AES-OTR v3

Kazuhiko Minematsu (NEC Corporation)

DIAC 2016, Nagoya Japan

AES-OTR (v3)

- OTR: a blockcipher mode of operation for Nonce-based AE (NAE) [M14]
  - Based on OCB, removing the need for blockcipher decryption function needed by OCB ("inverse-free")
- AES-OTR for CAESAR

Features:

- Rate-1 (needs one AES call for one block)
- Parallelizable for encryption and decryption
- No AES decryption (inverse-free)
  - Unique AES-based NAE CAESAR candidate featuring all of them

[M14] Minematsu. Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions, Eurocrypt 2014

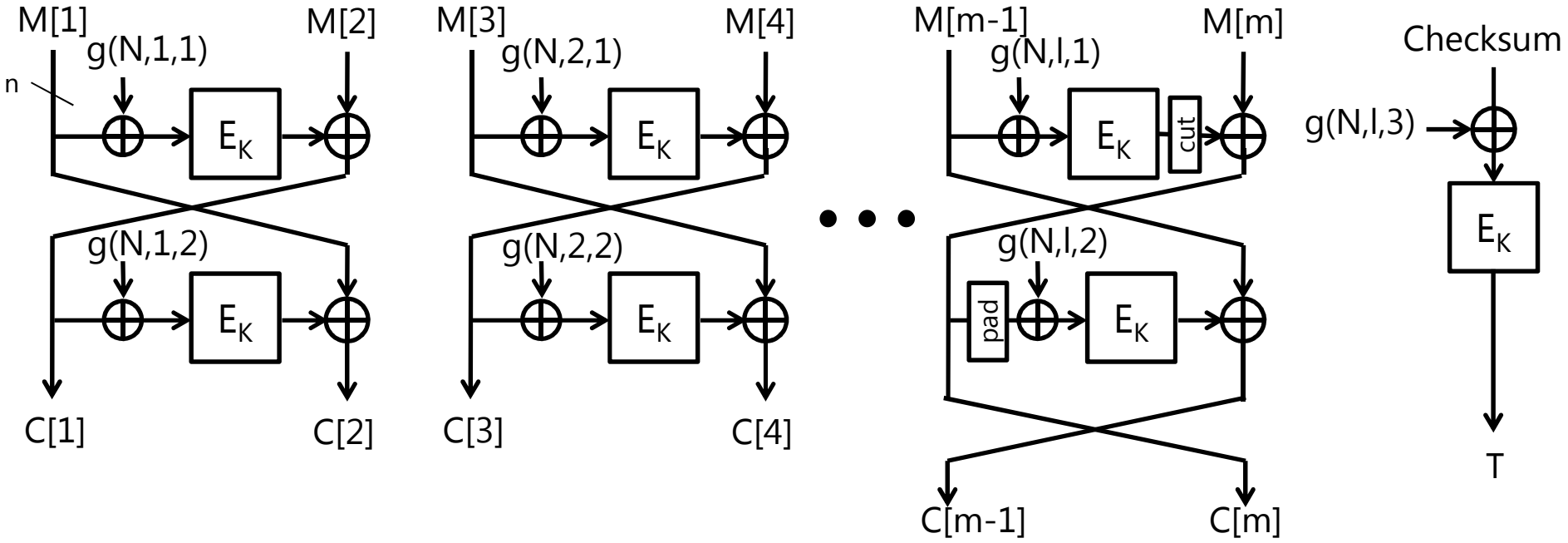
Limitations (as v2)

- NAE: nonce must be unique for encryption
  - No protection against nonce-misuse and decryption-misuse (RUP setting)
  - Needs outer protection such as [FJMV03]
- Theoretical limit speed and size : AES in counter mode

[FJMV03] Fouque, Joux, Martinet, Valette. Authenticated On-Line Encryption. SAC 2003

