PMAC: A Parallelizable Message Authentication Code

John Black
Department of Computer Science
University of Nevada, Reno
jrb@cs.unr.edu
http://www.cs.unr.edu/~jrb

Phillip Rogaway
Department of Computer Science
UC Davis + CMU
rogaway@cs.ucdavis.edu
http://www.cs.ucdavis.edu/~rogaway
+66 1 530 7620  +1 530 753 0987

NIST Modes of Operation Workshop 2 – Aug 24, 2001 - Santa Barbara, California
What is a MAC

\[ \text{MAC}^G: \text{generate authentication tag} \]
\[ \sigma = \text{MAC}_K^G( [IV], M ) \]

\[ \text{MAC}^V: \text{verify authentication tag:} \]
\[ \text{MAC}_K^V ( M, \sigma ) \]

- Security addresses an adversary’s \textbf{inability} to forge a \textbf{valid} authentication tag for some \textbf{new} message.
- Most MACs are \textbf{deterministic}—they need no nonce/state/IV/$$. In practice, such MACs are preferable. Deterministic MACs are usually PRFs.
CBC MAC
Inherently sequential

\[ \text{M[1]} \rightarrow E_K \rightarrow \text{M[2]} \rightarrow E_K \rightarrow \ldots \rightarrow \text{M[m]} \rightarrow E_K \rightarrow \text{Tag} \]
PMAC’s Goals

- A fully parallelizable alternative to the CBC MAC
- But without paying much for parallelizability in terms of serial efficiency
- While we’re at it, fix up other “problems” of the CBC MAC
  - Make sure PMAC applies to any bit string
  - Make sure it is correct across messages of different lengths
What is PMAC?

- A variable-input-length pseudorandom function (VIL PRF):
  \[ \text{PMAC: } \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^n \]
- That you make from
  a fixed-input-length pseudorandom function (FIL PRF) –
  invariably a block cipher such as E=AES:
  \[ \text{E: } \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \]
PMAC’s Properties

- Functionality: VIL PRF: \( \{0,1\}^* \rightarrow \{0,1\}^n \) Can’t distinguish \( \text{PMAC}_K (\cdot) \) from a random function \( \text{R}(\cdot) \)
- Customary use of a VIL PRF:
  A (stateless, deterministic) Message Authentication Code (MAC)
- PRFs make the most pleasant MACs because they are deterministic and stateless.
- Few block-cipher calls: \( \left\lceil \frac{|M|}{n} \right\rceil \) to PMAC message \( M \)
- Low session-setup cost: about one block-cipher call
- Fully parallelizable
- No n-bit addition or mod \( p \) operations – just xors and shifts
- Uses a single block-cipher key
- Provably secure: If \( E \) is a secure block cipher then \( \text{PMAC-}E \) is a good PRF
if $|M[m]| < n$ then 0
if $|M[m]| = n$ then $z[-1]$
Definition of PMAC \([E, t]\)

**algorithm** \(\text{PMAC}_K(M)\)

\[\begin{align*}
L(0) &= E_K(0) \\
L(-1) &= \text{lsb}(L(0)) \oplus (L(0) \gg 1) \oplus \text{Const43} : (L(0) \gg 1) \\
\text{for } i = 1, 2, \ldots \text{ do } L(i) &= \text{msb}(L(i-1)) \oplus (L(i-1) \ll 1) \oplus \text{Const87} : (L(i-1) \ll 1)
\end{align*}\]

Partition \(M\) into \(M[1] \ldots M[m]\) // each 128 bits, except \(M[m]\) may be shorter

Offset = 0

\[\begin{align*}
\text{for } i=1 \text{ to } m-1 \text{ do } \\
\quad \text{Offset} &= \text{Offset} \oplus L(\text{ntz}(i)) \\
\quad \Sigma &= \Sigma \oplus E_K(M[i] \oplus \text{Offset}) \\
\Sigma &= \Sigma \oplus \text{pad}(M[m]) \\
\text{if } |M[m]| = n \text{ then } \Sigma &= \Sigma \oplus L(-1) \\
\text{FullTag} &= E_K(\Sigma) \\
\text{Tag} &= \text{first } t \text{ bits of } \text{FullTag} \\
\text{return} \text{ Tag}
\end{align*}\]
Related Work

