GCM-SIV:
Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte

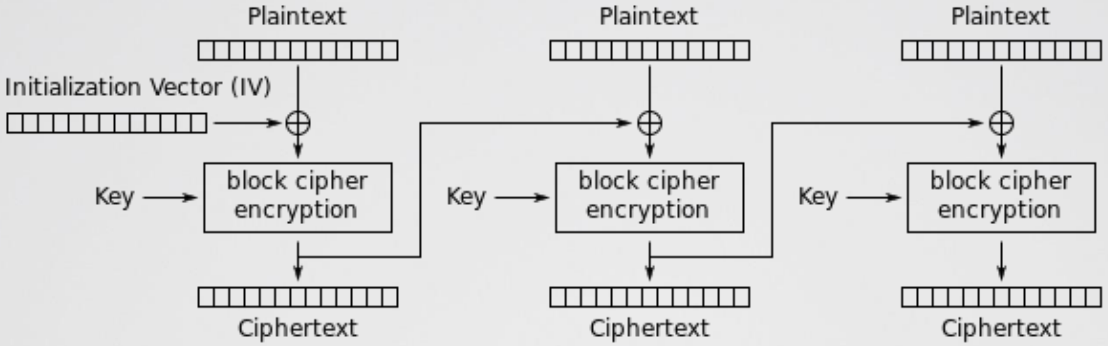
Shay Gueron
Haifa Univ. and Intel

Yehuda Lindell
Bar-Ilan University

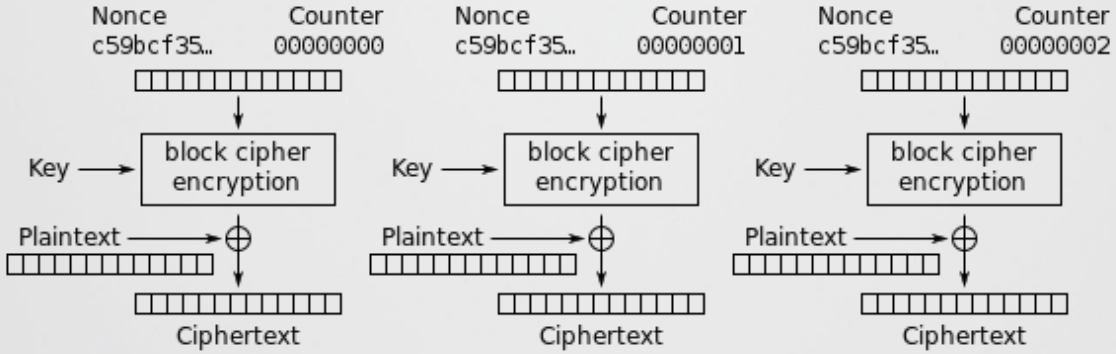
Center for Research in Applied Cryptography and Cyber Security

