Authenticated Encryption and Recent Trends in Cryptanalysis

Andrey Bogdanov
DTU Compute, Denmark

FRISC Winter School on Information Security
Finse, Norway, 23 April 2013

Outline

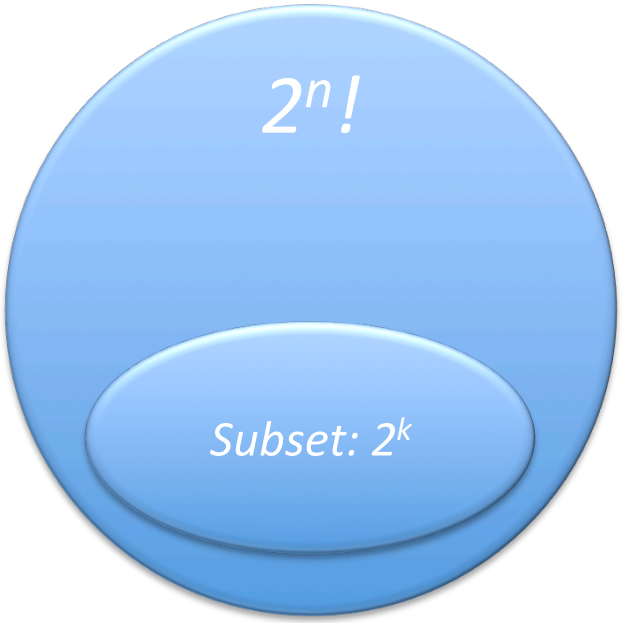
1. Symmetric-key cryptology and block ciphers
2. Lightweight cryptography
3. Authenticated encryption
4. Meet-in-the-middle attacks and biclique cryptanalysis

PART 1:
Symmetric-Key Cryptology and Block Ciphers

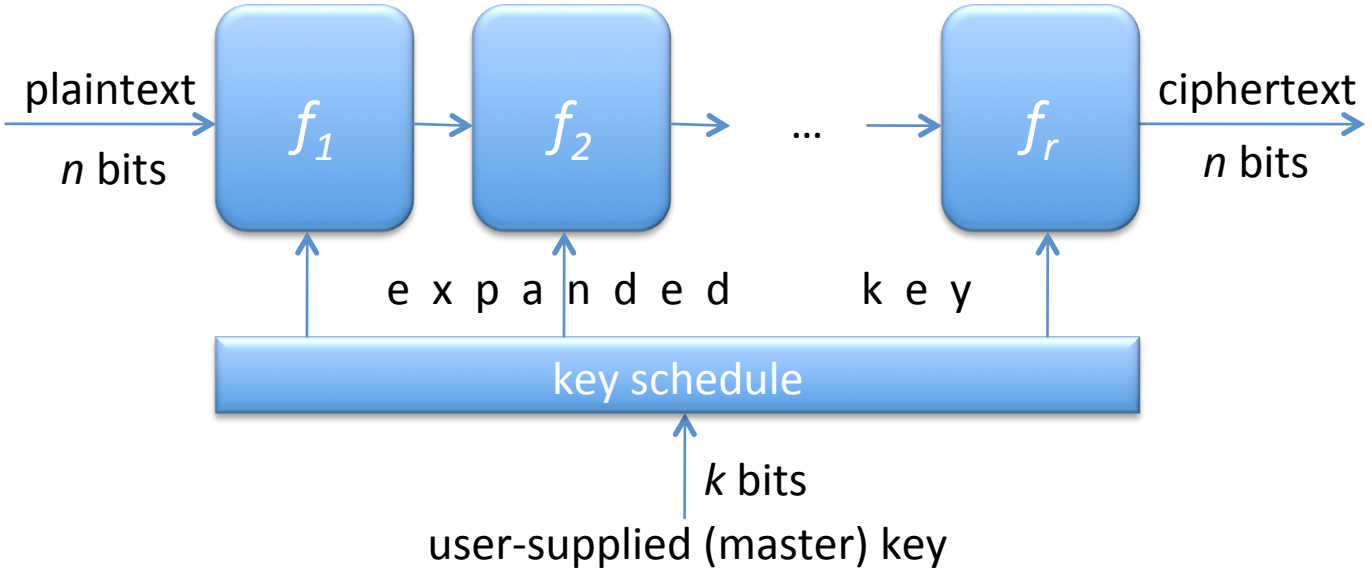
Symmetric-key cryptology: Block ciphers

Block cipher

A block cipher with n-bit block and k-bit key is a subset of 2^k permutations among all $2^n!$ permutations on n bits.



Symmetric-key cryptology: Iterative block ciphers



An iterative block cipher consists of r consecutive applications of simpler key-dependent transforms $f = f_r \circ f_{r-1} \circ \ldots f_2 \circ f_1$

Iterative block ciphers

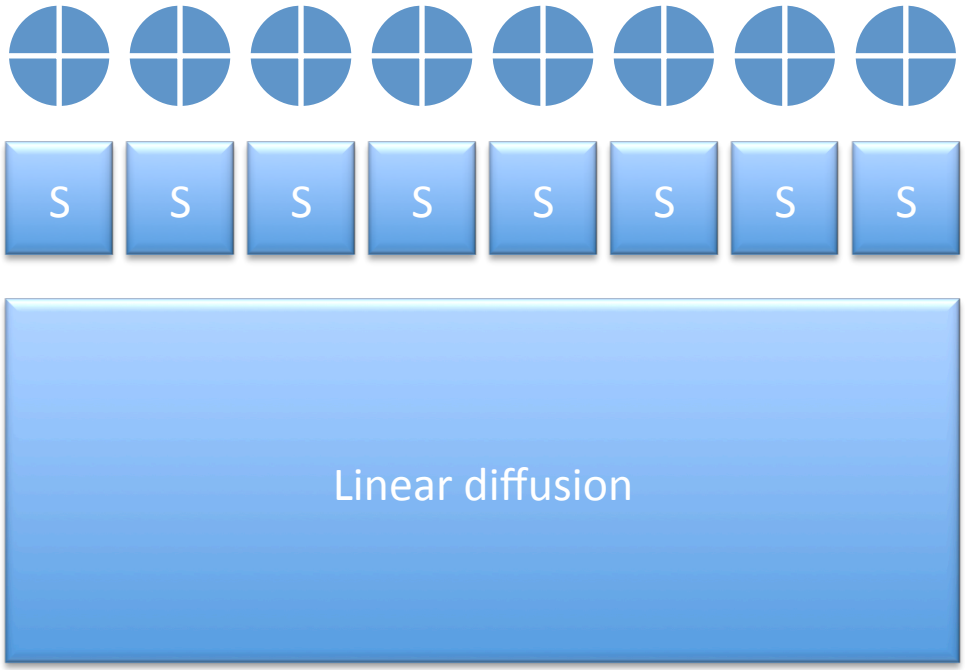
Luke O'Connor (IBM):

"Most ciphers are secure after sufficiently many rounds"

James L. Massey (ETH Zürich):

"Most ciphers are too slow after sufficiently many rounds"

Popular round structure: Substitution-Permutation (SP) network



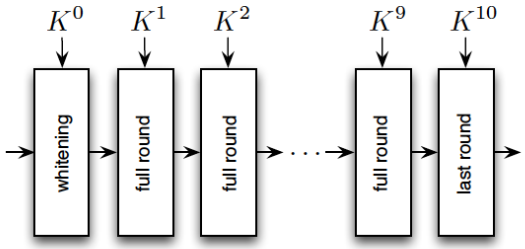
addition with subkey

local nonlinear functions

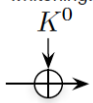
linear operation:
bit permutation,
matrix-vector mult.

