NIST Workshop on the Requirements for an Accordion Cipher Mode 2024

June 20-21, 2024

Double-Nonce-Derive-Key-GCM (DNDK-GCM)

General design paradigms and application

Shay Gueron

University of Haifa, Israel and **Meta, USA**

shay.gueron@gmail.com

AES-GCM

- Secure (under a single assumption on AES)
- Fast ~0.88cpb (Skylake); ~0.25 (Icelake)
- Ubiquitous (part of TLS 1.3)
- NIST standard

Limitations

- Short nonce (96 bits) leading to
- Short key lifetime in *random nonce* setting
  Prob (nonce collision) ≤ 2^{-32} ➔
  ➔ Rotate after 2^{32} messages
- Birthday bound
  - Only 2^{52} bytes allowed
    with indistinguishability advantage ≤ 2^{-32}
    even with counter nonce

- No key commitment
- Easy to find K1 ≠ K2 such that
- Enc (K1, N1, A1, M1) = Enc (K2, N2, A2, M2)

1. Encrypt ≥ 2^{64} bytes
   w/ random nonces (no maintained state)
1. Security based on a single assumption
2. Enjoy NIST compliance umbrella
3. Good performance
   Enjoy AES/CLMUL speedups

4. While we are here, get (optional) key commitment

5. Minimum effort (*reuse existing code bases*)

Why not just use AES-GCM with a longer nonce?

- **Short *random* nonce setting (96 bits)**

➔High collision probability to

➔ frequent key rotation

- **And the Birthday Bound still remains**
  - **Due to AES block size (16B)**

➔ limiting one key to ≤ 2^{52} bytes

for distinguishing advantage ≤ 2^{-32}

Even with a (stateful) counter nonce

But AES-GCM can consume nonces of arbitrary lengths

- The key rotation limitation is not fully alleviated by using longer than 12 bytes nonces even with a stateful counter.
- If length (nonce) ≠ 12B
  - counter ← GHASH (H, ϕ, N) incremented over 4 bytes

    ➔ similar collision probability issue
  - Lower than with 12-byte random nonces, but higher than our goal

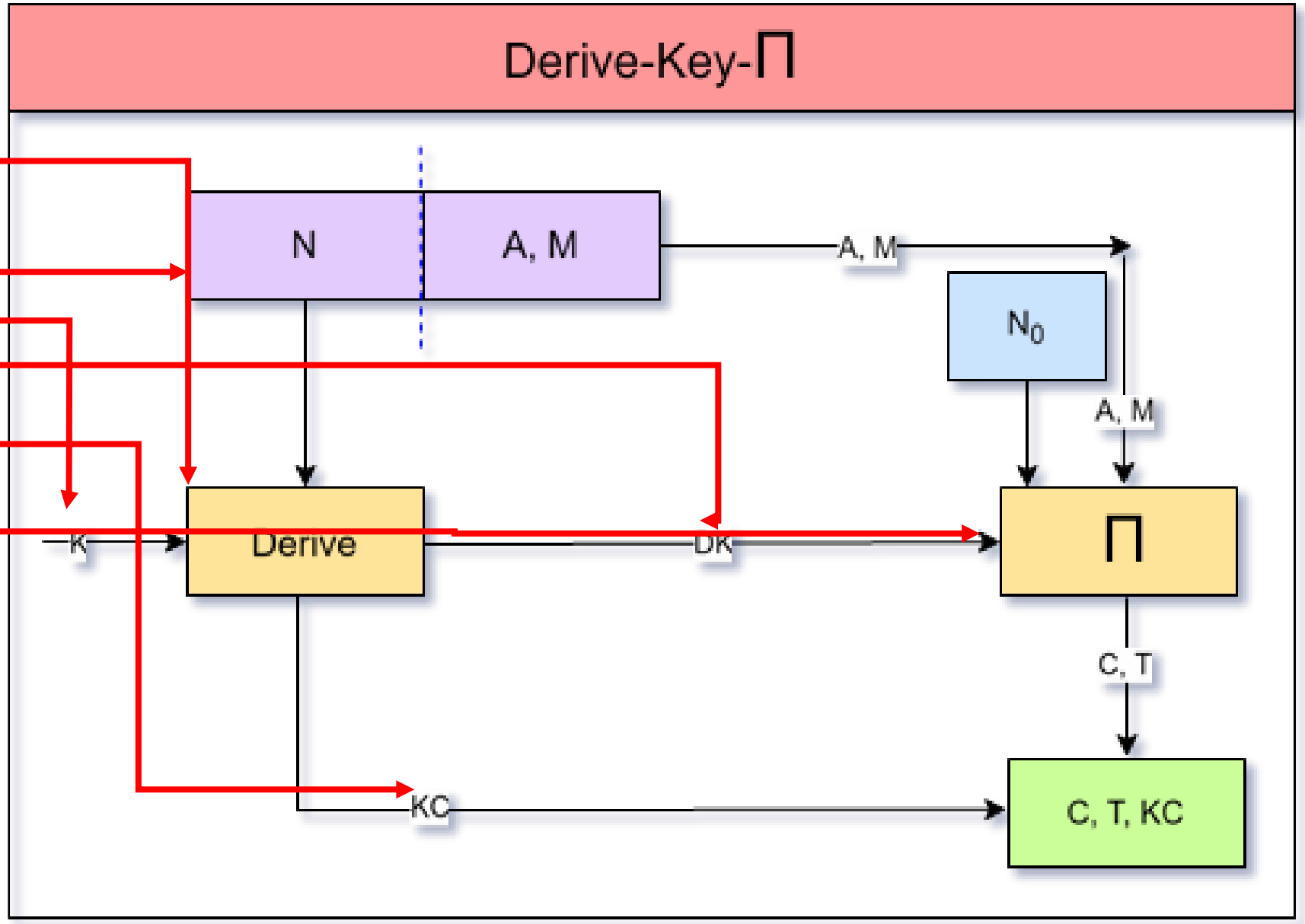
from AEAD Π to Derive-Key-Π

Derive: a PRF that takes nonce N and root key K
Outputs a fresh key DK (+ commitment value)

DK to be used with Π over the payload (& fixed nonce)

Generalization of [GL17] & AES-GCM-SIV [RFC8452]

Derive Key Concept



Derive-Key-Π is a general concept

- Can be mounted on any AEAD scheme
- In [GL17] has N0 the same as N
  - [GLL19] applied it to AES-GCM-SIV (with 12 bytes nonces)
- Generalization:
  - Decouple nonce N of Derive-Key-Π from the none N0 used with Π
  - Allow long N independently of the nonce length of Π

What nonce lengths to use for Derive-Key-Π?

What nonce lengths should we use? u = |N|

- Encrypt 2^{64} times with random nonces

  and keeping low nonce collision probability ➔

- Q^2/2^(u+1) ≤ 2^(-32) ➔ u = |N| ≥ 160

! we need u = |N| > 128 (=16B) does not fit in the AES block size

So?

