Authenticated Ciphers

Andrey Bogdanov
Technical University of Denmark
June 5, 2014

Part 1: AES-Based Authenticated Encryption Modes and High-Performance Software

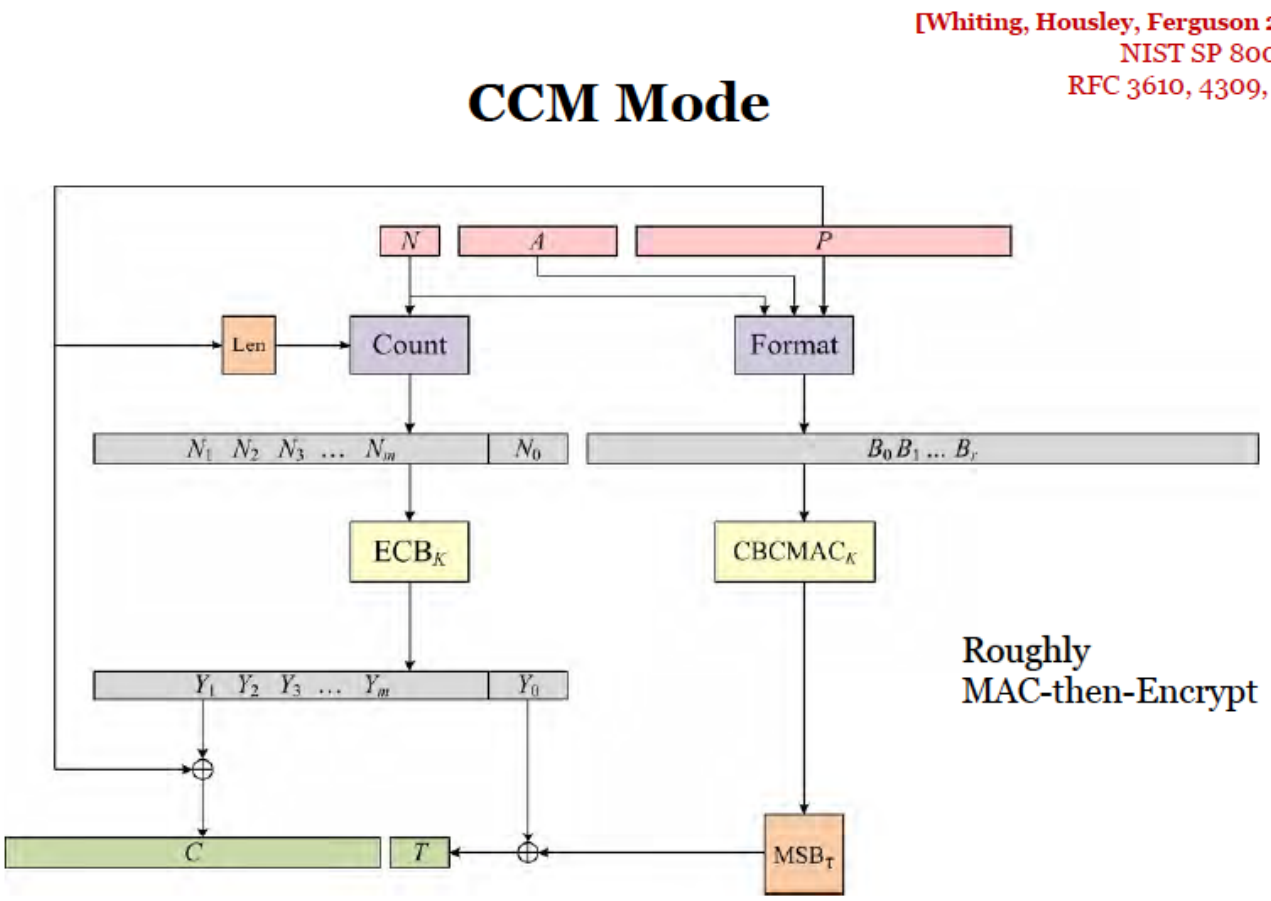
Based on joint work with Martin Lauridsen and Elmar Tischhauser

Scope

- Focus on modes of operation for block ciphers
- Underlying block cipher: AES-128
- Context: CAESAR competition

