CFRG Meeting

EUROCRYPT 2016, May 12, 2016

AES-GCM-SIV

Nonce Misuse-Resistant Authenticated Encryption

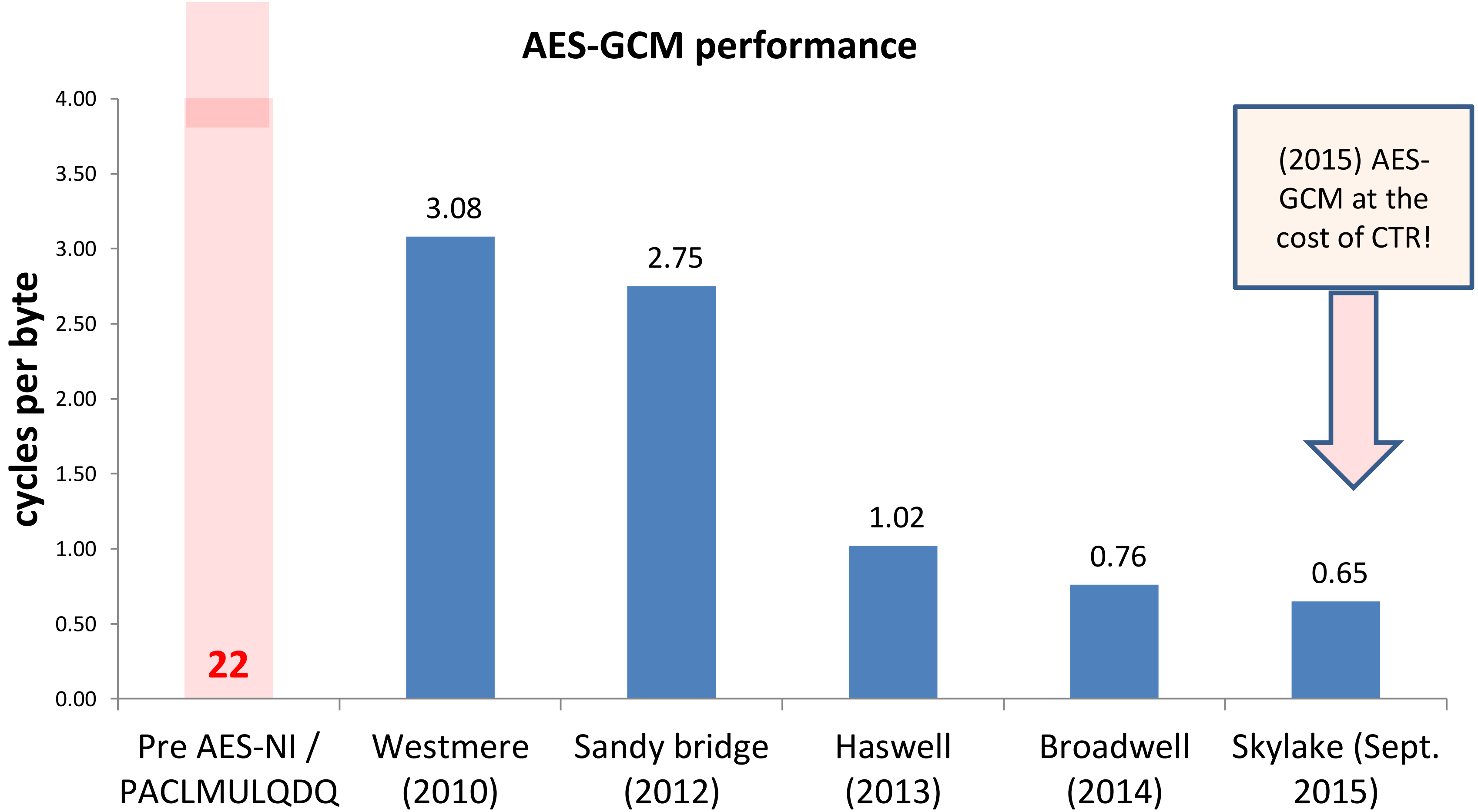
Shay Gueron
University of Haifa
and
Intel Corporation
●
Adam Langley
Google
●
Yehuda Lindell
Bar Ilan University

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)
    - https://datatracker.ietf.org/doc/draft-irtf-cfrg-gcmsiv/
- **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
  - 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

- *<u>Such hardware support is now ubiquitous: on 64-bit processors</u>*

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: $H = AES_K (0^{128})$

Setup initial counter: $\boxed{CTR = IV || 0^{31} || 1}$

Compute $MASK = AES_K (CTR)$

For $j = 1, 2, \ldots,$:

- $CTR = inc32 (CTR)$;
- $c_j = AES_K (CTR) \oplus m_j$
- inc32 increments the 32-bit counter inside the 128-bit block

Set $X_1=a_1, \ldots X_r = (a_r)', X_{r+1}=c_1, \ldots X_{r+s}= (c_s)', \quad X_{r+s+1} = (bitlen(M) \,||\, bitlen(A))$

- All X_j's are 128-bit blocks (possible 0 padding for $(a_r)'$, $(c_s)'$)

$GHASH_H = X_1 \bullet H^n \oplus X_2 \bullet H^{n-1} \oplus \ldots \oplus X_n \bullet H$

- $n = r+s+1$
- "$\bullet$" = 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)

$TAG = GHASH_H \oplus \boxed{MASK}$

$C = (c_1 , c_2 , \ldots c_s^*)$

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 "●":
  - $A \bullet B = A * B * x^{-128} \bmod P(x)$
  - $P(x) = x^{128} + x^{127} + x^{126} + x^{121} + 1$
  - Operations in In GF (2^{128}) [x] / P(x)
  - * is the field multiplication.
- Let $\mathbf{X_i}$ be a 128 bit block.
- Let **M** be a massage of n blocks ($M = X_1 \,||\, X_2 \,||\ldots||\, X_n$)
- Let **H** be a 128 bit block

$$\boxed{\mathrm{POLYVAL}_H(M) = X_1 \bullet H^n \oplus X_2 \bullet H^{n-1} \oplus \ldots \oplus X_n \bullet H}$$

For example:

- $\mathrm{POLYVAL}_H(X_1) = X_1 \bullet H$
- $\mathrm{POLYVAL}_H(X_1 \,||\, X_2) = X_1 \bullet H^2 \oplus X_2 \bullet H$

The relation between POLYVAL and GHASH

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

- GHASH in AES-GCM
  - $GHASH_H(M) = X_1 \bullet H^n \oplus X_2 \bullet H^{n-1} \oplus \ldots \oplus X_n \bullet H$
  - "$\bullet$" 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
  - "$\bullet$": $A \bullet B = A \times B \times x^{-128}$ in $GF(2^{128})[x] / Q(x)$
    - $Q(x) = x^{128} + x^{127} + x^{126} + x^{121} + 1$
    - ($\times$ is the field multiplication)

$$POLYVAL_{x \otimes H}(X_1, X_2, \ldots, X_\ell)) =$$

$$= ByteSwap\left(GHASH_H\left(ByteSwap(X_1), ByteSwap(X_2), \ldots, ByteSwap(X_\ell)\right)\right)$$

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 $\oplus$ N) [126:0])
  - $CTRBLK_i$ = 1||TAG[126:32]||TAG[31:0] $\boxplus$ i   (i is 32 bit long. i = 0,1 ... i< 2^{32} -1)
  - CT_i = $AES_{Record\_Enc\_key}$ ($CTRBLK_i$) $\oplus$ 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