Used in many ciphers (DES, AES, Serpent, Present, Camellia, Clefia,…) and hash functions (Whirlwind, Groestl, Spongent, Photon, …)

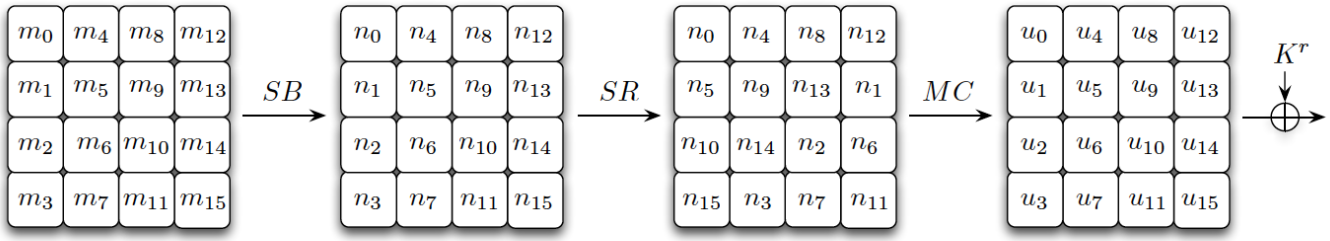
AES-128: data transform (2D)



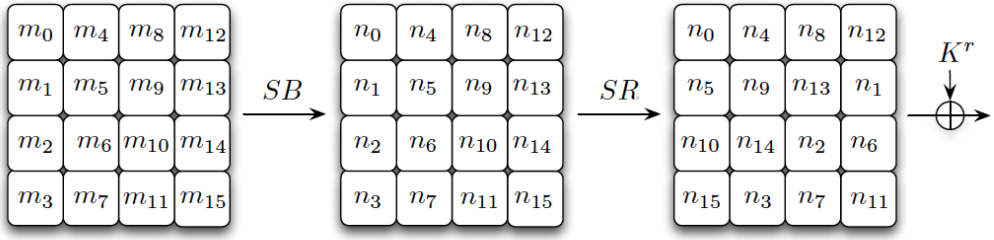
whitening:



full round:



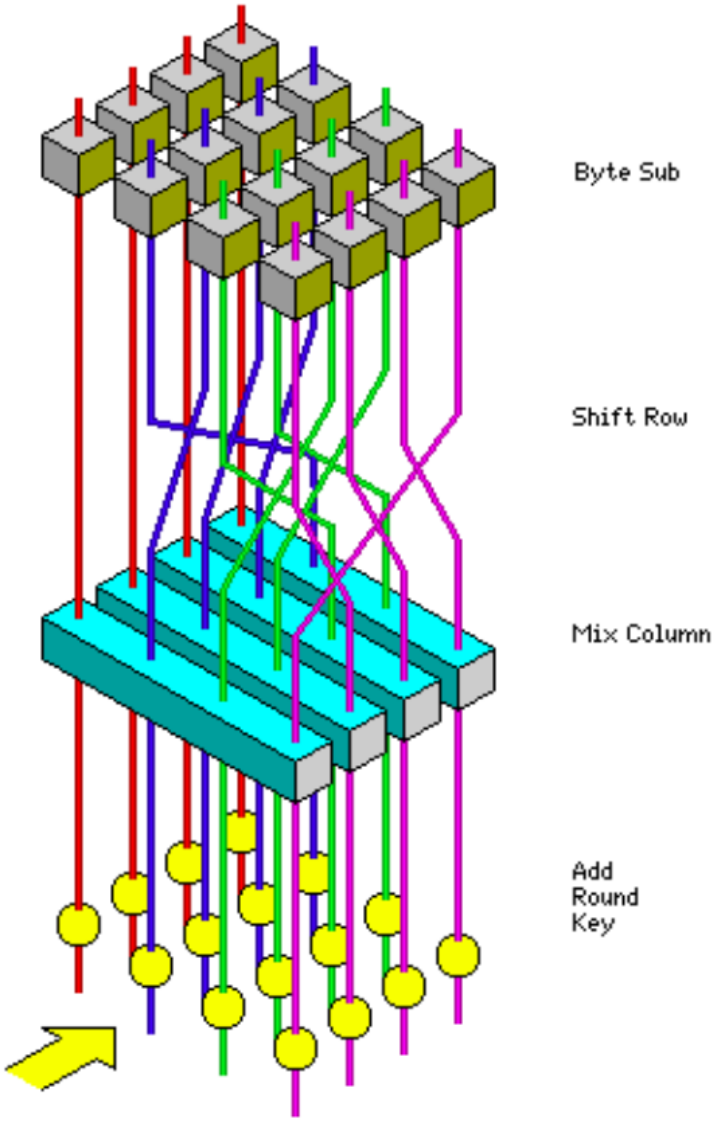
last round:



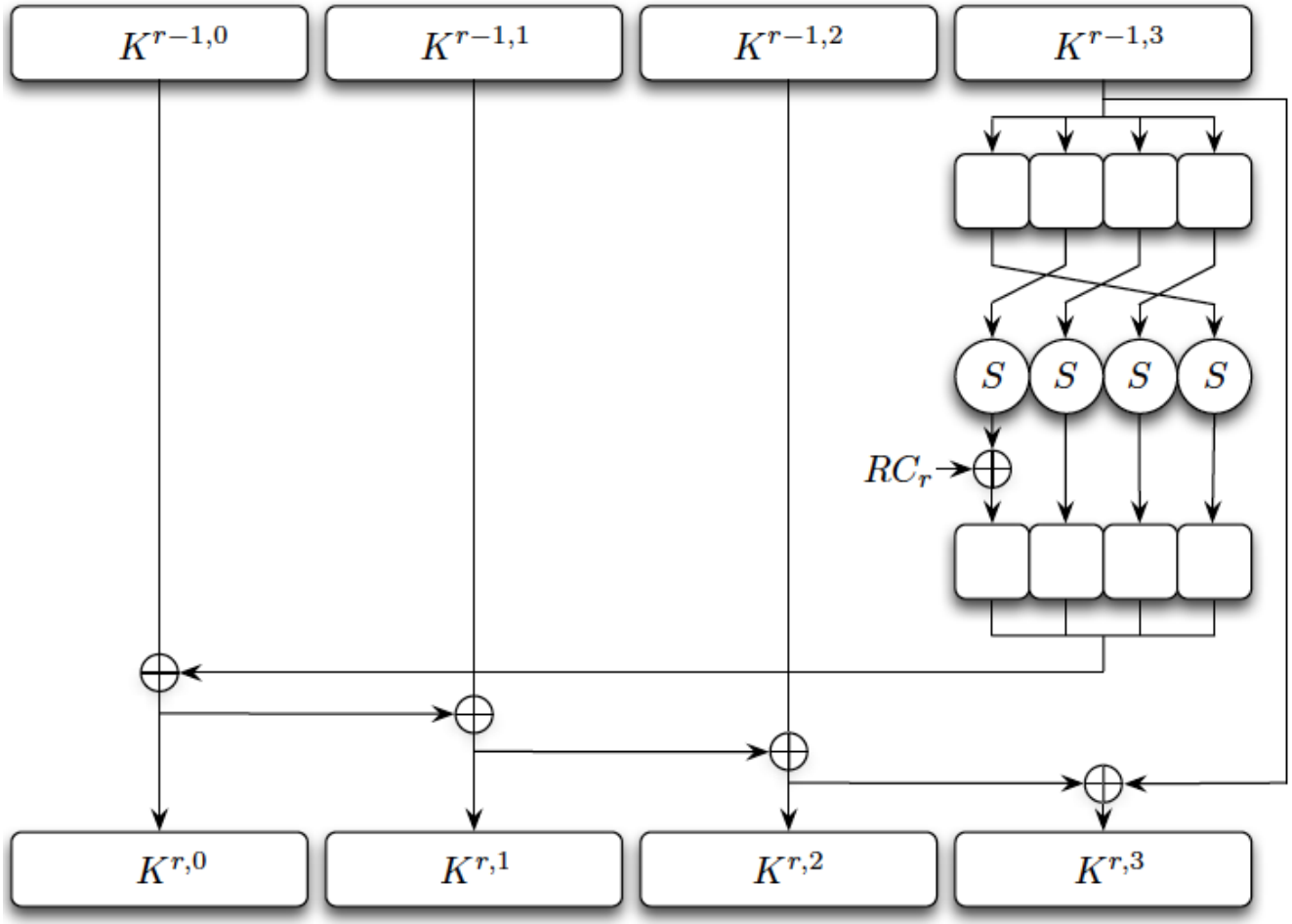
$$SB: \ n_i = S(m_i)$$

$$MC: \ (a_{4j}, a_{4j+1}, a_{4j+2}, a_{4j+3})^T = M \cdot (b_{4j}, b_{4j+1}, b_{4j+2}, b_{4j+3})^T$$

AES: data transform (3D)



AES-128: key schedule

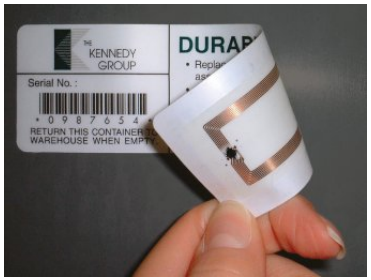


PART 2:
Lightweight Cryptography

Ubiquitous computing

Implants

Sensor networks



Logistics

Transportation

Access control

IDs

What about AES?

