AES-GCM-SIV
Nonce Misuse-Resistant Authenticated Encryption

Shay Gueron
University of Haifa
and
Intel Corporation

Adam Langley
Google

Yehuda Lindell
Bar Ilan University

Presented by Shay Gueron
AES-GCM-SIV in a nutshell

• **What:**
  • Full nonce misuse-resistant authenticated encryption at an extremely low cost
  • Almost at the performance of AES-GCM (can enjoy (almost) any optimization of AES-GCM)
• **Full proof of security and full implementation**
  • Updates for improved bounds – to be published
• **History:**
  • First version: Gueron and Lindell ACM CCS 2015
  • Extended version Gueron, Lindell, Langley (March 9, 2016)
• **Features:**
  • Nonce misuse resistance and high performance
  • Easily deployable:
    • Can utilizes existing hardware (for AES and for GHASH) and existing code primitives
      – No patents
      – Publicly available code (Reference, optimized asm, MAC OS asm, C intrinsics)
        • [https://github.com/Shay-Gueron/AES-GCM-SIV](https://github.com/Shay-Gueron/AES-GCM-SIV)
    – Soon to be integrated to BoringSSL
AES-GCM success: now the leading AEAD enjoying excellent performance on high end CPU’s

Westmere, Sandy bridge, Haswell, Broadwell, Skylake are Intel Architecture Codenames.
Codenames Haswell: 4th Generation Intel® Core Processor
Codenames Broadwell: 5th Generation Intel® Core Processor
Codenames Skylake: 6th Generation Intel® Core Processor
How did AES-GCM become so fast?

Hardware support and more...

CPU instructions

- AES-NI for encryption
- PCLMULQDQ (64-bit polynomial multiplication) for the GHASH of AES-GCM
- Improved performance of AES-NI / PCLMULQDQ across CPU generations

• Such hardware support is now ubiquitous: on 64-bit processors

Algorithms and optimizations for CTR encryption & GHASH computations (e.g., efficient reduction with PCLMULQDQ)

All contributed to OpenSSL and NSS
AES-GCM and nonce misuse

Derive hash key: \( H = \text{AES}_K(0^{128}) \)

Setup initial counter: \( \text{CTR} = IV | 0^{31} | 1 \)

Compute \( \text{MASK} = \text{AES}_K(\text{CTR}) \)

For \( j = 1, 2, \ldots, \):
- \( \text{CTR} = \text{inc32}(\text{CTR}) \)
- \( c_j = \text{AES}_K(\text{CTR}) \oplus m_j \)
- \( \text{inc32} \) increments the 32-bit counter inside the 128-bit block

Set \( X_1 = a_1, \ldots, X_r = (a_r)', X_{r+1} = c_1, \ldots, X_{r+s} = (c_s)', \quad X_{r+s+1} = (\text{bitlen}(M) | | \text{bitlen}(A)) \)
- All \( X_j \)'s are 128-bit blocks (possible 0 padding for \( (a_r)', (c_s)' \))

\( \text{GHASH}_H = X_1 \cdot H^n \oplus X_2 \cdot H^{n-1} \oplus \ldots \oplus X_n \cdot H \)
- \( n = r+s+1 \)
- “\( \cdot \)” = multiplication in \( \text{GF}(2^{128}) \) \([x]/P(x)\)
- \( P(x) = x^{128} + x^7 + x^2 + x + 1 \) (with reversed order of bits within the bytes)

\( \text{TAG} = \text{GHASH}_H \oplus \text{MASK} \)
\( C = (c_1, c_2, \ldots, c_s^*) \)

Repeating a nonce (with the same key) has a disastrous effect on both privacy and integrity

Our goal:
Enjoy the AES-GCM hardware / software support to define an analogous AEAD mode but with nonce misuse resistance:
Same nonce and same message: the result is the same ciphertext (inherent property)
Otherwise – full security of authenticated encryption (within the security margins)
POLYVAL
(a universal family of hash functions)

- The operation “●”:
  - \( A \ ● \ B = A * B * x^{-128} \mod P(x) \)
  - \( P(x) = x^{128} + x^{127} + x^{126} + x^{121} + 1 \)
  - Operations in \( \text{In GF} (2^{128}) [x] / P(x) \)
  - * is the field multiplication.