Algorithms of OTR

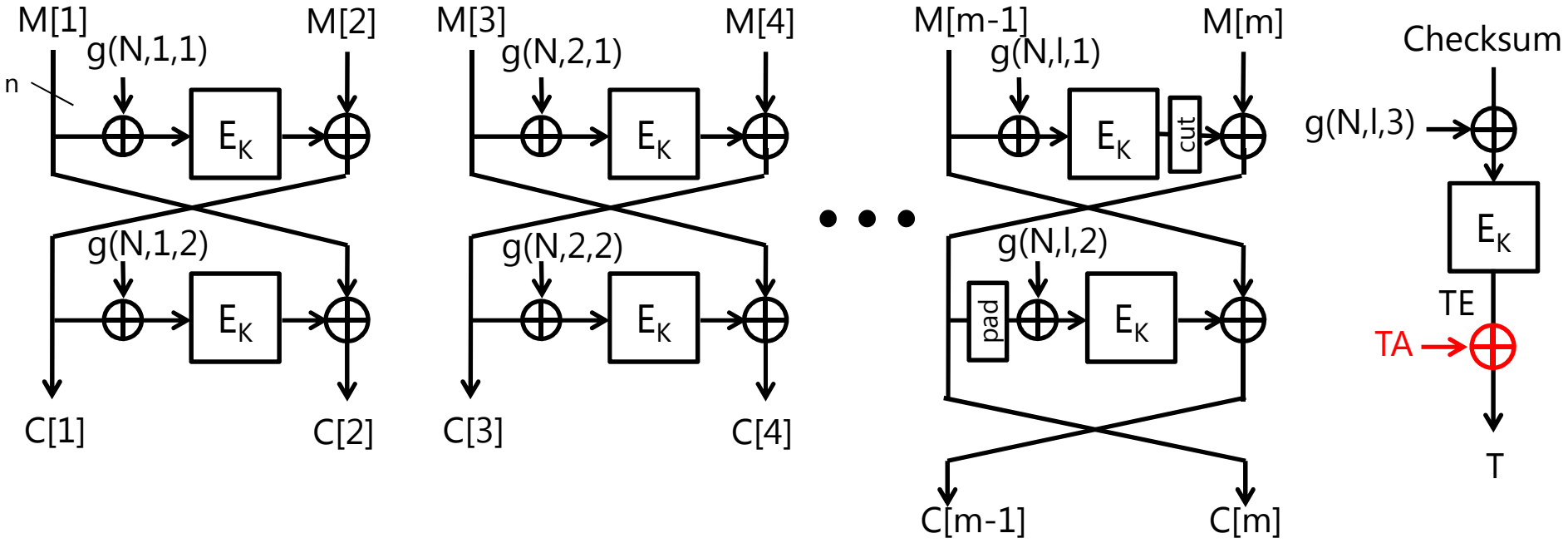
- Two-round Feistel for encryption
- Taking a sum of even plaintext blocks (checksum)
- Different operations for even and odd plaintext blocks



OTR (no AD, even blocks)

