(x) \neq$ `undefined`. By $\overline{\mathrm{Domain}}(\pi)$

```
**Initialization:**
01    $bad \leftarrow \texttt{false}$;   **for all** $x \in \{0,1\}^n$ **do** $\pi(x) \leftarrow \texttt{undefined}$
02    $L \xleftarrow{R} \{0,1\}^n$;   $\pi(0^n) \leftarrow L$

**When** $A$ **asks query** $(N, M)$:        //*q such queries will be asked*
10    Partition $M$ into blocks $M[1] \cdots M[m]$
11    $X[0] \leftarrow N \oplus L$;   $Y[0] \xleftarrow{R} \{0,1\}^n$
12    **if** $X[0] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow \texttt{true}$;   $Y[0] \leftarrow \pi(X[0])$ } **else**
13    **if** $Y[0] \in \text{Range}(\pi)$ **then** { $bad \leftarrow \texttt{true}$;   $Y[0] \xleftarrow{R} \overline{\text{Range}}(\pi)$ }
14    $\pi(X[0]) \leftarrow Y[0]$

15    **for** $i \leftarrow 1$ **to** $m$ **do** $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
16    **for** $i \leftarrow 1$ **to** $m-1$ **do** {
17          $X[i] \leftarrow M[i] \oplus Z[i]$;   $Y[i] \xleftarrow{R} \{0,1\}^n$
18          **if** $X[i] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow \texttt{true}$;   $Y[i] \leftarrow \pi(X[i])$ } **else**
19          **if** $Y[i] \in \text{Range}(\pi)$ **then** { $bad \leftarrow \texttt{true}$; $Y[i] \xleftarrow{R} \overline{\text{Range}}(\pi)$ }
20          $\pi(X[i]) \leftarrow Y[i]$;   $C[i] \leftarrow Y[i] \oplus Z[i]$ }

21    $X[m] \leftarrow \text{len}(M[m]) \oplus huge \cdot L \oplus Z[m]$;   $Y[m] \xleftarrow{R} \{0,1\}^n$
22    **if** $X[m] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow \texttt{true}$;   $Y[m] \leftarrow \pi(X[m])$ } **else**
23    **if** $Y[m] \in \text{Range}(\pi)$ **then** { $bad \leftarrow \texttt{true}$;   $Y[m] \xleftarrow{R} \overline{\text{Range}}(\pi)$ }
24    $\pi(X[m]) \leftarrow Y[m]$;   $C[m] \leftarrow M[m] \oplus Y[m]$

25    $\text{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
26    $X[m+1] \leftarrow \text{Checksum} \oplus Z[m]$;   $Y[m+1] \xleftarrow{R} \{0,1\}^n$
27    **if** $X[m+1] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow \texttt{true}$;   $Y[m+1] \leftarrow \pi(X[m+1])$ } **else**
28    **if** $Y[m+1] \in \text{Range}(\pi)$ **then** { $bad \leftarrow \texttt{true}$;   $Y[m+1] \xleftarrow{R} \overline{\text{Range}}(\pi)$ }
29    $\pi(X[m+1]) \leftarrow Y[m+1]$;   $T \leftarrow Y[m+1]$ [first $\tau$ bits]
30    **return** $\mathcal{C} \leftarrow C[1] \cdots C[m]\, T$
```

Figure 5: **Game A, part 1.** This game provides adversary A a perfect simulation of OCB[Perm(n), τ].

we mean $\{0,1\}^n \setminus \text{Domain}(\pi)$. Similarly, $\text{Range}(\pi)$ is the set of $y \in \{0,1\}^n$ such that there exists an $x \in \{0,1\}^n$ for which $\pi(x) = y$. And $\overline{\text{Range}}(\pi) = \{0,1\}^n \setminus \text{Range}(\pi)$.

An inspection of game A makes clear that it supplies to A a perfect simulation of $\text{OCB.Enc}_\pi(\cdot,\cdot)$. Game A simulates OCB in a somewhat unusual way, not only defining π point-by-point, but, when a value $\pi(x)$ is needed, for some new x, we get this value, in most cases, not by choosing $y \xleftarrow{R} \overline{\text{Range}}(\pi)$, as would seem natural, but by choosing $y \xleftarrow{R} \{0,1\}^n$, setting $\pi(x)$ to y if y is not already in the range of π, and "changing our minds," setting $\pi(x) \xleftarrow{R} \overline{\text{Range}}(\pi)$, otherwise. In the latter case, a flag *bad* is set to `true`. The flag *bad* is also set to `true` when the adversary successfully forges. Consequently, upperbounding the probability that *bad* gets set to `true` in game A serves to upperbound the adversary's forging probability.

Game A$'$. We begin by making a couple of quite trivial changes to game A. First, instead of setting $C[m] = M[m] \oplus Y[m]$ (in line 24 of game A), we set $C[m] = M[m]\,0^* \oplus Y[m]$, instead. That is, we imagine returning the "full" final-ciphertext-block instead of the truncated final-ciphertext-block. Clearly the extra bits given to the adverary can not make worse an optimal adversary's chance of successful forgery. Second, instead of returning (in line 30 of game A) a tag T which is the first τ bits of $Y[m+1]$, we return the full tag, $Y[m+1]$. Once again, the extra bits provided to the

```
When $A$ makes forgery attempt $(N,\ \mathcal{C})$:
50    Partition $\mathcal{C}$ into $C[1] \cdots C[m]\, T$
51    $X[0] \leftarrow N \oplus L$;   if $X[0] \in \mathrm{Domain}(\pi)$ then $Y[0] \leftarrow \pi(X[0])$ else $Y[0] \xleftarrow{R} \overline{\mathrm{Range}}(\pi)$
52    $\pi(X[0]) \leftarrow Y[0]$
53    for $i \leftarrow 1$ to $m$ do $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
54    for $i \leftarrow 1$ to $m-1$ do {
55        $Y[i] \leftarrow C[i] \oplus Z[i]$
56        if $Y[i] \in \mathrm{Range}(\pi)$ then $X[i] \leftarrow \pi^{-1}(Y[i])$ else $X[i] \xleftarrow{R} \overline{\mathrm{Domain}}(\pi)$
57        $\pi(X[i]) \leftarrow Y[i]$;   $M[i] \leftarrow X[i] \oplus Z[i]$ }

58    $X[m] \leftarrow \mathrm{len}(C[m]) \oplus \mathit{huge} \cdot L \oplus Z[m]$
59    if $X[m] \in \mathrm{Domain}(\pi)$ then $Y[m] \leftarrow \pi(X[m])$ else $Y[m] \xleftarrow{R} \overline{\mathrm{Range}}(\pi)$
60    $\pi(X[m]) \leftarrow Y[m]$
61    Checksum $\leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\, 0^* \oplus Y[m]$
62    $X[m+1] \leftarrow$ Checksum $\oplus Z[m]$
63    if $X[m+1] \in \mathrm{Domain}(\pi)$ then $Y[m+1] \leftarrow \pi(X[m])$ else $Y[m+1] \xleftarrow{R} \overline{\mathrm{Range}}(\pi)$
64    $T' \leftarrow Y[m+1]$ [first $\tau$ bits]
65    if $T = T'$ then $\mathit{bad} \leftarrow$ true
```

