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

Phillip Rogaway
UC Davis
This work done at Chiang Mai University
rogaway@cs.ucdavis.edu
http://www.cs.ucdavis.edu/~rogaway

Mihir Bellare
UC San Diego

John Black
University of Nevada

Ted Krovetz
UC Davis

CCS-8 — Philadelphia, USA — November 8, 2001
Principal Goals of Symmetric Cryptography

**Privacy**  What the Adversary sees tells her nothing of significance about the underlying message $M$ that the Sender sent

**Authenticity**  The Receiver is sure that the string he receives was sent (in exactly this form) by the Sender

---

**Authenticated Encryption**  Achieves both privacy and authenticity
Why Authenticated Encryption?

- **Efficiency**
  By merging privacy and authenticity one can achieve efficiency difficult to achieve if handling them separately.

- **Easier-to-correctly-use abstraction**
  By delivering strong security properties one may minimize encryption-scheme misuse.
Easier to correctly use because stronger security properties

- Idealized encryption
  - OCB
    - Authenticated encryption
      - IND-CPA + auth of ciphertexts
    - IND-CCA = NM-CCA
  - CTR, CBC$
    - IND-CPA
  - ECB

References:

- [Bellare, Rogaway]
- [Katz, Yung]
- [Bellare, Namprempre]
- [Goldwasser, Micali]
- [Bellare, Desai, Jokipii, Rogaway]
Right or Wrong?

It depends on what definition \( E \) satisfies
Generic Composition

Traditional approach to authenticated encryption

Glue together an encryption scheme ($E$) and a message authentication code (MAC)

Preferred way to do generic composition:
Generic Composition

+ Versatile, clean approach
+ Reduces design work
+ Quick rejection of forged messages if use optimized MAC (eg., UMAC)
+ Inherits the characteristics of the modes one builds from

- Cost ≈ (cost to encrypt) + (cost to MAC)
  For CBC Enc + CBC MAC, cost ≈ 2 × (cost to CBC Enc)
- Often done wrong
- Two keys
- Inherits characteristics of the modes one builds from
Trying to do Better

• Numerous attempts to make privacy + authenticity cheaper.
• One approach: stick with generic composition, but find cheaper. encryption schemes or MACs.
• Make authenticity an “incidental” adjunct to privacy within a conventional-looking mode:
  • CBC-with-various-checksums (wrong)
  • PCBC in Kerberos (wrong)
  • PCBC of [Gligor, Donescu 99] (wrong)
  • [Jutla 00] First correct solution
• Jutla described two modes, IACBC and IAPM.
• A lovely start, but many improvements possible.
• OCB: inspired by IAPM, but many new characteristics.
Additional Related Work

• [Halevi]—improved on Jutla’s IAPM proof and helped to clarify what was going on in the scheme.

• [Gligor, Donescu]—Proposed IACBC-like scheme, using mod $2^n$ addition.
What is OCB?

- Authenticated-encryption scheme
- Uses any block cipher (eg. AES)
- Computational cost $\approx$ cost of CBC
- OCB-AES good in SW or HW
- Lots of nice characteristics designed in:
  - Uses $\left\lceil \frac{|M|}{n} \right\rceil + 2$ block-cipher calls
  - Uses any nonce (needn’t be unpredictable)
  - Works on messages of any length
  - Creates minimum-length ciphertext
  - Uses a single block-cipher key, each block-cipher keyed with it
  - Quick key setup – suitable for single-message sessions
  - Essentially endian-neutral
  - Fully parallelizable
  - No $n$-bit additions
- Provably secure: if you break OCB-AES you’ve broken AES
- In IEEE 802.11 draft standard (wireless LANs)
Checksum = $M[1] \oplus M[2] \oplus \cdots \oplus M[m-1] \oplus C[m]0^* \oplus \text{Pad}$

$L = E_K(0)$
Pseudocode of OCB[E, τ]

algorithm OCB-Encrypt_ (Nonce, M)
L(0) = E_K (0)
L(-1) = lsb(L(0))? (L(0) >> 1) ⊕ Const43 : (L(0) >>1)
for i = 1, 2, … do L(i) = msb(L(i-1))? (L(i-1) << 1) ⊕ Const87 : (L(i-1) <<=1)
Partition M into M[1] … M[m] // each n bits, except M[m] may be shorter
Offset = E_K (Nonce ⊕ L(0))
for i=1 to m-1 do
    Offset = Offset ⊕ L(ntz(i))
    C[i] = E_K (M[i] ⊕ Offset) ⊕ Offset
Offset = Offset ⊕ L(ntz(m))
Pad = E_K (len(M[m]) ⊕ Offset ⊕ L(-1))
C[m] = M[m] ⊕ (first | M[m] | bits of Pad)
Checksum = M[1] ⊕ … ⊕ M[m-1] ⊕ C[m]0* ⊕ Pad
Tag = first τ bits of E_K(Checksum ⊕ Offset)
return C[1] … C[m] || Tag
Many natural-looking variants are wrong, eg.,
  • Eliminate post-whitening
  • Checksum = M[1] ⊕ M[2] ⊕ ... ⊕ M[m-1] ⊕ C[m]0*
  • Use offsets Z[1], ..., Z[m-1], Z[m], Z[m+1]
  • Use offsets Z[1], ..., Z[m-1], Z[m], Z[-m]
  ...
Assurance via Provable Security

- Provable security begins with [Goldwasser, Micali 82]
- Despite the name, one doesn’t really prove security
- Instead, one gives reductions: theorems of the form
  
  If a certain primitive is secure
  then the scheme based on it is secure.