Algorithms of OTR

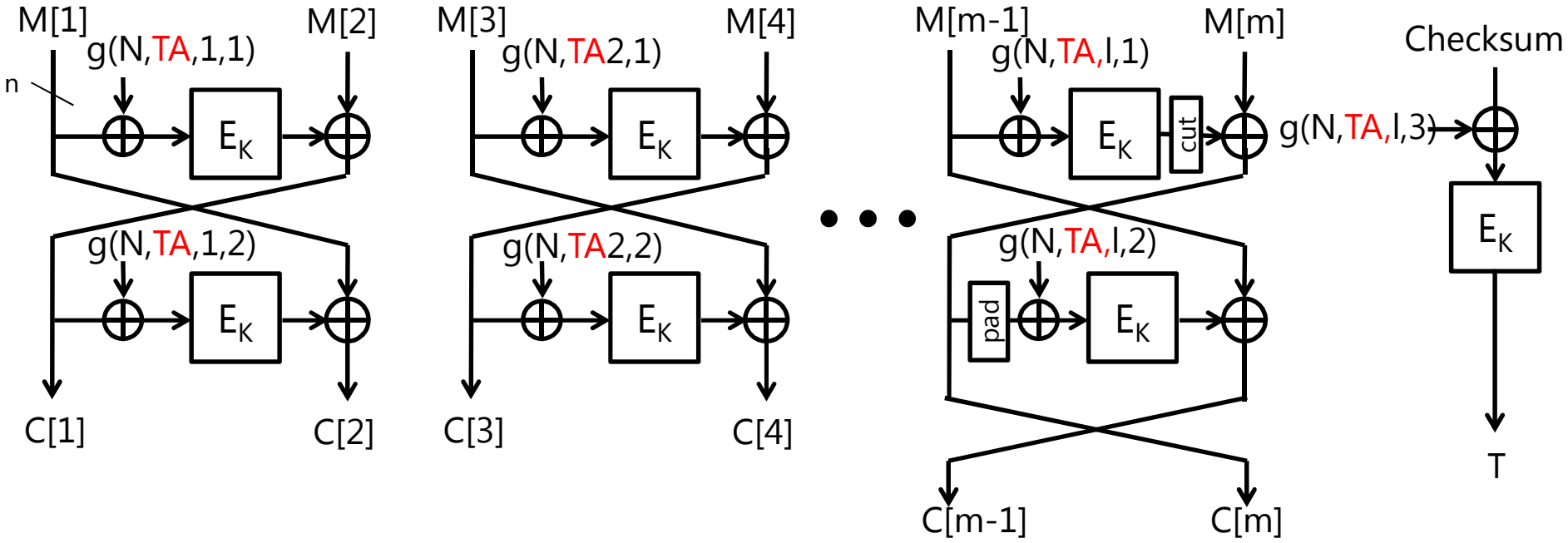
- Associated data processing, ADP
- ADP=P: PMAC output (TA) xored to tag
- ADP=S: CMAC output xored to the initial mask



OTR (ADP=P, even blocks)

Algorithms of OTR

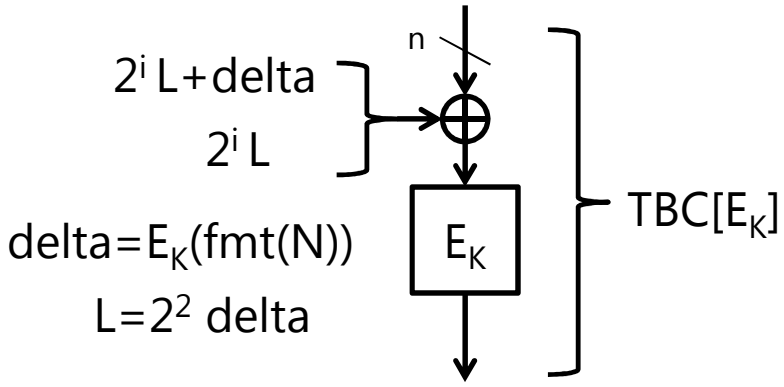
- Associated data processing, ADP
- ADP=P: PMAC output (TA) xored to tag
- ADP=S: CMAC output (TA) absorbed into the first mask generation



OTR (ADP=S, even blocks)

A change from v2

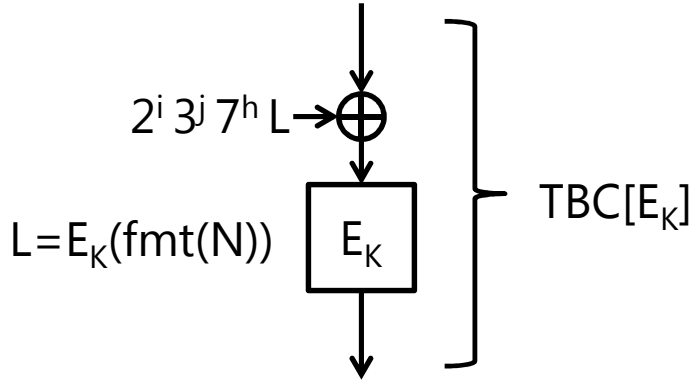
- An issue about tweaking by Bost and Sanders [BS16]
- Underlying n-bit TBC not exactly following XE [R04]
  - can increase the security bounds (=reduce the maximum acceptable amount of data per key).
  - The effect depends on the polynomial defining the field GF(2^n)
    - For v2, to some extent