What is a good Derive design for Derive-Key-Π?

- Options for Derive (K, N) → 64 bytes
  - `HMAC-SHA256 (K, N || 0x00) || HMAC-SHA256 (K, N || 0x01)`
  - `HMAC-SHA512 (K, N)`
  - `SHA256 (K, N || 0x00) || SHA256 (K, N || 0x01)`
  - `SHA512 (K, N)`
  - `Rijndael256 (K, N || 0x00*) || Rijndael256 (K, N || 0x01*)`
  - `Simpira256 in Even-Mansour mode`

- Reminder of the design goals:
  - "Have security based on a single assumption"
  - "Have good performance"

Derive-Key-Π – design decisions for choosing parameters

1. **(random) nonces are *long enough* and do not collide** — **e.g., 24 bytes**
2. **Derive is a BBB PRF**
3. **Derived keys are long enough s.t.** — **e.g., 32 bytes**
    1. **They do not collide in 2^64 derivations**
    2. **Multi-Key guessing (of derived keys) probability is negligible**

- **With properly chosen parameters for Derived-Key Π :**
    - **Every message is encrypted with Π under a one-time key (a special case of a multi key setting)**
- **So, we can decouple the Derived-Key Π nonce (IV) from the nonce used for Π**
    - **and use an arbitrary fixed N0**

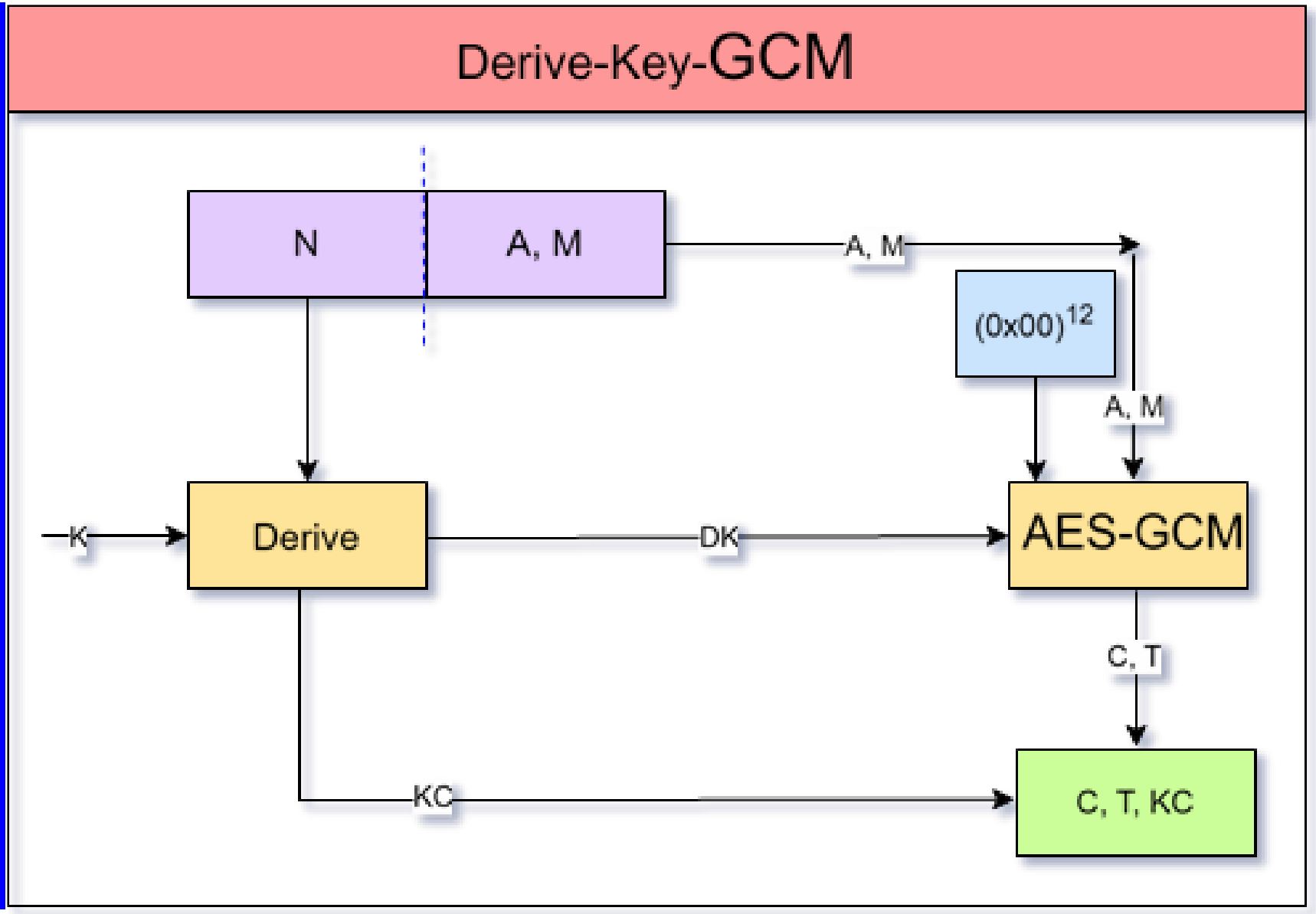
Derive-Key-GCM

Design questions:

- Nonce length?
- Root key size?
- Derived key size?

How to instantiate Derive?

What PRF
How to handle |N| > 16B



DNDK-GCM: transforms AES-GCM to a "multi key" setting

- **Double-Nonce-Derive-Key-AES-GCM** (K, N, A, M) $N \leftarrow_{\$}$
  - (KC, DK) = Derive (K, N) Double nonce: |N| = 24 B
  - (C, Tag) = AES256-GCM (DK, $(0x00)^{12}$, A, M)
  - Output: N, C, T, KC

Derive (K, N): some permutation-based (AES) BBB PRF
Achieve minimalism → only AES as a building block.