AES

- Suitable for most applications
  - especially in software
- Might be too expensive for tiny devices

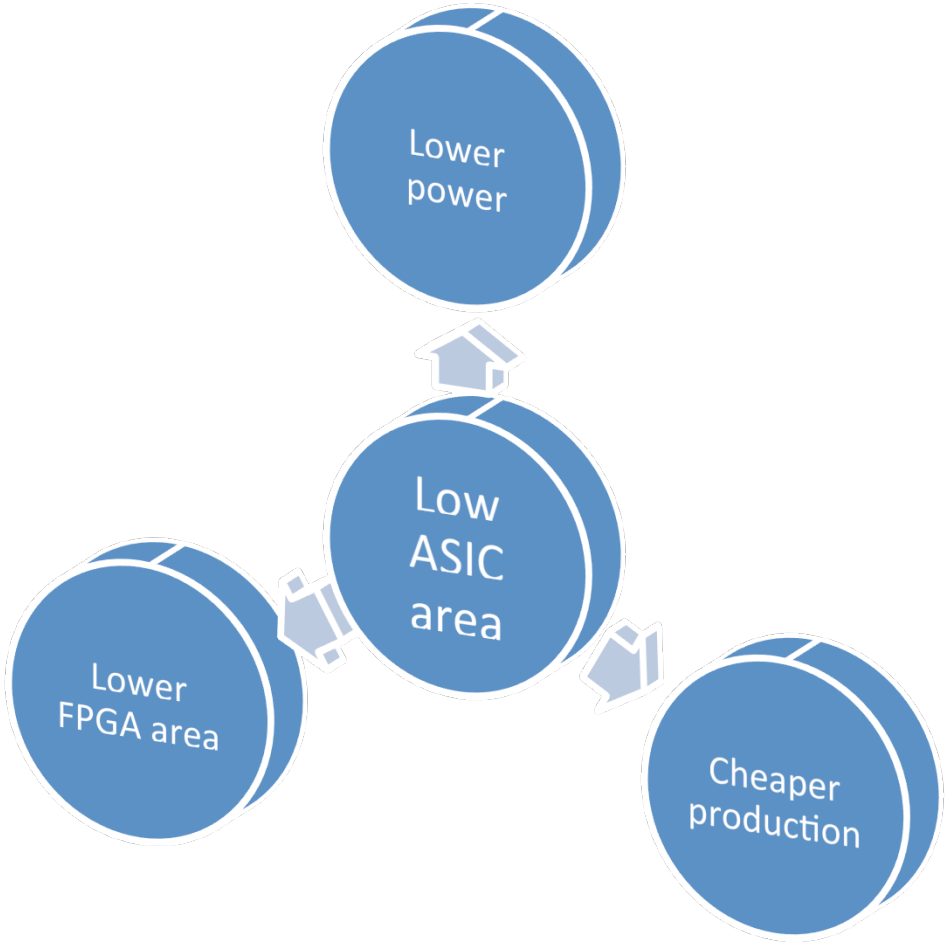
Lightweight = Low Cost

Lightweight = low development costs?
Lightweight = small code size?

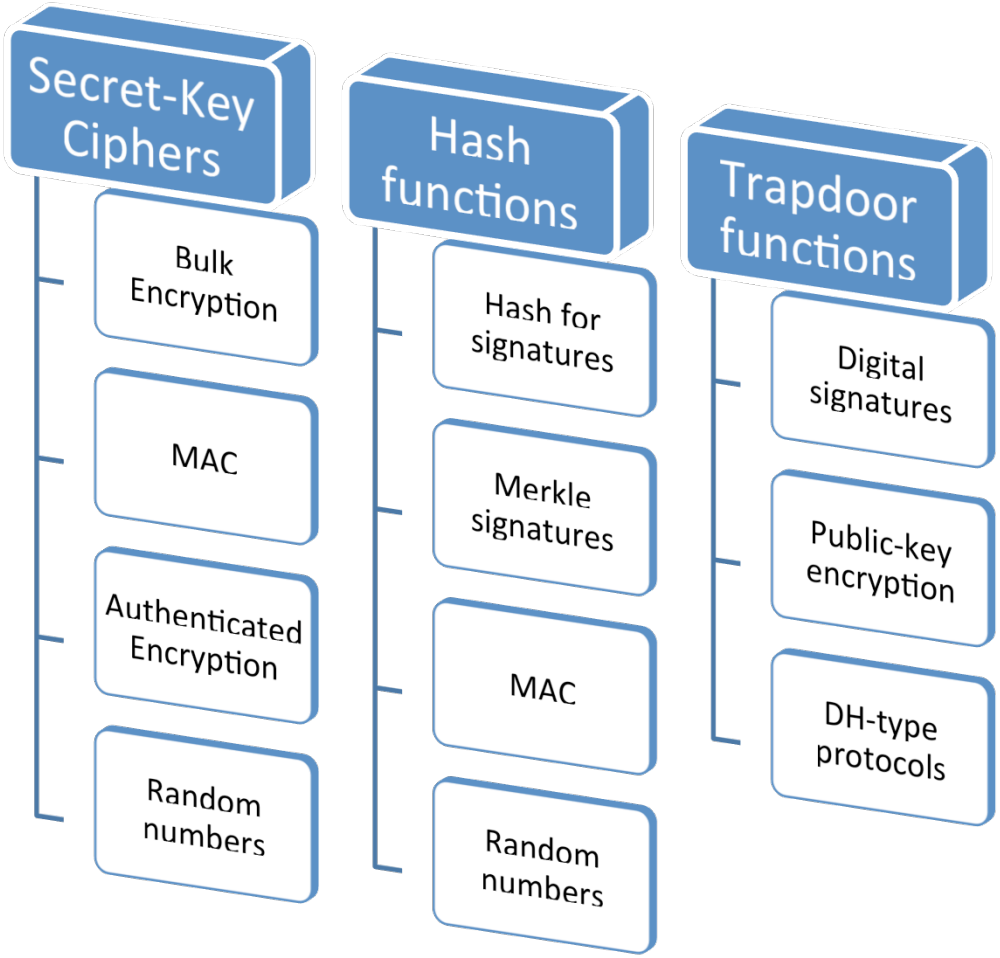
Lightweight = low ASIC area?
Lightweight = low FPGA area?

Lightweight = low power consumption?
Lightweight = low energy consumption?

