OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption *

Phillip Rogaway†
UC Davis & Chiang Mai Univ

Mihir Bellare‡
UC San Diego

John Black§
University of Nevada

Ted Krovetz¶
Digital Fountain

ABSTRACT

We describe a parallelizable block-cipher mode of operation that simultaneously provides privacy and authenticity. OCB encrypts-and-authenticates a nonempty string $M \in \{0,1\}^*$ using $\lceil |M|/n \rceil + 2$ block-cipher invocations, where n is the block length of the underlying block cipher. Additional overhead is small. OCB refines a scheme, IAPM, suggested by Charanjit Jutla. Desirable properties of OCB include: the ability to encrypt a bit string of arbitrary length into a ciphertext of minimal length; cheap offset calculations; cheap key setup; a single underlying cryptographic key; no extended-precision addition; a nearly optimal number of block-cipher calls; and no requirement for a random IV. We prove OCB secure, quantifying the adversary's ability to violate the mode's privacy or authenticity in terms of the quality of its block cipher as a pseudorandom permutation (PRP) or as a strong PRP, respectively.

Keywords

AES, authenticity, block-cipher usage, cryptography, encryption, integrity, modes of operation, provable security, standards

1. INTRODUCTION

BACKGROUND. 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 (often called an IV), 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 [7, 24].) 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. Many such schemes have been broken. Recently, however, Charanjit Jutla has proposed two authenticated-encryption schemes supported by a claim of provable security [21]. Virgil Gligor and Pompiliu Donescu have described a different authenticated-encryption scheme [15]. We continue in this line of work.

OCB MODE. This paper describes a new mode of operation, OCB, which refines one of Jutla's schemes, IAPM [21]. OCB (which stands for "offset codebook") retains the principal characteristics of IAPM: it is fully parallelizable and adds minor overhead compared to conventional, privacy-only modes. But OCB combines the following features:

- *Arbitrary-length messages + minimal-length ciphertexts*: Any string $M \in \{0,1\}^*$ can be encrypted; in particular, $|M|$ need not be a multiple of the block length n. What is more, the resulting ciphertexts are as short as possible;

* A full version of this paper is available as [33].

† Dept. of Comp. Sci., Eng. II Bldg., Univ. of California, Davis, CA 95616 USA; and Dept. of Comp. Sci., Faculty of Science., Chiang Mai University, Chiang Mai 50200 Thailand. e-mail: rogaway@cs.ucdavis.edu web: www.cs.ucdavis.edu/∼rogaway

‡ Dept. of Comp. Sci. & Eng., Univ. of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093 USA. e-mail: mihir@cs.ucsd.edu web: www-cse.ucsd.edu/users/mihir

§ Dept. of Comp. Sci., Univ. of Nevada, Reno, NV 89557 USA. e-mail: jrb@cs.unr.edu web: www.cs.unr.edu/∼jrb

¶ Digital Fountain, 600 Alabama Street, San Francisco, CA 94110 USA. e-mail: tdk@acm.org

CCS'01, November 5±8, 2001, Philadelphia, Pennsylvania, USA.

plaintexts are *not* padded to a multiple of n bits.

- *Nearly optimal number of block-cipher calls*: OCB uses $\lceil |M|/n \rceil + 2$ block-cipher invocations (excluding a block-cipher call assumed to be made during key setup). (It is possible to make do with $\lceil |M|/n \rceil + 1$ calls, but such alternatives seem to be more complex or require a random IV.) Keeping low the number of block-cipher calls is especially important when messages are short.
- *Minimal requirements on nonces*: Like other encryption modes, OCB requires a nonce. The entity that encrypts chooses a new nonce for every message with the only restriction that no nonce is used twice. Schemes that require non-repeating nonces are less likely to be misused, and often more efficient, than those requiring random IVs.
- *Efficient offset calculations*: As with [15, 21], we require a sequence of *offsets*. We generate these in a particularly cheap way, each offset requiring just a few machine cycles. We avoid the use of extended-precision addition, which would introduce endian dependency and might make the scheme less attractive for dedicated hardware.
- *Single underlying key*: The key used for OCB is a single block-cipher key, and all block-cipher invocations are keyed by this one key, saving space and key-setup time.

Achieving the properties above has required putting together a variety of "tricks" that work together in just the right way. Many earlier versions of the algorithm were rejected by the authors because attacks were found or a proof could not be pushed through. We have found schemes of this sort to be amazingly "fragile"—tweak them a little and they break. We have concluded that, if the goals above are ever to be sought, they must be carefully addressed from the start.

PERFORMANCE. On a Pentium III processor, experiments by Lipmaa [25] show that OCB is about 6.5% slower than the privacy-only mode CBC. The cost of OCB is about 54% of the cost of CBC encryption combined with the CBC MAC. These figures assume a block cipher of AES128 [34].

In settings where there is adequate opportunity for parallelism, OCB will be faster than CBC. Parallelizability is important for obtaining the highest speeds from special-purpose hardware, and it may become useful on commodity processors. For special-purpose hardware, one may want to encrypt-and-authenticate at speeds near 10 Gbits/second—an impossible task, with today's technology, for modes like CBC encryption and the CBC MAC. (One could always create a mode that interleaves message blocks fed into separate CBC encryption or CBC MAC calculations, but that would be a new mode, and one with many drawbacks.) For commodity processors, 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.