192-bit random nonces do not collide
256-bit (derived) keys do not collide
Derive is a (permutation based) BBB PRF
One assumption: AES is a good PRP

➔ **Every message is encrypted with a one-time key (multi key setting)**
Every key is used for 1 message ($|M| \leq 2^{32}-1$ blocks)

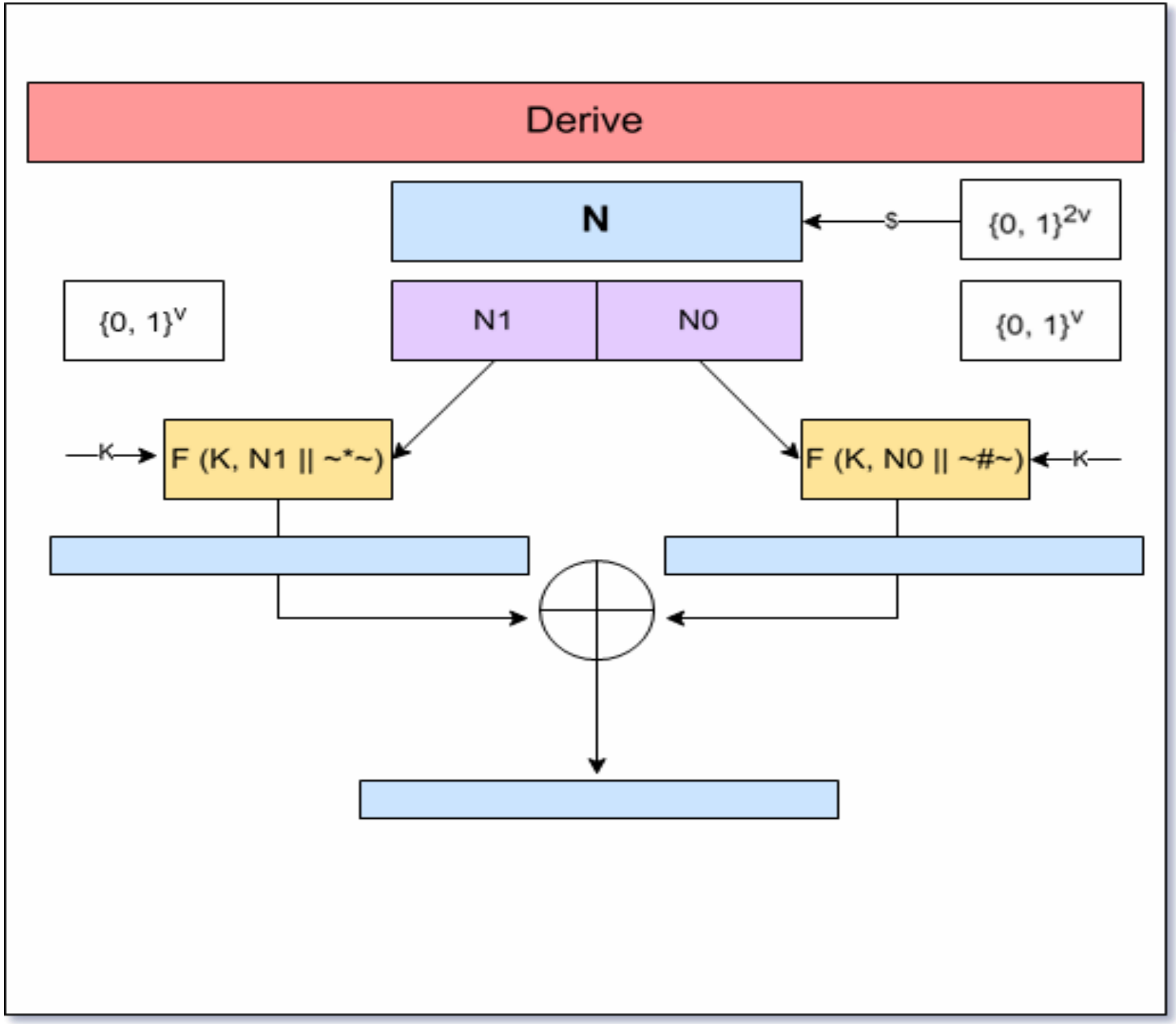
Indistinguishability with Q messages of L blocks dominated by $Q \times (L+2)^2 / 2^{129}$

AES-based Double-Nonce-Derive-Key

Split (**random**) double nonce
into 2 halves
Apply twice
AES based) PRF F
XOR 2 PRF outputs

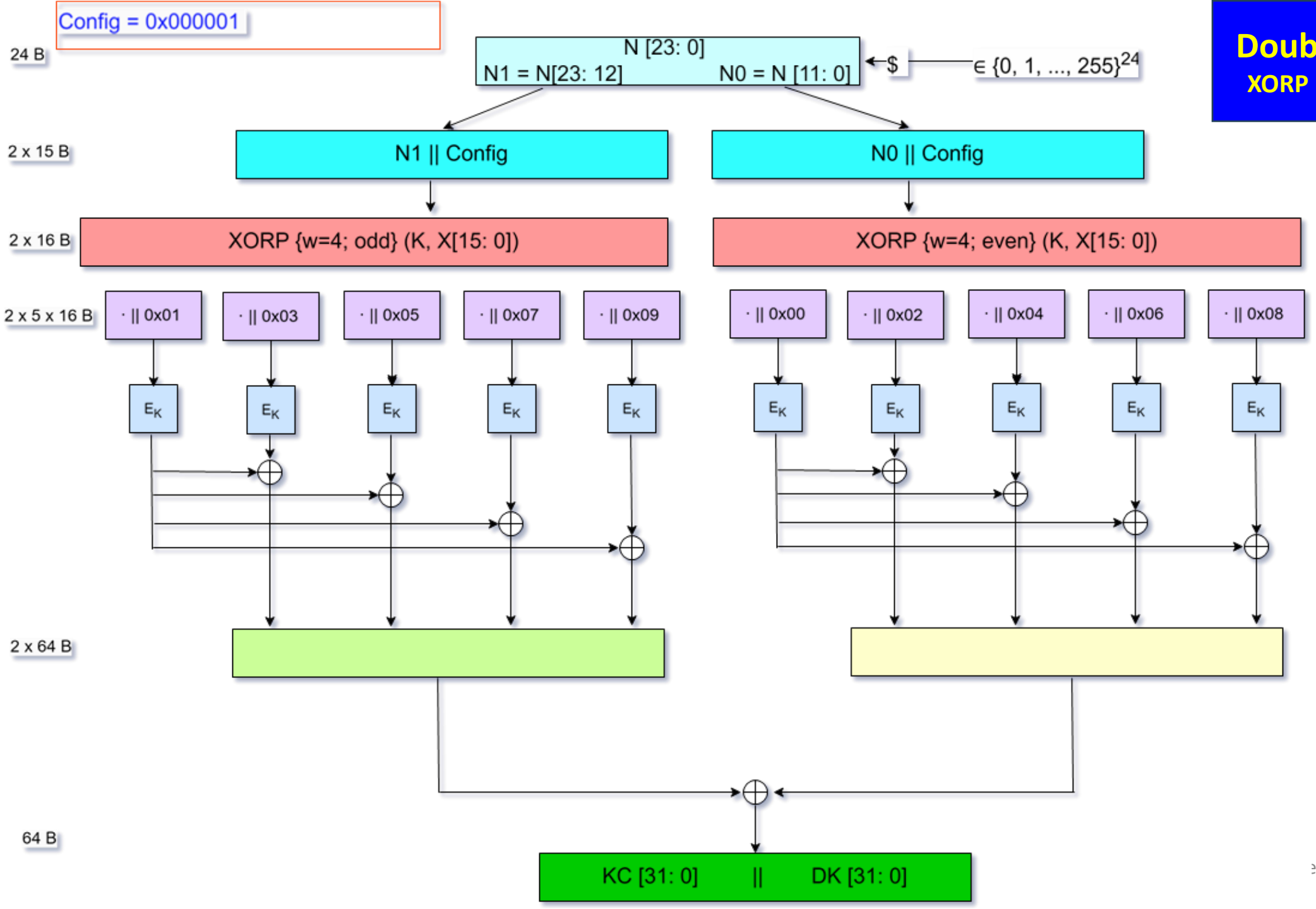
57*Q^2/2^(2v)