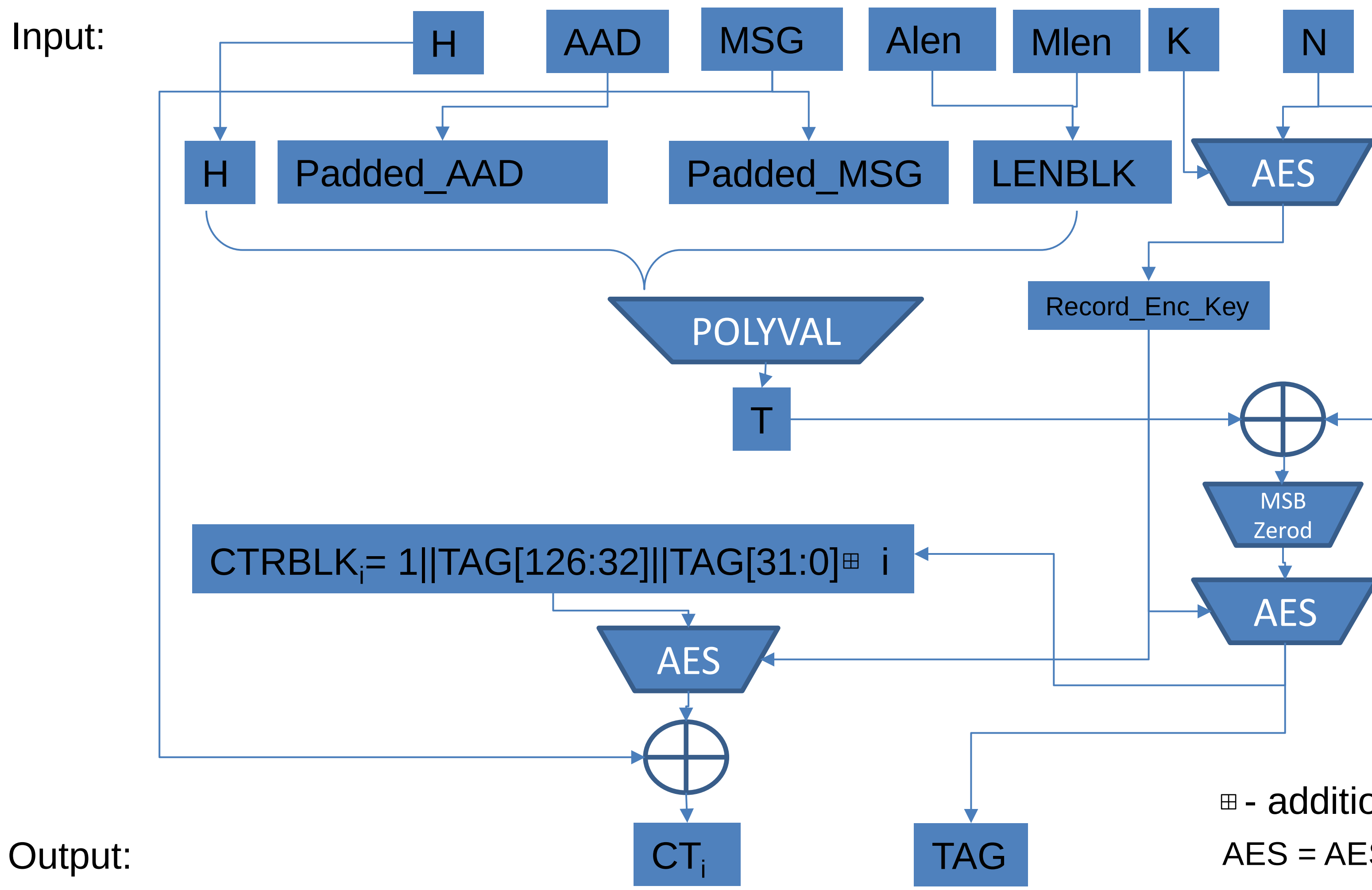
$\boxplus$ - addition modulo 2^{32}

AES-GCM-SIV 256 flow (encryption)

- Input:
  - In_AAD, in_MSG
  - K, H, N
- Derive (as described before):
  - AAD
  - MSG = M_1 || M_2 || M_3 … ||M_n
  - LENBLK
- Calculate:
  - T = $POLYVAL_H$ (AAD||MSG||LENBLK)
  - Record_Enc_key[255:128] = AES_K (N) (AES= AES 256)
  - Record_Enc_key [127:0] = AES_K (Record_Enc_key[255:128]) (AES= AES 256)
  - TAG = $AES_{Record\_Enc\_key}$ (0||(T $\oplus$ N) [126:0]) (AES= AES 256)
  - $CTRBLK_i$ = 1||TAG[126:32]||TAG[31:0] $\boxplus$ i (i is 32 bits long. i = 0,1 ... i< 2^{32} -1)
  - CT_i = $AES_{Record\_Enc\_key}$ ($CTRBLK_i$) $\oplus$ M_i (AES= AES 256)
  - 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