- Let \( X_i \) be a 128 bit block.
- Let \( M \) be a massage of \( n \) blocks (\( M = X_1 || X_2 || ... || X_n \))
- Let \( H \) be a 128 bit block

\[
\text{POLYVAL}_H(M) = X_1 \ ● \ H^n \oplus X_2 \ ● \ H^{n-1} \oplus ... \oplus X_n \ ● \ H
\]

For example:
- \( \text{POLYVAL}_H(X_1) = X_1 \ ● \ H \)
- \( \text{POLYVAL}_H(X_1 || X_2) = X_1 \ ● \ H^2 \oplus X_2 \ ● \ H \)
The relation between POLYVAL and GHASH

- $X_i$ and $H$ be 128 bit blocks; $M = \text{message of } n \text{ blocks } (M = X_1 || X_2 || \ldots || X_n)$

- GHASH in AES-GCM
  - $\text{GHASH}_H(M) = X_1 \cdot H^n \oplus X_2 \cdot H^{n-1} \oplus \ldots \oplus X_n \cdot H$
  - “$\cdot$” denotes multiplication in GF $(2^{128}) [x] / P(x)$
    - $P(x) = x^{128} + x^7 + x^2 + x + 1$ (with reversed order of bits within the bytes)

- POLYVAL in GCM-SIV
  - No need to reverse the order of bits within the bytes
  - “$\cdot$”:
    - $A \cdot B = A \times B \times x^{-128}$ in GF $(2^{128}) [x] / Q(x)$
    - $Q(x) = x^{128} + x^{127} + x^{126} + x^{121} + 1$
    - ($\times$ is the field multiplication)

$$\text{POLYVAL}_{x \otimes H}(X_1, X_2, \ldots, X_{\ell}) = $$
$$= \text{ByteSwap} (\text{GHASH}_H (\text{ByteSwap}(X_1), \text{ByteSwap}(X_2), \ldots, \text{ByteSwap}(X_{\ell})))$$
AES-GCM-SIV 128 flow (encryption)

- Input:
  - in_AAD, in_MSG
  - K, H, N

- Message / AAD padding:
  - AAD = Pad in_AAD to d blocks
  - MSG = pad in_MSG to n blocks (M_1 | | M_2 | | M_3 ... | | M_n)
  - Define LENBLK
  - Padded AAD/MSG = AAD | | MSG | | LENBLK (consists of d+n+1 blocks)

- Calculate:
  - T = POLYVAL_H (AAD | | MSG | | LENBLK)
  - Record_Enc_key = AES_K (N)
  - TAG = AES_{Record_Enc_key} (0 | | (T ⊕ N) [126:0])
  - CTRBLK_i = 1 | | TAG[126:32] | | TAG[31:0] ▸ i (i is 32 bit long. i = 0,1 ... i< 2^{32} -1 )
  - CT_i = AES_{Record_Enc_key} (CTRBLK_i) ⊕ M_i
  - Define CT = (CT_1, CT_2, ... CT_n)
  - If length(in_MSG) != length(CT) - chop lsbits of CT so that length(in_MSG) == length(CT)

- Output: CT = (CT_1, CT_2, ... CT_n), TAG

* ▸ - addition modulo 2^{32}
AES-GCM-SIV 256 flow (encryption)

- Input:
  - In_AAD, in_MSG
  - K, H, N

- Derive (as described before):
  - AAD
  - MSG = M₁ || M₂ || M₃ ... || Mₙ
  - LENBLK