Figure 6: **Games A, A′ B, B′, and C, part 2.**

adverary can only improve an optimal adversary's chance of success. Let game A′ denote this new, "easier" game. We will bound the probability that *bad* gets set to `true` in game A′.

Game B. Next we eliminate from game A′ the statement which immediately follows *bad* being set to `true` in each of lines 12, 13, 18, 19, 22, 23, 27, 28. The **else** statements are also eliminated. This new game, game B, is shown in Figure 7. This new game is different from game A′, and an adversary A having queries answered according to game B will not be seeing the same view as one whose queries are answered according to A′. Still, game B has been constructed so that it behaves identically to game A′ until the flag *bad* is set to `true`. Only at that point do the two games diverge. As a consequence, regardless of the behavior of A, the probaiblity that *bad* will get set to `true` when A plays game B is identical to the probability that *bad* gets set to `true` when A plays game A′. Now we are interested in upperbounding the probability of forgery in game A, which we do by upperbound the probability that *bad* gets set to `true` in game A′, which is just the probability that *bad* gets set to `true` in game B.

Note that we are not claiming that the probability of the adversary forging in game B (meaning that *bad* gets set to `true` at line 65 of game B) is the same as the probability of the adversary forging in A′ (meaning that *bad* gets set to `true` in the last line of that game). Claims of this sort are tempting to make, but they are untrue.

Bounding Y-collisions in Game B. We next bound the probability that *bad* will be set to `true` in any of lines 13, 19, 23, or 28 of game B. In each of these lines, a random n-bit string was just chosen and then it is tested for membership in the growing set Range(π). In the course of game B the size Range(π) starts off at 0 and then grows one element at a time until it reaches a final size of $\sigma + 2q + 1$ elements. Therefore the probability that, in growing Range(π), there is a repetition as we add in random points is at most $(1 + 2 + \cdots + \sigma + 2q)/2^n \le (\sigma + 2q + 1)^2/2^{n+1}$. We note this for future reference:

$$\Pr[A \text{ causes } \mathit{bad} \text{ to be set in any of lines 13, 19, 23 or 28 of game B}] \;\le\; \frac{(\sigma + 2q + 1)^2}{2^{n+1}} \qquad (1)$$

```
Initialization:
01    $bad \leftarrow$ false;   for all $x \in \{0,1\}^n$ do $\pi(x) \leftarrow$ undefined
02    $L \xleftarrow{R} \{0,1\}^n$;   $\pi(0^n) \leftarrow L$

When $A$ asks query $(N, M)$:        //$q$ such queries will be asked
10    Partition $M$ into blocks $M[1] \cdots M[m]$
11    $X[0] \leftarrow N \oplus L$;   $Y[0] \xleftarrow{R} \{0,1\}^n$
12    if $X[0] \in \text{Domain}(\pi)$ then $bad \leftarrow$ true
13    if $Y[0] \in \text{Range}(\pi)$ then $bad \leftarrow$ true
14    $\pi(X[0]) \leftarrow Y[0]$
15    for $i \leftarrow 1$ to $m$ do $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
16    for $i \leftarrow 1$ to $m-1$ do {
17          $X[i] \leftarrow M[i] \oplus Z[i]$;   $Y[i] \xleftarrow{R} \{0,1\}^n$
18          if $X[i] \in \text{Domain}(\pi)$ then $bad \leftarrow$ true
19          if $Y[i] \in \text{Range}(\pi)$ then $bad \leftarrow$ true
20          $\pi(X[i]) \leftarrow Y[i]$;   $C[i] \leftarrow Y[i] \oplus Z[i]$ }
21    $X[m] \leftarrow \text{len}(M[m]) \oplus huge \cdot L \oplus Z[m]$;   $Y[m] \xleftarrow{R} \{0,1\}^n$
22    if $X[m] \in \text{Domain}(\pi)$ then $bad \leftarrow$ true
23    if $Y[m] \in \text{Range}(\pi)$ then $bad \leftarrow$ true
24    $\pi(X[m]) \leftarrow Y[m]$;   $C[m] \leftarrow M[m]\,0^* \oplus Y[m]$
25    $\text{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
26    $X[m+1] \leftarrow \text{Checksum} \oplus Z[m]$;   $Y[m+1] \xleftarrow{R} \{0,1\}^n$
27    if $X[m+1] \in \text{Domain}(\pi)$ then $bad \leftarrow$ true
28    if $Y[m+1] \in \text{Range}(\pi)$ then $bad \leftarrow$ true
29    $\pi(X[m+1]) \leftarrow Y[m+1]$
30    return $\mathcal{C} \leftarrow C[1] \cdots C[m]\; Y[m+1]$
```

Figure 7: **Game B, part 1.**

Having bounded the probability that *bad* will be set in the four indicated lines, we may imagine eliminating these four lines, forming a new game, game B′. The probability that *bad* is set in game B is at most the computed bound more than than the probability that *bad* is set in game B′. Thus we may continue the analysis using game B′ as long as we compensate the final bound by adding in the term given by Equation (1).