Appeared at ACM CCS 2015

How to Encrypt with a Block Cipher



Cipher Block Chaining (CBC) mode encryption



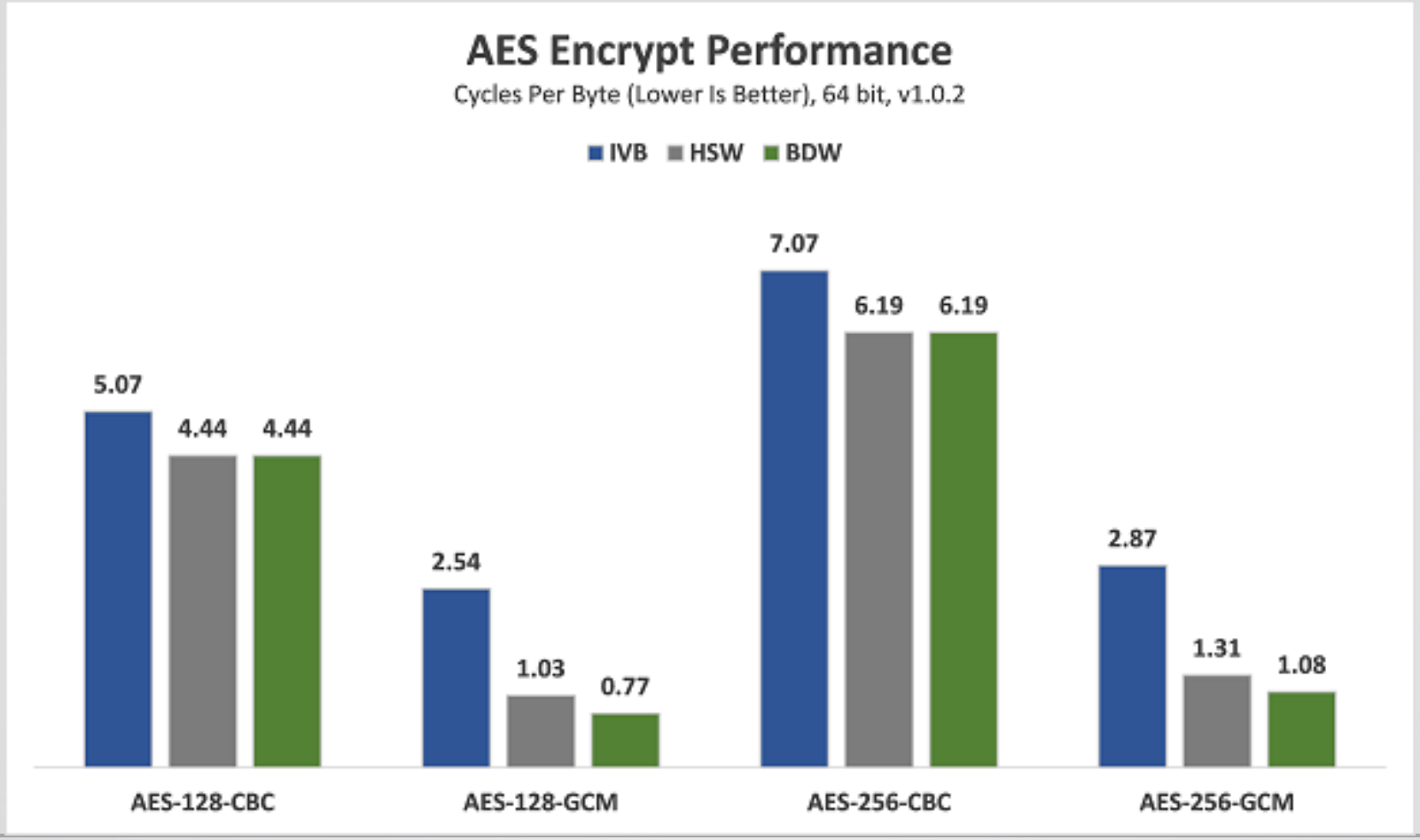
Counter (CTR) mode encryption

CBC vs CTR

- **Efficiency:**
  - CBC – encryption is strictly **sequential**
  - CTR – encryption can be parallelized

- **Does this matter?**
  - The Intel AES-NI instruction is fully pipelineable
  - AES-CTR encryption with AES-NI is 7 times faster!

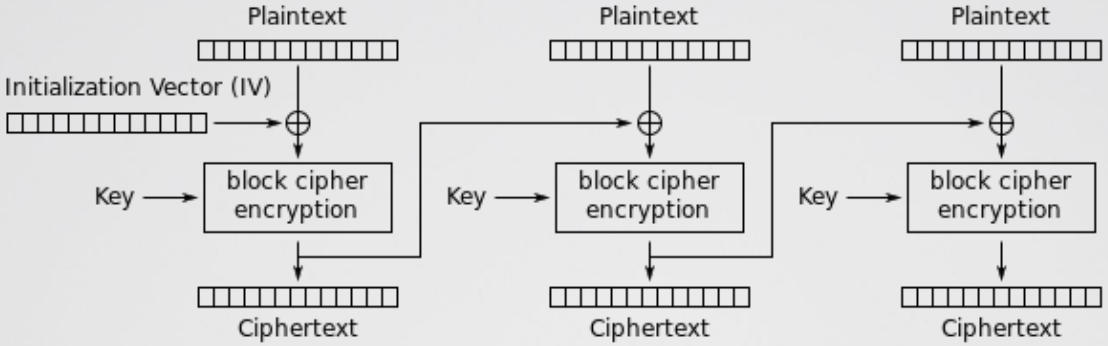
CBC vs CTR


AES Encrypt Performance
Cycles Per Byte (Lower Is Better), 64 bit, v1.0.2
IVB
HSW
BDW
5.07
4.44
4.44
2.54
1.03
0.77
7.07
6.19
6.19
2.87
1.31
1.08
AES-128-CBC
AES-128-GCM
AES-256-CBC
AES-256-GCM