[R04] Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. Asiacrypt 2004
[BS16] Bost and Sanders. Trick or Tweak: On the (In)security of OTR's Tweaks. Asiacrypt 2016

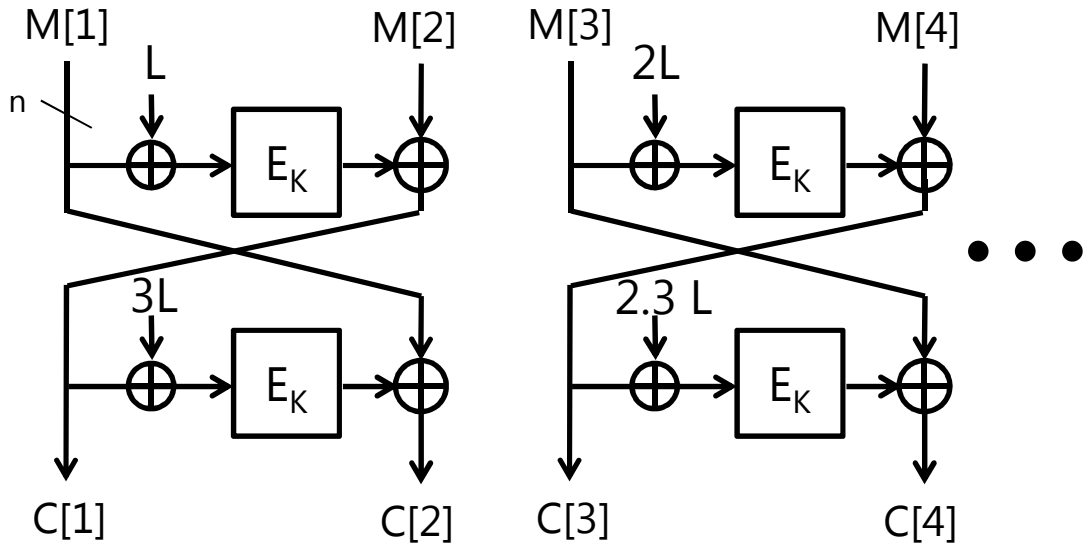
Tweaking for v3

- TBC of v3 now strictly follows XE (of 128-bit block)
- Proofs unchanged once TBC is established
- Security bounds are the same as initially claimed at v2



Tweak constants

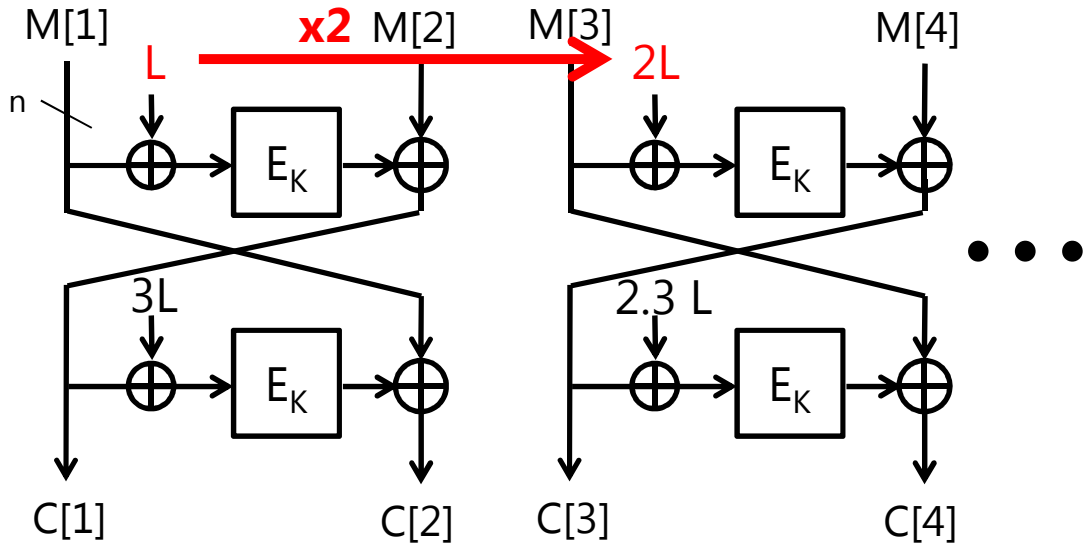
- 1 GF doubling per 2 blocks
- Suitable for parallel processing
- Our choice is similar to one suggested by [BS16]
- Keeping the same efficiency as tweaking in v2