Game C. In game B′, consider the distribution strings returned to the adversary in response to a query (N, M), where $m = \|M\|_n$. The adversary receives a ciphertext $\mathcal{C} = C[1] \cdots C[m-1]C[m]\; Y[m+1]$. Since each block of this string is a uniform random value xor'ed with some other, independent value, we have that $\mathcal{C}$ is uniformly distributed and independent of the query M, apart from its length. As a consequence, when a query of N, M is made, where M has m blocks, we can return a random answer $\mathcal{C}$ (of $nm + n$ bits) and do no more at that time. Later, when the adversary is done making its q queries, we can set the remaining random values, make the associated assignments to π, and set the flag *bad*, as appropriate. This is what has been done in Game C of Figure 8. From the adversary's point of view, game B′ and game C are identical. Furthermore, the probability that *bad* gets set to `true` is identical in the two games.

Game D. We have reduced the problem of upperbounding the forging probability to the problem of upperbounding the probability that *bad* gets set to `true` in game C. This probability is over the coins used in line 11 of game C (which defines the $\mathcal{C}_r$-values) and over the additional coins used subsequently in the program. We must show that, over this sequence of coins (remember that the

```
When A asks its r-th query, $(N_r,\ M_r)$:        //r will range from 1 to q
10      Partition $M_r$ into blocks $M_r[1]\cdots M_r[m_r]$
11      $C_r[1],\ldots,C_r[m_r],\ Y_r[m+1] \xleftarrow{R} \{0,1\}^n$
12      return $\mathcal{C}_r \leftarrow C_r[1]\cdots C_r[m_r]\ Y_r[m_r+1]$

When A is done making oracle queries:
20      $bad \leftarrow$ false;   for all $x \in \{0,1\}^n$ do $\pi(x) \leftarrow$ undefined
21      $L \xleftarrow{R} \{0,1\}^n$;   $\pi(0^n) \leftarrow L$

30      for $r \leftarrow 1$ to $q$ do {
31          $X_r[0] \leftarrow N_r \oplus L$;   $Y_r[0] \xleftarrow{R} \{0,1\}^n$
32          for $i \leftarrow 1$ to $m_r$ do $Z_r[i] \leftarrow \gamma_i \cdot L \oplus Y_r[0]$
33          for $i \leftarrow 1$ to $m_r - 1$ do { $X_r[i] \leftarrow M_r[i] \oplus Z_r[i]$;   $Y_r[i] \leftarrow C_r[i] \oplus Z_r[i]$ }
34          $X_r[m_r] \leftarrow \mathrm{len}(M[m_r]) \oplus huge \cdot L \oplus Z_r[m_r]$ ;   $Y_r[m_r] \leftarrow C_r[m_r] \oplus M_r[m_r]\,0^*$
35          $\mathrm{Checksum}_r \leftarrow M_r[1] \oplus \cdots \oplus M_r[m_r-1] \oplus C_r[m_r]\,0^* \oplus Y_r[m_r]$
36          $X_r[m_r+1] \leftarrow \mathrm{Checksum}_r \oplus Z_r[m_r]$ }

37      $\mathcal{X} \leftarrow (X_1[0], X_1[1], \ldots, X_1[m_1+1],\ \ldots,\ X_q[0], X_q[1], \ldots, X_q[m_q+1])$
38      $\mathcal{Y} \leftarrow (Y_1[0], Y_1[1], \ldots, Y_1[m_1+1],\ \ldots,\ Y_q[0], Y_q[1], \ldots, Y_q[m_q+1])$
39      if some string is repeated in $\mathcal{X} \cup \{0^n\}$ then $bad \leftarrow$ true
40      for $i \leftarrow 1$ to $|\mathcal{X}|$ do $\pi(\mathcal{X}[i]) \leftarrow \mathcal{Y}[i]$
```

Figure 8: **Game C, part 1.** This game provides adversary A with the same view as game B, and sets *bad* with the same probability. But it defers some random choices.

adversary is deterministic) the flag *bad* is rarely set.

We will show something stronger: that even if one fixes all of the coins used in line 11 (the $\mathcal{C}_r$-values) and takes the probability over just the remaining coins, still the probability that *bad* gets set to `true` is small. The virtue of ths change is that it effectively eliminates the q interactive queries from the game. Namely, since the adversary A is deterministic and each response $\mathcal{C}_r$ has been fixed, the adversary can be imagined to "know" all of the queries $N_1, M_1,\ \ldots,\ N_q, M_q$ that it would ask and all of the answers $\mathcal{C}_1, \ldots, \mathcal{C}_q$ that it would receive. All the adversary has left to do is to output the forgery attempt $(N, C\ T)$. This value too is now pre-determined, as our adversary is deterministic. So the adversary is effectively gone, and we are left to claim that for any $N_1, M_1, \ldots, N_q, M_q,\ \mathcal{C}_1, \ldots, \mathcal{C}_q,\ N, C, T$, the flag *bad* will rarely be set if we run game C starting at line 20. The new game is called game D. It depends on $N_1, M_1, \ldots, N_q, M_q,\ \mathcal{C}_1, \ldots, \mathcal{C}_q,\ N, C, T$, which are now just constants. The constants are not quite arbitrary: the N_r are still required to be distinct. The lengths of $M_1, \ldots, M_q$ are $m_1, \ldots, m_q$ blocks. The length of C is $m = m^*$ blocks (at this point we start to use m as a synonym for m^*).