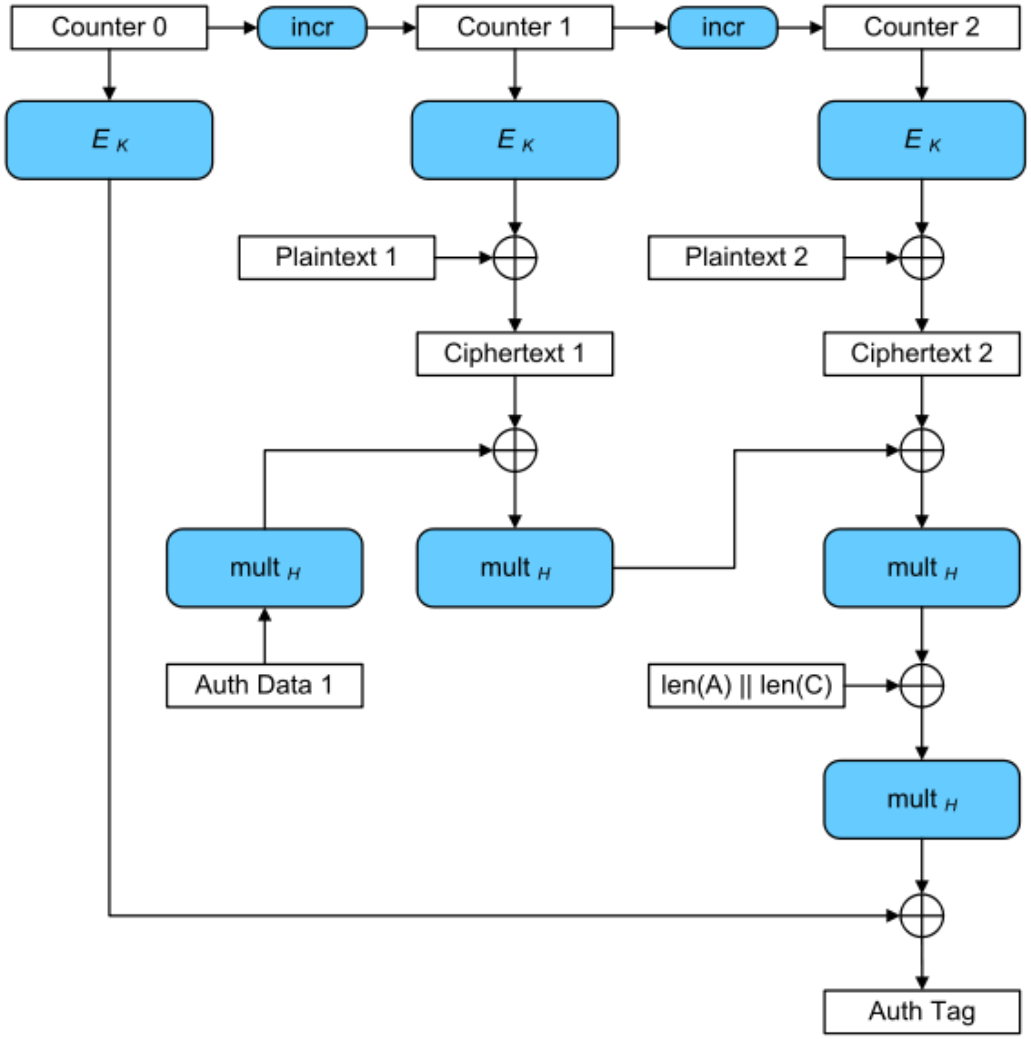
CCM: Counter with CBC-MAC

[Whiting, Housley, Ferguson 2002]
NIST SP 800-38C
RFC 3610, 4309, 5084



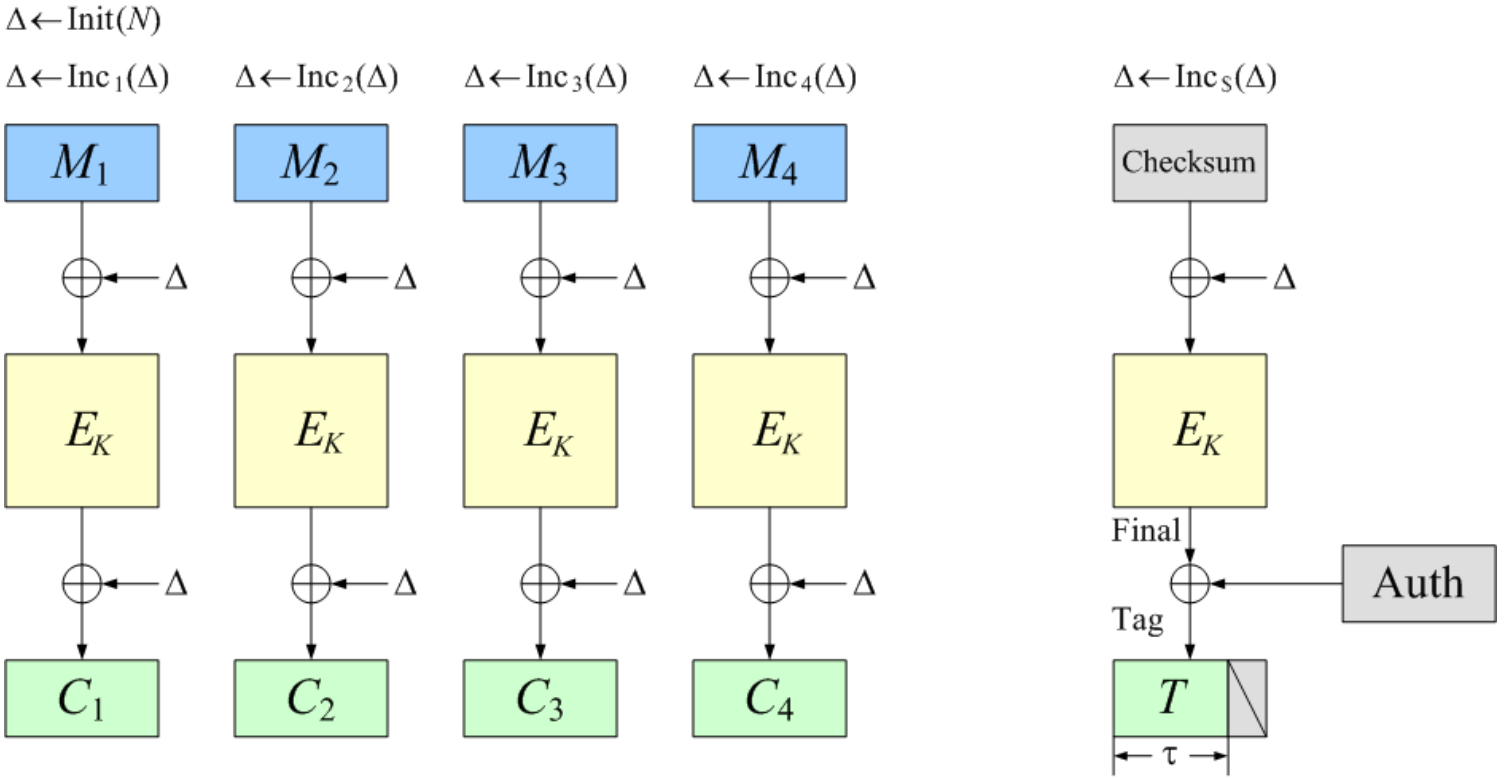
[MV04]

GCM: Galois/Counter Mode



[RBBK01]
[BR02]
[R02]
[R04]
[KR11]

OCB: Offset Codebook Mode



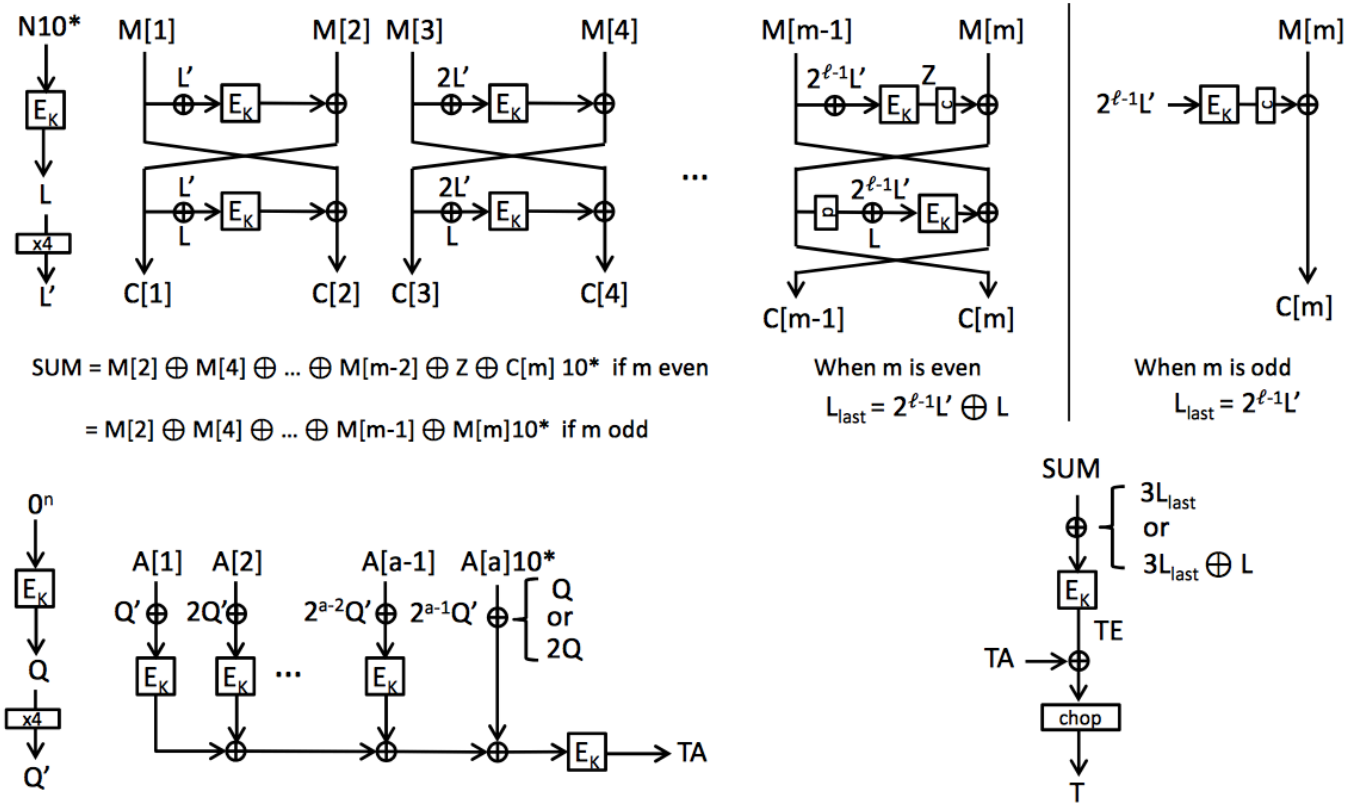
+

- 1 AES-128 call per block
- well parallelizable

-

- enc/dec different
- (patents pending)

