OCB : Parallelizable Authenticated Encryption

PMAC : Parallelizable Message Authentication Code

Phillip Rogaway

UC Davis (USA) and
CMU (Thailand)

rogaway@cs.ucdavis.edu
www.cs.ucdavis.edu

with assistance from Mihir Bellare (UCSD)
and John Black (UNR)

NIST Modes of Operation Workshop - 20 October 2000
What I'm doing

**OCB** - Refining a parallelizable scheme recently suggested by [Jutla] for authenticated encryption (privacy+authenticity)

**PMAC** - Improving on [Bellare, Guerin, Rogaway], [Bernstein], [Gligor, Donescu] for a parallelizable MAC.
OCB (Offset CodeBook) Mode

Security Goals

(1) The adversary can't understand anything about plaintexts
Formalized as $\text{IND - CPA}$ [GM, BDJR]

Formalized as:

\[
\mathcal{E}_K (0^{\left| M \right|}) \xrightarrow{M} C \xrightarrow{A} \mathcal{E}_K (\cdot)
\]

(2) The adversary can't produce valid ciphertexts
Formalized as $\text{Integrity of Ciphertexts}$ [KY, BR, BN]

Formalized as:

\[
\mathcal{E}_K (\cdot) \xrightarrow{\mathcal{E}_K (\cdot)} M_1 \xrightarrow{C_1} A \xrightarrow{\mathcal{E}_K (\cdot)} \ldots \\
C^* \xrightarrow{\mathcal{E}_K (\cdot)} M_q \xrightarrow{C_q}
\]
Why is Integrity-of-Ciphertexts important?

Because users of encryption often assume, wrongly, that they have it! Achieving IND-CPA + integrity-of-ciphertexts implies IND-CCA [BN] and non-malleability-CCA, so an encryption scheme with Integrity-of-Ciphertexts is far less likely to be misused.

This sort of encryption-scheme usage, to bind together a private message, is very common in the literature and in practice. But is completely bogus when using IND-CPA encryption.
**OCB (full final block)**

```
Nonce

M[1] + 1L + R

E_K

M[2] + 2L + R

M[3] + 3L + R

E_K


E_K


R

C[1]

C[2]

C[3]

C[4]

Tag
```
OCB (short final block)
procedure Encrypt(  , Nonce,  M)

$L = E_K(1^{128}) \lor 0^{127}1$  // Do during key-setup
$R = E_K(Nonce)$

Let $m = \max\{1, \lfloor |M|/128 \rfloor\}$
Let $M[1], \ldots, M[m]$ be strings s.t. $M[1] \cdot \cdot M[m] = M$ and $|M[i]| = 128$ for $1 \leq i < m$
Offset = $L + R$
for $i = 1$ to $m - 1$ do
   $C[i] = E_K(M[i] \oplus$ Offset $) +$ Offset
   Offset = Offset + $L$
   if $|M[m]| = 128$ then
      Mask = $E_K($ Offset $) +$ Offset
      $C[m] = M[m] \oplus$ Mask
      Offset = Offset + $L$
      PreTag = $M[1] \oplus \cdot \cdot \oplus M[m - 1] \oplus M[m] +$ Offset
      Tag = $E_K($ PreTag $)$
   else
      $W =$ pad($M[m]$)
      Mask = $E_K($ Offset $) +$ Offset
      $C[m] = M[m] \oplus ($ last $|M[m]|$ bits of Mask $) +$ Offset
      Offset = Offset + $L$
      PreTag = $M[1] \oplus \cdot \cdot \oplus M[m - 1] \oplus W +$ Offset
      Offset = Offset + $L$
      Tag = $E_K($ PreTag $) +$ Offset

return (Nonce,  $C[1] \cdot \cdot C[m]$,  $T[1..tagLen]$ )
OCB Advantages

(1) Fully parallelizable - important for HW and SW
(2) Arbitrary domain - any bitstring can be encrypted
(3) Short ciphertexts - $|M| + |Nonce| + |T|$
(4) Fewer block-cipher calls - $\text{ceiling}\{\frac{|M|}{n}\} + 2$
(5) Nonces - counter is fine - needn't be unpredictable
(6) Short key - OCB defined as using one AES key
(7) Fast key setup - one AES invocation to make $L$
(8) Addition version - three 128-bit adds per block
   128-bit xor per block
(9) XOR version - four 128-bit xors per block,
Addition is less pleasant than you might think
- Add-with-carry unavailable from C
- Dependency among instructions slows things down

Offset($i+1$) = Offset($i$) $\text{xor } L(\text{ntz}(i))$
where $L(0) = L$ and

$$L(j+1) = \begin{cases} 
L(j) \ll 1 & \text{if lsb}(L(j)) = 0 \\
L(j) \ll 1 \text{ xor CONST} & \text{otherwise}
\end{cases}$$
PMAC (full final block)

\[ M[1] + 1L \]

\[ E_K \]
\[ C[1] \]

\[ M[2] + 2L \]

\[ E_K \]
\[ C[2] \]

\[ M[3] + 3L \]

\[ E_K \]
\[ C[3] \]

\[ M[4] + \]

\[ E_K \]

\[ \text{Final} (L) \]
\[ \text{Tag} \]
PMAC  (short final block)
PMAC Advantages

(1) Fully parallelizable - important for HW and SW
(2) Arbitrary domain - any bitstring can be MACed
(3) Deterministic - uses no nonces or random values
(4) Short MACs - up to 128 bits, but 64 bits is enough
(5) Fewer block-cipher calls - $\left\lceil \frac{|M|}{n} \right\rceil$
(6) Short key - PMAC defined as using one AES key
(7) Fast key setup - one AES invocation to make $L$
(8) Addition version - two 128-bit adds per block
(9) XOR version - three 128-bit xors per block,

one 128-bit xor