The MMcoll term. At this point we make the observation that *bad* will be set to `true` in line 40 of game D if and only if there is some pair r, s, where $r < s$, such that the resulting random variables

$$0^n,\ X_r[0], \ldots X_r[m_r+1],\ X_s[0], \ldots X_s[m_s+1]$$

contains a repetition. But the probability of this event is at most $\mathrm{MMcoll}(m_r, m_s)$. Indeed this was our definition of $\mathrm{MMcoll}(m_r, m_s)$. Therefore the probability that *bad* is set to `true` in line 40 is at most

$$\sum_{1 \le r < s \le q} \mathrm{MMcoll}(m_r, m_s) \qquad (2)$$

```
20  $bad \leftarrow$ false;  for all $x \in \{0,1\}^n$ do $\pi(x) \leftarrow$ undefined
21  $L \xleftarrow{R} \{0,1\}^n$;  $\pi(0^n) \leftarrow L$

30  for $r \leftarrow 1$ to $q$ do {
31      $X_r[0] \leftarrow N_r \oplus L$;  $Y_r[0] \xleftarrow{R} \{0,1\}^n$
32      for $i \leftarrow 1$ to $m_r$ do $Z_r[i] \leftarrow \gamma_i \cdot L \oplus Y_r[0]$
33      for $i \leftarrow 1$ to $m_r - 1$ do { $X_r[i] \leftarrow M_r[i] \oplus Z_r[i]$;  $Y_r[i] \leftarrow C_r[i] \oplus Z_r[i]$ }
34      $X_r[m_r] \leftarrow \mathrm{len}(M[m_r]) \oplus huge \cdot L \oplus Z_r[m_r]$ ;  $Y_r[m_r] \leftarrow C_r[m_r] \oplus M_r[m_r]\, 0^*$
35      $\mathrm{Checksum}_r \leftarrow M_r[1] \oplus \cdots \oplus M_r[m_r - 1] \oplus C_r[m_r]\, 0^* \oplus Y_r[m_r]$
36      $X_r[m_r + 1] \leftarrow \mathrm{Checksum}_r \oplus Z_r[m_r]$ }
37  $\mathcal{X} \leftarrow (X_1[0], X_1[1], \ldots, X_1[m_1+1],\ \ldots,\ X_q[0], X_q[1], \ldots, X_q[m_q+1])$
38  $\mathcal{Y} \leftarrow (Y_1[0], Y_1[1], \ldots, Y_1[m_1+1],\ \ldots,\ Y_q[0], Y_q[1], \ldots, Y_q[m_q+1])$
39  for $i \leftarrow 1$ to $|\mathcal{X}|$ do $\pi(\mathcal{X}[i]) \leftarrow \mathcal{Y}[i]$
40  if some string is repeated in $\mathcal{X} \cup \{0^n\}$ then $bad \leftarrow$ true

50  $X[0] \leftarrow N \oplus L$;  if $X[0] \in \mathrm{Domain}(\pi)$ then $Y[0] \leftarrow \pi(X[0])$ else $Y[0] \xleftarrow{R} \overline{\mathrm{Range}}(\pi)$
51  $\pi(X[0]) \leftarrow Y[0]$
52  for $i \leftarrow 1$ to $m$ do $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
53  for $i \leftarrow 1$ to $m - 1$ do {
54      $Y[i] \leftarrow C[i] \oplus Z[i]$
55      if $Y[i] \in \mathrm{Range}(\pi)$ then $X[i] \leftarrow \pi^{-1}(Y[i])$ else $X[i] \xleftarrow{R} \overline{\mathrm{Domain}}(\pi)$
56      $\pi(X[i]) \leftarrow Y[i]$;  $M[i] \leftarrow X[i] \oplus Z[i]$ }
57  $X[m] \leftarrow \mathrm{len}(C[m]) \oplus huge \cdot L \oplus Z[m]$
58  if $X[m] \in \mathrm{Domain}(\pi)$ then $Y[m] \leftarrow \pi(X[m])$ else $Y[m] \xleftarrow{R} \overline{\mathrm{Range}}(\pi)$
59  $\pi(X[m]) \leftarrow Y[m]$
60  $\mathrm{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\, 0^* \oplus Y[m]$
61  $X[m+1] \leftarrow \mathrm{Checksum} \oplus Z[m]$
62  if $X[m+1] \in \mathrm{Domain}(\pi)$ then $Y[m+1] \leftarrow \pi(X[m])$ else $Y[m+1] \xleftarrow{R} \overline{\mathrm{Range}}(\pi)$
63  $T' \leftarrow Y[m+1]$ [first $\tau$ bits]
64  if $T = T'$ then $bad \leftarrow$ true
```

Figure 9: **Game D.** This game depends on $N_1, \ldots, N_q$, $M_1, \ldots, M_q$, $C_1, \ldots, C_q$, $Y_1[m_1+1], \ldots, Y_q[m_q+1]$, N, C and T.

We are left now to focus on the probability that *bad* gets set to `true` in line 64 of Game D.

Game E. We modify the second half of game D (lines 20–40 are unchanged). First, we simplify lines 50, 55 and 58, and 62 by choosing a random value in $\{0,1\}^n$ as opposed to a value in the co-range, co-domain, co-range, and co-range of π, respectively. By similar reasoning to that used before, this new game may decrease the probability that *bad* gets sets to `true`, but by an amount that is at most

$$\frac{(m+2)(\sigma + 2q + m + 3)}{2^n} \tag{3}$$

Second, we modify the game so as to "give up" (set *bad*) if the condition of line 62 is satisfied. (Here is where pretag-collisions would begin to cause extra complications.) In doing this, we may again decrease the probability that *bad* will be set to `true`. But the decrease is at most $1/2^\tau$ since, when the **else** clause of the new line 62 is executed (that is, $Y[m+1] \xleftarrow{R} \{0,1\}^n$), T will equal T' with probability exactly $1/2^\tau$. Finally, we modify the game to give up (set *bad*) whenever $N \notin \{N_1, \ldots, N_q\}$ but $X[0] = N \oplus L$ is already in $\mathrm{Domain}(\pi)$ when this is checked at line 50. The new game is called game E and it is shown in Figure 10. We note for future reference:

```
50    $X[0] \leftarrow N \oplus L$
51    **if** $N \neq N_r$ for any $r$ **and** $X[0] \in \text{Domain}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
52    **if** $N = N_r$ for some $r$ **then** $Y[0] \leftarrow Y_r[0]$ **else** $Y[0] \xleftarrow{R} \{0,1\}^n$
53    $\pi(X[0]) \leftarrow Y[0]$
54    **for** $i \leftarrow 1$ **to** $m$ **do** $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
55    **for** $i \leftarrow 1$ **to** $m-1$ **do** {
56        $Y[i] \leftarrow C[i] \oplus Z[i]$
57        **if** $Y[i] \in \text{Range}(\pi)$ **then** $X[i] \leftarrow \pi^{-1}(Y[i])$ **else** $X[i] \xleftarrow{R} \{0,1\}^n$
58        $\pi(X[i]) \leftarrow Y[i]$;   $M[i] \leftarrow X[i] \oplus Z[i]$ }
59    $X[m] \leftarrow \text{len}(C[m]) \oplus \mathit{huge} \cdot L \oplus Z[m]$
60    **if** $X[m] \in \text{Domain}(\pi)$ **then** $Y[m] \leftarrow \pi(X[m])$ **else** $Y[m] \xleftarrow{R} \{0,1\}^n$
61    $\pi(X[m]) \leftarrow Y[m]$
62    $\text{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
63    $X[m+1] \leftarrow \text{Checksum} \oplus Z[m]$
64    **if** $X[m+1] \in \text{Domain}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
```