OTR: Offset Two-Round



+

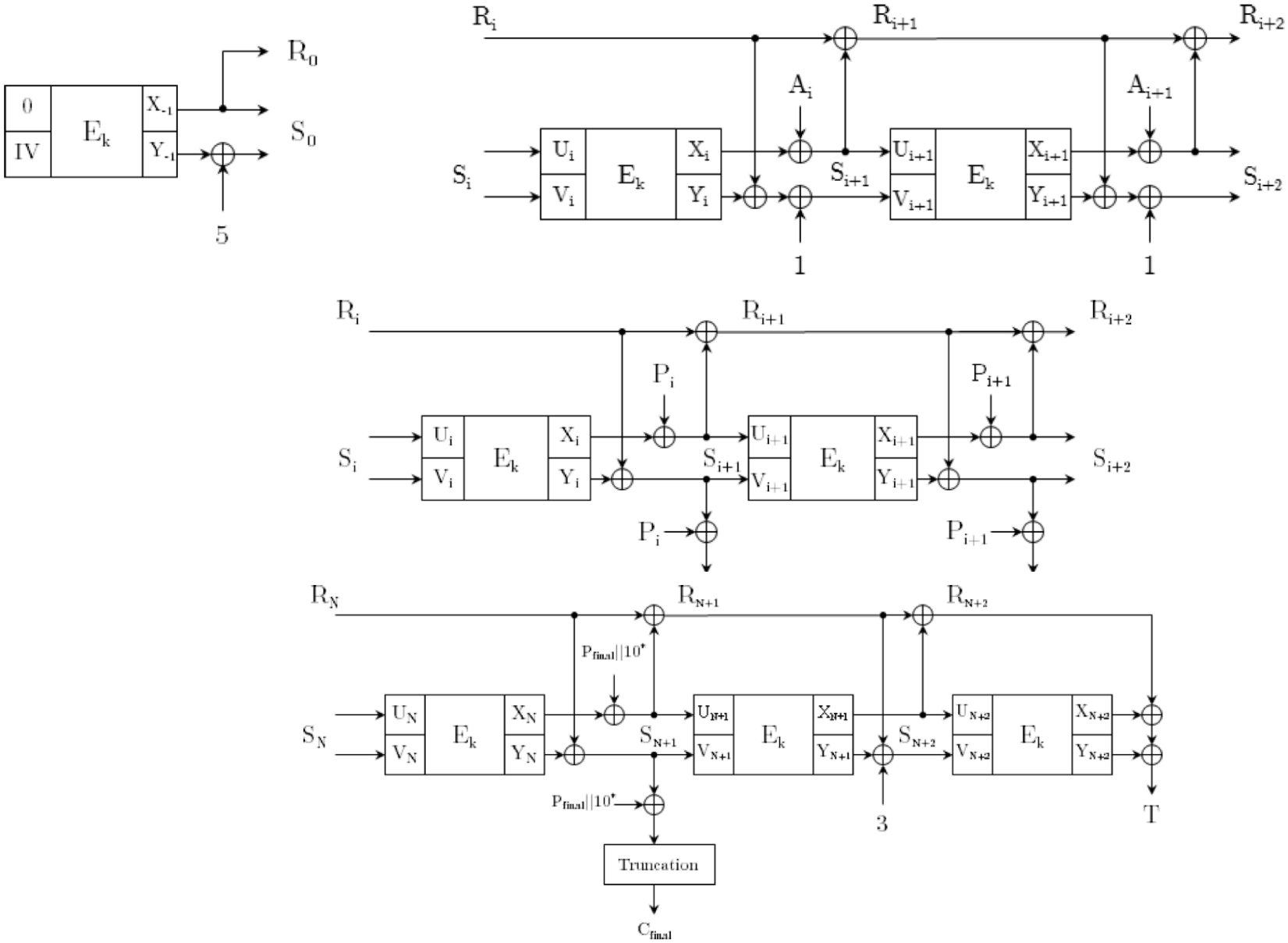
- 1 AES-128 call per block
- well parallelizable
- only BC enc is needed for both OTR enc & dec

-

- Feistel limits parallelizability

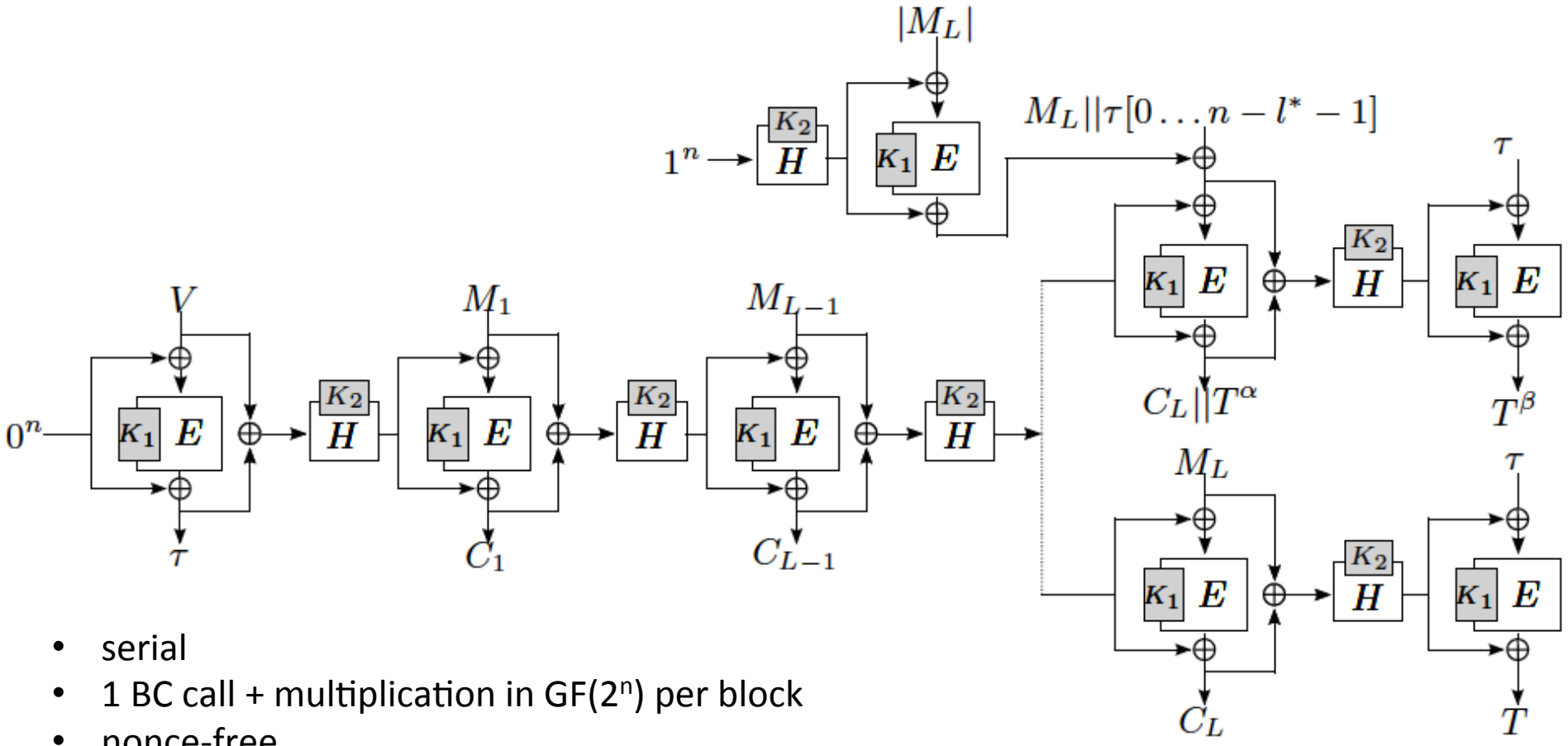
[WH14]