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

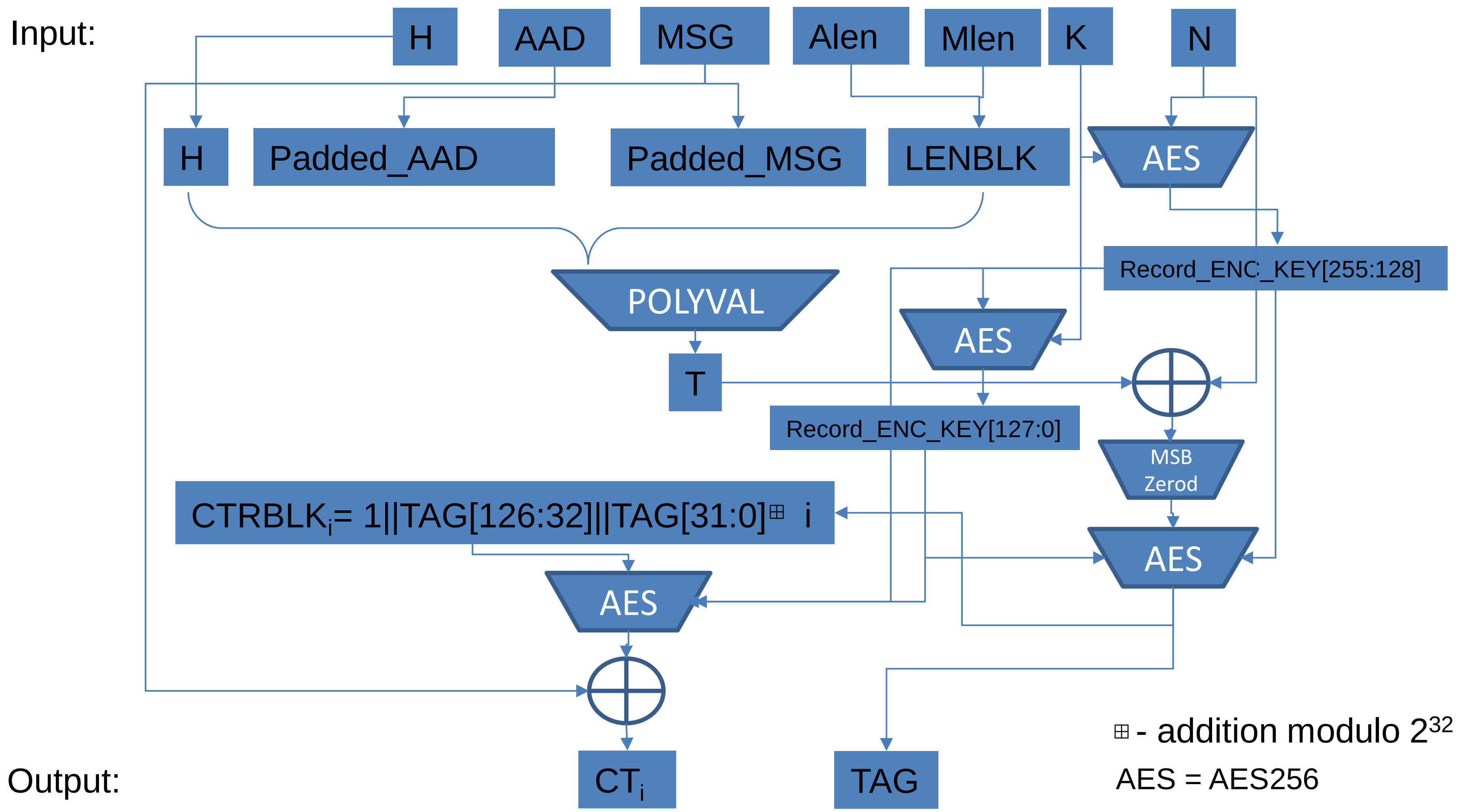
AES-GCM-SIV 128 flow (encryption)



⊞ - addition modulo 2^{32}

AES = AES128

AES-GCM-SIV 256 flow (encryption)



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

AES_GCM_SIV_Encryption (128 bit)

	1KB	2KB	4KB	8KB	16KB
HSW	1.50	1.37	1.30	1.27	1.26
BDW	1.16	1.03	0.96	0.93	0.91
SKL	1.11	1.02	0.97	0.95	0.94

AES_GCM_SIV_Decryption (128 bit)

	1KB	2KB	4KB	8KB	16KB
HSW	1.47	1.30	1.27	1.23	1.22
BDW	1.00	0.85	0.81	0.77	0.76
SKL	0.83	0.71	0.68	0.65	0.64

GCM-SIV 256 Performance (in C/B)

AES_GCM_SIV_Encryption (256 bit)

	1KB	2KB	4KB	8KB	16KB
HSW	1.90	1.71	1.61	1.56	1.54
BDW	1.60	1.37	1.26	1.20	1.17
SKL	1.53	1.32	1.25	1.22	1.20

AES_GCM_SIV_Decryption (256 bit)

	1KB	2KB	4KB	8KB	16KB
HSW	1.98	1.68	1.54	1.49	1.46
BDW	1.49	1.20	1.13	1.07	1.04
SKL	1.19	1.02	0.96	0.92	0.90

GCM-SIV Short Messages Performance[Cycles]

- **AES_GCM_SIV 128 bit (encryption)**

Input Size	16B	32B	64B
HSW	293	349	470
BDW	249	292	379
SKL	194	228	293

- **AES_GCM_SIV 256 bit (encryption)**

Input Size	16B	32B	64B
HSW	430	460	557
BDW	430	483	557
SKL	316	350	422

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 $\mathcal{A}'$ for F such that for every $\mathcal{A}$ attacking Construction 3.1 making q_E encryption queries and q_d decryption queries of overall length L:*

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

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

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)
    - https://datatracker.ietf.org/doc/draft-irtf-cfrg-gcmsiv/
- **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
  - Soon to be integrated to BoringSSL

Thank you.