[BS16] Bost and Sanders. Trick or Tweak: On the (In)security of OTR's Tweaks. Asiacrypt 2016

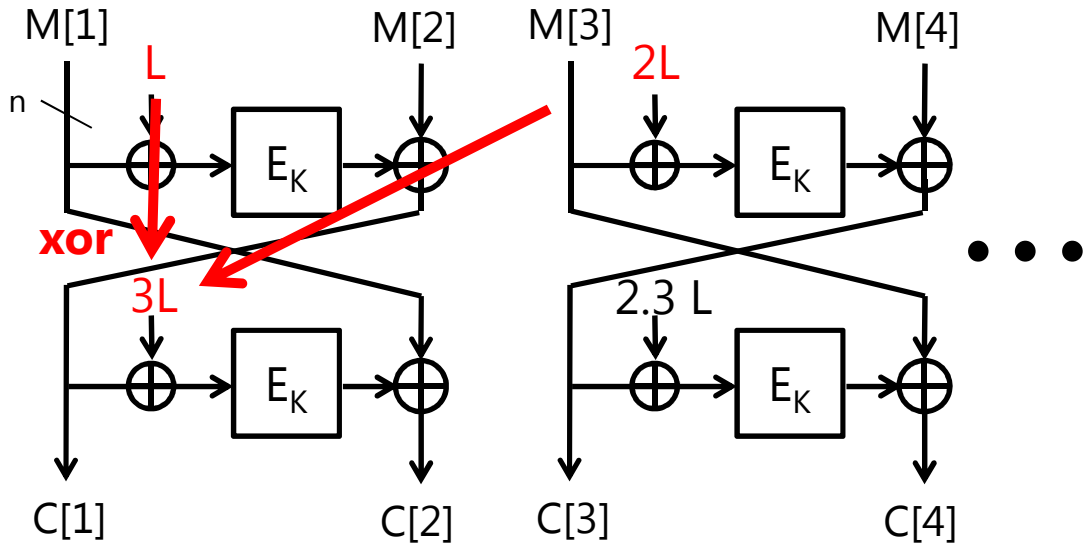
Tweak constants

- 1 GF doubling per 2 blocks
- Suitable for parallel processing
- Our choice is similar to one suggested by [BS16]
- Keeping the same efficiency as tweaking in v2