JAMBU


0
IV
E_k
X_{-1}
Y_{-1}
R_0
S_0
5
R_i
R_{i+1}
R_{i+2}
A_i
A_{i+1}
S_i
U_i
V_i
X_i
Y_i
S_{i+1}
U_{i+1}
V_{i+1}
X_{i+1}
Y_{i+1}
S_{i+2}
1
1
P_i
P_{i+1}
R_N
R_{N+1}
R_{N+2}
$P_{final}||10^*$
S_N
U_N
V_N
X_N
Y_N
S_{N+1}
U_{N+1}
V_{N+1}
X_{N+1}
Y_{N+1}
S_{N+2}
U_{N+2}
V_{N+2}
X_{N+2}
Y_{N+2}
3
T
Truncation
C_{final}

[FFLW12]

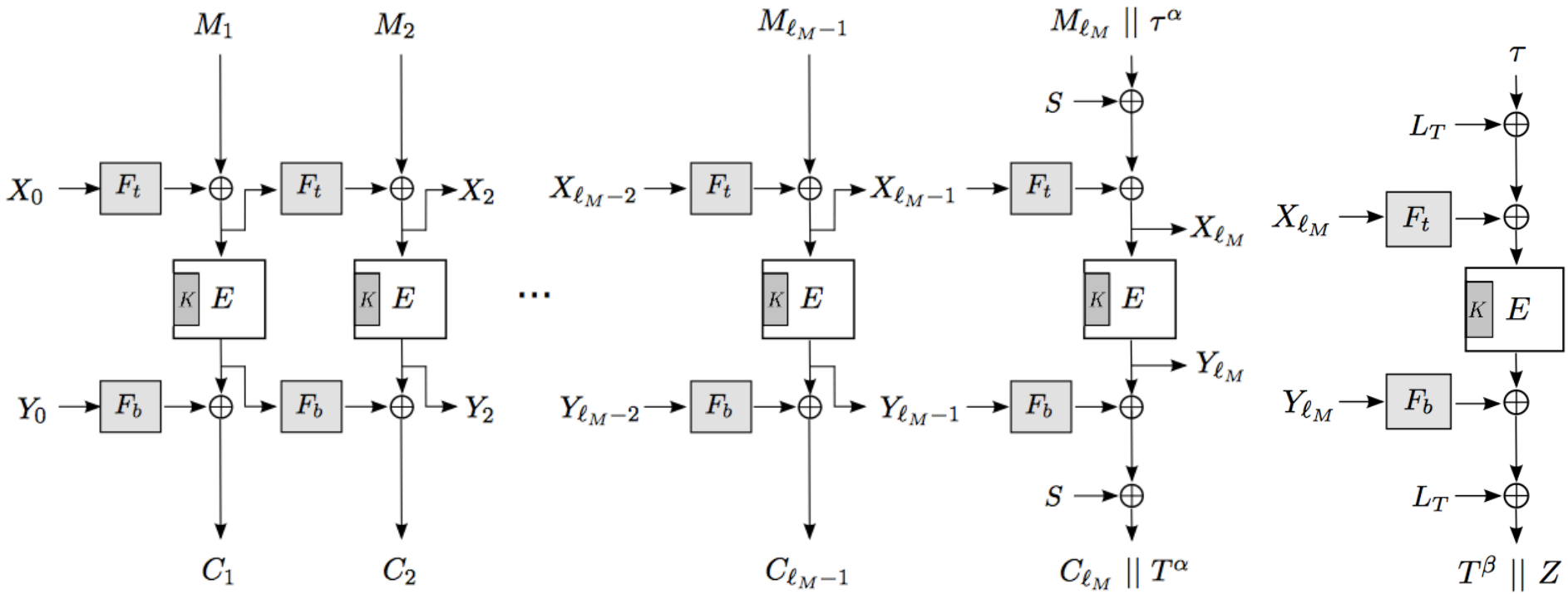
MCoE-G: nonce-misuse resistant & one-pass



- serial
- 1 BC call + multiplication in GF(2^n) per block
- nonce-free
- one-pass
- common-prefix preservation

[AFFFLLMW14]

POET: Nonce-free one-pass

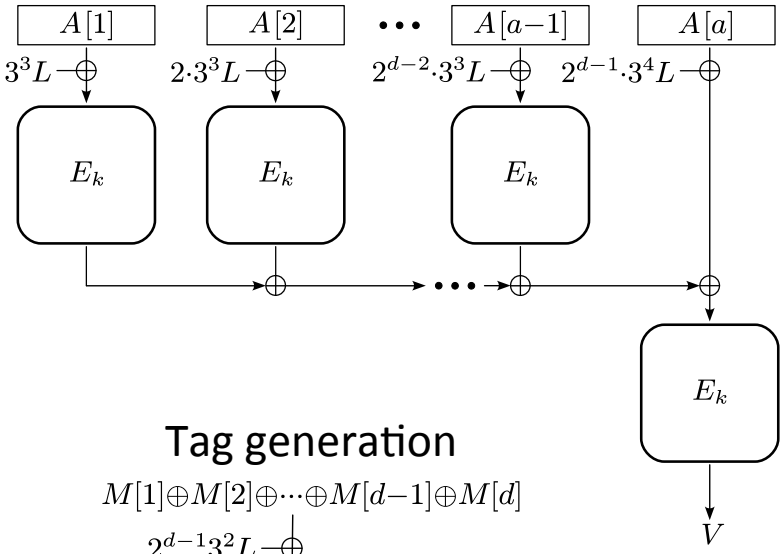


- 1 BC call + 2 multiplications in GF(2^n) per block or 3 BC calls per block
- nonce-misuse resistant
- one-pass
- common-prefix preservation
- “pipelinable” but rather serial

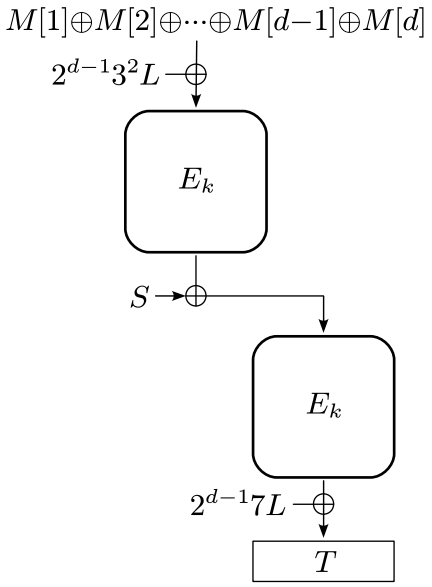
COPA:
Parallelizable single-pass nonce-free AE

[ABLMTY'13]