Bellare, Goldreich, Krawczyk 1999, “Stateless evaluations of PRFs”



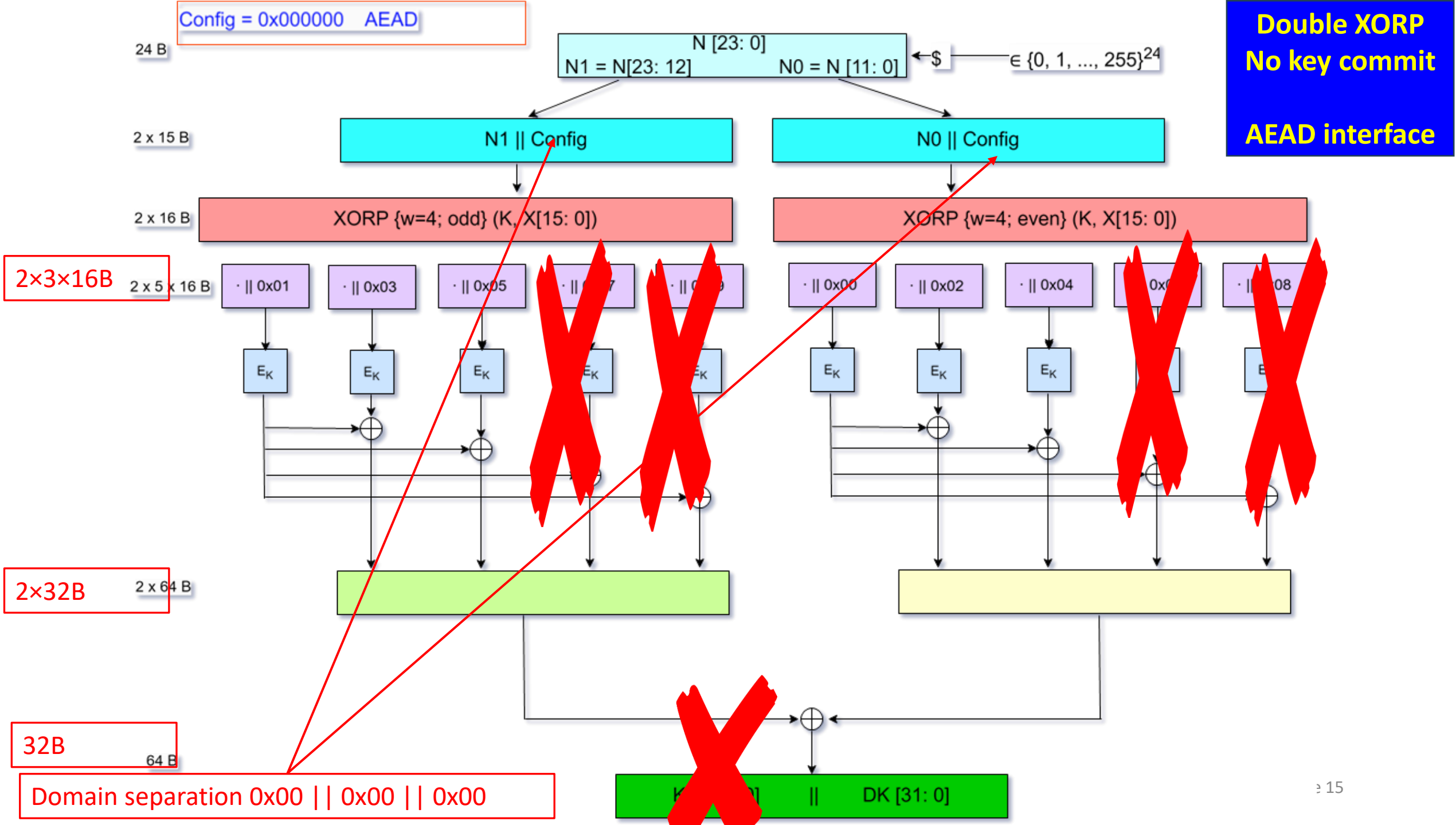
Double XORP

XORP [Iwata 06]



Other variants of DNDK-GCM

- I don’t need key commitment in my usage
- I want a standard AEAD interface
  - Why should I invoke AES 10 times?


Config = 0x000000 AEAD
24 B
N [23: 0]
N1 = N[23: 12]
N0 = N [11: 0]
$
∈ {0, 1, ..., 255}^24
Double XORP
No key commit
AEAD interface
2 x 15 B
N1 || Config
N0 || Config
2 x 16 B
XORP {w=4; odd} (K, X[15: 0])
XORP {w=4; even} (K, X[15: 0])
2×3×16B
2 x 5 x 16 B
· || 0x01
· || 0x03
· || 0x05
· || 0x00
· || 0x02
· || 0x04
2×32B
2 x 64 B
32B
64 B
DK [31: 0]
Domain separation 0x00 || 0x00 || 0x00

Counter nonce instead of random nonce (IV)?

- Can I just use a 24B counter nonces instead of a random IV?
  - No!
  - (at least without constraints on their selection).
- Free choice of non-repeating but non-random nonces could lead to a distinguisher (zero XOR sum) in the derived keys.
  - Example: F (K, N1 ||) F(K, N0 ||)

    (0x11)^12 || (0x21)^12

    (0x11)^12 || (0x22)^12

    (0x12)^12 || (0x21)^12

    (0x12)^12 || (0x22)^12

Counter nonce instead of random nonce (IV)?

Sure we want this at all?