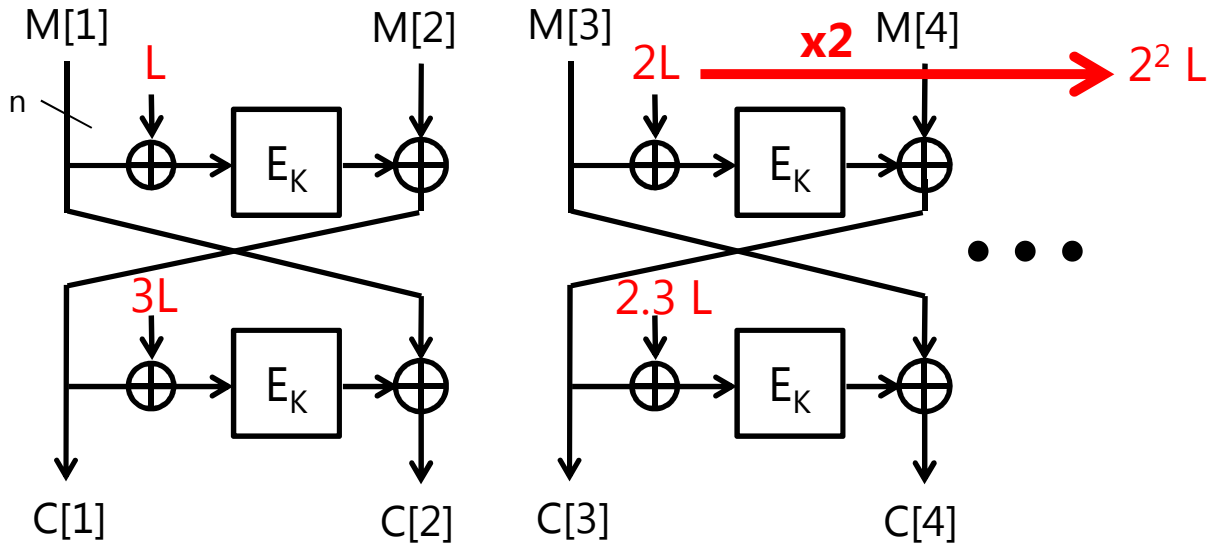
Tweak constants

- 1 GF doubling per 2 blocks
- Suitable for parallel processing
- Our choice is similar to one suggested by [BS16]
- Keeping the same efficiency as tweaking in v2



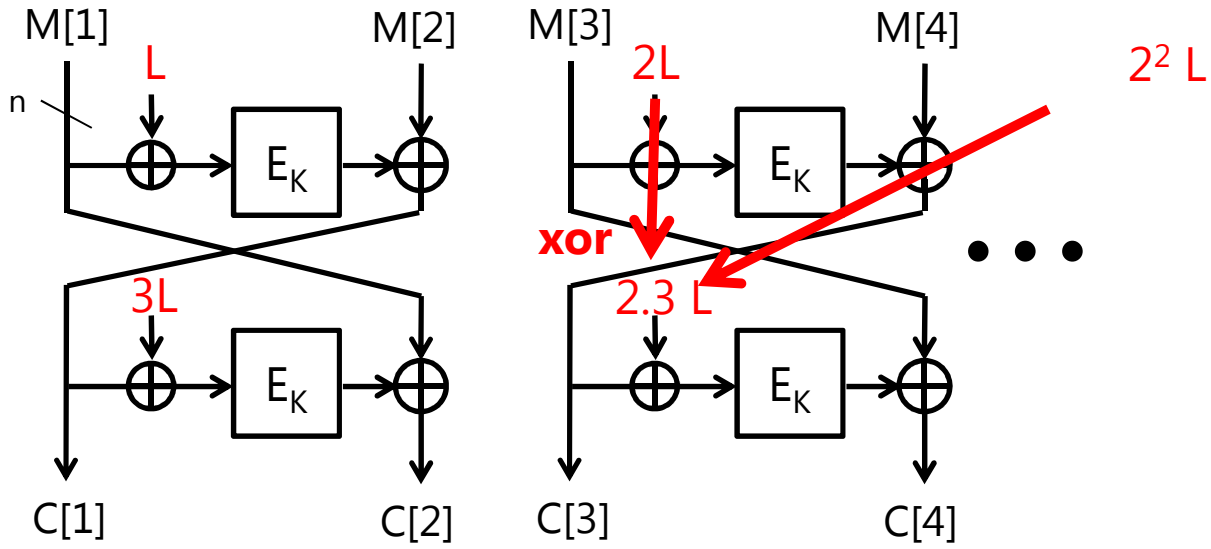
Tweak constants

- 1 GF doubling per 2 blocks
- Suitable for parallel processing
- Our choice is similar to one suggested by [BS16]
- Keeping the same efficiency as tweaking in v2



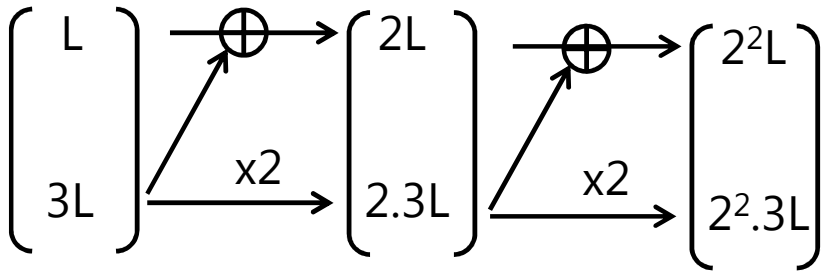
Tweak constants

- 1 GF doubling per 2 blocks
- Suitable for parallel processing
- Our choice is similar to one suggested by [BS16]
- Keeping the same efficiency as tweaking in v2