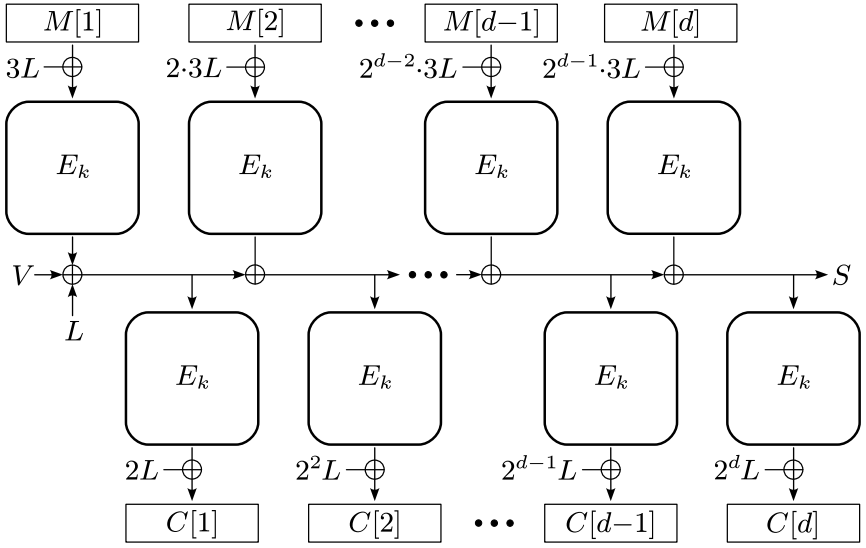
Associated data



Tag generation



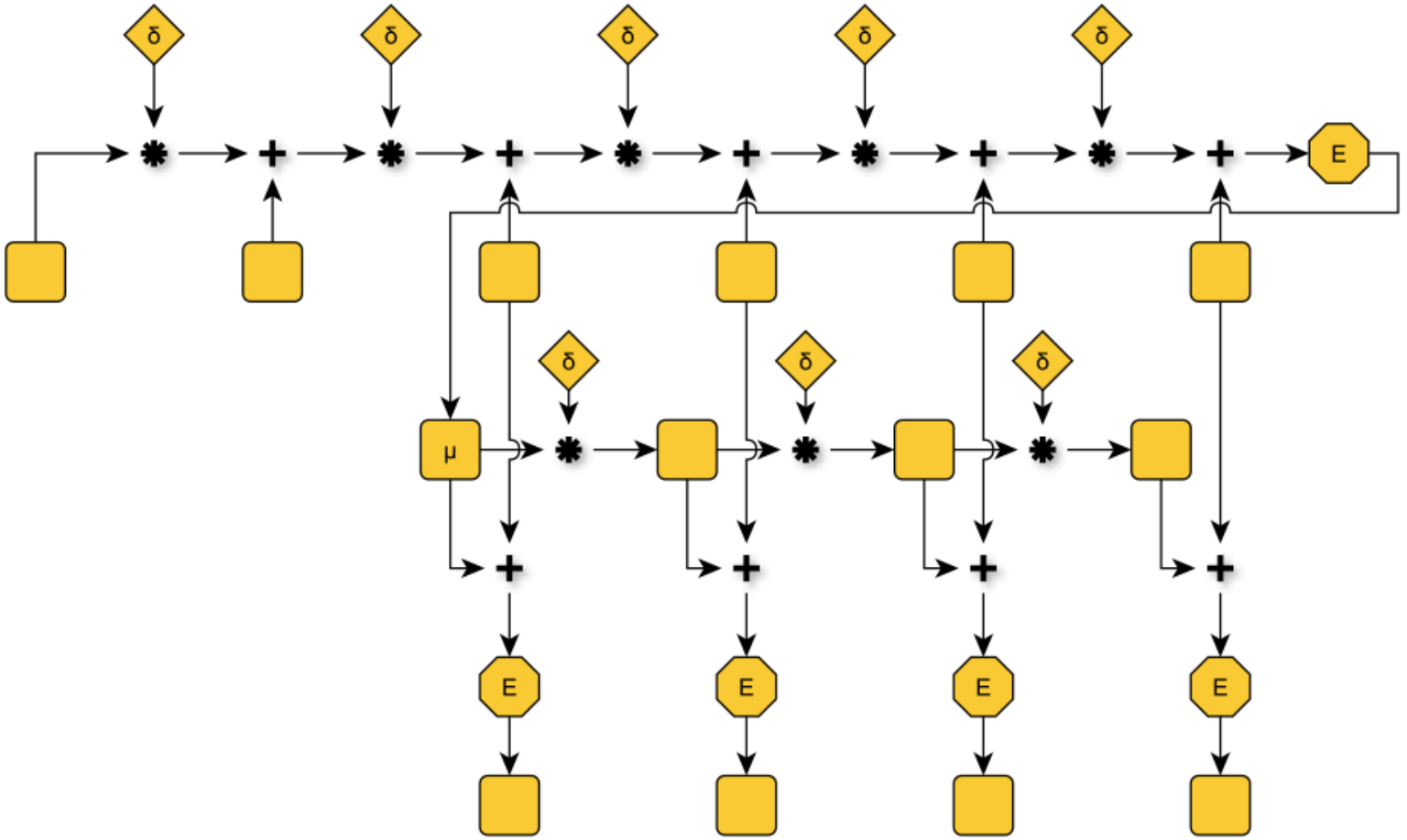
Message (COPE)



- well parallelizable
- two BC calls per block
- nonce-misuse resistant
- one-pass
- common-prefix preservation

Collaboration with Elena Andreeva, Atul Luykx, Bart Mennink, Elmar Tischhauser, Kan Yasuda

Julius-ECB



Intel's Haswell: AES and field multiplication

Instruction	L	T^{-1}	Instruction	L	T^{-1}
`aesenc`	7	1	`aesimc`	14	2
`aesdec`	7	1	`aeskeygenassist`	10	8
`aesenclast`	7	1	`pclmulqdq`	7	2
`aesdeclast`	7	1			

- Latency *L*
- Inverse throughput T^{-1}
- pclmulqdq largely improved (Haswell vs Sandy Bridge):
  - *L*=7 vs *L*=14
  - T^{-1}=2 vs T^{-1}=8

Single-message processing vs multiple-message processing

- In a typical networking environment one deal with many shorter packets simultaneously
- Bimodal distribution:
  - 44% of packets between 40 and 100 bytes long
  - 37% between 1400 and 1500 bytes long
- Most data is transmitted in packets of medium size between 1 and 2 KB
- Setting proposed at FSE'12

CBC encryption: Single- vs multiple-message

		multiple messages (# msgs.)						
	single msg.	2	3	4	5	6	7	8
AES-CBC	4.28	2.15	1.43	1.08	0.88	0.74	0.64	0.63
Relative speed-up	×1.00	×1.99	×2.99	×3.96	×4.86	×5.78	×6.69	×6.79

- CBC encryption is serial
- CBC decryption is parallel

Main performance comparison

(a) Nonce-based AE modes

Mode	# msgs.	Message length (bytes) 128	256	512	1024	2048
		single message				
CCM		5.35	5.19	5.14	5.11	5.10
GCM		2.09	1.61	1.34	1.20	1.14
OCB3		2.19	1.43	1.06	0.87	0.81
OTR		2.97	1.34	1.13	1.02	0.96
CLOC		4.50	4.46	4.44	4.46	4.44
COBRA		4.41	3.21	2.96	2.83	2.77
JAMBU		9.33	9.09	8.97	8.94	8.88
SILC		4.57	4.54	4.52	4.51	4.50
	# msgs.	multiple messages				
CCM	8	1.51	1.44	1.40	1.38	1.37
GCM	13	1.81	1.72	1.68	1.65	1.64
OCB3	7	1.59	1.16	**0.94**	**0.83**	**0.77**
OTR	8	**1.28**	**1.08**	0.98	0.94	0.92
CLOC	7	1.40	1.31	1.26	1.24	1.23
COBRA	8	2.04	1.88	1.80	1.76	1.75
JAMBU	14	2.14	1.98	1.89	1.85	1.82
SILC	7	1.43	1.33	1.28	1.25	1.24