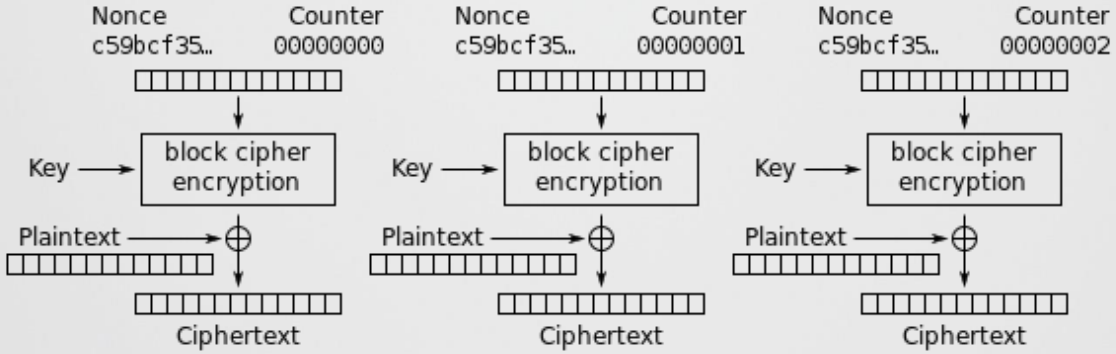
CBC vs CTR – Security

- **Security bounds**
  - CTR has better security bounds – the counter is a nonce and security is preserved as long as it doesn't repeat
  - CBC breaks at the birthday bound since "random" values are input to the block cipher
- ~~**Integrity**~~
  - ~~CBC is harder to tamper with~~
- **IV/nonce reuse**
  - CBC – reveals common prefix
  - **CTR – completely broken**

IV/Nonce Reuse



Cipher Block Chaining (CBC) mode encryption



Counter (CTR) mode encryption

Why Should an IV Repeat?

- **Randomness is much harder than it should be**
  - Intel has RDRAND and RDSEED on all new chips
- **Not used inside Linux /dev/random**

"We cannot trust" Intel and Via's chip-based crypto, FreeBSD developers say

Following NSA leaks from Snowden, engineers lose faith in hardware randomness.

by **Dan Goodin** - Dec 10, 2013 3:00pm IST

This post was updated on December 16 to make clear that for most of FreeBSD's history, it wasn't possible to use RDRAND and Padlock as the sole source of random numbers fed to the /dev/random engine.

Bad Randomness

- **In 2008, a bug in Debian Linux was found**
  - In 2006, code that was crucial for RNG reseeding was commented out

```
MD_Update(&m,buf,j);
[ .. ]
MD_Update(&m,buf,j); /* purify complains */
```

Bad Randomness

- **PlayStation 3**
  - In 2010, the ECDSA private key used by Sony to sign software for PlayStation 3 was recovered because Sony failed to generate a new random nonce for each signature

RSA Keys – Lenstra et al. 2012

- **Collected 6.4 million RSA keys from the web**
  - 71,052 occurred more than once
    - Different owners can decrypt each other's traffic
    - Some of the moduli repeated thousands of times (no entropy)
  - 12,934 had a common factor
    - Computed $GCD(N, N')$ where $N = pq$ and $N' = p'q$
    - Factor both moduli

- **We use this for entropy estimation**

Entropy Estimation via RSA Keys

- The expected number of collisions in q samples from a domain of size N is $\binom{q}{2}/N \approx q^2/2N$
- We have $q = 12,800,000$ (number of primes is double)
- We have number of collisions = 12,934
- So, $\frac{12,800,000^2}{2N} = 12,934$ giving $N \approx 2^{32.56}$

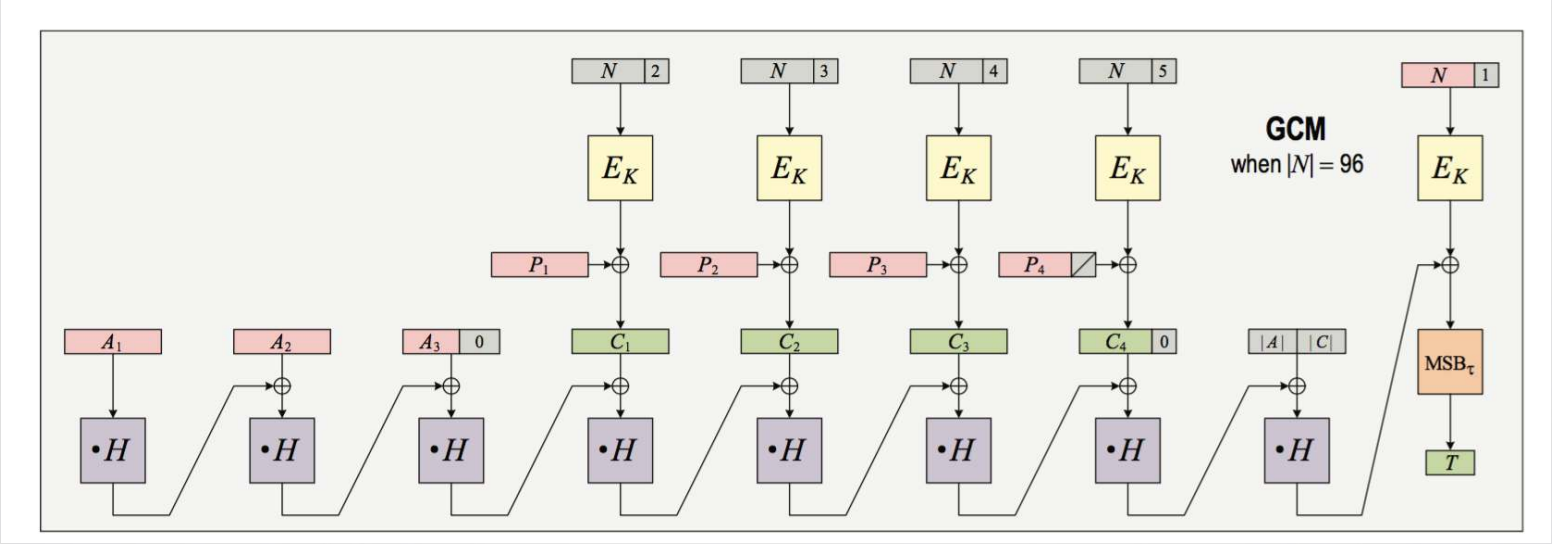
- Conclusion: an "average" of 33 bits of entropy