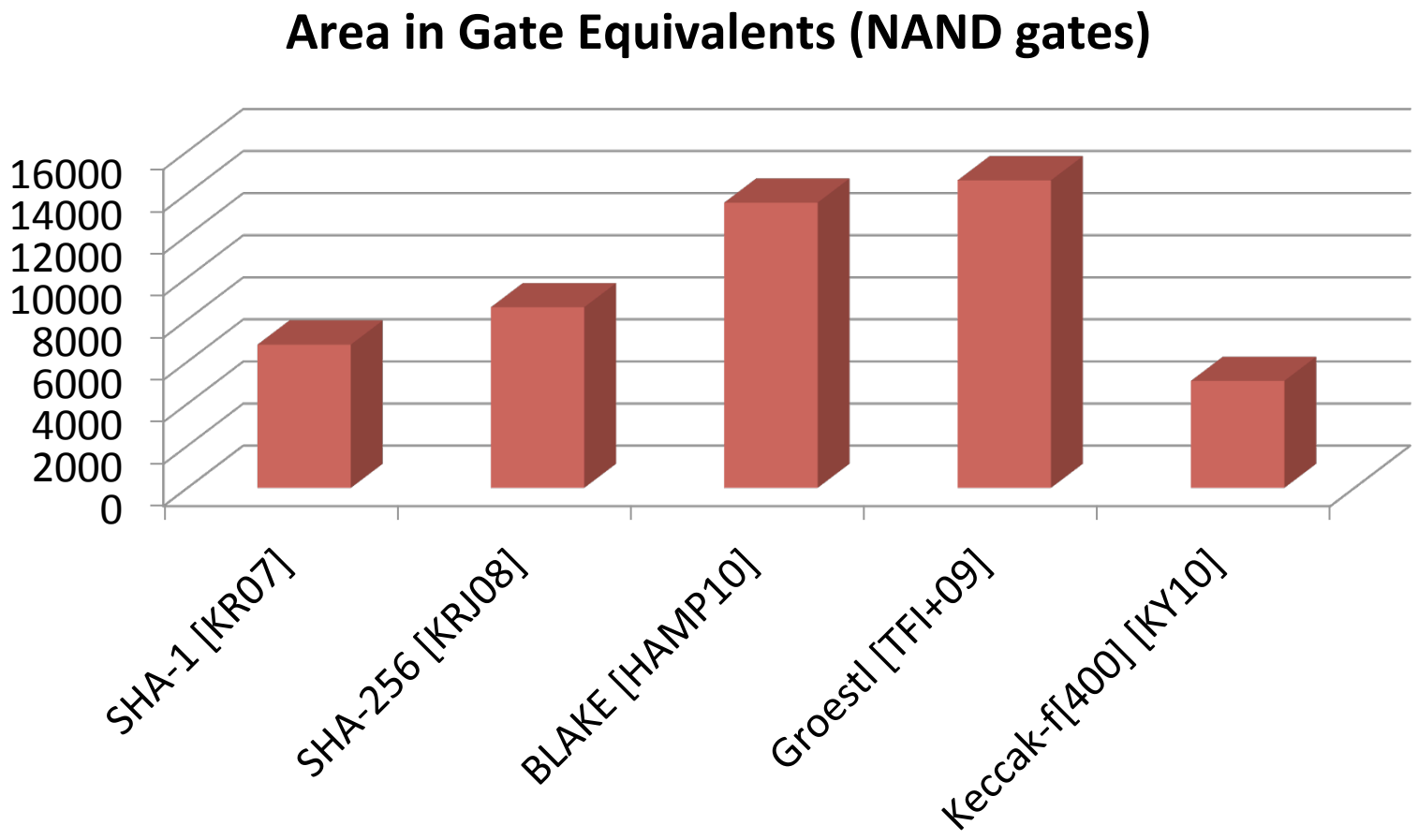
Lightweight =Low ASIC Area



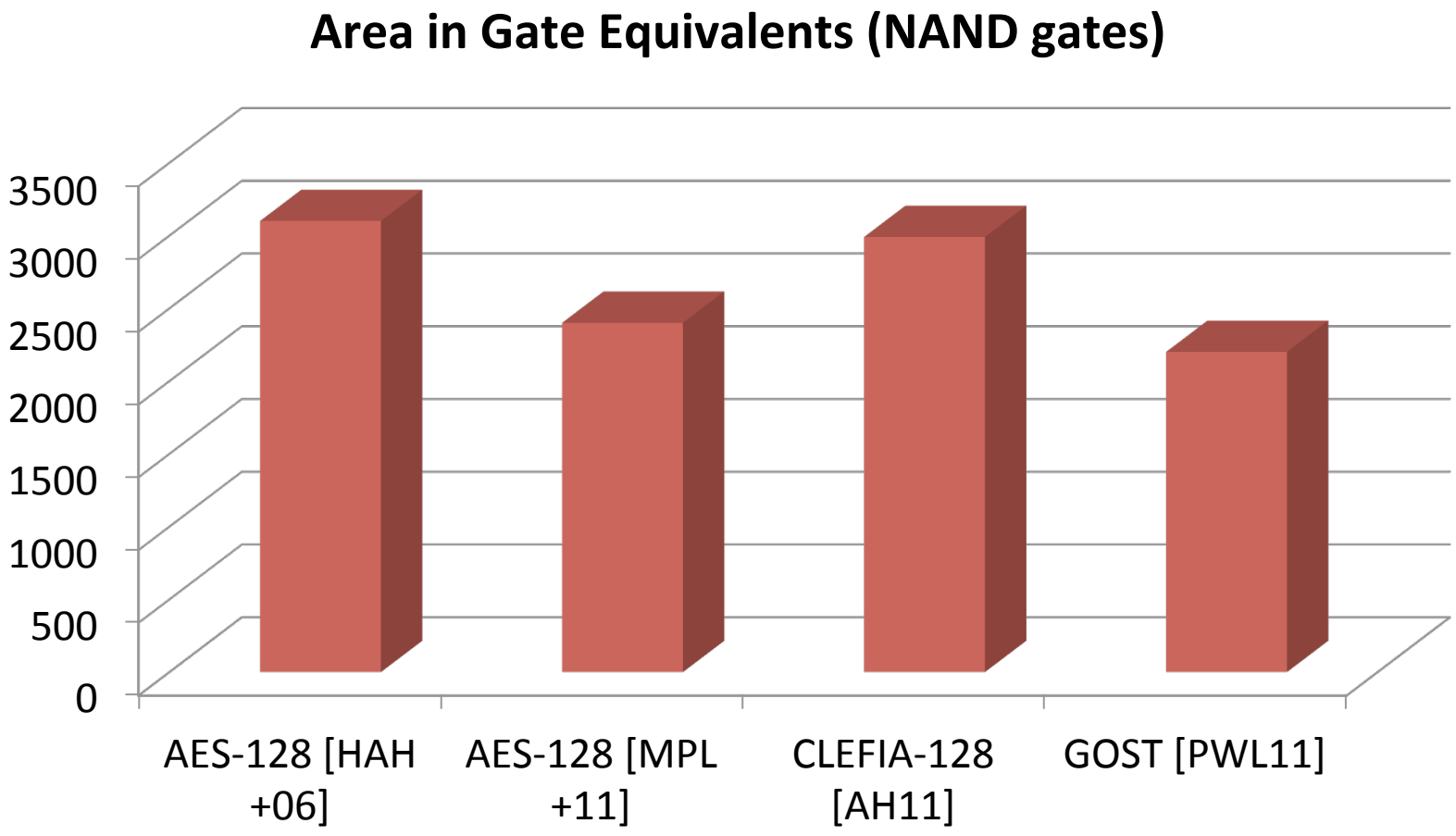
Crypto Algorithms to Meet Basic Security Needs