SECURITY PROPERTIES. We prove OCB secure, in the sense of reduction-based cryptography. Specifically, we prove indistinguishability under chosen-plaintext attack [3, 16] and authenticity of ciphertexts [7, 8, 22]. As shown in [7, 22], this combination implies indistinguishability under chosen-ciphertext attack (CCA) which, in turn, is equivalent to non-malleability [10] under CCA [4, 23]. (Non-malleability refers to an adversary's inability to modify a ciphertext in a way that makes related the two underlying plaintexts.) Our proof of privacy assumes that the underlying block cipher is good in the sense of a pseudorandom permutation (PRP) [6, 26], while our proof of authenticity assumes that the block cipher is a strong PRP [26]. Our results are quantitative; the security analysis is in the concrete-security paradigm.

We emphasize that OCB has stronger security properties than standard modes. In particular, non-malleability and indistinguishability under CCA are not achieved by CBC, or by any other standard mode, but these properties are achieved by OCB. We believe that the lack of strong security properties has been a problem for the standard modes of operation, because many users of encryption implicitly assume these 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, or 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 incorrectly apply the standard modes, or to invent or select wrong ways to encrypt with authenticity (a well-known example is the use of PCBC mode [27] in Kerberos v.4 [29]). We believe that a mode like OCB is less likely to be misused because the usual abuses of privacy-only 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 promptly upgrade its scheme [9, 28]. 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.

THE FUTURE. We believe that *most* of the time privacy is desired, authenticity is too. As a consequence, fast authenticated encryption may quickly catch on. OCB has already appeared in one draft standard—the wireless LAN standard IEEE 802.11—and it is also under consideration by NIST.

2. PRELIMINARIES

NOTATION. If a and b are integers, $a \le b$, then $[a..b]$ is the set $\{a, a+1, \ldots, b\}$. If $i \ge 1$ 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 $\varepsilon \oplus A = A \oplus \varepsilon = \varepsilon$). So, for example, $1001 \oplus 11 = 01$. If

$A = a_{n-1} \cdots a_1 a_0 \in \{0,1\}^n$ 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 $A{\ll}1$ is the n-bit string $a_{n-2}a_{n-3} \cdots a_1 a_0 0$ which is a left shift of A by one bit (the first bit of A disappearing and a zero coming into the last bit), while $A{\gg}1$ is the n-bit string $0a_{n-1}a_{n-2} \ldots a_2 a_1$ 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 $C = \mathcal{C}$ [first $|\mathcal{C}| - \tau$ bits], let $T = \mathcal{C}$[last τ 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. Let $\mathrm{GF}(2^n)$ denote the field with 2^n points. 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 an 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 an abstract point in $\mathrm{GF}(2^{128})$.

To add two points in $\mathrm{GF}(2^n)$, take their bitwise xor. We denote this operation by $a \oplus b$. To multiply two points in the field, first fix an irreducible polynomial $p_n(\mathtt{x})$ having binary coefficients and degree n: say the lexicographically first polynomial among the irreducible degree n polynomials having a minimum number of nonzero coefficients. For $n = 128$, the indicated polynomial is $p_{128}(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$. To multiply $a, b \in \mathrm{GF}(2^n)$, 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 $c(\mathtt{x})$ over $\mathrm{GF}(2)$, and take the remainder one gets when dividing $c(\mathtt{x})$ by $p_n(\mathtt{x})$.

It is computationally simple to multiply $a \in \{0,1\}^n$ by $\mathtt{x}$. We illustrate the method for $n = 128$, in which case multiplying $a = a_{n-1} \cdots a_1 a_0$ by $\mathtt{x}$ yields $a_{n-1}\mathtt{x}^n + a_{n-2}\mathtt{x}^{n-1} + a_1\mathtt{x}^2 + a_0\mathtt{x}$. 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 $p_{128}(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1 = 0$ 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 $n = 128$,

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

It is similarly easy to divide $a \in \{0,1\}^{128}$ by $\mathtt{x}$ (i.e., to multiply a by the multiplicative inverse of $\mathtt{x}$). 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 that $\mathtt{x}^{-1} = \mathtt{x}^{127} + \mathtt{x}^6 + \mathtt{x} + 1 = 10^{120}1000011$. In summary, when $n = 128$,

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

If $L \in \{0,1\}^n$ and $i \ge -1$, we write $L(i)$ as shorthand for $L \cdot \mathtt{x}^i$. Using the equations just given, we have an easy way to compute from L the values $L(-1)$, $L(0)$, $L(1), \ldots, L(\mu)$, where μ is small number.

Gray codes. For $\ell \ge 1$, 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 $\gamma^1 = (0 \;\; 1)$ and, for $\ell > 0$, $\gamma^{\ell+1} = (0\gamma_0^\ell \; 0\gamma_1^\ell \; \cdots \; 0\gamma_{2^\ell-2}^\ell \; 0\gamma_{2^\ell-1}^\ell \; 1\gamma_{2^\ell-1}^\ell \; 1\gamma_{2^\ell-2}^\ell \; \cdots \; 1\gamma_1^\ell \; 1\gamma_0^\ell)$. 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.

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

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 $\gamma_i \cdot L = (\gamma_{i-1} \oplus (0^{n-1}1{\ll}\mathsf{ntz}(i))) \cdot L = (\gamma_{i-1} \cdot L) \oplus (0^{n-1}1{\ll}\mathsf{ntz}(i)) \cdot L = (\gamma_{i-1} \cdot L) \oplus (L \cdot \mathtt{x}^{\mathsf{ntz}(i)}) = (\gamma_{i-1} \cdot L) \oplus L(\mathsf{ntz}(i))$. 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. THE SCHEME

Parameters. To use OCB one must specify a block cipher and a tag length. The *block cipher* 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* is an integer $\tau \in [0..n]$. 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 [34]. As for the tag length, a suggested default of $\tau = 64$ is reasonable. Tags of 32 bits are standard in retail banking. Tags of 96 bits are used in IPSec.