Bad Randomness

- **Given that randomness can repeat and does repeat, what should we do?**
- **CBC still reveals common prefixes, but is better than CTR…**
- **Can we do better? Efficiently?**

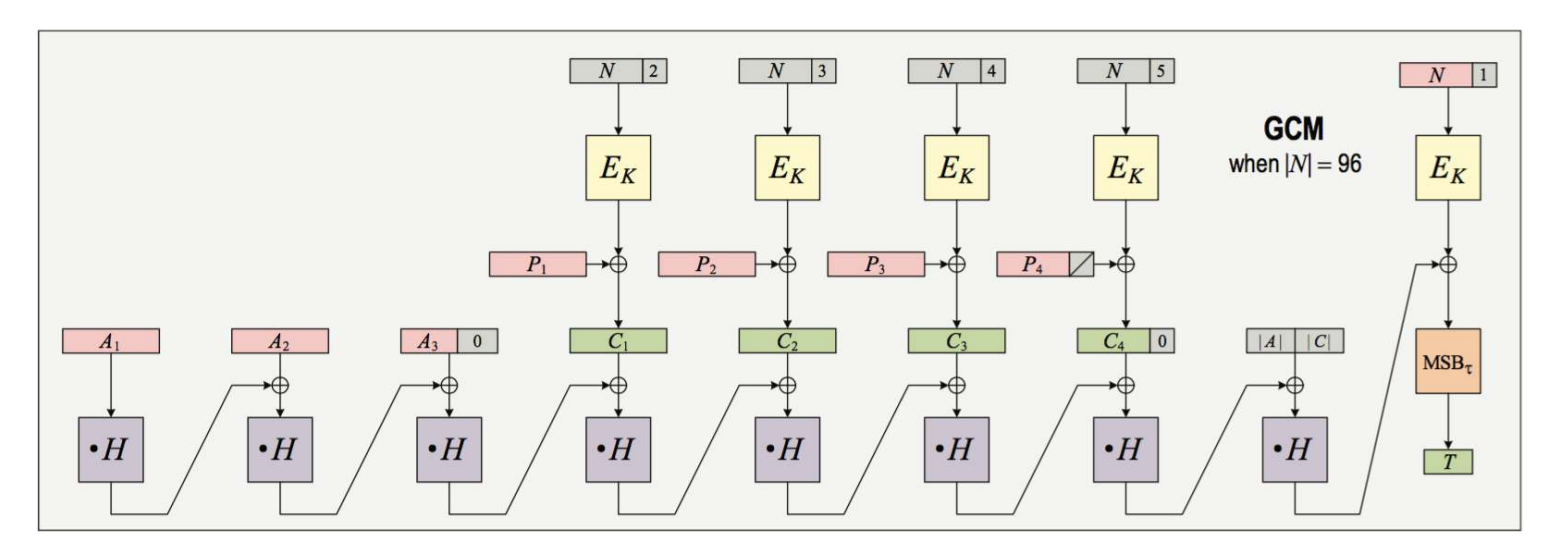
What About Authenticated Encryption?

- **CCM:**
  - CBC-MAC followed by CTR encryption: **slow due to CBC-MAC** and **vulnerable due to CTR encryption**
- **GCM:**



What About Authenticated Encryption?