Figure 10: **Game E, part 2.** The first half of this game is lines 20–40 of Game D.

$$\begin{aligned}
&\Pr[\mathit{bad} \text{ gets set in game D}] \\
&\quad\leq \;\; \Pr[\mathit{bad} \text{ gets set in game E}] + \frac{(m^*+2)(\sigma+2q+m^*+3)^2}{2^n} + \frac{1}{2^\tau} \qquad (4)
\end{aligned}$$

Game F. We now examine game E and relate it to a final game, F. If *bad* is set to `true` in game E the reason is either that $X[0] = N \oplus L$ was found to be in the domain of π even though N is a new nonce, or else $X[m+1]$ was found to be in the domain of π when this was checked. In the latter case, how did $X[m+1]$ come to be in the domain of π? At least one of the following must be true:

- $X[m+1] = 0^n$. (The value 0^n was added to the domain of π at line 21.)
- For some $r \in [1..q]$, for some $j \in [0..m_r+1]$, $X[m+1] = X_r[j]$. (These values were added to the domain of π at line 39.)
- For some $i \in [0..m]$, $X[m+1] = X[i]$. (These values were added to the domain of π at lines 53, 57, and 61).

When *bad* is set to `true` we will assign responsibility for this event to exactly one index $r \in [1..q]$. We say that the *responsible index* is r where:

- If N is a new nonce and $X[0] \in \text{Domain}(\pi)$ at line 51, then the responsible index is the least $r \in [1..q]$ such that $X_r[j] = X[0]$ for some j. Otherwise,
- If $X[m+1] = 0^n$, then the responsible index is $r = 1$. Otherwise,
- If there is an $r \in [1..q]$ such that, for some $j \in [0..m_r+1]$, $X[m+1] = X_r[j]$, then the responsible index is the least such value r. Otherwise,
- The responsible index is $r = 1$. (This last case can happen when $X[m+1] = X[i]$ for some $i \in [0..m]$.)

Partition the coins used in the running of game E into: the coins s_0 used in the initialization step (line 21); the coins $s_1, \ldots, s_q$ used for processing message $M_1, \ldots, M_q$, respectively (line 31); and the coins s used to process the forgery attempt C (lines 52, 57, and 60). Suppose we eliminate the **for** statement at line 30, and execute lines 31–36 for some specific value of r. Call this game E_r. We make the crucial observation that if *bad* is set to `true` in game E using coins $(s_0, s_1, \ldots, s_q, s)$

then *bad* will *still* be set to `true` in game E_r using coins (s_0, s_r, s) when the responsible index is r. This follows from our definition of the responsible index. The only observation that is needed is that when $X[m+1] = X[i]$ for some $i \in [0..m]$, then, considering the least such i, if $X[i]$ was selected by assigning to it an already-selected $X_s[j]$-value, then the third case in the definition of the responsible index will result in the selection of an index r that forces *bad* to `true`.

By what we have said, one can bound the probability that *bad* gets set to `true` in game E by summing the probabilities that *bad* gets set to `true` in game E_r, where $1 \le r \le q$.

Game E_r is precisely the game that was used to define the CMcoll; in particular, the probability that *bad* is set in E_r is $\mathrm{CMcoll}(m^*, m_r)$. We conclude that the probability that *bad* is set to `true` in game E_r is at most $\mathrm{CMcoll}(m^*, m_r)$. Thus the probability that *bad* gets set to `true` in game E is at most

$$\sum_{r=1}^{q} \mathrm{CMcoll}(m^*, m_r) \tag{5}$$

Summing Equations (1), (2), (4) and (5) gives that the adversary's chance of forgery is at most

$$\sum_{1\le r<s\le q} \mathrm{MMcoll}(q_r, q_s) + \sum_{r=1}^{q} \mathrm{CMcoll}(m^*, q_s) + \frac{(\sigma+2q+1)^2 + 2(m^*+2)(\sigma+2q+m^*+3)}{2^{n+1}} + \frac{1}{2^\tau}$$

Using that $(\sigma+\Delta)^2 - \sigma^2 = 2\sigma\Delta + \Delta^2$, we can increase σ by a small amount in order to compensate for the lower-order terms and clean up the expression. Namely, increasing σ by $2q+1$ is enough to take care of the first addend, while increasing σ by m^*+2 plus $2(m^*+2)$ plus $\sqrt{2}(m^*{+}2)$ plus $\sqrt{6}\sqrt{m^*+2}$, is enough to take care of the second addend. So increasing σ by $2q+8m^*+16$ will take care of both. Letting $\bar{\sigma} = 2q+8m^*+16$ we thus have that the adversary's chance of forgery is at most

$$\sum_{1\le r<s\le q} \mathrm{MMcoll}(q_r, q_s) + \sum_{r=1}^{q} \mathrm{CMcoll}(m^*, q_s) + \frac{\bar{\sigma}^2}{2} \cdot \frac{1}{2^n} + \frac{1}{2^\tau}$$

This completes the proof of the structure lemma. ▮

A.2 Proof of the MM-collision Bound (Lemma 2)

Proof: There are $m + \bar{m} + 5$ values in the set

$$S = \{0^n,\ X[0], \ldots, X[m+1],\ \bar{X}[0], \ldots, \bar{X}[\bar{m}+1]\}$$

and we want to bound the probability of a collision among any pair of them. We claim that, for any particular pair of elements in this set, the probability that they collide is at most $1/2^n$. Since there are $\binom{m+\bar{m}+5}{2}$ pairs of elements in S, the lemma follows.