- [Bellare, Guerin, Rogaway 95] – the XOR MAC.
  Not a PRF, but introduced central element of the construction
- [Bernstein 99] – A PRF-variant of the XOR MAC
- [Gligor, Donescu 00, 01] – Another descendent of the XOR MAC.
  Introduced the idea of combining message blocks with a sequence of offsets as an alternative to encoding. Not a PRF
- [Black, Rogaway 00] – Tricks for optimal handing of arbitrary input lengths (XCBC method you have just seen)
- [Carter-Wegman 79, 81] – A completely different approach that can achieve the same basic goals.
- Tree MAC (a la Merkle) – Another approach, not fully parallelizable.
The CBC MAC is in its “raw” form. Code is Pentium 3 assembly under gcc. This CBC MAC figure is inferior to Lipmaa’s OCB results, indicating that PMAC and OCB add so little overhead that quality-of-code differences contribute more to measured timing differences than algorithmic differences across CBC – CBCMAC – PMAC – OCB. Since Lipmaa obtained 15.5 cpb for the CBC MAC, adding 8% to this, 16.7 cpb, is a conservative estimate for well-optimized Pentium code.
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
For us:
  If AES is a secure block cipher
  then PMAC-AES is a secure authenticated-encryption scheme
Equivalently:
  If some adversary A does a good job at breaking PMAC-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.
Block-Cipher Security
Security as a FIL PRP

\[ \text{Adv}^{\text{prp}}(B) = \Pr[B^{E_K} = 1] - \Pr[B^\pi = 1] \]
PMAC’s Security
Security as a VIL PRF

$\text{Adv}^{\text{prf}}(A) = \Pr[A_{\text{PMAC}_K} = 1] - \Pr[A^R = 1]$
**PMAC Theorem**

<table>
<thead>
<tr>
<th>Suppose $\exists$ an adversary $A$ that breaks <strong>PMAC</strong>-$E$ with:</th>
<th>Then $\exists$ an adversary $B$ that breaks block cipher $E$ with:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$time = t$</td>
<td>$time \approx t$</td>
</tr>
<tr>
<td>$total$-$num$-$of$-$blocks = \sigma$</td>
<td>$num$-$of$-$queries \approx \sigma$</td>
</tr>
<tr>
<td>$adv = \text{Adv}^{\text{prf}}(A) \frac{\sigma^2}{2^n}$</td>
<td>$\text{Adv}^{\text{prp}}(B) \approx \text{Adv}^{\text{prf}}(A) - \sigma^2 / 2^{n-1}$</td>
</tr>
</tbody>
</table>

(To wrap up,

it is a standard result that any $\tau$-bit-output PRF can be used as a MAC, where the forging probability will be at most $\text{Adv}^{\text{prf}}(A) + 2^{-\tau}$)

[Goldreich, Goldwasser, Micali]
[Bellare, Kilian, Rogaway])
<table>
<thead>
<tr>
<th>Domain</th>
<th>PRF</th>
<th>MAC length</th>
<th>Parallelizable</th>
<th>#calls</th>
<th>Key bits</th>
<th>/ blk overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBCMAC</td>
<td>${0,1}^m$</td>
<td>$\checkmark$</td>
<td>$\tau$</td>
<td>$</td>
<td>M</td>
<td>/n$</td>
</tr>
<tr>
<td>XCBC</td>
<td>${0,1}^*$</td>
<td>$\checkmark$</td>
<td>$\tau$</td>
<td>$\left\lceil</td>
<td>M</td>
<td>/n \right\rceil$</td>
</tr>
<tr>
<td>[BR 00]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XECB-MAC</td>
<td>${0,1}^*$</td>
<td>$\tau + \nu$</td>
<td>$\checkmark$</td>
<td>$\left\lceil</td>
<td>M</td>
<td>/n \right\rceil + \text{varies}$</td>
</tr>
<tr>
<td>(3 versions)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GD 00,01]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMAC</td>
<td>${0,1}^*$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\left\lceil</td>
<td>M</td>
<td>/n \right\rceil$</td>
</tr>
<tr>
<td>[BR 00,01]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For More Information

- PMAC web page → www.cs.ucdavis.edu/~rogaway
  Contains FAQ, papers, reference code, test vectors...
- Feel free to call or send email
- Or grab me now!