- **GCM – if the nonce repeats, then:**
  - **As with CTR plaintexts can be recovered**
  - Much more seriously – **H can be recovered**
- **This means that integrity is lost forever!**



Preliminaries: IV vs Nonce Encryption

- **IV (initial vector) encryption:**
  - IV must be randomly chosen
- **Nonce-based encryption:**
  - Only require that nonce is unique

- **CBC encryption:** need random IV; nonce not good enough
- **CTR encryption:** suffices to have a unique nonce
  - In AES-CTR, use a nonce of length 96 bits and counter of length 32 bits

Nonce Misuse Resistance [Rogaway-Shrimpton]

- Denote nonce by N
- **Security property**
  - If N is same and message is same – the result is the same ciphertext
    - This is inherent
  - Otherwise – full security (authenticated encryption):
    - Even if N is the same and the message is not
    - Even if N is different and the message the same
- **This cannot be achieved for online encryption**
  - If two long messages differ only in the last bit, when same N is used, must have same prefix in online

Abstract SIV Encryption [Rogaway-Shrimpton]

- **Input:** message M and nonce N
- **Step 1:**
  - Apply a PRF F with key $K1$ to (N, M); denote result by T
- **Step 2:**
  - Encrypt M with key $K2$ using nonce T; denote result by C
- **Output** (N, M, T)

- **Decryption:** $M \leftarrow Dec_{K2}(C)$ with nonce T; check $T = F_{K1}(N, M)$

SIV Encryption Security

- **Encryption:**

$$T = F_{K1}(N, M); C \leftarrow Enc_{K2}(M) \text{ with nonce } T$$

- **Security**
  - If nonce N is different, then by PRF the value T is pseudorandom
  - If nonce N is the same but M is different, then by PRF the value T is pseudorandom
  - The value T also serves as a valid MAC and so have authenticated encryption

Efficient Instantiations

- **Option 1 – apply a PRF based on AES**
  - What PRFs do we have? CBC-MAC
  - Very expensive

- **Option 2 – construct a more efficient PRF using simpler primitives**
  - Let H be an ϵ-XOR universal hash function

$$\forall x, y, z : \Pr[H_{K1}(x) \oplus H_{K1}(y) = z] \leq \epsilon(n)$$

- **Claim:** $F_{K1,K2}(N, M) = F_{K2}(H_{K1}(M) \oplus N)$ **is a PRF**

Universal-Hash Based PRF

- **The construction:** $F_{K1,K2}(N, M) = F_{K2}(H_{K1}(M) \oplus N)$
- **Proof idea:**
  - By the PRF property of F, can distinguish only if it queries $(N, M), (N', M')$ where $H_{K1}(M) \oplus N = H_{K1}(M') \oplus N'$
  - Equivalently: if $H_{K1}(M) \oplus H_{K1}(M') = N \oplus N'$
  - By the ϵ-XOR property, this happens with probability only ϵ for each pair
  - Therefore, secure PRF for negligible ϵ

The GCM-SIV Instantiation

- **The GHASH function H in GCM is an ϵ-XOR universal hash function (for negligible ϵ)** [McGrew-Viega]
- **The PRF used is AES (only need a single block)**
- **Encryption is AES-CTR**