Eg:

If AES is a secure block cipher
then OCB-AES is a secure authenticated-encryption scheme.

Equivalently:

If some adversary A does a good job at breaking OCB-AES
then some comparably efficient B does a good job to break AES.

- Actual theorems quantitative: they measure how much security is “lost” across the reduction.
Privacy
IND$-CPA: Indistinguishability from Random Bits

\[ \text{Adv}^{\text{priv}}(A) = \Pr[A^{\text{Real}} = 1] - \Pr[A^{\text{Rand}} = 1] \]
Authenticity
Authenticty of Ciphertexts

Adversary A forges if she outputs Nonce C s.t.
- C is valid (it decrypts to a message, not to invalid)
- there was no earlier query Nonce Mi that returned C

\[ \text{Adv}^{\text{auth}}(A) = \Pr[A \text{ forges}] \]
Block-Cipher Security

PRP and Strong PRP

\[ \text{Adv}^{\text{prp}}(B) = \Pr[B^{E_K} = 1] - \Pr[B^{\pi} = 1] \]

\[ \text{Adv}^{\text{sprp}}(B) = \Pr[B^{E_K E_K^{-1}} = 1] - \Pr[B^{\pi \pi^{-1}} = 1] \]
OCB Theorems

Privacy theorem:

Suppose \( \exists \) an adversary \( A \) that distinguishes \( \text{OCB}[E, \tau] \) in:

- \( \text{time} = t \)
- \( \text{total-num-of-blocks} = \sigma \)
- \( \text{adv} = \text{Adv}^{\text{priv}}(A) \)

Then \( \exists \) an adversary \( B \) that breaks block cipher \( E \) with:

- \( \text{time} \approx t \)
- \( \text{num-of-queries} \approx \sigma \)
- \( \text{Adv}^{\text{prp}}(B) \approx \text{Adv}^{\text{priv}}(A) - 1.5 \sigma^2 / 2^n \)

Authenticity theorem:

Suppose \( \exists \) an adversary \( A \) that forges \( \text{OCB}[E, \tau] \) with:

- \( \text{time} = t \)
- \( \text{total-num-of-blocks} = \sigma \)
- \( \text{adv} = \text{Adv}^{\text{auth}}(A) \)

Then \( \exists \) an adversary \( B \) that breaks block cipher \( E \) with:

- \( \text{time} \approx t \)
- \( \text{num-of-queries} \approx \sigma \)
- \( \text{Adv}^{\text{sprp}}(B) \approx \text{Adv}^{\text{auth}}(A) - 1.5 \sigma^2 / 2^n - 2^{-\tau} \)
Assembly Speed

Data from Helger Lipmaa  www.tcs.hut.fi/~helger  helger@tcs.hut.fi

// Best Pentium AES code known.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Speed</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCB-AES</td>
<td>16.9 cpb</td>
<td>(271 cycles)</td>
</tr>
<tr>
<td>CBC-AES</td>
<td>15.9 cpb</td>
<td>(255 cycles)</td>
</tr>
<tr>
<td>ECB-AES</td>
<td>14.9 cpb</td>
<td>(239 cycles)</td>
</tr>
<tr>
<td>CBCMAC-AES</td>
<td>15.5 cpb</td>
<td>(248 cycles)</td>
</tr>
</tbody>
</table>

6.5 % slower

1 Kbyte messages—pure Pentium 3 assembly—AES128.
Overhead so small that AES with a C-code CBC wrapper is slightly more expensive than AES with an assembly OCB wrapper.

C Speed

Data from Ted Krovetz. Compiler is MS VC++. Uses rijndael-alg-fst.c ref code.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Speed</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCB-AES</td>
<td>28.1 cpb</td>
<td>(449 cycles)</td>
</tr>
<tr>
<td>CBCMAC-AES</td>
<td>26.8 cpb</td>
<td>(428 cycles)</td>
</tr>
</tbody>
</table>

4.9 % slower
Why I like OCB 😊

- **Ease-of-correct-use.** Reasons: all-in-one approach; any type of nonce; parameterization limited to block cipher and tag length
- **Aggressively optimized**: ≈ optimal in many dimensions: key length, ciphertext length, key setup time, encryption time, decryption time, available parallelism; SW characteristics; HW characteristics; …
- **Simple but non-obvious**
- Ideal setting for **practice-oriented provable security**

For More Information

  Contains FAQ, papers, reference code, ...