(b) Nonce-misuse resistant AE modes

Mode	# msgs.	Message length (bytes) 128	256	512	1024	2048
		single message				
McOE-G		7.77	7.36	7.17	7.07	7.02
COPA		3.37	2.64	2.27	2.08	1.88
POET		5.30	4.93	4.75	4.68	4.62
Julius		4.18	4.69	3.24	3.08	3.03
	# msgs.	multiple messages				
McOE-G	7	1.91	1.76	1.68	1.64	1.62
COPA	15	**1.62**	**1.53**	**1.48**	**1.46**	**1.45**
POET	8	3.24	2.98	2.86	2.79	2.75
Julius	7	2.53	2.27	2.16	2.09	2.06

Speed-up: Ratio single to multiple message performance

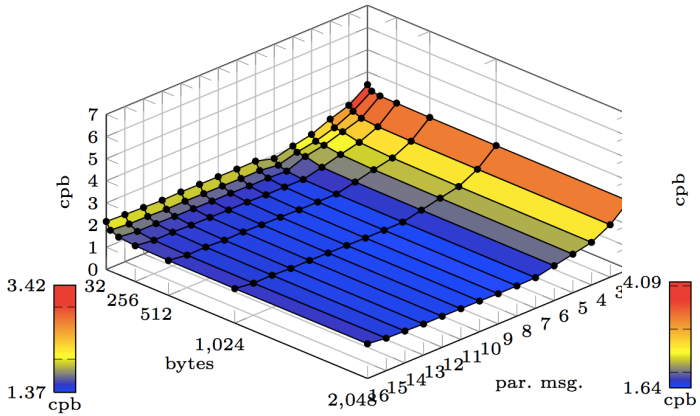
(a) Nonce-based AE modes

Mode	Message length (bytes) 128	256	512	1024	2048
CCM	×3.54	×3.60	×3.67	×3.70	×3.72
GCM	×1.15	×0.94	×0.80	×0.73	×0.70
OCB3	×1.38	×1.23	×1.13	×1.05	×1.05
OTR	×2.32	×1.24	×1.15	×1.09	×1.04
CLOC	×3.21	×3.40	×3.52	×3.60	×3.61
COBRA	×2.16	×1.71	×1.64	×1.61	×1.58
JAMBU	**×4.36**	**×4.59**	**×4.75**	**×4.83**	**×4.88**
SILC	×3.20	×3.41	×3.53	×3.61	×3.63

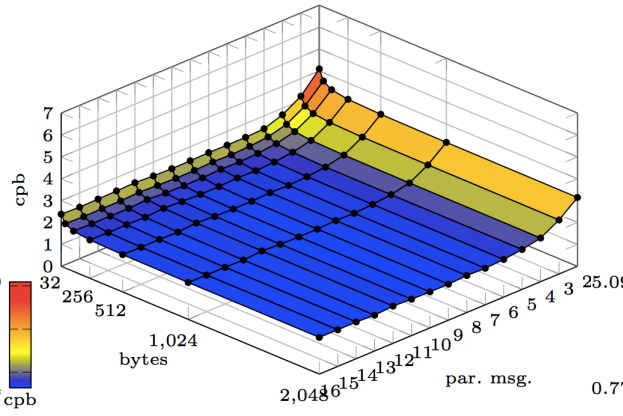
(b) Nonce-misuse resistant AE modes

Mode	Message length (bytes) 128	256	512	1024	2048
McOE-G	**×4.07**	**×4.18**	**×4.27**	**×4.31**	**×4.33**
COPA	×2.08	×1.73	×1.53	×1.42	×1.30
POET	×1.64	×1.65	×1.66	×1.68	×1.45
Julius	×1.65	×2.07	×1.50	×1.47	×1.47

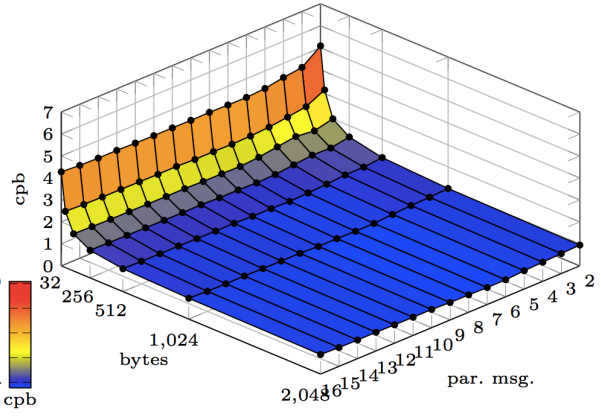
Performance of nonce-based modes in multiple-message setting