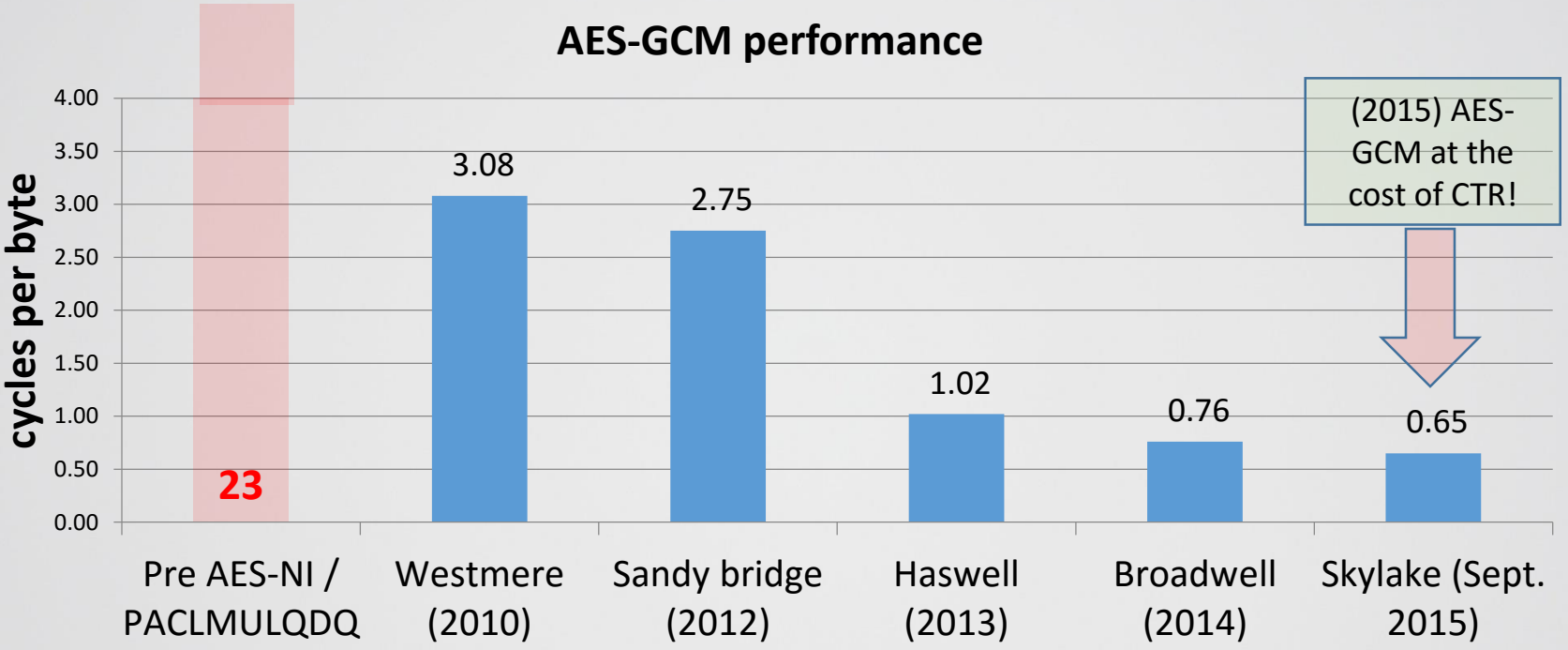
- **Versions:**
  - Three different keys (for GHASH, PRF, CTR-ENC)
  - Two keys: use same key for PRF and CTR-ENC
  - One key: derive the two keys using AES itself

The GCM-SIV Instantiation

- **A very important property: all the elements here are identical to the existing AES-GCM**
  - We only change the order of operations

- **Why is this important?**
  - Efficiency
  - Deployment ease (use existing code bases)