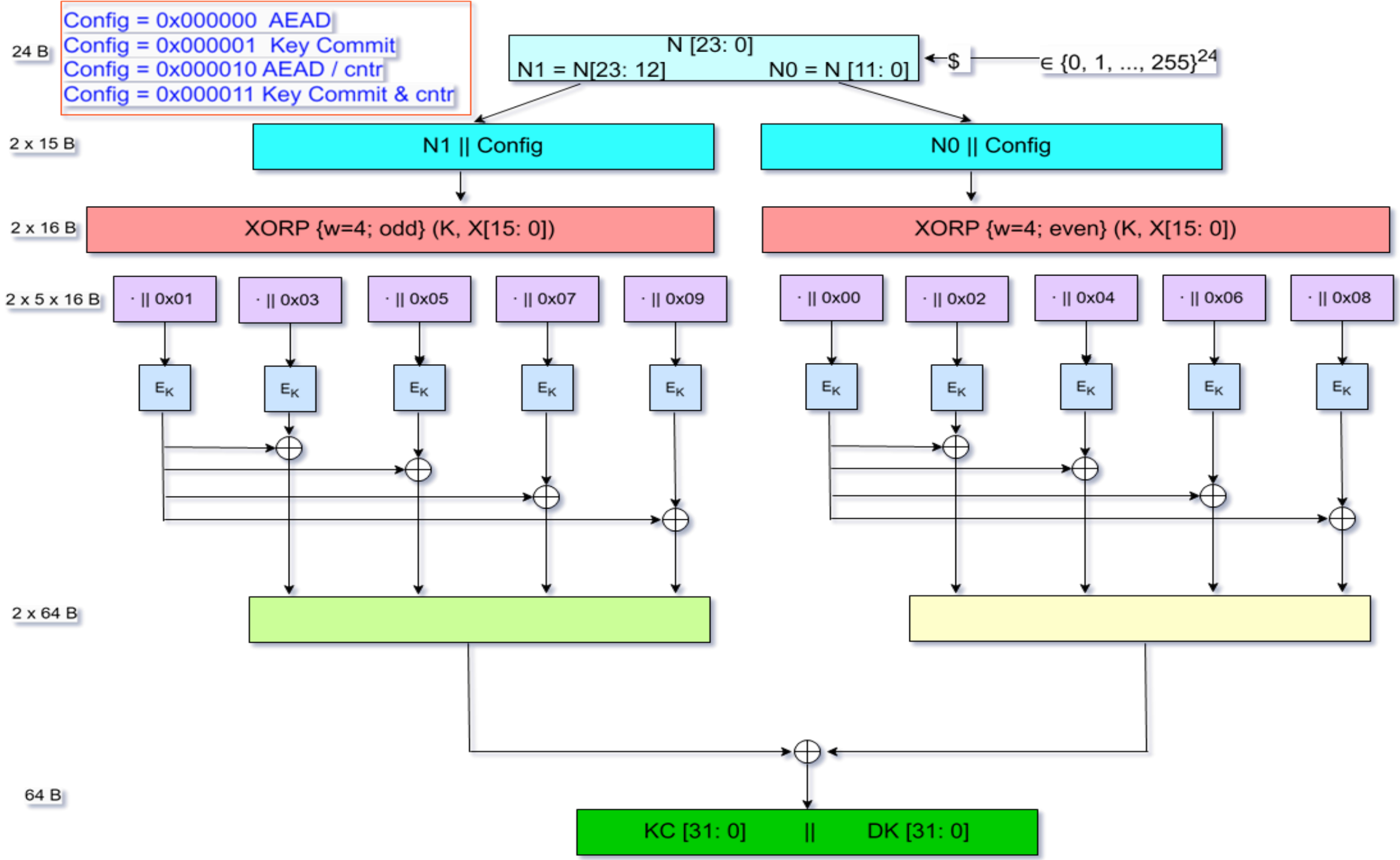
- This unique but nonrandom nonce setting would work:
  (though DNDK is intended to use random IVs)
- A “degenerated” double nonce
  - Freeze half of the 24B nonce to some fixed value
  - Populate the other half with a 12-byte running counter
  - Noce format: (0xff)^12 || counter
    - (0xff)^12 || 0x000000000000000000000001
    - (0xff)^12 || 0x000000000000000000000002
    - (0xff)^12 || 0x000000000000000000000003
    - …

Variants of DNDK-GCM

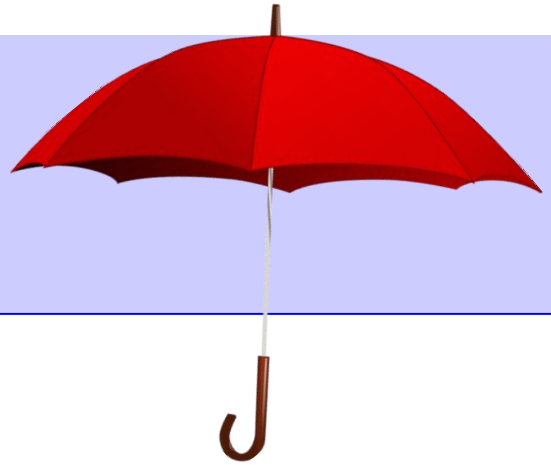
Controlled by the Config value (3 bytes)

Different derived keys

Config = 0x000000	**Standard AEAD interface**
Config = 0x000001	**With Key Commit**
Config = 0x000010	**AEAD interface with “half” counter**
Config = 0x000011	**With Key Commit and “half” counter**


24 B
Config = 0x000000 AEAD
Config = 0x000001 Key Commit
Config = 0x000010 AEAD / cntr
Config = 0x000011 Key Commit & cntr
N [23: 0]
N1 = N[23: 12]
N0 = N [11: 0]
$
∈ {0, 1, ..., 255}^24
2 x 15 B
N1 || Config
N0 || Config
2 x 16 B
XORP {w=4; odd} (K, X[15: 0])
XORP {w=4; even} (K, X[15: 0])
2 x 5 x 16 B
· || 0x01
· || 0x03
· || 0x05
· || 0x07
· || 0x09
· || 0x00
· || 0x02
· || 0x04
· || 0x06
· || 0x08
E_K
2 x 64 B
64 B
KC [31: 0] || DK [31: 0]

What about compliance?



- Is DNDK-GCM covered under the NIST umbrella?

- Derived-Key Π paradigm only converts an underlying Π to a multi key scenario
  - (every message is encrypted with Π under a one-time key)
- DNDK-GCM is *not* really a new AEAD. Rather,
  - *DNDK-GCM is just a way to use the NIST standard AES-GCM*
  - (with a pseudorandom set of keys – one per message)