Tweak constants

- 1 GF doubling per 2 blocks
- Suitable for parallel processing
- Similar to one suggested by [BS16]
- Keeping the same efficiency as tweaking in v2



Goal of OTR

- Fast operations for a wide variety of platforms
  - Good for heterogeneous networks
- Inverse-freeness can contribute to
  - Size
  - Speed, but not always (e.g. AESNI)
  - Security (PRP)
  - Ease of implementation

Performance of OTR : AESNI

- Intel/AMD CPUs with AES instructions (AESNI)
- Using intrinsic
- Software pipeline (i.e. way to efficiently compute AES in parallel)
  - Common trick applicable to parallelizable modes
- Batch GF-doubling optimization [A13][MSK15]

- After some efforts…
- Result on Skylake processor (Intel Core i5-6600) with AES-128
- 0.68 c/b for long message at SUPERCOP
  - OCB : 0.64 c/b
  - Raw AES in theory : 10/16 = 0.625 c/b

[A13] Aoki, Optimization of mode implementations on Sandy Bridge. SCIS 2013 (in Japanese)
[MSK15] M, Shigeri, Kubo. AES-OTR v2. DIAC 2015

Performance of OTR : ARMv7 [DIAC15]

- Beaglebone Black (Cortex-A8 1GHz)

- Combining single-block T-table AES with Bitslice AES taken from SUPERCOP
  - BS-AES by Kasper and Schwabe [KS09]
  - Bitslice processes 8 blocks in parallel
    - Only AES-Encryption code available, which is sufficient for OTR
  - Single-block for Nonce/tag encryption
- Use NEON SIMD engine
- Use intrinsic

[KS09] Kaesper, Schwabe. Faster and Timing-Attack Resistant AES-GCM. CHES 2009

Performance of OTR : ARMv7 [DIAC15]

- Peak speed : ~23.5 c/b (+7% of AESBS)
- For reference, in Gouvea-Lopez's GCM runs 32.8 c/b, using BS-AES, on Cortex A9 [GL15]

AES-OTR (AES-128, no AD)

Msglen (byte)	Enc (median c/b)	AESBS ratio	Dec (median c/b)	AESBS ratio
1056	25.42	1.14	25.42	1.14
2080	24.19	1.07	24.2	1.07
16416	23.5	1.07	23.51	1.07

AES Bit-slice

Msglen (byte)	Enc (med c/b)
1056	22.24
2080	22.58
16416	21.87

AES T-table

Msglen (byte)	Enc (med c/b)
1056	41.31
2080	43.56
16416	43.54

[GH15] Gouvea, Lopez Hernandez. Implementing GCM on ARMv8. CTRSA 2015

Performance of OTR : AVR [ASK14]

- 8-bit Atmel AVR (ATmega 128)
- Assembly AES from open source (AVRAES), runs at 156 cpb for enc, 196 cpb for dec
- Mode written in assembly
- ~240 cpb for 256 input bytes, for both Enc/Dec
- ~2100 ROM bytes, ~180 RAM bytes including stack etc.

Future plan

- Software targets
  - ARMv7 and AVR : porting to v3
  - Cortex-M (e.g. use AES from Schwabe and Stoffelen [SS16])
  - Non-NI Intel/AMD CPUs
- Hardware targets (both FPGA/ASIC)
  - Optimized reference : one in ATHENa is immature
  - Hi-speed (Pipeline/multi-core)
  - SCA-resistant
- Make them public when ready

[SS16] Schwabe and Stoffelen. All the AES You Need on Cortex-M3 and M4. SAC 2016.

On Serial ADP

- Current Serial ADP has limited value
- We are trying to make it better
  - AD parallelizable
  - More hardware-friendly parameter
  - Simplified logic, reusing components etc.

Thank you !