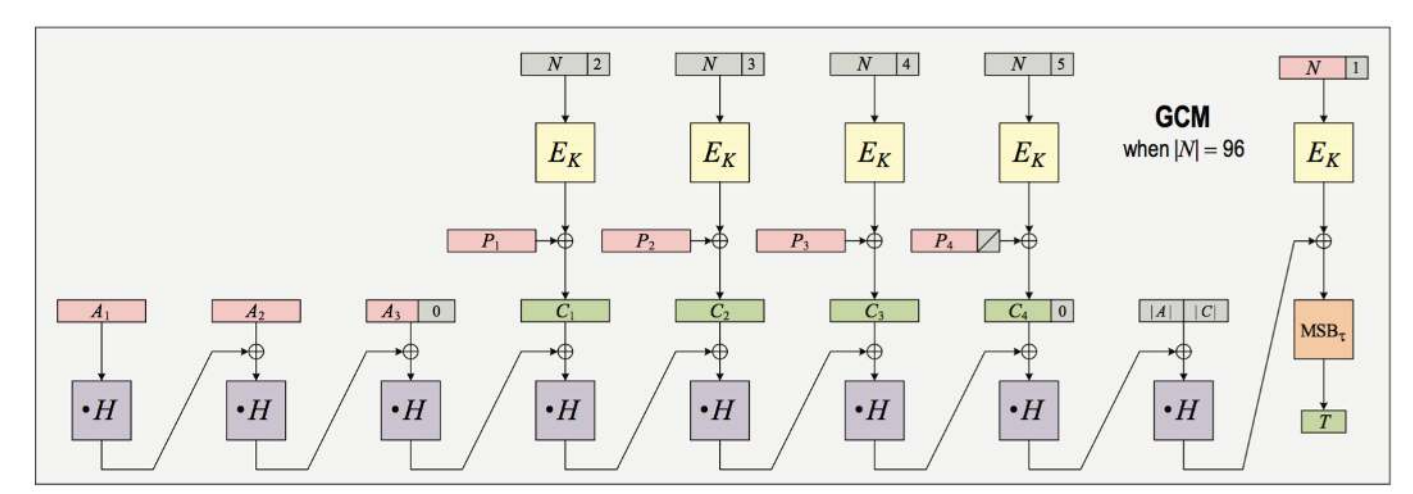
AES-GCM Across Intel CPU Generations



Use AES-NI for CTR and PCLMULQDQ for GHASH

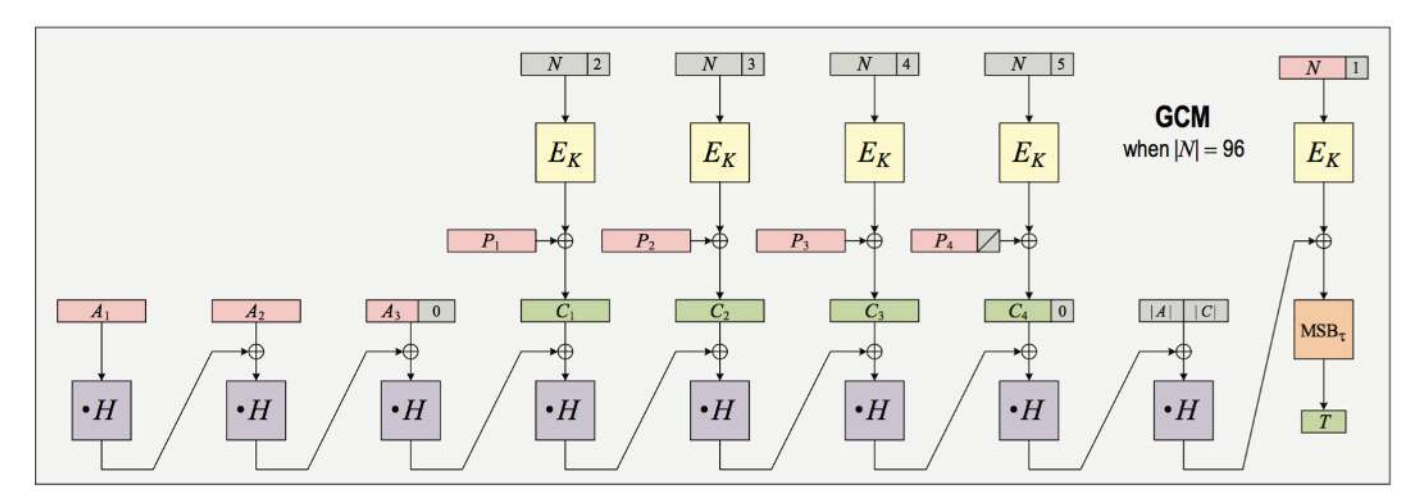
Efficiency of GCM vs GCM-SIV

- **Encryption**
  - In **GCM**, CTR-ENC and GHASH are interleaved and run in parallel
  - In **GCM-SIV**, GHASH must be finished before CTR-ENC can begin **(cannot be done in parallel)**