Performance: DNDK-GCM vs. plain AES-GCM

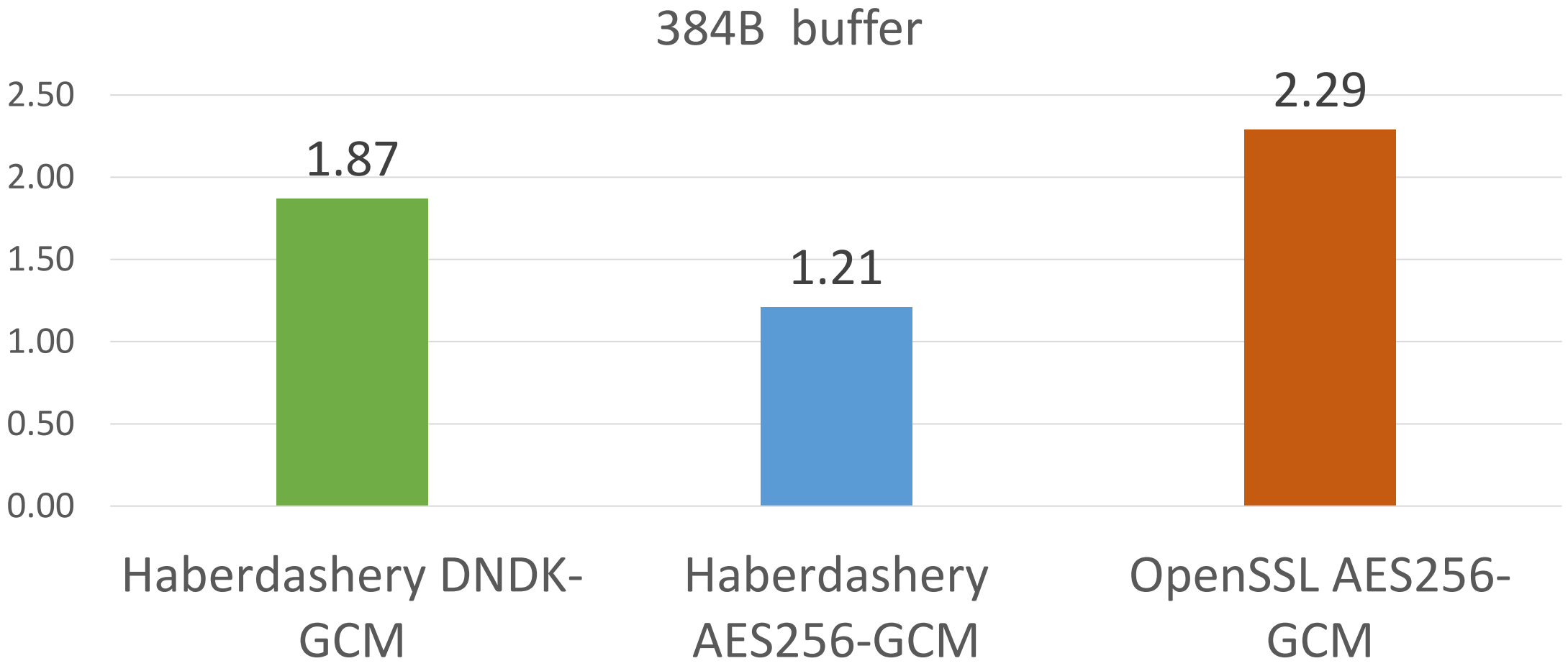
These are the overheads on top of AES-GCM:

1. Derive (+ commitment) takes 10 AES calls over independent blocks

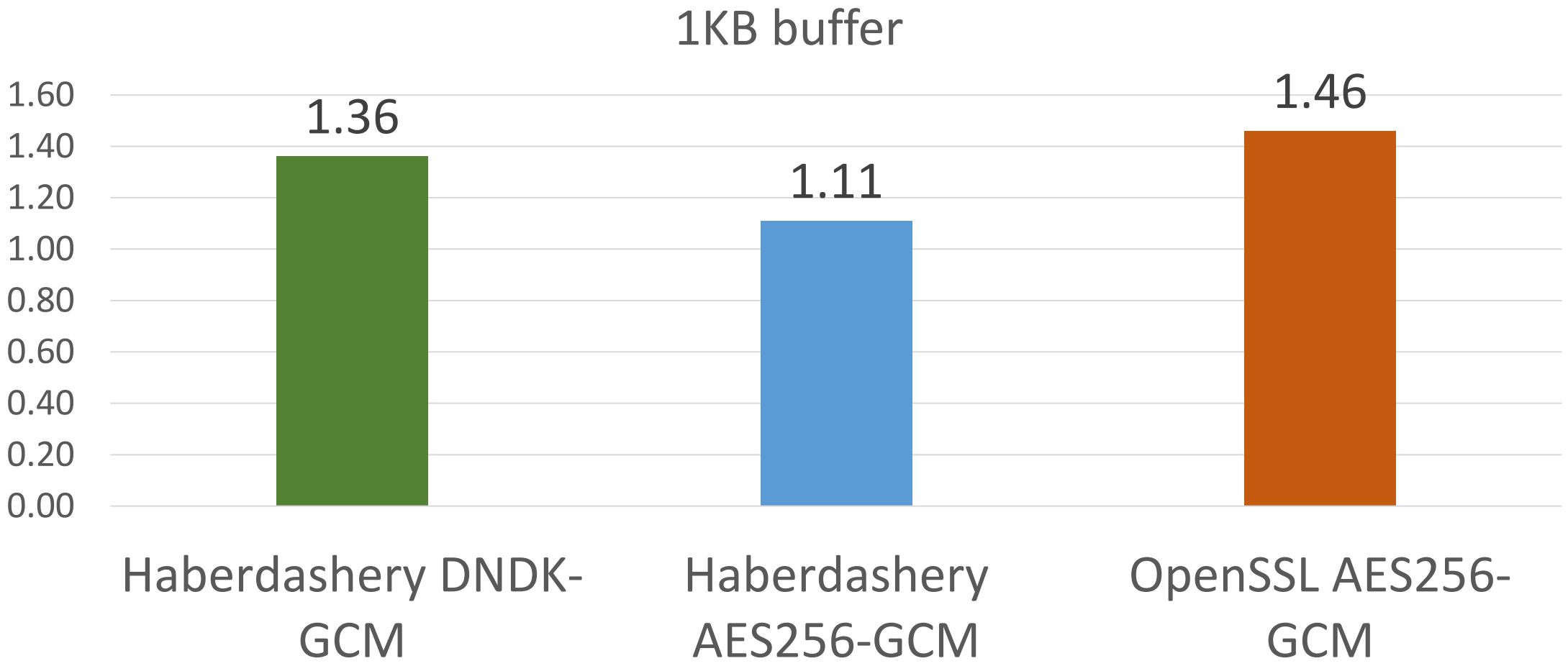
   Without commitment: only 6 AES calls
2. AES256 setup (key expansion, H = AES (K, 0^{128}), H powers (Htable)
3. Bandwidth: +12B (24B double nonce) +32B (or less) for KC