There are a few cases to consider. Below, remember that $L, R, \bar{R}$ are random, and everything else is constant. The probabilities are over $L, R, \bar{R}$. We let $i, i' \in [1..m]$ and $j, j' \in [1..\bar{m}]$ where $i \ne j$ and $i' \ne j'$.

- $\Pr[0^n = M[0] \oplus L] = 1/2^n$ and $\Pr[0^n = M[i] \oplus \gamma_i \cdot L \oplus R] = 1/2^n$ and $\Pr[0^n = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R] = 1/2^n$. Similarly for $\bar{M}$.
- $\Pr[M[0] \oplus L = M[i] \oplus \gamma_i \cdot L \oplus R] = 1/2^n$ and $\Pr[M[0] \oplus L = M[m] \oplus (\gamma_m + \mathit{huge}) \cdot L \oplus R] = 1/2^n$. Similarly for $\bar{M}$.
- $\Pr[M[i] \oplus \gamma_i \cdot L = M[i'] \oplus \gamma_{i'} \cdot L] = \Pr[M[i] \oplus M[i'] = (\gamma_i \oplus \gamma_{i'}) \cdot L] = 1/2^n$ because $\gamma_i \ne \gamma_{i'}$. Similarly for $\bar{M}$.

- $\Pr[M[i] \oplus \gamma_i \cdot L \oplus R = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R] = \Pr[M[i] \oplus \gamma_i \cdot L = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L] = \Pr[M[i] \oplus M[m] = (\gamma_m \oplus \mathit{huge} \oplus \gamma_i) \cdot L] = 1/2^n$ because $\gamma_i \oplus \gamma_m \neq \mathit{huge}$. The reason that $\gamma_i \oplus \gamma_m \neq \mathit{huge}$ is that *huge* has a 1 in bit position 1, while neither γ_i nor γ_m do, because $i, m < 2^{n-2}$ and $\gamma_i \leq 2i$, $\gamma_m \leq 2m$. Here we are assuming that $m < 2^{n-2}$ because if $m \geq 2^{n-2}$ then the lemma statement gives a non-result (a probability bound already exceeding 1). Similarly for $\bar{M}$.
- $\Pr[M[0] \oplus L = \bar{M}[0] \oplus L] = 0$, since $M[0] \neq \bar{M}[0]$. $\Pr[M[0] \oplus L = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$, and $\Pr[M[0] \oplus L = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R}] = 1/2^n$.
- $\Pr[M[i] \oplus \gamma_i \cdot L \oplus R = \bar{M}[0] \oplus L] = 1/2^n$, $\Pr[M[i] \oplus \gamma_i \cdot L \oplus R = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$, and $\Pr[M[i] \oplus \gamma_i \cdot L \oplus R = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R}] = 1/2^n$.
- $\Pr[M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R = \bar{M}[0] \oplus L] = 1/2^n$. $\Pr[M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$, and $\Pr[M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R}] = 1/2^n$.

The only "interesting" case was the fourth, which more significantly used the characteristics of the Gray-code sequence. ▌

A.3 Proof of the CM-Collision Bound (Lemma 3)

Proof: At the top level, we consider two cases: $N \neq \bar{N}$ and $N = \bar{N}$. The second of these will be analyzed by breaking into three subcases.

Case 1: $N \neq \bar{N}$. In this case there are two ways for *bad* to be set to `true`: it can happen at line 31 or line 44 in the game that defines the CMcoll collision probability (Figure 3). Let us first calculate the probability that *bad* is set to `true` at line 31, which is

$$\Pr[\mathit{bad} \text{ is set at line 31}] = \Pr[N \oplus L \in \{0^n,\ \bar{X}[1], \ldots, \bar{X}[\bar{m}+1]\}]$$

One point in the domain of π has been omitted from set $B = \{0^n,\ \bar{X}[1], \ldots, \bar{X}[\bar{m}], \bar{X}[\bar{m}+1]\}$: $\bar{X}[0] = \bar{N} \oplus L$, which we know is different from $N \oplus L$ since $N \neq \bar{N}$. The probability above is taken over L and $\bar{R}$, where each $X[i]$ implicitly depends on both. We claim that for each of the $\bar{m}+2$ values in S, the probability that $N \oplus L$ is equal to this particular value is exactly $1/2^n$. This is verified by:

- $\Pr[N \oplus L = 0^n] = 1/2^n$.
- For any $j \in [1..\bar{m}]$, $\Pr[N \oplus L = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$ because of the influence of the random $\bar{R}$.
- Similarly, $\Pr[N \oplus L = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R}] = 1/2^n$ because of the random $\bar{R}$.

We conclude that

$$\Pr[\mathit{bad} \text{ is set at line 31}] \quad \leq \quad \frac{\bar{m}+2}{2^n} \tag{6}$$

We next show that

$$\Pr[X[m] \in \text{Domain}(\pi) \text{ at line 40}] \quad \leq \quad \frac{m+\bar{m}+3}{2^n} \tag{7}$$

For this, let us define S to be

$$S = \{0^n,\ \bar{X}[0], \bar{X}[1], \ldots, \bar{X}[\bar{m}+1],\ X[0], X[1], \ldots, X[m-1]\}$$

This is the domain of π at the time that line 40 is executed. The set has $m+\bar{m}+3$ points and we shall use the sum bound to see that the probability that $X[m]$ is one of these is at most $(m+\bar{m}+3)/2^n$. Namely,

- $\Pr[X[m]=0^n]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=0^n]=1/2^n$ as the right-hand side of the equality sign does not depend on R.
- $\Pr[X[m]=\bar{X}[0]]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=\bar{N}\oplus L]=1/2^n$ for the same reason.
- For $j\in[1..\bar{m}-1]$, $\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=\bar{M}[j]\oplus\gamma_j\cdot L\oplus\bar{R}]=1/2^n$ for the same reason.
- $\Pr[X[m]=X[\bar{m}]]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=\bar{M}[\bar{m}]\oplus(\gamma_{\bar{m}}\oplus \mathit{huge})\cdot L\oplus\bar{R}]=1/2^n$ for the same reason.
- $\Pr[X[m]=\bar{X}[\bar{m}+1]]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=\text{Checksum}'\oplus\gamma_{\bar{m}}\cdot L\oplus\bar{R}]=1/2^n$.
- $\Pr[X[m]=X[0]]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=N\oplus L]=1/2^n$ for the same reason.
- For $i\in[1..m-1]$, $X[i]$ is determined in one of two possible ways: either it is a value already placed into the Domain(π) (the **then** clause at line 37 was executed) or else it is a randomly selected value in $\{0,1\}^n$ (the **else** clause was executed). In the former case, the sum bound has already accounted for the probability of a collision with $X[i]$. In the latter case, the chance of collision with $X[m]=\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R$ is $1/2^n$.