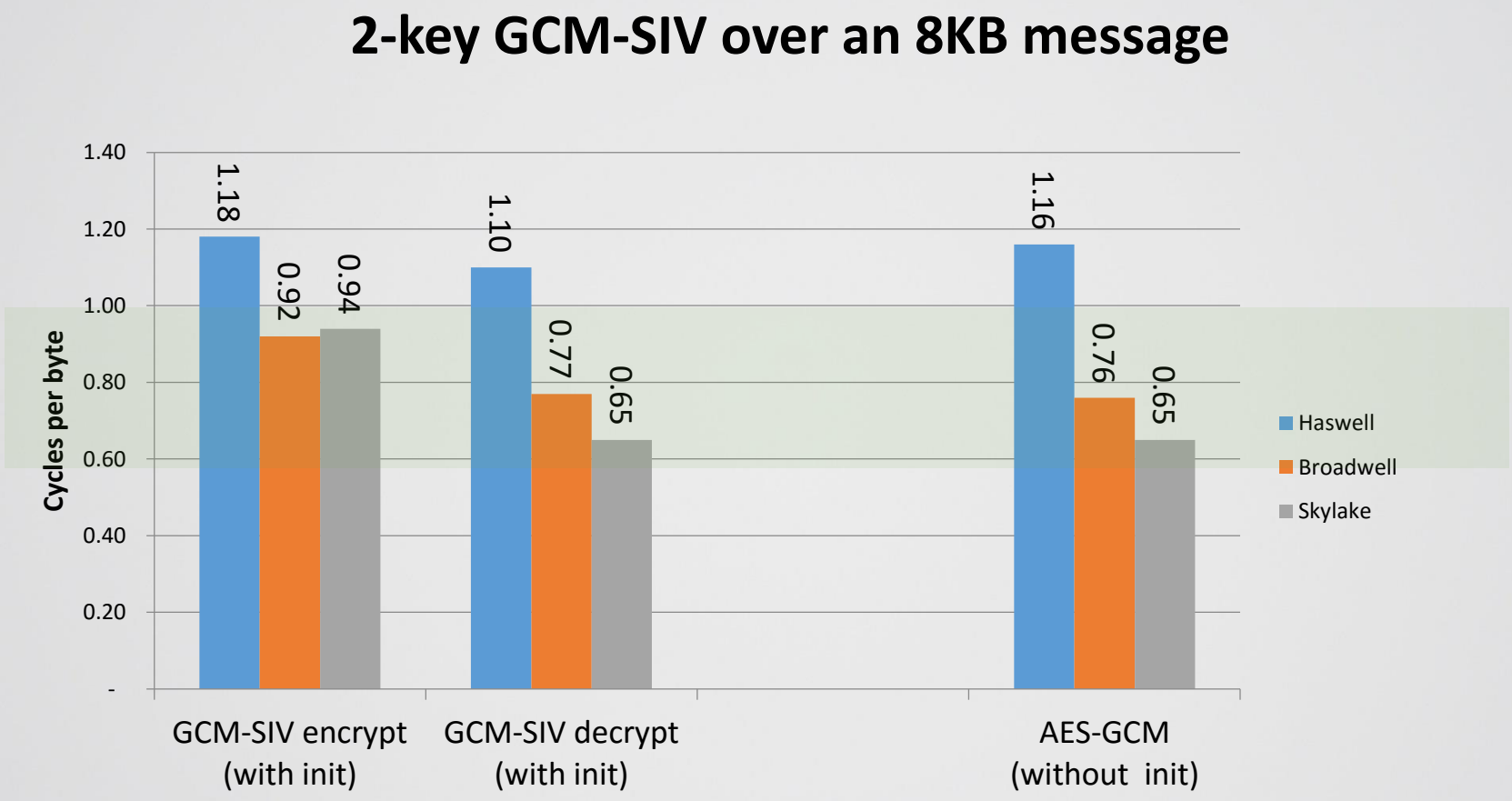


Efficiency of GCM vs GCM-SIV

- **Decryption:**
  - In **GCM**, once again CTR-DEC and GHASH interleaved
  - In **GCM-SIV**, can also interleave (decryption cost "should be" the same as the original GCM)



GCM-SIV Performance – Highlights



Time Comparison to AES-GCM

- **GCM-SIV (our implementation) is faster than (OpenSSL's best) AES-GCM for short messages, due to a new software optimization**

Full				No Init		
	Cycles			Cycles		
HSW/BDW	GCM-SIV	GCM-SIV	AES-GCM	GCM-SIV	GCM-SIV	AES-GCM
# bytes	Two keys	One key		Two keys	One key	
16	149 / 136	297 / 241	1289 / 1263	133 / 121	133 / 121	178 / 172
32	198 / 171	318 / 284	1277 / 1318	178 / 153	178 / 153	219 / 217
64	322 / 281	444 / 417	1292 / 1335	319 / 278	319 / 278	236 / 238
128	516 / 440	645 / 568	1415 / 1371	282 / 262	282 / 262	293 / 266
256	674 / 566	800 / 694	1558 / 1417	426 / 401	426 / 401	421 / 385
512	966 / 796	1093 / 930	1808 / 1730	722 / 626	722 / 626	760 / 651
1,024	1566 / 1252	1695 / 1385	2312 / 2108	1315 / 1085	1315 / 1085	1252 / 989
1,536	2159 / 1713	2274 / 1843	2816 / 2416	1907 / 1544	1907 / 1544	1714 / 1305
2,048	2751 / 2171	2869 / 2300	3372 / 2842	2498 / 1996	2498 / 1996	2287 / 1765
4,096	5118 / 4005	5244 / 4136	5332 / 4354	4867 / 3837	4867 / 3837	4296 / 3243
8,192	9862 / 7666	9994 / 7782	9521 / 7388	9611 / 7498	9611 / 7498	8399 / 6289
	C/B			C/B		
8,192	1.2/0.94	1.22/0.95	1.16/0.9	1.17/0.92	1.17/0.92	1.03/0.77

GCM-SIV Performance Comparison

- **GCM-SIV significantly outperforms all other implemented nonce-misuse resistant schemes**
  - Including all CAESAR round 1 candidates
  - Based on published authors' optimized implementations
  - When measured on modern x64 processors

- **The only exception is AEZ, which is based on a non-standard use of AES**

Summary

- Full nonce misuse-resistant authenticated encryption at an **extremely low cost** (almost AES-GCM)
- **Full proof of security** and **full implementation**
- **Easily deployable:**
  - Utilizes existing hardware
  - Utilize existing code and software (AES-GCM implementations)
- Detailed specifications, reference code and Open Source optimized code implementations coming soon
- Unpatented
- We hope to see it adopted

Thank You