- Calculate:
  - T = POLYVALₜₜₜ(H (AAD | MSG | LENBLK))
  - Record_Enc_key[255:128] = AESₜₜₜ(K (N))  \( (AES= AES 256) \)
  - Record_Enc_key [127:0] = AESₜₜₜ(K (Record_Enc_key[255:128]))  \( (AES= AES 256) \)
  - TAG = AESₜₜₜ(Record_Enc_key (0 | (T ⊕ N) [126:0]))  \( (AES= AES 256) \)
  - CTRBLKᵢ = 1 | | TAG[126:32] | | TAG[31:0]  \( (i \text{ is 32 bits long. } i = 0,1 ... i< 2^{32} -1 ) \)
  - CTᵢ = AESₜₜₜ(Record_Enc_key (CTRBLKᵢ) ⊕ Mᵢ)  \( (AES= AES 256) \)
  - Define CT = (CT₁, CT₂, ... CTₙ)
  - If length(in_MSG) != length(CT) - chop Isbits of CT so that length(in_MSG) == length(CT)

- Output:
  - CT = (CT₁, CT₂, ... CTₙ)
  - TAG
AES-GCM-SIV 128 flow (encryption)

Input:

- $H$
- $AAD$
- $MSG$
- $Alen$
- $Mlen$
- $K$
- $N$

$H$, $Padded_{AAD}$, $Padded_{MSG}$, $LENBLK$, $AES$

$POLYVAL$

$T$

$CTRBLK_i = 1 || TAG[126:32] || TAG[31:0] \pmod{2^{32}}$

$AES$

$CT_i$, $TAG$

Output:

$\pmod{}$ - addition modulo $2^{32}$

$AES = AES128$
AES-GCM-SIV 256 flow (encryption)

Input:
- $H$
- $AAD$
- $MSG$
- $Alen$
- $Mlen$
- $K$
- $N$

Output:
- $CT_i$
- $TAG$

$\text{CTRBLK}_i = 1 || \text{TAG}[126:32] || \text{TAG}[31:0] \oplus i$

$\oplus$ - addition modulo $2^{32}$

$AES = AES256$
AES-GCM-SIV 128 Performance (in C/B)

**AES_GCM_SIV_Encryption (128 bit)**

<table>
<thead>
<tr>
<th></th>
<th>1KB</th>
<th>2KB</th>
<th>4KB</th>
<th>8KB</th>
<th>16KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSW</td>
<td>1.50</td>
<td>1.37</td>
<td>1.30</td>
<td>1.27</td>
<td>1.26</td>
</tr>
<tr>
<td>BDW</td>
<td>1.16</td>
<td>1.03</td>
<td>0.96</td>
<td>0.93</td>
<td>0.91</td>
</tr>
<tr>
<td>SKL</td>
<td>1.11</td>
<td>1.02</td>
<td>0.97</td>
<td>0.95</td>
<td>0.94</td>
</tr>
</tbody>
</table>

**AES_GCM_SIV_Decryption (128 bit)**

<table>
<thead>
<tr>
<th></th>
<th>1KB</th>
<th>2KB</th>
<th>4KB</th>
<th>8KB</th>
<th>16KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSW</td>
<td>1.47</td>
<td>1.30</td>
<td>1.27</td>
<td>1.23</td>
<td>1.22</td>
</tr>
<tr>
<td>BDW</td>
<td>1.00</td>
<td>0.85</td>
<td>0.81</td>
<td>0.77</td>
<td>0.76</td>
</tr>
<tr>
<td>SKL</td>
<td>0.83</td>
<td>0.71</td>
<td>0.68</td>
<td>0.65</td>
<td>0.64</td>
</tr>
</tbody>
</table>
## GCM-SIV 256 Performance (in C/B)

### AES_GCM_SIV_Encryption (256 bit)

<table>
<thead>
<tr>
<th></th>
<th>1KB</th>
<th>2KB</th>
<th>4KB</th>
<th>8KB</th>
<th>16KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSW</td>
<td>1.90</td>
<td>1.71</td>
<td>1.61</td>
<td>1.56</td>
<td>1.54</td>
</tr>
<tr>
<td>BDW</td>
<td>1.60</td>
<td>1.37</td>
<td>1.26</td>
<td>1.20</td>
<td>1.17</td>
</tr>
<tr>
<td>SKL</td>
<td>1.53</td>
<td>1.32</td>
<td>1.25</td>
<td>1.22</td>
<td>1.20</td>
</tr>
</tbody>
</table>