- Cost of DNDK-GCM “init”: Derive + AES256-GCM setup : ~175 cycles

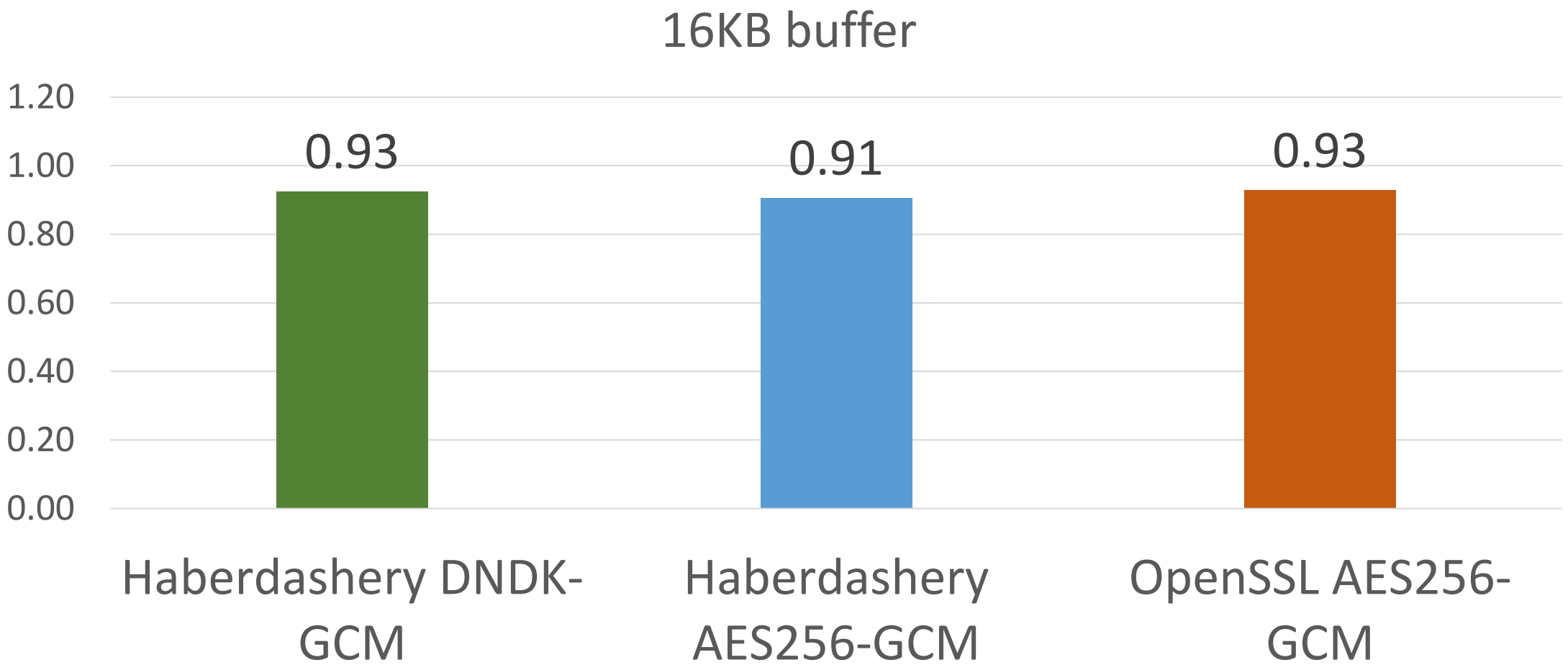
Performance on a short buffer (cpb; lower is better)



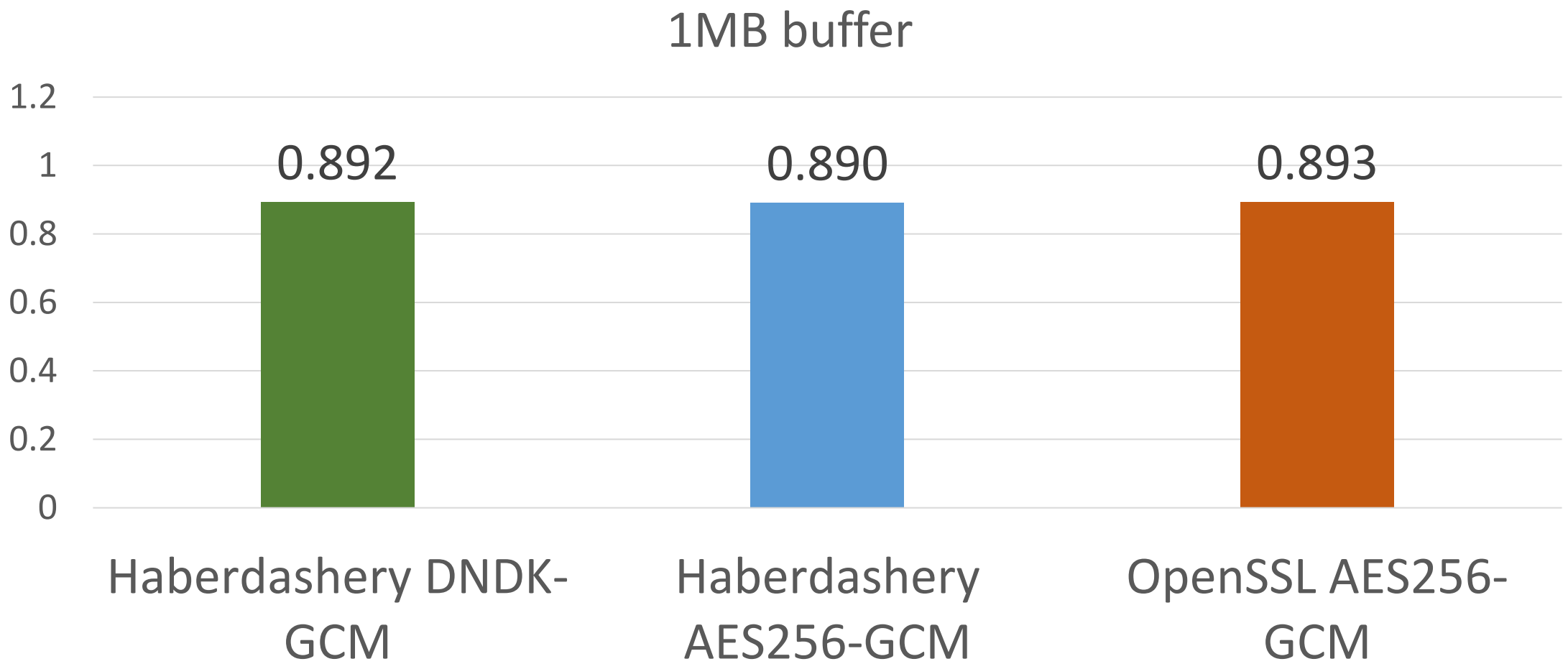
Performance on a medium length buffer (cpb; lower is better)



Performance on a long buffer (cpb; lower is better)



Performance: asymptotic (cpb; lower is better)



Other variants of DNDK-GCM

Other variants of DNDK-GCM? DNDK-*?
Yes! (following the same paradigm)

Other variants of DNDK-GCM

- Different optional parameters
  - Shorter root key L1 bytes
  - Shorter derived keys 2v bytes
  - Shorter nonces L2 bytes
  - Key commitment L3 bytes
  - DNDK-GCM with KC has Derive-32-24-32-32
- Other options :
  - 24B root key; 24B nonce; 24B derived key; no KC
    - Derive-24-24-24-0
  - Performance:
    - Derive-24-24-24-0 is 1.44x faster than Derive-32-24-32-0
    - For 3KB buffer: 0.87cpb vs. 0.99 cpb