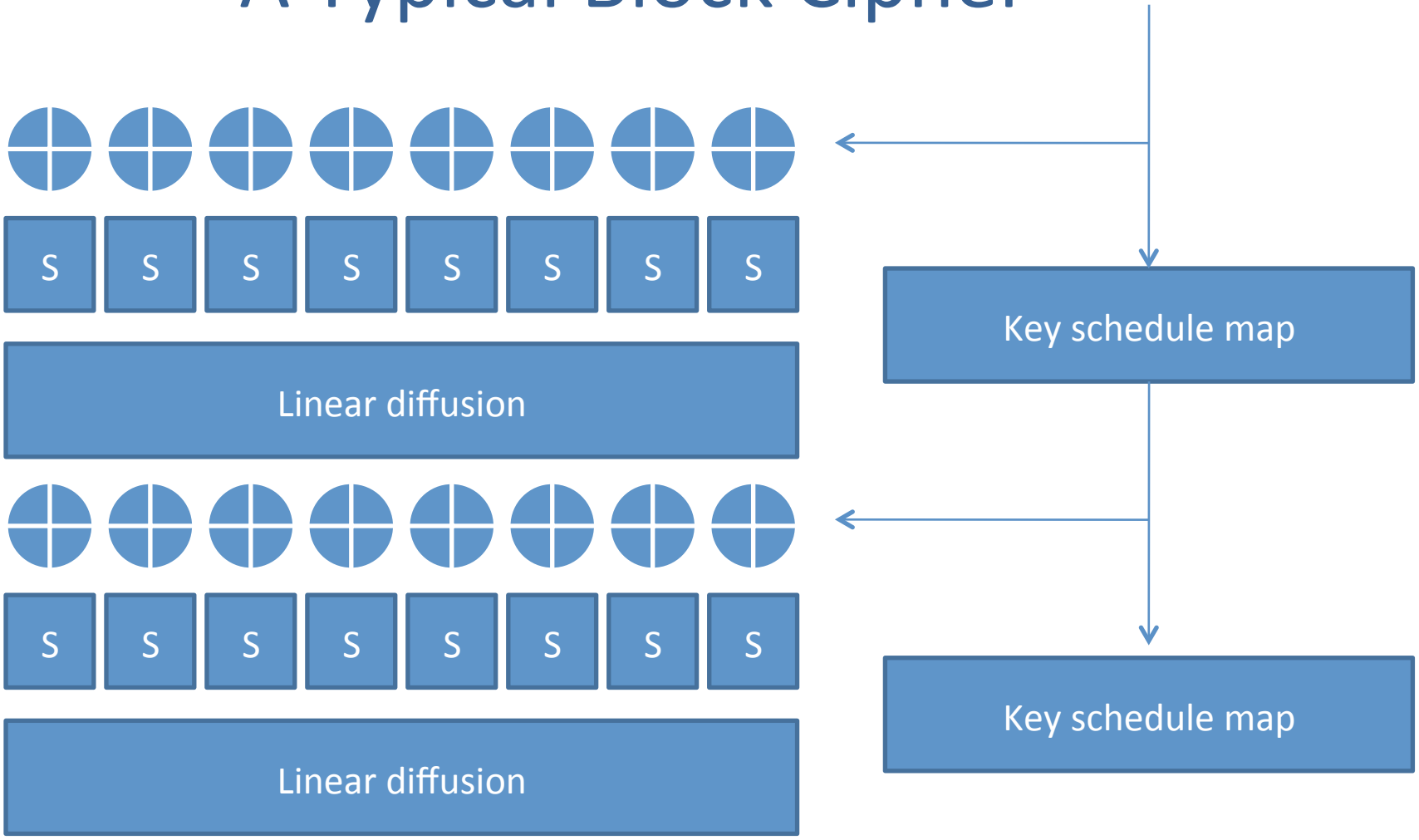
General-Purpose Hash Functions



General-Purpose Block Ciphers

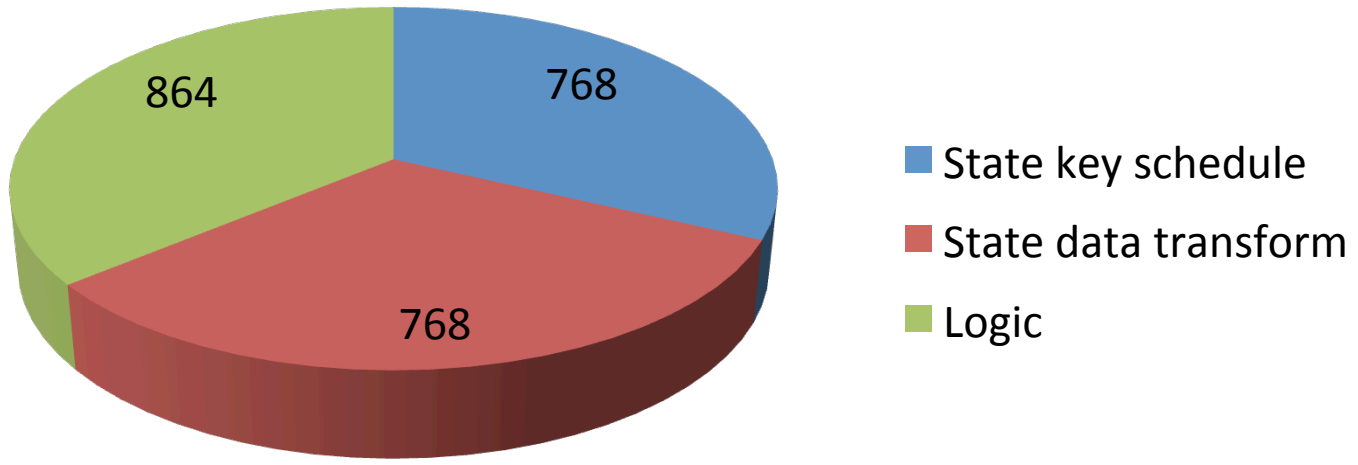


Substitution-Permutation: A Typical Block Cipher



Contribution of Building Blocks

Approximate Area of AES-128 [MPL+11] in GE



How to optimize the design of a cipher for low area?

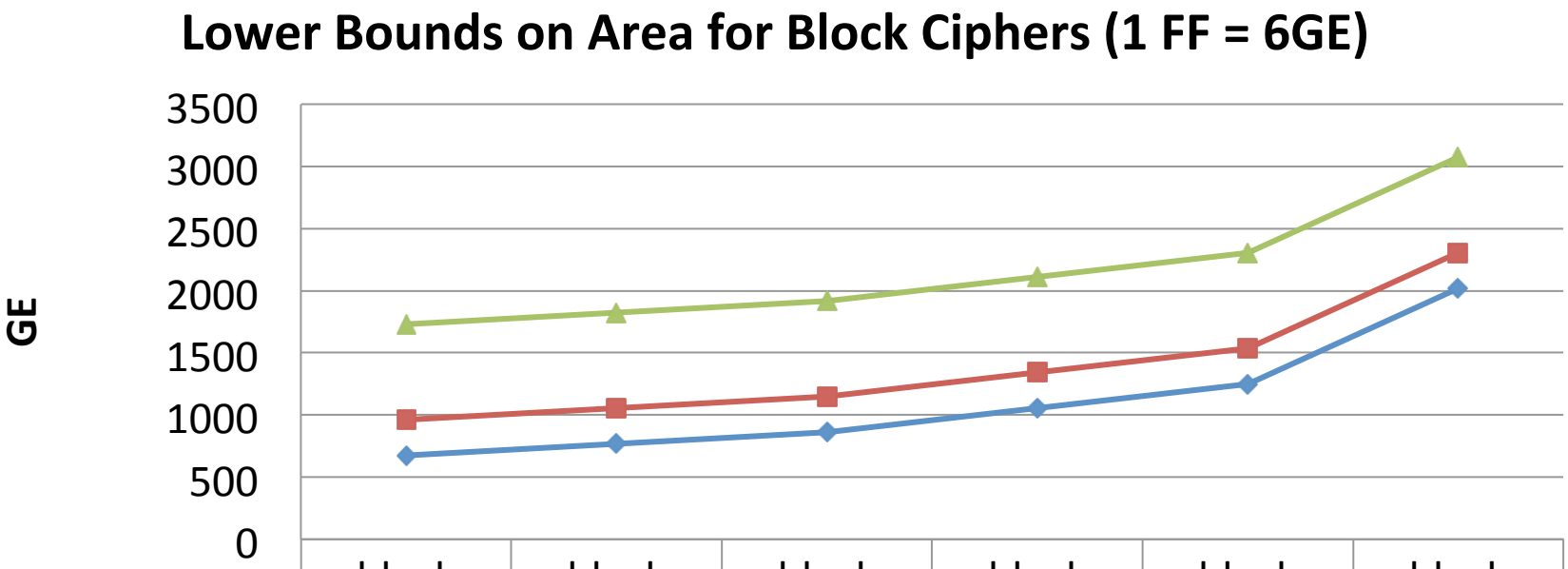
For given block and key sizes, minimize logic!

