How Fast Can a Two-Pass Mode Go?  
A Parallel Deterministic Authenticated Encryption Mode for AES-NI

Kazumaro Aoki       Tetsu Iwata       Kan Yasuda

NTT       Nagoya Univ       NTT

July 5, 2012 @ DIAC
## Contents

- Background and design rationale
- New mode
- Security
- Implementation issues
- Conclusion
Recent blockcipher-based modes try to reduce $GF(2^n)$ arithmetics (other than the blockcipher calls). The effect is very limited.

Intel announced AES-NI, and a current standard PC can use AES-NI.

Before AES-NI, CTR is slower than 7 cpb, and CTR is faster than 1 cpb after AES-NI.

Even reducing the number of XORs becomes effective on AES-NI environment.
Authenticated encryption (AE) schemes has much attention. An AE should be used instead of using encryption only schemes. Recent attacks regarding BEAST and XML encryption might be avoided.

AE modes can be classified into two categories: Nonce-based AE (NAE) and Deterministic AE (DAE). From the viewpoint of misuse-resistant, DAE is more preferable than NAE.
Background: Polynomial hash

- A polynomial hash can be used for integrity, e.g., GCM and BTM.
- Intel also provides PCLMUL instruction, but it is not faster than expected.
- According to the tutorial at Indocrypt 2011, CTR performs 0.83 cpb, while GCM performs 2.59 cpb on Intel Sandy Bridge.

Question: How fast can a DAE go, if the DAE is fully parallelizable and entirely blockcipher-based?

Our answer: PMAC + CTR
A new mode
Associated data part

$0^n \rightarrow E_K \rightarrow L \rightarrow E_K \rightarrow 2L \rightarrow E_K \rightarrow 2^2L \rightarrow E_K \rightarrow 2^{a-1}3L \rightarrow E_K \rightarrow W$

$A[a] \| 10^*$

$2^{a-1}3^2L \rightarrow E_K \rightarrow W$

$M[1] \rightarrow E_K \rightarrow L \rightarrow 2L \rightarrow E_K \rightarrow 2^2L \rightarrow E_K \rightarrow 2^{m-1}L \rightarrow E_K \rightarrow M[m]$
Tag generation

\[ 2^{a-1}3^2L \rightarrow \oplus \rightarrow E_K \rightarrow W \]

\[
\begin{align*}
L & \rightarrow & 2L & \rightarrow & 2^2L & \rightarrow & 2^{m-1}L & \rightarrow & 2^mL \\
E_K & \rightarrow & E_K & \rightarrow & E_K & \rightarrow & E_K & \rightarrow & E_K \text{ (trunc)} \rightarrow T \\
W & \rightarrow & W & \rightarrow & W & \rightarrow & W & \rightarrow & W \oplus T \rightarrow T \| 0^* \\
0 & \rightarrow & 1 & \rightarrow & 2 & \rightarrow & m-1 & \rightarrow & \text{Copyright 2012 ©NTT – p.8/17}
\end{align*}
\]
Confidentiality part

\[ W \oplus T \parallel 0^* \quad W \oplus T \parallel 0^* \quad W \oplus T \parallel 0^* \quad W \oplus T \parallel 0^* \]

\[ 0 \quad 1 \quad 2 \quad m-1 \]

\[ E_K \quad E_K \quad E_K \quad \cdots \quad E_K \]

\[ M[1] \quad M[2] \quad M[3] \quad M[m] \]

\[ C[1] \quad C[2] \quad C[3] \quad C[m] \]
DAE security

DAE is “secure” if \((E_K, D_K)\) and \((R, \bot)\) are indistinguishable.

Intuition: DAE is “secure” if the output of encryption looks random (as long as the input is “new”), and the forgery is hard.
Security claim

Our PMAC+CTR is a secure DAE under the assumption that $E$ is a pseudorandom permutation (up to the standard birthday bound security).

A proof is not simple, and we are still developing the details of the proof, but the probability of “bad events” can be shown to be small.
**AES-NI**

- **AES-NI** $\ni$ aesenc, aesenclast.

<table>
<thead>
<tr>
<th>Core</th>
<th>Latency</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nehalem</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Sandy Bridge</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

Unit: clock cycle

- **AES-128 can be implemented as** $\text{pxor} + \text{aesenc} \times 9 + \text{aesenclast}$.  

<table>
<thead>
<tr>
<th>Core</th>
<th>Sequential</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nehalem</td>
<td>3.813</td>
<td>1.271</td>
</tr>
<tr>
<td>Sandy Bridge</td>
<td>5.063</td>
<td>0.646</td>
</tr>
</tbody>
</table>

Unit: clock cycle per byte
AES-NI on Sandy Bridge

- Repeating independent \texttt{aesenc} achieves the throughput of 1 clock cycle.

- Adding any instruction even which is zero cycle instruction such as \texttt{xor rax, rax} requires more clock cycles.

- That is, \texttt{aesenc} seems MSROM instruction, which can only be decoded 1 instruction per cycle. It means that it is hard to mix other instructions in the sequence of AES encryption.
Incrementing Gray Code and “Doubling”

Gray code:

$$L \leftarrow L \oplus \text{tbl}[\text{ntz(ctr + +)}]$$

Doubling:

$$L \leftarrow (L\ll1) \oplus (\text{msb}(L) \cdot 0x87)$$
Optimizing Doubling

1: movmskpd rax, xmm0
2: add rax, rax
3: paddq xmm0, xmm0
4: pxor xmm0, tbl[8 * rax]
5: vpshufb xmm2, xmm0, xmm1

movmskpd rax, xmm0:
(h, l) ← xmm0
rax ← 0 · · · 0∥msb(h)∥msb(l)

tbl = [0, 1≪64, 0x87, 1≪64 ⊕ 0x87]
## Performance Comparison on Sandy Bridge

<table>
<thead>
<tr>
<th>Mode</th>
<th>cycles per byte</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTR</td>
<td>0.916</td>
<td>OpenSSL 1.0.1c</td>
</tr>
<tr>
<td></td>
<td>0.787</td>
<td>Ours</td>
</tr>
<tr>
<td>GCM</td>
<td>2.900</td>
<td>OpenSSL 1.0.1c</td>
</tr>
<tr>
<td></td>
<td>2.59</td>
<td>Gueron@IndoC’11</td>
</tr>
<tr>
<td>PMAC+CTR</td>
<td>2.047</td>
<td>Ours</td>
</tr>
</tbody>
</table>
Conclusion

- PMAC+CTR is faster than GCM on Sandy Bridge.
- PMAC+CTR is DAE, fully parallelizable, and uses single key, and internal blockcipher is used only for forward direction.
- OCB3 seems an efficiency-optimal NAE mode, and considering better DAE modes with AES-NI seems an interesting topic.