Equation (7) has now been established.

Next we observe that

$$\Pr[X[m+1]\in\text{Domain}(\pi)\text{ at line 44 }\mid X[m]\notin\text{Domain}(\pi)\text{ at line 40 }]\ \le\ \frac{m+\bar{m}+4}{2^n}\qquad(8)$$

The reason is that, when the conditioning event happens, $Y[m]$ is selected as a random point in $\{0,1\}^n$ at line 40, which results in Checksum being a random value independent of the points in the domain of π, which results in $X[m+1]$ being a random value independent of the points in the domain of π. Since the domain of π has at most $1+\bar{m}+2+m+1=m+\bar{m}+4$ points at this time, Equation (8) follows. Now, summing Equations (6), (7) and (8) gives us that

$$\Pr[\mathit{bad}\text{ gets set }\mid\text{ Case 1 }]\ \le\ \frac{3\bar{m}+2m+9}{2^n}\qquad(9)$$

Case 2a: $N=\bar{N}$ and $m\ne\bar{m}$. The next case we consider is when $N\ne\bar{N}$ and $m\ne\bar{m}$. Redefine S to be

$$S=\{0^n,\ \bar{X}[0],\ldots,\bar{X}[\bar{m}+1],\ X[1],\ldots,X[m-1]\}$$

This is Domain(π) at the time line 40 is executed. We show that

$$\Pr[X[m]\in S\ \mid\ \text{Case 2a }]\ \le\ \frac{m+\bar{m}+2}{2^n}\qquad(10)$$

To show this, one has as before to go through the $m+\bar{m}+2$ points of S:

- $\Pr[X[m]=0^n]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=0^n]=1/2^n$.
- $\Pr[X[m]=N\oplus L]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=N\oplus L]=1/2^n$.
- For $j\in[1..\bar{m}-1]$, $\Pr[X[m]=\bar{X}[j]]=\Pr[\text{len}(C[m])\oplus(\gamma_m\oplus \mathit{huge})\cdot L\oplus R=\bar{M}[j]\oplus\gamma_j\cdot L\oplus R]=\Pr[\text{len}(C[m])\oplus\bar{M}[j]=(\gamma_j\oplus\gamma_m\oplus \mathit{huge})\cdot L]=1/2^n$ since $\gamma_j\oplus\gamma_m\ne \mathit{huge}$ for the reason already explained.

- $\Pr[X[m] = \bar{X}[\bar{m}]] = \Pr[\text{len}(C[m]) \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R = \text{len}(\bar{M}[\bar{m}]) \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus R] = \Pr[\text{len}(C[m]) \oplus \text{len}(\bar{M}[\bar{m}]) = (\gamma_m \oplus \gamma_{\bar{m}}) \cdot L] = 1/2^n$ since $\gamma_m \neq \gamma_{\bar{m}}$
- $\Pr[X[m] = \bar{X}[\bar{m}+1]] = \Pr[\text{len}(C[m]) \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R = \text{Checksum}' \oplus \gamma_{\bar{m}} \cdot L \oplus R] = \Pr[\text{len}(C[m]) \oplus \text{Checksum}' = (\gamma_m \oplus \mathit{huge} \oplus \gamma_{\bar{m}}) \cdot L] = 1/2^n$ as before.
- For $i \in [1..m-1]$, either $X[i]$ was selected as a value already in Domain(π), in which case the sum bound has already accounted for the probability of a collision with $X[m]$, or else $X[i]$ was selected as a new random value, in which case it has a $1/2^n$ chance of colliding with $X[m]$.

We have established (10). Next, as before, if $X[m] \notin S$ then $Y[m]$ is chosen at random, making Checksum random, and making $X[m+1]$ random. Thus

$$\Pr[X[m+1] \in \text{Domain}(\pi) \text{ at line 44} \quad | \quad X[m] \notin \text{Domain}(\pi) \text{ at line 40 }] \quad \leq \quad \frac{m+\bar{m}+3}{2^n} \tag{11}$$

since the size of the domain of π at line 44 is at most $m + \bar{m} + 3$. Adding Equations (10) and (11) we have that

$$\Pr[\mathit{bad} \text{ gets set} \quad | \quad \text{Case 2a}] \quad \leq \quad \frac{2m+2\bar{m}+5}{2^n} \tag{12}$$

Case 2b: $N = \bar{N}$ and $m = \bar{m}$ and $\exists\, a,\ a < m$, s.t. $C[a] \neq \bar{C}[a]$. In this case, let $a \geq 1$ be the smallest index such that $C[a] \neq \bar{C}[a]$. We claim that $Y[a]$ is almost certainly not in the range of π when this point is examined at line 37, when $i = a$. In fact, we claim something stronger: that $Y[a]$ is almost certainly different from all of

$$S = \{L,\ \bar{Y}[0], \ldots, \bar{Y}[m+1],\ Y[1], \ldots, Y[a-1],\ Y[a+1], \ldots, Y[m-1]\}$$

In particular,

$$\Pr[Y[a] \in S] \quad \leq \quad \frac{m+\bar{m}}{2^n} \tag{13}$$

This is verified by going through each point in S, exactly as before. This time, for each point in S except $\bar{Y}[a]$, the probability that this point coincides with $Y[a]$ is exactly $1/2^n$. The probability that $\bar{Y}[a] = Y[a]$ is 0, since $C[a] \neq \bar{C}[a]$.