Use domain separation Config in Derive to distinguish options and avoid misuse

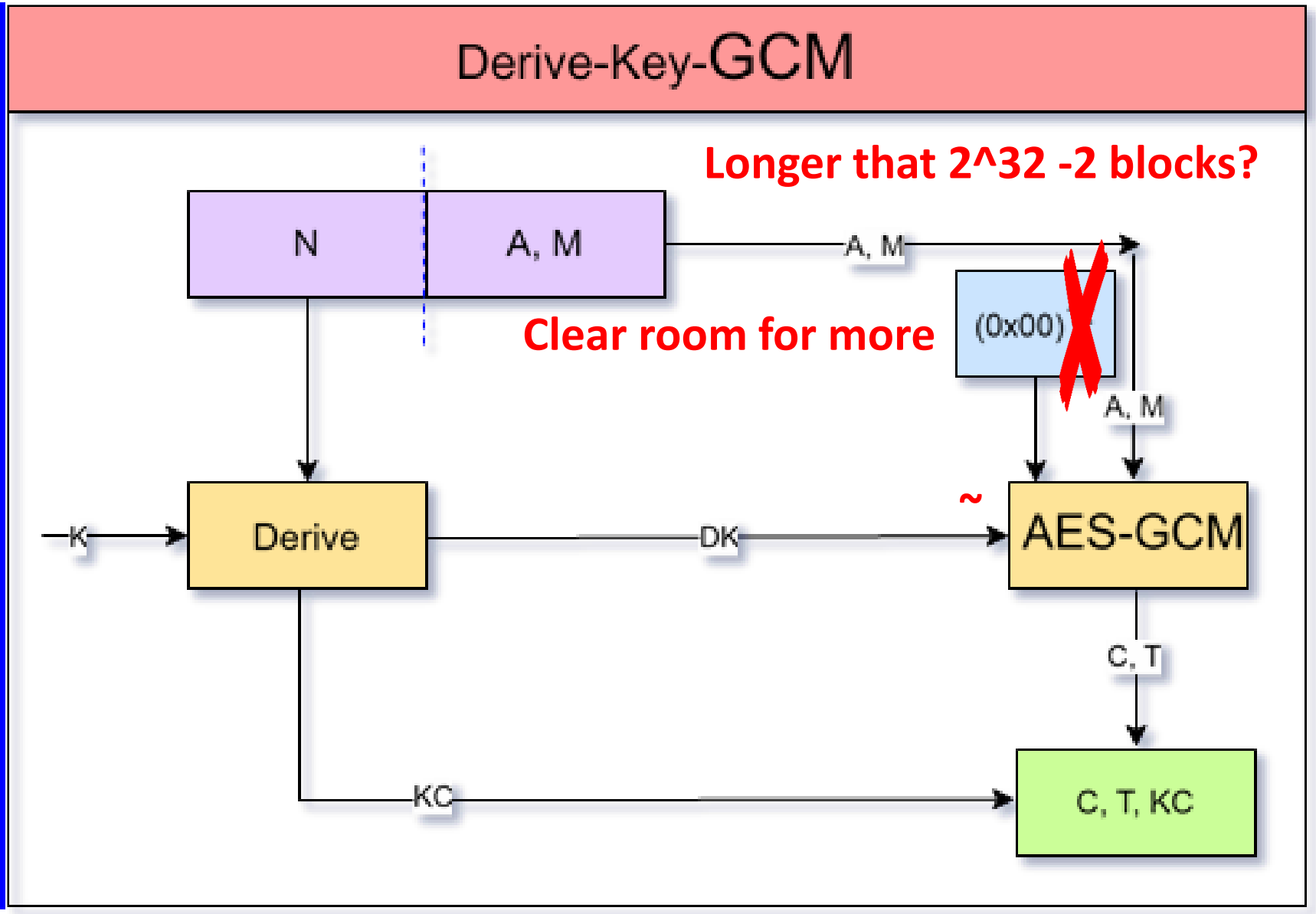
Derive-Key-GCM

Design questions:

- Nonce length?
- Root key size?
- Derived key size?

How to instantiate Derive?

What PRF
How to handle |N| > 16B



Where can I get details & DNDK-GCM (optimized) code?

- **Haberdashery:** an open-source library of high performance (assembly) implementations of a collection of (symmetric) algorithms:
  - AES-GCM, AES-GCM-SIV, **DNDK-GCM**, SIV-MAC
- Get it from https://github.com/facebookincubator/haberdashery
- Internet RFC draft:
- https://datatracker.ietf.org/doc/draft-gueron-cfrg-dndkgcm/

Summary: DNDK-GCM Double-Nonce-Derive-Key-AES-GCM

- (K, N, A, M) Double nonce: |N| = 24 B
  - (KC, DK) = Derive (K, N) → (C, Tag) = AES256-GCM (DK, $(0x00)^{12}$, A, M)

- Solving the AES-GCM (random nonce) limitations
  - To be used with random nonces (no maintained state)
  - Extended key lifetime & better security bounds $Q \times (L+2)^2 / 2^{129}$ **Can process 2^{64} blocks**
  - Key commitment value (optional)
    - Controlled through a "Config" value

- Relatively small overhead
- Only one security assumption ("AES is a good PRP")
- Simple design. Trivial code addition over AES-GCM (can reuse existing AES-GCM code)
- Not a new AEAD. Rather, *DNDK-GCM is just way to use the NIST standard AES-GCM*

DNDK is the default encryption mode at Meta

Thank you

Acknowledgements

(in alphabetic order of last names)

- **Sasha Frolov:** rollout of DNDK-GCM in production at Meta
- **Isaac Elbaz:** facilitation support
- **Ed Knapp:** Haberdashery implementation

References

[BGK99] Bellare, M., Goldreich, O., Krawczyk, H. (1999). Stateless Evaluation of Pseudorandom Functions: Security Beyond the Birthday Barrier. In: Wiener, M. (eds) Advances in Cryptology — CRYPTO' 99. CRYPTO 1999. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48405-1_17

[GL17] Gueron, S. and Y. Lindell, "Better Bounds for Block Cipher Modes of Operation via Nonce-Based Key Derivation", Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017. [Online]. Available: https://doi.org/10.1145/3133956.3133992

[GLL17] Gueron, S., Langley, A., & Lindell, Y. (2017). AES-GCM-SIV: Specification and Analysis. *Cryptology ePrint Archive, Paper 2017/168*. URL: https://eprint.iacr.org/2017/168

[Iwa06] Iwata, T., "New Blockcipher Modes of Operation with Beyond the Birthday Bound Security", Fast Software Encryption, FSE 2006, Revised Selected Papers, Lecture Notes in Computer Science, vol. 4047, pp. 310–327, Springer, 2006. [Online]. Available: https://doi.org/10.1007/11799313_20

[RFC8452] Gueron, S. Langley, A, and Lindell, Y., "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption," RFC 8452, RFC Editor, April 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8452