We let OCB-E denote the OCB mode of operation using block cipher E and an unspecified tag length. 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 an illustration of OCB, and see Figure 2 for the definition. The latter fig-

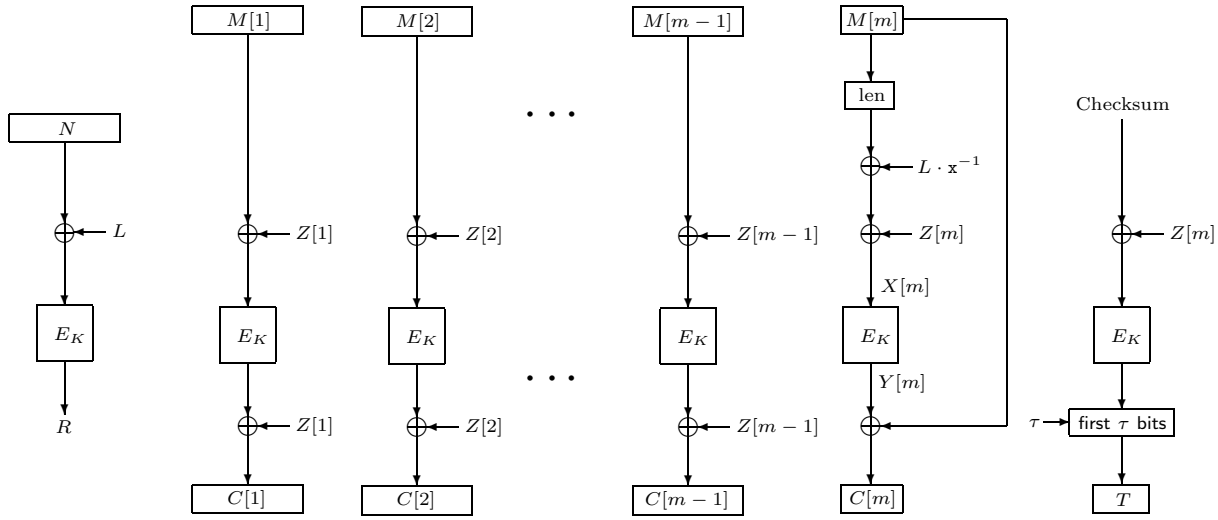

Figure 1: Illustration of OCB encryption. The message is $M = M[1]M[2]\cdots M[m-1]M[m]$ and the nonce is N. The resulting ciphertext is $\mathcal{C} = C[1]C[2]C[3]\cdots C[m-1]C[m]\,T$. The Checksum **is $M[1]\oplus\cdots\oplus M[m-1]\oplus C[m]\,0^*\oplus Y[m]$. Define $L = E_K(0^n)$, $Z[1] = L \oplus R$, and, for $i \geq 2$, $Z[i] = Z[i-1] \oplus L(\mathsf{ntz}(i))$.**

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** $C[i] \leftarrow E_K(M[i] \oplus Z[i]) \;\oplus\; Z[i]$
$X[m] \leftarrow \mathrm{len}(M[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$C[m] \leftarrow Y[m] \oplus M[m]$
$C \leftarrow C[1]\cdots C[m]$
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 τ bits]
return $\mathcal{C} \leftarrow C \parallel T$

Algorithm OCB.Dec$_K$ $(N, \mathcal{C})$

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** $M[i] \leftarrow E_K^{-1}(C[i] \oplus Z[i]) \;\oplus\; Z[i]$
$X[m] \leftarrow \mathrm{len}(C[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$M[m] \leftarrow Y[m] \oplus C[m]$
$M \leftarrow M[1]\cdots M[m]$
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 τ bits]
if $T = T'$ **then return** M
else return Invalid

Figure 2: Definition of OCB. Encryption and decryption are specified, while key generation chooses a random element from the key space $\mathcal{K}$ of the block cipher.

ure defines OCB encryption and decryption, while the key space $\mathcal{K}$ is the key space for the underlying block cipher E.

An equivalent description. The following description may clarify what a typical implementation might do.

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

Key setup: For the party that encrypts, do any key setup associated to block-cipher enciphering. For the party that decrypts, do any key setup associated to block-cipher enciphering and deciphering. Let $L \leftarrow E_K(0^n)$. 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 $i \in [1..\mu]$, compute $L(i) \leftarrow L(i-1) \cdot \mathtt{x}$ 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 $L(-1)$, $L(0)$, $L(1), \ldots, L(\mu)$ in a table.