Now we modify the game which defines CMcoll so that $X[a]$ is always selected at random from $\{0,1\}^n$. If we bound the probability that *bad* gets set in this new game and then add to it the bound of Equation (13), the result bounds the probability that *bad* gets set in Case 2b. From now on in this case analysis, assume this new game.

Next we claim that $X[m]$ is almost certainly different from $X[a]$:

$$\Pr[X[m] = X[a]] \quad = \quad \frac{1}{2^n} \tag{14}$$

This is clear because, in the modified game we have described, $X[a]$ is now chosen at random, independent of $X[m] = \text{len}(C[m]) \oplus (\mathit{huge} \oplus \gamma_m) \cdot L \oplus R$.

As before, we may now modify the game once again so that $Y[m]$ is selected at random even in the case that $X[m] = X[a]$. Bounding the probability of *bad* being set in the new game, and adding in the bound of (14), serves to bound the probability of *bad* being set in the prior game.

Now we can look at the probability that $X[m+1] \in \text{Domain}(\pi)$ when this is checked in the modified game.

At this point the domain of π contains the $m + \bar{m} + 3$ points

$$\text{Domain}^* \;=\; \{0^n,\ \bar{X}[0], \ldots, \bar{X}[\bar{m}+1],\ X[1], \ldots, X[a], \ldots, X[m]\}$$

We want to know the probability that $\text{Checksum} \oplus \gamma_m \cdot L \oplus R$ is in this set. But Checksum now contains the point $Y[m]$, which, in the modified game, has just been selected at random and independent of the points above. So

$$\Pr[X[m+1] \in \text{Domain}(\pi) \text{ in the modified game}] \;\leq\; \frac{m + \bar{m} + 3}{2^n} \tag{15}$$

Summing Equations (13), (14), and (15), we conclude that

$$\Pr[\mathit{bad} \text{ gets set } \mid \text{ Case 2b }] \;\leq\; \frac{2m + 2\bar{m} + 4}{2^n} \tag{16}$$

Case 2c: $N = \bar{N}$ and $m = \bar{m}$ and $C[i] = \bar{C}[i]$ for all $1 \leq i < m$ and $|C[m]| = |\bar{C}[m]|$. In this case, necessarily $C[m] \neq \bar{C}[m]$. Note that Checksum has a known value, which is different from Checksum$'$, being exactly $\text{Checksum}' \oplus \bar{C}[m]\, 0^* \oplus C[m]\, 0^*$. The values $M[1], \ldots, M[m-1]$ are likewise known, being identical to $\bar{M}[1], \ldots, \bar{M}[m] - 1$, respectively. We are interested in

$$\Pr[X[m+1] \in \{0^n, X[0], \ldots, X[m], \bar{X}[m+1]\}$$

One goes through each of the points, as before, and sees that the probability that $X[m+1] = \text{Checksum} \oplus \gamma_m \cdot L \oplus R$ is any one of them is $1/2^n$, except for the last point, for which the probability that they coincide is 0. Thus

$$\Pr[\mathit{bad} \text{ gets set } \mid \text{ Case 2c }] \;\leq\; \frac{m+2}{2^n} \tag{17}$$

Case 2d: $N = \bar{N}$ and $m = \bar{m}$ and $C[i] = \bar{C}[i]$ for all $1 \leq i < m$ and $|C[m]| \neq |\bar{C}[m]|$. For this case, we first claim that $X[m]$ is almost certainly not in the domain of π when this is inspected at line 40 of Figure 3. The method is as before. The point $X[m]$ is certain to be different from $\bar{X}[m]$, by construction, and its chance of coinciding with any of the $m+2$ points $0^n, X[0], X[1], \ldots, X[m-1], \bar{X}[m+1]$ is easily verified to be $1/2^n$. Thus

$$\Pr[X[m] \in \text{Domain}(\pi) \text{ at line 39 }] \;\leq\; \frac{m+2}{2^n} \tag{18}$$

Proceeding as before,

$$\Pr[X[m+1] \in \text{Domain}(\pi) \text{ at line 44 } \mid X[m] \notin \text{Domain}(\pi) \text{ at line 38 }] \;\leq\; \frac{m+3}{2^n} \tag{19}$$

since $m+3$ bounds the size of the domain when line 44 is executed, and the conditioning even ensures a random value for $X[m+1]$ which is independent of these points. Summing the bounds of Equation (18) and (19) gives

$$\Pr[X[m+1] \in \text{Domain}(\pi) \text{ at line 44 }] \;\leq\; \frac{2m+5}{2^n} \tag{20}$$

Conclusion. Taking the maximum from Equations (9), (12), (16), (17), and (20) we have

$$\Pr[\mathit{bad} \text{ gets set }] \;\leq\; \frac{3\bar{m} + 2m + 9}{2^n}$$

which is the lemma. ▮

A.4 Proof of the Privacy Bound (Lemma 4)

The proof is straightforward compared to authenticity, so we quickly go though it. We begin by following the proof of the Structure Lemma (Appendix A.1). Games A to D are defined as before, except that

- The second half of each game is omitted, since there is no forgery attempt in this context.
- Return the truncated final-ciphertext-blocks, instead of the full final-ciphertext blocks, as the games specify.
- Do not eliminate lines 13, 19, 23 and 28 from game B; instead, check for that collision in game C and D.

Focus on the (modified) game C, where we have now returned to the adversary A a random string of $|M_r| + \tau$ bits whenever a query M_r is asked. Furthermore, the behavior of game C coincides with the behavior of the original game A unless the flag *bad* is set to `true`, at which point the two games diverge. Thus we can bound $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$ by bounding the probability that the flag *bad* is set to `true` in (the modified) game C, which is at most the probability that it gets set in Game D. From the same reasoning as in the structure lemma, this is at most

$$\frac{(\sigma + 2q + 1)^2}{2^{n+1}} + \sum_{1 \le r < s \le q} \mathrm{MMcoll}(m_r, m_s)$$

which is precisely the bound given by the the lemma. ▮

B Test Vectors

OCB-AES test vectors and reference code, using AES-128, AES-192, and AES-256, are available at `http://www.cs.ucdavis.edu/~rogaway/`

C Document History

- October 16, 2000. First version of the OCB document submitted to NIST [22].
- April 1, 2001. Second version of the OCB document. Submitted to NIST.