### AES_GCM_SIV_Decryption (256 bit)

<table>
<thead>
<tr>
<th></th>
<th>1KB</th>
<th>2KB</th>
<th>4KB</th>
<th>8KB</th>
<th>16KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSW</td>
<td>1.98</td>
<td>1.68</td>
<td>1.54</td>
<td>1.49</td>
<td>1.46</td>
</tr>
<tr>
<td>BDW</td>
<td>1.49</td>
<td>1.20</td>
<td>1.13</td>
<td>1.07</td>
<td>1.04</td>
</tr>
<tr>
<td>SKL</td>
<td>1.19</td>
<td>1.02</td>
<td>0.96</td>
<td>0.92</td>
<td>0.90</td>
</tr>
</tbody>
</table>
GCM-SIV Short Messages Performance [Cycles]

- **AES_GCM_SIV 128 bit (encryption)**

<table>
<thead>
<tr>
<th>Input Size</th>
<th>16B</th>
<th>32B</th>
<th>64B</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSW</td>
<td>293</td>
<td>349</td>
<td>470</td>
</tr>
<tr>
<td>BDW</td>
<td>249</td>
<td>292</td>
<td>379</td>
</tr>
<tr>
<td>SKL</td>
<td>194</td>
<td>228</td>
<td>293</td>
</tr>
</tbody>
</table>

- **AES_GCM_SIV 256 bit (encryption)**

<table>
<thead>
<tr>
<th>Input Size</th>
<th>16B</th>
<th>32B</th>
<th>64B</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSW</td>
<td>430</td>
<td>460</td>
<td>557</td>
</tr>
<tr>
<td>BDW</td>
<td>430</td>
<td>483</td>
<td>557</td>
</tr>
<tr>
<td>SKL</td>
<td>316</td>
<td>350</td>
<td>422</td>
</tr>
</tbody>
</table>
Proven security statement
(Gueron Lindell CCS 20)15

Theorem 4.3  (2-Key GCM-SIV). Consider the above variant of Construction 3.1 with one key for the pseudorandom function $F$ and one key for the hash function GHASH. Then, the result is a nonce misuse-resistant authenticated encryption scheme, and there exists an adversary $A'$ for $F$ such that for every $A$ attacking Construction 3.1 making $q_E$ encryption queries and $q_d$ decryption queries of overall length $L$:

$$\text{Adv}_{\Pi}^{\text{mrAE}}(A)$$

$$< 2 \cdot \text{Adv}_{F}^{\text{prf}}(A') + \frac{q_E(A)^2}{2^{n-k-2}} + \frac{q_E(A)^2 + q_d(A)}{2^{n-1}}$$

where $t(A') \leq 6 \cdot t(A)$ and $q_f(A') \leq 2q_E(A) + 2q_d(A) + \frac{L}{n}$.

Security of GCM-SIV is equivalent to that of AES-GM (with 96-bit IV)

Improved bound will be published soone
Summary: AES-GCM-SIV in a nutshell

• **What:**
  • Full nonce misuse-resistant authenticated encryption at an extremely low cost
  • Almost at the performance of AES-GCM (can enjoy (almost) any optimization of AES-GCM)
• Full proof of security and full implementation
  • Updates for improved bounds – to be published
• **History:**
  • First version: Gueron and Lindell ACM CCS 2015
  • Extended version Gueron, Lindell, Langley (March 9, 2016)
• **Features:**
  • Nonce misuse resistance and high performance
  • Easily deployable:
    • Can utilizes existing hardware (for AES and for GHASH) and existing code primitives
      – No patents
      – Publicly available code (Reference, optimized asm, MAC OS asm, C intrinsics)
        • [https://github.com/Shay-Gueron/AES-GCM-SIV](https://github.com/Shay-Gueron/AES-GCM-SIV)
    – Soon to be integrated to BoringSSL
Thank you.