Encryption: To encrypt plaintext $M \in \{0,1\}^*$ using key K and nonce $N \in \{0,1\}^n$, obtaining a ciphertext $\mathcal{C}$, do the following. 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 $i \in [1..m-1]$. Let Offset $\leftarrow E_K(N \oplus L)$. Let Checksum $\leftarrow 0^n$. For $i \leftarrow 1$ to $m-1$, do the following: let Checksum $\leftarrow$ Checksum $\oplus\, M[i]$; let Offset $\leftarrow$ Offset$\oplus L(\mathsf{ntz}(i))$; let $C[i] \leftarrow E_K(M[i] \oplus \text{Offset}) \oplus \text{Offset}$. Let Offset $\leftarrow$ Offset $\oplus\, L(\mathsf{ntz}(m))$. Let $Y[m] \leftarrow E_K(\mathrm{len}(M[m]) \oplus L(-1) \oplus \text{Offset})$. Let $C[m] \leftarrow M[m]$ xored with the first $|M[m]|$ bits of $Y[m]$. Let Checksum $\leftarrow$ Checksum $\oplus\, Y[m] \oplus C[m]\,0^*$. Let T be the first τ bits of $E_K(\text{Checksum} \oplus \text{Offset})$. 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 ciphertext $\mathcal{C} \in \{0,1\}^*$ using key K and nonce $N \in \{0,1\}^n$, obtaining a plaintext $M \in \{0,1\}^*$ or an indication Invalid, do the following. 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 $i \in [1..m-1]$. Let Offset $\leftarrow E_K(N \oplus L)$. Let Checksum $\leftarrow 0^n$. For $i \leftarrow 1$ to $m-1$, do the following: let Offset $\leftarrow$ Offset $\oplus L(\mathsf{ntz}(i))$; let $M[i] \leftarrow E_K^{-1}(C[i] \oplus \text{Offset}) \oplus$ Offset; let Checksum $\leftarrow$ Checksum $\oplus M[i]$. Let Offset $\leftarrow$ Offset $\oplus L(\mathsf{ntz}(m))$. Let $Y[m] \leftarrow E_K(\text{len}(C[m]) \oplus L(-1) \oplus$ Offset). Let $M[m] \leftarrow C[m]$ xored with the first $|C[m]|$ bits of $Y[m]$. Let Checksum $\leftarrow$ Checksum $\oplus Y[m] \oplus C[m]\,0^*$. Let T' be the first τ bits of $E_K(\text{Checksum} \oplus \text{Offset})$. If $T \neq T'$ then return INVALID (the ciphertext has been rejected). Otherwise, the plaintext is $M = M[1] \cdots M[m-1]M[m]$.

4. DISCUSSION

OCB has been designed to have a variety of desirable properties. Some of these have been discussed in the Introduction. We extend that discussion here.

ARBITRARY-LENGTH MESSAGES AND NO CIPHERTEXT EXPANSION. One of the key characteristics of OCB is that any string $M \in \{0,1\}^*$ can be encrypted, and doing this yields a ciphertext $\mathcal{C}$ of length $|M| + \tau$. That is, the length of the "ciphertext core"—the portion $C = C[1] \cdots C[m]$ of the ciphertext that excludes the tag—is the same as the length of the message M. This is better, by up to n bits, than what one gets with conventional padding.

SINGLE BLOCK-CIPHER KEY. OCB makes use of just one block-cipher key, K. While $L = E_K(0^n)$ functions rather like a key and would normally be computed at key-setup time, and while standard key-separation techniques can always be used to obtain many keys from one, the point is that, in OCB, all block-cipher invocations use the one key K. Thus only one block-cipher key needs to be setup, saving on storage space and key-setup time.

WEAK NONCE REQUIREMENTS. We believe that modes of operation that require a random IV are often misused. As an example, consider CBC mode, where $C[i] = E_K(M[i] \oplus C[i-1])$ and $C[0] = \text{IV}$. Many standards and many books (e.g., Schneier, *Applied Cryptography*, 2nd edition, p. 194]) suggest that the IV may be a fixed value, a counter, a timestamp, or the last block of ciphertext from the previous message. But if it is any of these things one certainly will not achieve any of the standard definitions of security [3, 16].

It is sometimes suggested that a mode which needs a random IV is preferable to one that needs a nonce: it is said that *state* is needed for a nonce, but not for making random bits. We find this argument wrong. First, a random value of sufficent length can always be used as a nonce, but a nonce can not be used as a random value. Second, the manner in which systems provide "random" IVs is invariably stateful anyway: unpredictable bits are too expensive to harvest for each IV, so one does this rarely, using state to generate pseudorandom bits from unpredictable bits harvested before. Third, the way to generate pseudorandom bits needs to use cryptography, so the prevalence of non-cryptographic pseudorandom number generators routinely results in implementation errors. Next, nonce-based schemes facillitate replay-detection with constant space and no added cryptography. Finally, nonces can be communicated using fewer bits, without additional cryptography.

ON-LINE. OCB encryption and decryption are "on line" in the sense that one does not need to know the length of the message in advance of encrypting or decrypting it. Instead, messages can be processed as one goes along, using constant memory, continuing until there is an indication that the message is over.

ENDIAN NEUTRALITY. In contrast to a scheme based on mod-p arithmetic (for p a prime just less than 2^n) or mod-2^n arithmetic, there is almost no endian-favoritism implicit in the definition of OCB. (The exception is that, because of our use of standard mathematical conventions, the left shift used for forming $L(i+1)$ from $L(i)$ is more convenient under a big-endian convention, as is the right shift used for forming $L(-1) = L \cdot \mathtt{x}^{-1}$ from L.)

OPTIONAL PRE-PROCESSING. Implementations can choose how many $L(i)$ values to precompute. As only one block-cipher call is needed to compuet these values from K, plus some shifts and conditional xors, it is feasible to do no preprocessing: OCB-AES is appropriate even when each session is a single, short message.

PROVABLE SECURITY. Provable security has become a popular goal for practical protocols. This is because it provides the best way to gain assurance that a cryptographic scheme does what it is should. For a scheme which enjoys provable security one does not need to consider attacks on the scheme, since successful ones imply successful attacks on some simpler object.

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

Provable security begins with Goldwasser and Micali [16], 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 [3, 5, 6].

It is not enough to know that there is a 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 roughly summarized as follows. An adversary can always forge with probability $1/2^\tau$. Beyond this, the maximal added advantage is at most $\sigma^2/2^n$, 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 [3, 6].