Area of Elementary Blocks

Logical operation	Cost in ASIC hardware
NAND(x,y)	**1.00 GE**
AND(x,y)	1.25 GE
OR(x,y)	1.25 GE
XOR(x,y)	2.25 GE
MUX(x,y;c)	2.50 GE
AND(x,y,z)	1.50 GE
MAJ(x,y,z)	2.25 GE
XOR(x,y,z)	**4.00 GE**

State (flip-flop)	Cost in ASIC hardware
1 bit	5.50 -7.50 GE

Minimum Area for Security Parameters



	block size 32	block size 48	block size 64	block size 96	block size 128	block size 256
key size 80	672	768	864	1056	1248	2016
key size 128	960	1056	1152	1344	1536	2304
key size 256	1728	1824	1920	2112	2304	3072

Lessons Learned: Design Ideas

Small State: Minimize block and key sizes

Each bit removed saves 6 GE	Tailor sizes to your exact needs

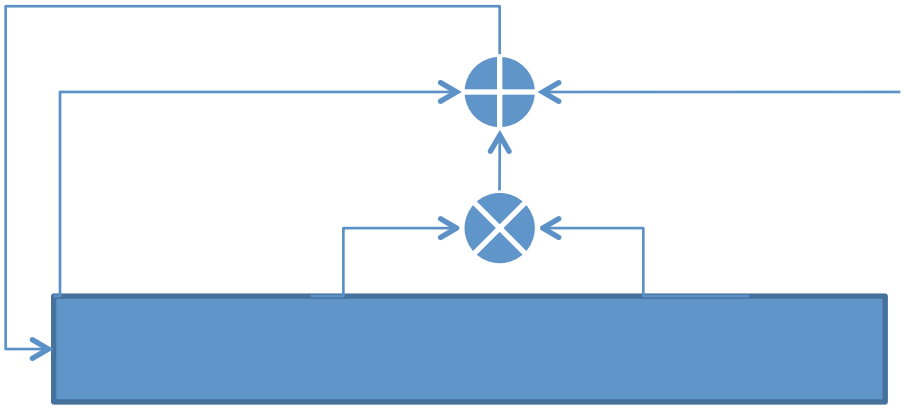
Small Logic: Minimize algorithm description

Min nonlinear = 1 GE per NAND	Avoid linear = 2.25 GE per XOR

Import expanded key: Get rid of the on-the-fly key schedule

Can save up to 50% of area	Depends on the environment

The Extreme Lightweight Cipher



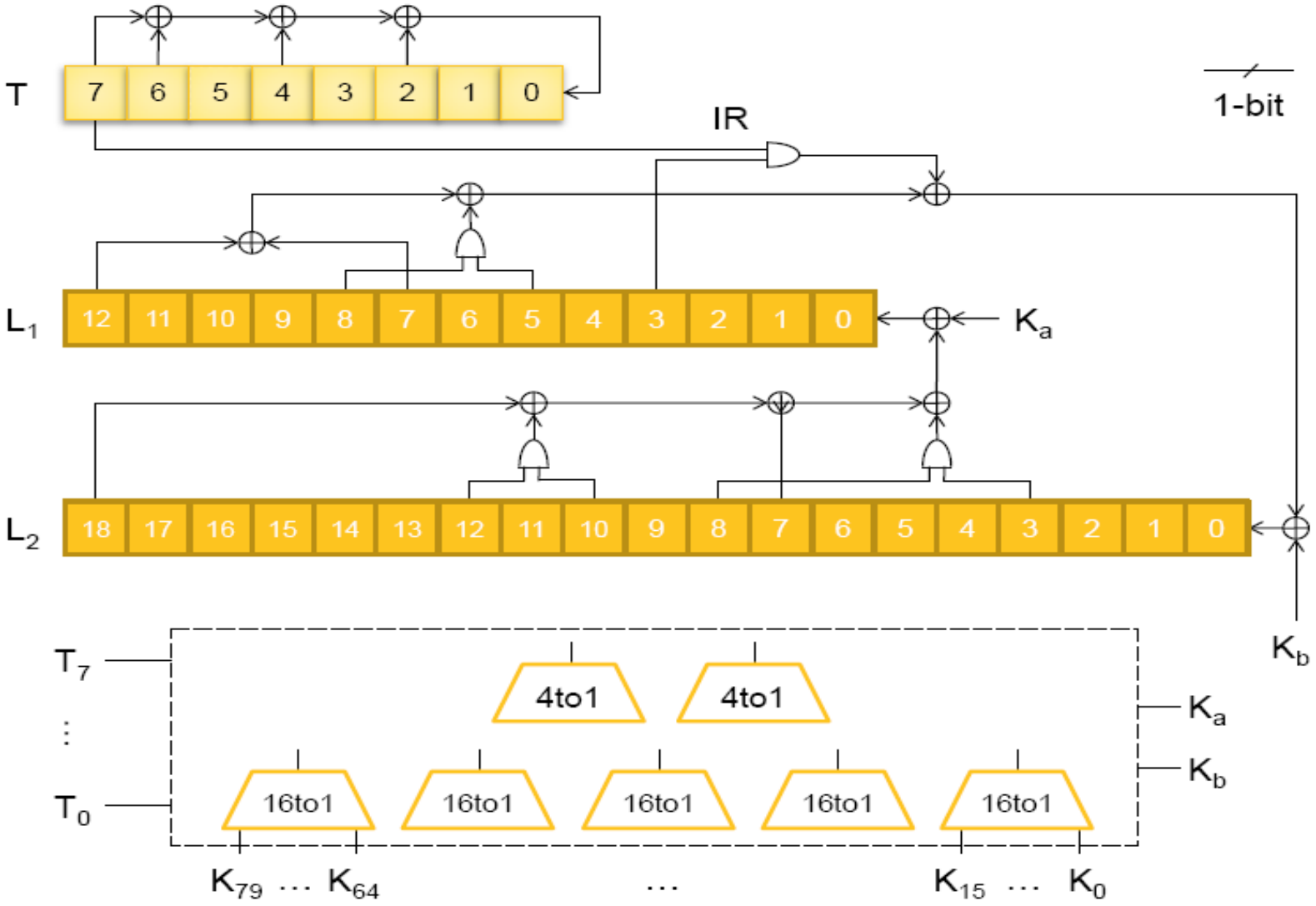
imported expanded key
with round constants