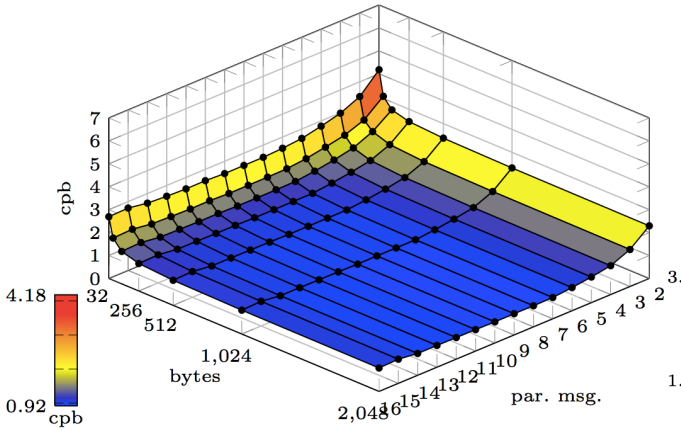
(a) CCM



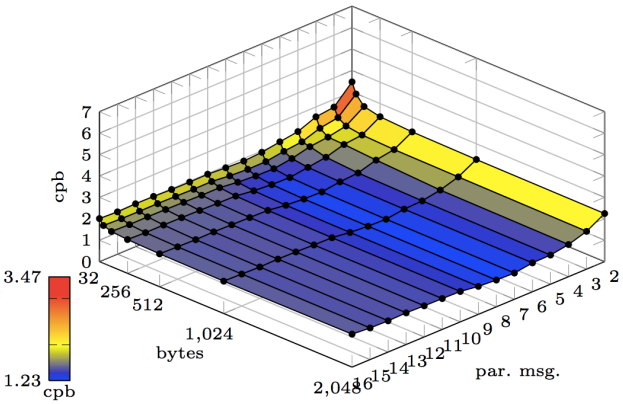
(b) GCM



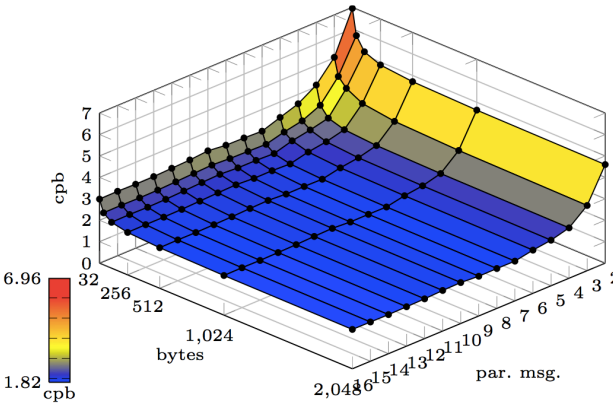
(c) OCB3



(d) OTR

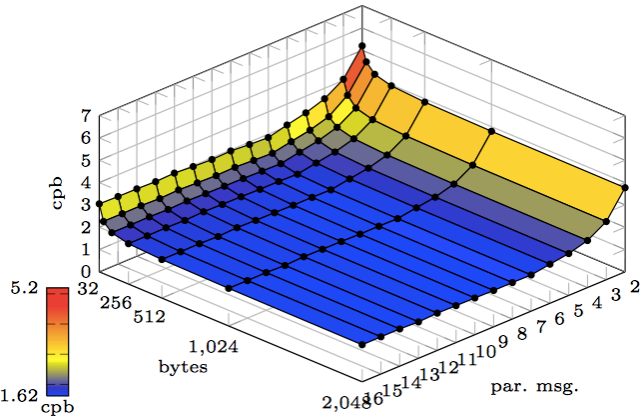


(f) CLOC

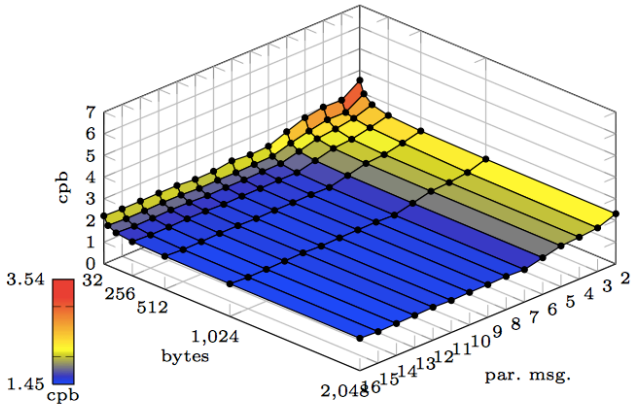


(h) JAMBU

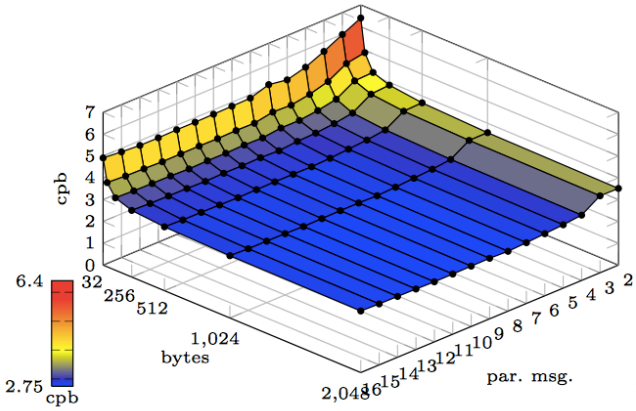
Performance of nonce-free modes in multiple-message setting



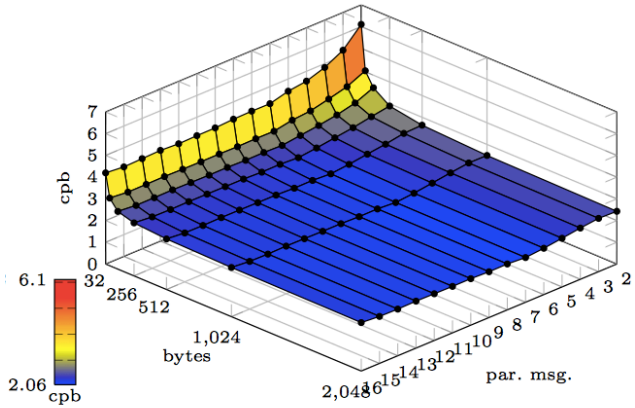
(a) McOE-G



(b) COPA



(c) POET



(d) Julius

Wrap-up

Nonce-based:

- OCB and OTR are especially good (around 0.7-0.9 cpb at several K)
- CCM, CLOC/SILC, and JAMBU get significant speedups with multiple messages

Nonce-misuse resistant:

- COPA is especially good (down to 1.4 cpb at several K)
- McOE-G and POET profit a lot from multiple-message setting

Part 2: Polynomial-based AE: Weak keys, Forgery, Twisted Polynomials and Sliding GCM

Joint work with Mohamed Abdelraheem, Peter Beelen and Elmar Tischhauser

Scope

- Classes of weak keys
- Explicit forgery polynomials
- Full coverage of the key space

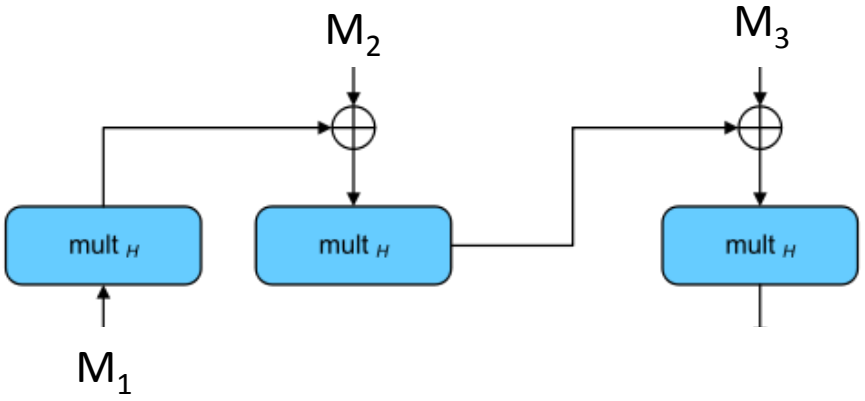
Weak keys

- A class of weak keys of size M if it's possible (Crypto'08):
  - To test the membership in less than M effort and queries

Polynomial Authentication

Definition 1 (Polynomial-based Authentication Schemes). *A polynomial hash-based authentication scheme processes an input consisting of a key H and plaintext/ciphertext $M = (M_1||M_2||\cdots||M_l)$, where each $M_i \in \mathbb{F}_2^{128}$, by evaluating the polynomial*

$$h_H(M) := \sum_{i=1}^{l} M_i H^i \in \mathbb{F}_2^{128}.$$



Weak keys and forgeries for polynomial authentication

- Saarinen's cycling forgery (FSE'12):
  - If *H* belongs to a (cyclic) subgroup of order t, then

$$H^{t+1} = H$$

    and the attacked can swap M_i and M_{i+t}
- Procter and Cid's forgery framework (FSE'13):
  - For any polynomial $q(X) = \sum_{i=1}^{r} q_i X^i$ with $q(H) = 0$

$$h_H(M) = \sum_{i=1}^{r} M_i H^i = \sum_{i=1}^{l} M_i H^i + \sum_{i=1}^{r} q_i H^i = \sum_{i=1}^{r} (M_i + q_i) H^i = h_H(M + Q)$$

  - Forgery polynomial *q*

Forgery polynomials

- How to construct a forgery polynomial?
  - X^t-*1* due to Saarinen, only for $t \mid (2^{128} - 1)$:
    - The only known explicit construction
    - Limits the sizes and coverage of weak key classes
  - "Manual" construction with $(X - \alpha_1) \cdots (X - \alpha_\ell)$
    - Usually not sparse -> heavy queries
    - High effort on the adversarial side

Forgery polynomials

- Explicit constructions of forgery polynomials?
- Complete disjoint coverage of the entire key space by forgery polynomials?