COMPARISON WITH JUTLA'S BOUND. More precisely, but still ignoring lower-order terms, our privacy and authentic-

ity bounds are $1.5\,\sigma^2/2^n$, while Jutla's authenticity bound is insignificantly worse at $2\,\sigma^2/2^n$ and his privacy bound, rescaled to $[0, 1]$, looks insignificantly worse at $3\,\sigma^2/2^n$ [20]. Magnifying the latter difference is that the privacy results assume different defintions. Jutla adopts the find-then-guess definition of privacy [3, 16], while we use an indistinguishability-from-random-bits definition. The former captures an adversary's inability to distinguish ciphertexts for a pair adversarilly-selected, equal-length plaintexts. The latter captures an adversary's inability to distinguish a ciphertext from a random string of the same length. Indistinguishability-from-random-bits implies find-then-guess security, and by a tight reduction, but find-then-guess secure does not imply indistinguishability-from-random-bits. Still, Jutla's scheme probably satisfies the stronger definition.

Simplicity. Simplicity has been a central design goal. Some of OCB's characteristics that contribute to simplicity are:

- Short and full final-message-blocks are handled without making a special case: the treatment of all messages is uniform, regardless of their length.
- 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. This 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 GF(2^n).
- In forming the sequence of offsets, Gray-code coefficients are taken monotonically, starting at 1 and stopping at m. One never goes back to some earlier offset, uses a peculiar starting point, or forms more offsets than there are blocks.

Not fixing how the nonce is communicated. We do not specify how the nonce is chosen or communicated. Formally, it is not part of the ciphertext (though the receiving party needs it to decrypt). In many contexts, there is already a natural value to use as a nonce (e.g., a sequence number already present in a protocol flow, or implicit because the parties are communicating over a reliable channel). Even when a protocol is designed from scratch, the number of bits needed to communicate the nonce will vary.

Not fixing the tag length. The number of bits necessary for the tag vary according to the application. In a context where the adversary obtains something quite valuable from a successful 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 many frames to have a major impact on the image, an 8-bit tag may be appropriate.

Forming R using a block-cipher call. During our work we discovered that there are methods for authenticated-encryption which encrypt M using $\lceil |M|/n \rceil + 1$ block-cipher calls, as opposed to our $\lceil |M|/n \rceil + 2$ calls. Shai Halevi has also made this finding [17]. However, the methods we know to shave off a block-cipher call either require an unpredictable IV instead of a nonce, or they add conceptual and computational complexity to compute the initial offset R by non-cryptographic means (e.g., using a finite-field multiplication of the nonce and a key variant).

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 [15]. 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 several 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 and we wanted OCB-AES to inherit this attribute.
- Addition is unpleasant for implementations using high-level languages, where one normally has no access to the add-with-carry instruction the machine may have.
- Addition needs more chip area than xor.
- Addition 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.
- The concrete security bound appears to be worse with addition than xor (though still not bad). The degradation would seem to be $\Theta(\lg \bar{m})$, where $\bar{m}$ is the maximal message length.

We eventually came to feel that even the simplicity benefit of addition was not quite real: these schemes seem harder to understand, to prove correct, and to implement well.

Lazy mod-p addition. Let p be the largest prime less than 2^n. An earlier design [31] allowed one to produce offset $Z[i]$ from $Z[i-1]$ by "lazy mod-p addition": add L to $Z[i-1]$, mod 2^n, and then add $\delta = 2^n - p$ whenever the first addition generates a carry. Now $X[m]$ would be defined by $\mathrm{len}(M[m]) \oplus \overline{Z[m]}$, say, where $\overline{Z[m]}$ is the bitwise complement of $Z[m]$. It appears that, unlike a mod-2^n scheme, xors can still be used to combine offsets with message blocks and enciphered message blocks. This might make lazy mod-p approach more attractive than a mod-2^n approach. But in order to propagate a single scheme, avoid endian favoritism, and avoid complicating an already complex proof, we chose not to propagate lazy mod-p-addition.

Definition of the Checksum. An initially odd-looking aspect of OCB's definition is the definition of 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 $M[1] \oplus \cdots M[m-1] \oplus M[m]$. We comment that these two definitions are identical in the case that $|M[m]| = n$. What is more, the definition 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 [31], allowed the adversary to force a "pretag collision." Recall that we compute the tag T by computing a "pretag" $X[m+1] = \text{Checksum} \oplus \text{SomeOffset}$, forming a value $Y[m+1] = E_K(X[m+1])$, and then forming T by doing further processing to $Y[m+1]$. 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 encrypted, getting $\bar{C}\,\bar{T}$, and then a forgery attempt N, $C\,T$ can be made such that, in it,

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 designed OCB so that an adversary can not force pretag collisions. The presence of pretag collisions substantially complicates proofs, since one can not follow a line of argument that shows that tags are unpredictable because each pretag-value is almost certainly new. For schemes like IAPM, where pretag collisions can be forced, this intuition is simply wrong. Beyond this, in the presence of pretag collisions one must modify $Y[m+1]$ by an amount Δ that depends on at least the key and nonce. Say that the modification is by xor, and one wants to be able to pull off an arbitrary bit as a 1-bit authentication tag. Then every bit of Δ will have to be adversarially unpredictable. This is unfortunate, as many natural ways to make Δ fail to have this property. Suppose, for example, the first couple bits of L are forced to zero, as suggested by [31], and $\Delta = L \cdot (m+1)$. Then, for small m, the first bit of Δ will be zero. This can be exploited to give an attack on the xor-based scheme of [31] when $\tau = 1$. Similarly, for i a power of two, $\Delta = iL \bmod 2^n$ ends in a 0-bit, so had [31] taken the tag to be the last τ bits instead of the first τ bits, one would again have an attack on 1-bit tags. A scheme would be arcane, at best, if certain bits of the full tag are usable and other bits are not.