n-bit block

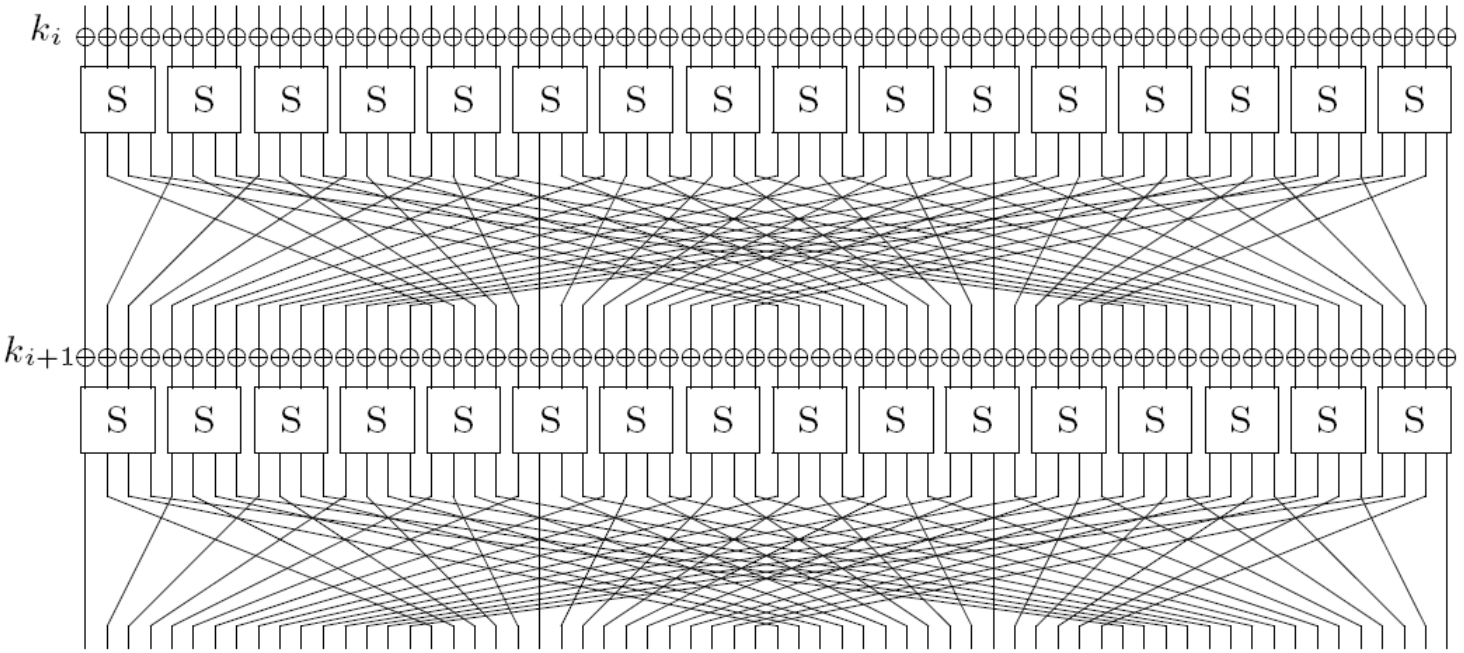
At least several thousand rounds required to attain security!

block size n	area: 5+6n GE
32	192
48	293
64	384
96	581
128	773

KTANTAN, 254 R [DDK09], 462 GE

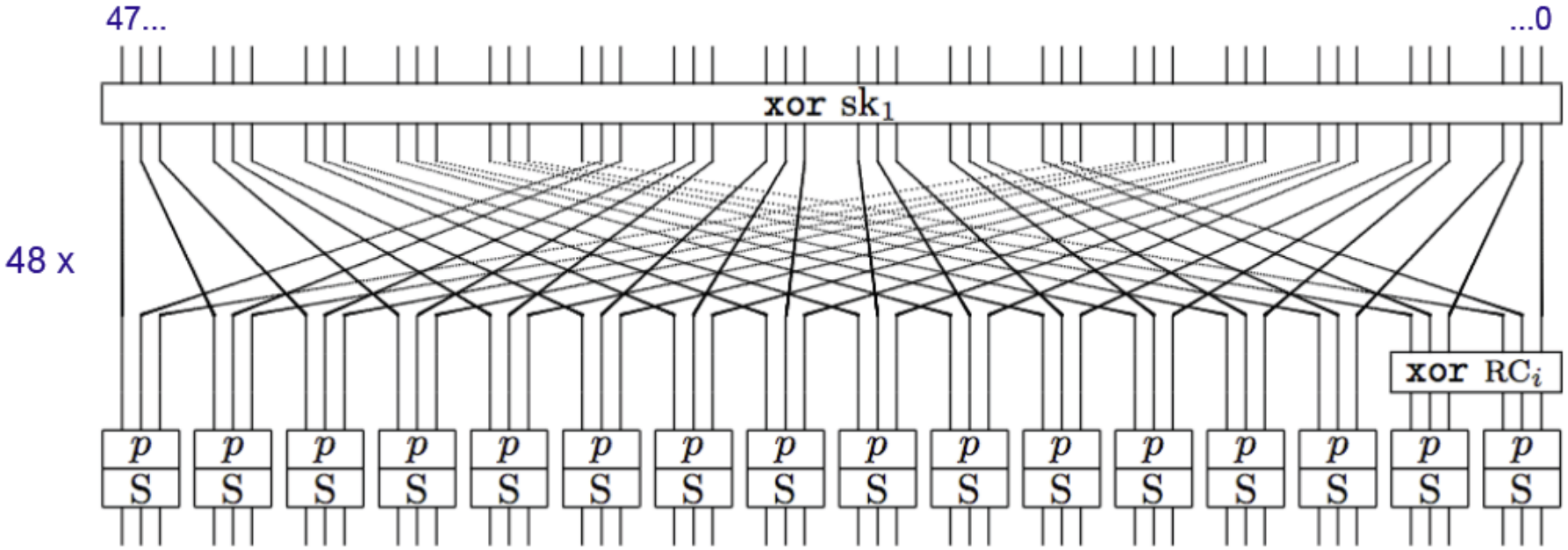


PRESENT, 31 R [BKL+07], 1075 GE



1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \texttt{round\_counter}$

PRINTcipher [KLP+10], 48/96 R, 402 GE



key in sk1 and secret bit permutations p

no key schedule (beyond round constant addition)

PART 3:
Authenticated Encryption

Why block ciphers?

- Most basic security primitive in nearly all security solutions, e.g. used for constructing
  - stream ciphers,
  - hash functions,
  - message authentication codes,
  - **authenticated encryption algorithms,**
  - entropy extractors, …
- Probably the best understood cryptographic primitives
- All U.S. symmetric-key encryption standards and recommendations have block ciphers at their core: DES, AES

Authenticated Encryption (AE)

- Is cryptography about encryption?
  - Yes, but not only!
  - Encryption alone is not enough in numerous applications
  - Is authentication actually more important?

- Authenticated encryption

  AE: (P,K) -> (C,T)
  with T authentication tag

- Authenticated encryption with associated data

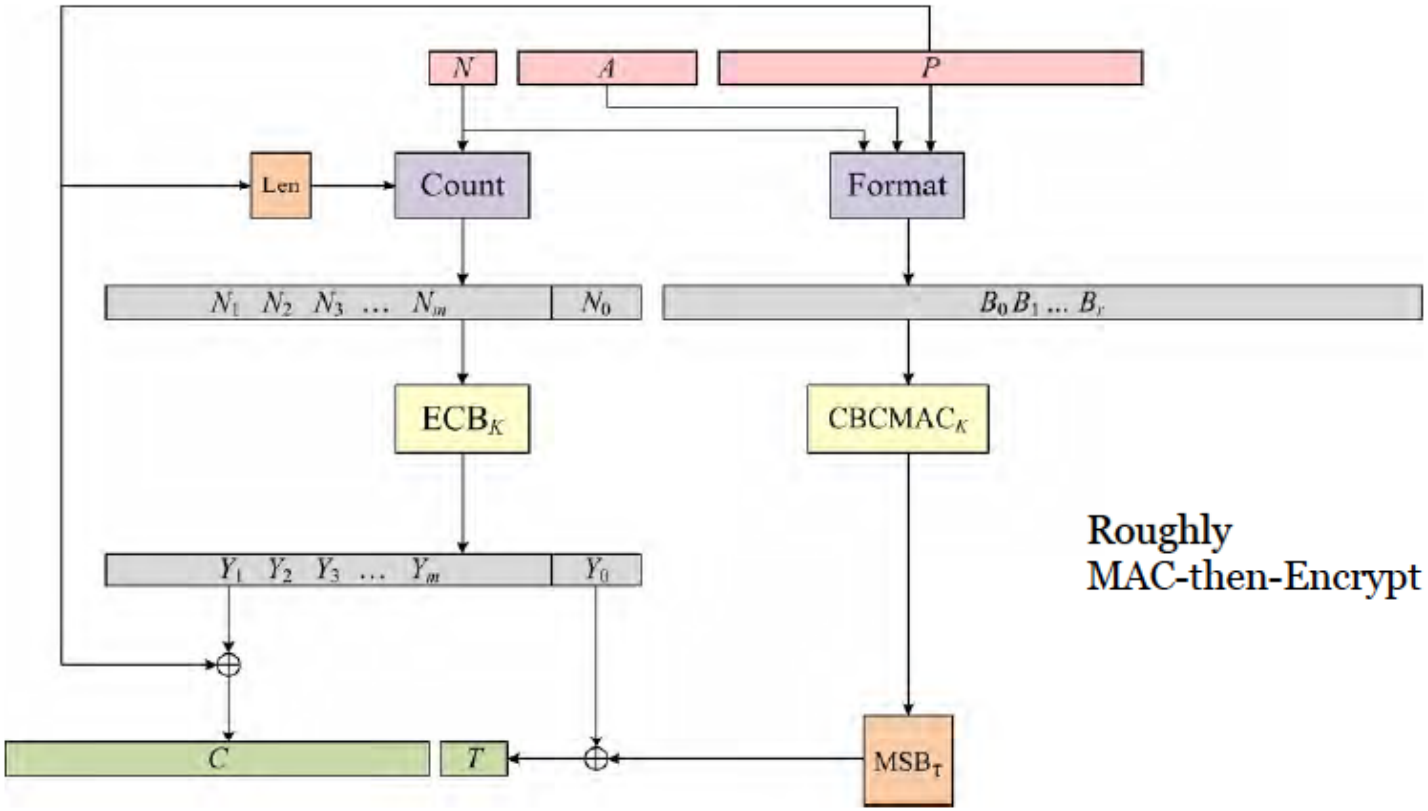
  AEAD: (A,P,K) -> (A,C,T)
  with A associated data transmitted in plaintext