Twisted polynomials

- Theory based upon the Ore rings

Definition 2. *Let $\mathbb{F}_q$ be a field of characteristic p. The* twisted polynomial *or* Ore *ring $\mathbb{F}_q\{\tau\}$ is defined as the set of polynomials in the indeterminate τ having coefficients in $\mathbb{F}_q$ with the usual addition, but with multiplication defined by the relation $\tau\alpha = \alpha^p\tau$ for all $\alpha \in \mathbb{F}_q$.*

- Provides a nice way to study linear maps from a binary field to itself as a vector space over F_2
- Given a vector subspace of V of $\mathbb{F}_{2^{128}}$, there is a fast procedure to compute a polynomial whose roots are exactly all elements of V

Multiplication in the Ore ring

$$(a+b\tau)(c+d\tau) = a(c+d\tau) + b\tau(c+d\tau) = ac + ad\tau + bc^p\tau + bd^p\tau^2$$

Construction of twisted polynomials

Algorithm 5.1 Construction of twisted polynomials

Input: basis $\mathcal{B} = \{\beta_1, \ldots, \beta_d\}$ of $V \subset \mathbb{F}_2^n$
Output: polynomials $p_{V^{(i)}}(X)$ having $\mathrm{span}\{\beta_1, \ldots, \beta_i\}$ as set of roots

1: Set $a_1 \leftarrow 1$
2: Set $a_i \leftarrow 0$ for $2 \leq i \leq d+1$
3: **for** $i = 1$ to d **do**
4: $\quad v \leftarrow \sum_{k=1}^{d} a_k \beta_i^{2^k}$
5: $\quad c_1 \leftarrow v \cdot a_1$
6: $\quad$ **for** $j = 2$ to $d+1$ **do**
7: $\quad\quad c_j \leftarrow a_{j-1}^2 + v \cdot a_j$
8: $\quad$ **end for**
9: $\quad p_{V^{(i)}} \leftarrow \sum_{k=1}^{d+1} c_k X^{2^{k-1}}$
10: **end for**
11: **return** polynomials $p_{V^{(1)}}(X), \ldots, p_{V^{(d)}}(X)$

A twisted polynomial of degree 2^{31}

i	e_i
1	576613647098987897394216259339443067 7
2	88640585123887860771282360281650849369
3	228467699759147933517306066079059941262
4	60870920642211311860125058878376239967
5	69981393859668264373786090851403919597
6	25545984420946355543584553897450020639 7
7	26357650066876523783054124192974030658 6
8	37167015149451472008716003077656492621
9	58043277378748107723324135119415484405
10	32176745583540153056725736641961423402 3
11	45033888451450737621429712394846444657
12	25842598508630980312235783230842151056 4
13	10583198952623274771783766826982534077 9
14	26746436017707187638674502455719932075 6
15	28064437275465890987288066203470862928 4
16	10500032685625069761543140328935770860 9
17	45825818359460611542283225368908192857
18	82845961308169259876601267127459416989
19	44217989936194208472522353821220861115
20	69062943960552309089842983129403174217
21	26846201940483608935933493977622068151 1
22	30001648942113240212113555293749765514
23	66973785438248799773654620388105644 9
24	12795885646825695604418987200045120323 5
25	27716223867823996583521968314331884840 0
26	13466249895416637311254280711306634255 4
27	21927841517524076258824088326661943647 0
28	21619747601031123010525953473090915868 2
29	28178300576761366713038004453626425182 9
30	18148313163977765640319841215141540492 9
31	38384836687611426333051602240884584792
32	0

A twisted polynomial of degree 2^{61}

i	e_i	i	e_i
1	205269631350267731197715294199912473 27	32	109604555581389038896555752982244394616
2	264546851691026540251722618719245777504	33	119482829110451460647031381779266776526
3	792797323058334749028936479677215949 21	34	165259785861038013124994816644344468967
4	325712555585908542291537560181869632351	35	155444340258770748055544634836807134293
5	281140838798434203589324885475612499 13	36	869821844387300458212740258310619614 30
6	271147943451442547572675283203493325775	37	104870645496065737272877350967826010844
7	335255520823733252020392488407731432338	38	562812815790023183370379193561271053 69
8	671801688290763317086056756932989527 3	39	100068518982837928471870587740499831 41
9	255889065981883867903019621991013125435	40	936879200755548123588902448980883454 49
10	494576877216014637126401892177554742 30	41	698326729003034322484017536582625335 06
11	311579005442569730277030755228683616807	42	246360754285298743574294101515912517720
12	227984510405461964893924913268809066393	43	895678936019042717674614594480764049 68
13	324660953045118328235538900161997992161	44	337681726780870315172220356080972321854
14	101370059745789285127519397790494215441	45	210317547004302372764274348440690947691
15	335840777837142047555650075244373419708	46	158574321133010145534802861165087620178
16	314588499802672014617473470717109075 23	47	291559826228649927512447763293001897434
17	339477818976914242962960654286547702007	48	156351243312442316097609527177914577 46
18	267056244491330957618685443721979120206	49	196562458398036090488379086660199368109
19	115274327651619347046091793992432007152	50	308779188958300135859037769338975723488
20	309606471838332610868454369483105904888	51	311961723579011854596575128443762996895
21	314728319634705433804935434967329297 63	52	153505386496968503239745640447605550270
22	191332595597193424626322329032056378009	53	266880473479137548264080346617303001989
23	189553913431309255614514163550670075672	54	325361660912502344542873376867973189476
24	224617322052671248319257827067474740867	55	756486261013747940931759163320432850 57
25	630412303067880329731111455333070515 62	56	122904035765598179315104311504496672627
26	221576606272152354153350739375040337239	57	240654849065616783877381099532333510366
27	291799903540006289220245045188573741192	58	717747464603164639815429745582806718 65
28	290489624437950764499707232619770186293	59	318833970371431372762935716012099244730
29	263754726506046639985479240660603777000	60	176351990917361872511208705771673004140
30	451608074361673079906891507920526707 07	61	227372417807158122619428517134408021585
31	336308819059966309252377016229504259 50	62	0

Complete disjoint coverage with twisted polynomials

- Cosets *a+V* of *V*: $a + V := \{a + v \mid v \in V\}$

Proposition 2. *Let $q = r^e$ and let V be a linear subspace of $\mathbb{F}_q$ of over the field $\mathbb{F}_r$ of dimension d. Moreover denote by $p_V(x)$ be the linearized polynomial associated to V and define $W := \{p_V(x) \mid x \in \mathbb{F}_q\}$.*

Then for any $a \in \mathbb{F}_q$, the polynomial $p_V(x) - p_V(a)$ has as sets of roots exactly $a + V$. Moreover, the sets of roots of the polynomials $p_V(x) - b$ with $b \in W$ partition $\mathbb{F}_q$.

- An efficient way to partition the key space into classes of (weak) keys!

Key recovery with twisted polynomials

Algorithm 5.2 Key recovery using twisted polynomials

Input: message M, polynomial $p_V(X)$ s.t. $h_H(M) = h_H(M + P_V)$,basis $\mathcal{B} = \{\beta_1, \ldots, \beta_d\}$ of d-dimensional linear subspace $V \subset \mathbb{F}_2^n$.

Output: authentication key H.

$b_i \leftarrow 0, \quad 1 \leq i \leq d$
Call Alg. 5.1 on V, obtain $p_{V^{(1)}}, \ldots, p_{V^{(d)}}$
for $i = d$ downto 1 **do**
    Denote $U^{(i)} = \text{span}\{\beta_1, \ldots, \beta_{i-1}\}$, so that $p_{U^{(i)}} = p_{V^{(i-1)}}$
    $\alpha \leftarrow p_{U^{(i)}}(\sum_{j=i}^{d} b_j \beta_j)$
    if $h_H(M) = h_H(M + P_{U^{(i)}} + \alpha)$ **then**
        $b_i \leftarrow 0$
    else
        $b_i \leftarrow 1$
    end if
end for
return key $H = \sum_{i=1}^{d} b_i \beta_i$

Applications

- In a weak-key setting, universal forgeries for:
  - GCM
  - POET-G (withdrawn upon our weak-key analysis)
  - Julius
- A note on GCM:
  - One can produce forgeries even in the nonce-respecting setting by using sliding techniques
  - This requires a nonce length other than 96 bits

Wrap-up

- Algebraic structure of polynomial authentication can lead to attacks in a weak-key setting
- Explicit forgery polynomials can be constructed efficiently from Ore rings (linearized polynomials)
- Disjoint coverage of the complete key space by sparse linearized polynomials
- New results for GCM, POET and Julius