Block-cipher circuit-depth. One further efficiency measure is the circuit depth of an encryption scheme as measured in terms of block-cipher gates. For OCB encryption, this number is three: a call to form R; calls to form the ciphertext core; and a call to compute the tag. Block-cipher circuit-depth serves as a lower bound for latency in an agressively parallel environment. Reducing the block-cipher circuit-depth to one or two is possible, but the benefit does not seem worth the associated drawbacks.

5. THEOREMS

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 [3]. A (nonce-using, symmetric) encryption scheme 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 $K \in \mathcal{K}$, $N \in \{0,1\}^n$, and $M \in \{0,1\}^*$, and returns a string $\mathcal{C} \leftarrow \mathcal{E}_K(N, M)$. Decryption algorithm $\mathcal{D}$ takes $K \in \mathcal{K}$, $N \in \{0,1\}^n$, 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 give a particularly strong definition of privacy, one asserting indistinguishability from random strings. This notion is easily seen to imply more standard definitions [3], 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 $\mathcal{C} \leftarrow \mathcal{E}_K(N, M)$. Assume that $|\mathcal{C}| = \ell(|M|)$ depends only on $|M|$. A fake encryption oracle, $\$(\cdot,\cdot)$, takes as input N, M and returns a random string $\mathcal{C} \xleftarrow{R} \{0,1\}^{\ell(|M|)}$. Given adversary A and encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, define $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) = \Pr[K \xleftarrow{R} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot)} = 1] - \Pr[K \xleftarrow{R} \mathcal{K} : A^{\$(\cdot,\cdot)} = 1]$.

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 [7, 8, 22]. 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. Adversary A *forges* (in this run) if A is nonce-respecting, A outputs $(N, \mathcal{C})$ where $\mathcal{D}_K(N, \mathcal{C}) \neq$ Invalid, and A made no earlier query (N, M) which resulted in a response $\mathcal{C}$. Let $\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A) = \Pr[K \xleftarrow{R} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot)}$ 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 $\mathrm{Rand}(n)$ denote the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$ and let $\mathrm{Perm}(n)$ denote the set of all permutations from $\{0,1\}^n$ to $\{0,1\}^n$. These sets can be regarded as function families by imagining that each member is specified by a string. For $\pi \in \mathrm{Perm}(n)$, let $\pi^{-1}(Y)$ be the unique string X such that $\pi(X) = Y$. Let $\mathbf{Adv}^{\mathrm{prf}}_{E}(A) = \Pr[A^{E_K(\cdot)} = 1] - \Pr[A^{\rho(\cdot)} = 1]$, $\mathbf{Adv}^{\mathrm{prp}}_{E}(A) = \Pr[A^{E_K(\cdot)} = 1] - \Pr[A^{\pi(\cdot)} = 1]$, and $\mathbf{Adv}^{\mathrm{sprp}}_{E}(A) = \Pr[A^{E_K(\cdot), E_K^{-1}(\cdot)} = 1] - \Pr[A^{\pi(\cdot), \pi^{-1}(\cdot)} = 1]$, where the probability is over $K \xleftarrow{R} \mathcal{K}$, $\rho \xleftarrow{R} \mathrm{Rand}(n)$, and $\pi \xleftarrow{R} \mathrm{Perm}(n)$.

5.2 Theorem Statements

We give information-theoretic bounds on the authenticity and the privacy of OCB. Proofs are in the full paper [33].

Theorem 1. *Fix OCB parameters n and τ. 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 at most c blocks. Let $\bar{\sigma} = \sigma + 2q + 5c + 11$. Then*

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \;\leq\; \frac{1.5\,\bar{\sigma}^2}{2^n} + \frac{1}{2^\tau}$$

The aggregate length of queries $M_1, \ldots, M_q$ means the number $\sigma = \sum_{r=1}^{q} \|M_r\|_n$.

It is standard to pass to a complexity-theoretic analog of Theorem 1, but in doing this 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 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 at most c blocks. Let $\bar{\sigma} = \sigma + 2q + 5c + 11$. Let $\delta = \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[E,\tau]}(A) - 1.5\,\bar{\sigma}^2/2^n - 1/2^\tau$. Then there is an adversary B for attacking block cipher E that achieves advantage $\mathbf{Adv}^{\mathrm{sprp}}_{E}(B) \geq \delta$. Adversary B asks at most $q' = \sigma + 2q + c + 3$ 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 $\alpha n\bar{\sigma}$, where the constant α depends only on details of the model of computation.

The privacy of OCB is given by the following result.

THEOREM 2. *Fix OCB parameters n and τ. Let A be an adversary that asks q queries, these having aggregate length of σ blocks. Let $\bar{\sigma} = \sigma + 2q + 3$. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \;\leq\; \frac{1.5\,\bar{\sigma}^2}{2^n}$$

It is standard to pass to a complexity-theoretic analog of Theorem 2. 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 A be an adversary that asks q queries, these having aggregate length of σ blocks. Let $\bar{\sigma} = \sigma + 2q + 3$. Let $\delta = \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[E,\tau]}(A) - 1.5\,\bar{\sigma}^2/2^n$. 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 $\alpha n\bar{\sigma}$, where the constant α depends only on details of the model of computation.

5.3 Proofs

To prove Theorem 1 (Theorem 2 is comparatively easy) we give a *structure lemma* that relates the authenticity of OCB to three functions: the M-collision probability, the MM-collision probability, and the CM-collision probability. Very informally, these measure the probability of trouble when the adversary asks a single query, some pair of queries, and when adversary tries to forge some ciphertext following a single earlier query. Due to space limitations, all of these definitions, lemmas and proofs are removed from this proceedings paper. They can be found in the full paper [33].

6. PERFORMANCE

ABSTRACT ACCOUNTING. OCB uses $\lceil |M|/n \rceil + 2$ block-cipher calls to encrypt a nonempty message M. (The empty string takes three block-cipher calls.) We compare this with CBC encryption and CBC encryption plus a CBC MAC:

- "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 sets IV $= E_K(N)$ for CBC encryption (so both schemes use a not-necessarily-random nonce), and uses obligatory 10^* padding (so 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 many 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

Algorithm	64 B	256 B	1 KB	4 KB
OCB encrypt	24.7	18.5	**16.9**	16.7
ECB encrypt	15.1	15.0	**14.9**	14.9
CBC encrypt	15.9	15.9	**15.9**	15.9
CBC mac	19.2	16.3	**15.5**	15.3

Figure 3: Performance results from Lipmaa [25], in cycles per byte on a Pentium III. The block cipher is AES128. Code is written in assembly.

As with any mode, there is 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 precomputed, 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 shift 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 $\sum_{i=1}^{m} \mathtt{ntz}(i)$, which is less than m. The above assumes that one does not retain or precompute any $L(i)$ value beyond $L = L(0)$. Suppose that one precomputes $L(-1)$, $L(0)$, $L(1)$, $L(2)$, $L(3)$. Computing and saving the four values beyond $L = L(0)$ is cheaper than computing L itself, which required an application of E_K. But now the desired multiple of L will have be available at least $1/2+1/4+1/8+1/16 \approx 94\%$ of the time. When it has not been precomputed it must be calculated, starting from $L(3)$, so the amortized number of multiplications by $\mathtt{x}$ has been reduced to $\sum_{i=1}^{\infty} = i/2^{i+4} = 0.125$.

EXPERIMENTAL RESULTS. In Table 3 we report, with permission, some experimental results by Helger Lipmaa [25]. On a Pentium III, in optimized assembly, Lipmaa implemented OCB encryption, ECB encryption, CBC encryption, and the CBC MAC. The last three modes were implemented in their "raw" forms, where one does no padding and assumes that the message acted on is a positive multiple of the block length. For CBC encryption, the IV is fixed. The underlying block cipher is AES128.

Focusing on messages of 1 KByte, OCB incurs about 6.4% overhead compared to CBC encryption, and the algorithm takes about 54% of the time of a CBC encryption + CBC MAC. Lipmaa points out that overhead is so low that, in his experiments, an assembly AES128 with a C-code CBC-wrapper is slightly slower than the same AES128 with an assembly OCB-wrapper. Lipmaa's (size-unoptimized) code is 7.2 KBytes, which includes unrolling AES128 (2.2 KBytes) three times.

Some aspects of the experiments above are unfavorable to OCB, making the performance estimates conservative. In particular, the "raw" CBC MAC 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.

And when combined with CBC encryption, the CBC MAC should be taken over the full ciphertext, including the nonce, which would add an extra block-cipher call. Finally, an extra block-cipher call would normally be performed by CBC to correctly compute the IV from a nonce.

The results above are for a serial execution environment. In settings with plenty of registers and multiple instruction pipes, OCB, properly implemented, will be faster than CBC.

Acknowledgments

At CRYPTO '00, Virgil Gligor described [12] to Rogaway, Charanjit Jutla gave a rump-session talk on [18], and Elaine Barker announced a first modes-of-operation workshop organized by NIST. These events inspired [31], which evolved into the current work. After the first workshop NIST made a second call for proposals, and OCB took its final form in response to this call [32]. We appreciate NIST's effort to solicit and evaluate modern modes of operation. Elaine Barker, Morris Dworkin, and Jim Foti are among those involved.

We received useful feedback from Michael Amling, Paulo Barreto, Johan Håstad, Hugo Krawczyk, Helger Lipmaa, David McGrew, Silvio Micali, Ilya Mironov, Alberto Pascual, Bart Preneel, Tom Shrimpton, and David Wagner. Special thanks to Michael and Ilya for their careful proofreading, and Helger for doing a state-of-the-art assembly implementation, along with providing associated timing data. Thanks to the anonymous ACM CCS referees for their comments.

This work was carried out 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 research, but the submitter gratefully acknowledges a recent gift by Cisco Systems. Many thanks to Cisco for their support.

7. REFERENCES

[1] J. An and M. Bellare. Does encryption with redundancy provide authenticity? *Advances in Cryptology – EUROCRYPT 2001*. Lecture Notes in Computer Science, vol. 2045, B. Pfitzmann, ed., Springer-Verlag, 2001. www-cse.ucsd.edu/users/mihir

[2] K. Aoki and H. Lipmaa. Fast implementations of AES candidates. Third AES Candidate Conference, New York City, USA, Apr 2000, pp. 106–120. www.tml.hut.fi/∼helger

[3] 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. www.cs.ucdavis.edu/∼rogaway

[4] 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. www.cs.ucdavis.edu/∼rogaway

[5] 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. www.cs.ucdavis.edu/∼rogaway

[6] 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.

[7] 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, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. www-cse.ucsd.edu/users/mihir

[8] 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, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. www.cs.ucdavis.edu/∼rogaway

[9] 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. www.bell-labs.com/user/bleichen

[10] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. on Comp.*, vol. 30, no. 2, pp. 391–437, 2000.

[11] V. Gligor and P. Donescu. Integrity-aware PCBC encryption schemes. *Security Protocols, 7th International Workshop, Cambridge, UK, April 1999 Proceedings*. Lecture notes in Computer Science, vol. 1796, Springer-Verlag, pp. 153–171, 2000.

[12] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Manuscript, Aug 18, 2000. Formerly available from www.eng.umd.edu/∼gligor.

[13] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Contribution to NIST, Oct 27, 2000. csrc.nist.gov/encryption/aes/modes

[14] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Contribution to NIST. Mar 30, 2001, rev. Apr 20, 2001. csrc.nist.gov/encryption/modes/proposedmodes

[15] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. *Fast Software Encryption*, Lecture Notes in Computer Science, Springer-Verlag, Apr 2001.

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

[17] S. Halevi. An observation regarding Jutla's modes of operation. Cryptology ePrint archive, reference number 2001/015, submitted Feb 22, 2001, revised Apr 2, 2001. eprint.iacr.org

[18] C. Jutla. Encryption modes with almost free message integrity. Cryptology ePrint archive, report 2000/039, Aug 1, 2000. eprint.iacr.org

[19] C. Jutla. Encryption modes with almost free message integrity. Contribution to NIST. Undated manuscript, appearing Oct 2000 at

csrc.nist.gov/encryption/modes/workshop1

[20] C. Jutla. Encryption modes with almost free message integrity. Contribution to NIST. Posted May 24, 2001 at csrc.nist.gov/encryption/modes/proposedmodes

[21] C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology – EUROCRYPT 2001*. Lecture Notes in Computer Science, vol. 2045, B. Pfitzmann, ed., Springer-Verlag, 2001.

[22] 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.

[23] J. Katz and M. Yung. Complete characterization of security notions for probabilistic private-key encryption. STOC 2000, pp. 245–254, 2000.

[24] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). Advances in Cryptology — CRYPTO '01. Springer-Verlag, 2001. Earlier version as ePrint report 2001/045, Jun 6, 2001. eprint.iacr.org/20001/045

[25] H. Lipmaa. Personal communications, Jul 2001. Further information at www.tcs.hut.fi/∼helger

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

[27] M. Matyas and S. Matyas. *Cryptography: A new dimension in computer data security*. John Wiley & Sons, New York, 1982.

[28] RSA Laboratories. PKCS #1: RSA encryption standard, Version 1.5, Nov 1993; and PKCS #1: RSA cryptography specifications, Version 2.0, Sep 1998, B. Kaliski and J. Staddon. www.rsasecurity.com/rsalabs/pkcs/pkcs-1

[29] J. Steiner, C. Neuman, and J. Schiller. Kerberos: an authentication service for open network systems. *Proceedings of the Winter 1988 Usenix Conference*, pp. 191–201, 1988.

[30] 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.

[31] P. Rogaway. OCB mode: Parallelizable authenticated encryption. Contribution to NIST, Oct 16, 2000. (Preliminary version of the OCB algorithm.) csrc.nist.gov/encryption/modes/workshop1

[32] P. Rogaway (submitter) and M. Bellare, J. Black, and T. Krovetz (auxiliary submitters). OCB mode. Contribution to NIST. Cryptology ePrint archive, report 2001/26, Apr 1, 2001, revised Apr 18, 2001. ePrint.iacr.org and csrc.nist.gov/encryption/modes/proposedmodes.

[33] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. Full version of this paper. Aug 2001. www.cs.ucdavis.edu/∼rogaway

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

APPENDIX

A. BRIEF HISTORY

Jutla, Gligor-Donescu, Rogaway. An April 1999 paper by Gligor and Donescu gives an authenticated-encryption scheme called PCBC [11]. The mode is wrong, as pointed out by Jutla [18], but it may have contributed to the subsequent development of correct modes. Jutla's paper [18] gives the first apparently correct schemes, IACBC and IAPM. Shortly after that paper appeared, Gligor and Donescu described a different scheme, XCBC [12], which is similar to IACBC. The most conspicuous difference between XCBC and IACBC is the former's use of mod-2^n addition where the latter uses xor or mod-p addition, for p a prime just less than 2^n.

A first call by NIST for modes of operation brought contributions [13, 19] based on [12, 18], and a contribution by Rogaway [31] that built on [18]. In [19], Jutla employs a Gray-code ordering for combining basis offsets, a refinement independently introduced, along with further tricks, in [31].

A second call by NIST gave rise to [14, 20, 32], which were revisions to [13, 19, 31], respectively. In [20], Jutla emphasized IAPM over IACBC, and he adopted lazy mod-p addition, first described in [31]. In [14], Gligor and Donescu describe four authenticated-encryption modes, one of which, XECBS-XOR, is parallelizable. The modes adopt some features introduced in [31] to deal with messages of arbitrary length and to use a single block-cipher key. In [32], Rogaway et al. settled on one mechanism to make offsets (three are described in [31]) and made further refinements to [31].

Briefly comparing OCB and IAPM, the latter uses two separate keys and is defined only for messages which are a multiple of the block length. Once a padding regime is included, say obligatory 10^* padding, ciphertexts will be longer than OCB's by 1 to n bits. IAPM supports offset-production using either lazy mod-p addition or an xor-based scheme. The latter is not competitive with OCB in terms of key-setup costs.

The initial version of Jutla's work [18] claimed a proof, and included ideas towards one. A subsequent writeup by Halevi [17] was more rigorous.

Patents. The history above ignores associated patent applications. Jutla/IBM, Gligor/VDG, and Rogaway have all indicated that there were such filings. All parties have provided statements to NIST promising reasonable and nondiscriminatory licensing.

Definitions. Though the authenticated-encryption goal is folklore, provable-security treatments of it are recent. The first definition for authenticated encryption is due to Bellare and Rogaway [8] and, independently, Katz and Yung [22]. Bellare and Namprempre were the first to seriously investigate the properties of authenticated-encryption and the generic-composition paradigm [7].