CAESAR competition for authenticated ciphers

- Announced in Jan 2013 at the Early Symmetric Crypto seminar in Mondorf-les-Bains, Luxembourg
- CAESAR = Competition for Authenticated Encryption: Security, Applicability, and Robustness
- Submissions due: 2014
- **A great deal of attention**

CCM

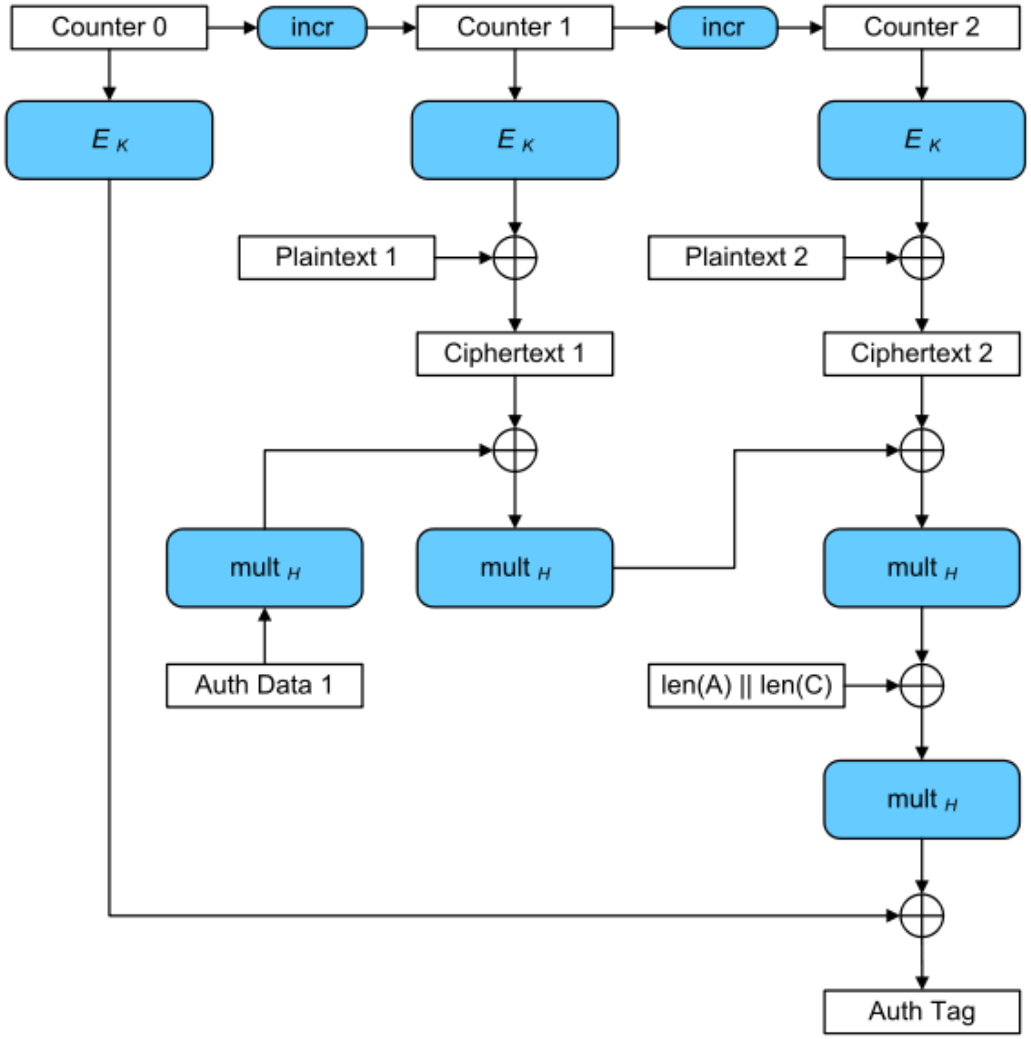
[Whiting, Housley, Ferguson 2002]
NIST SP 800-38C
RFC 3610, 4309, 5084



Courtesy of P. Rogaway, DIAC'12

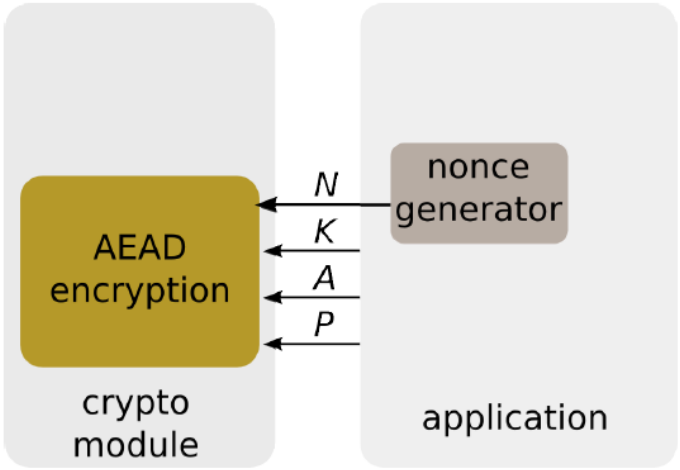
[MV04]

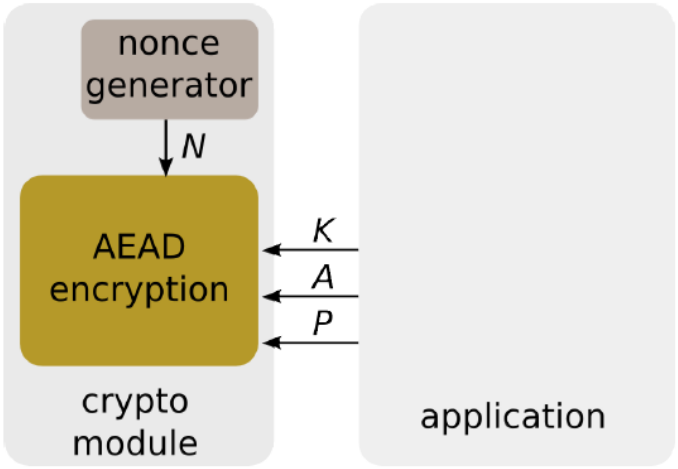
GCM


Counter 0
incr
Counter 1
incr
Counter 2
E K
E K
E K
Plaintext 1
Plaintext 2
Ciphertext 1
Ciphertext 2
mult H
mult H
mult H
Auth Data 1
len(A) || len(C)
mult H
Auth Tag

Nonce-free vs nonce-based

- Nonce N = number used once, freshness
- Nice but might be difficult to enforce in sometimes





- Good news: Nonce can be “just” a counter!

Courtesy of D. McGrew, DIAC’12

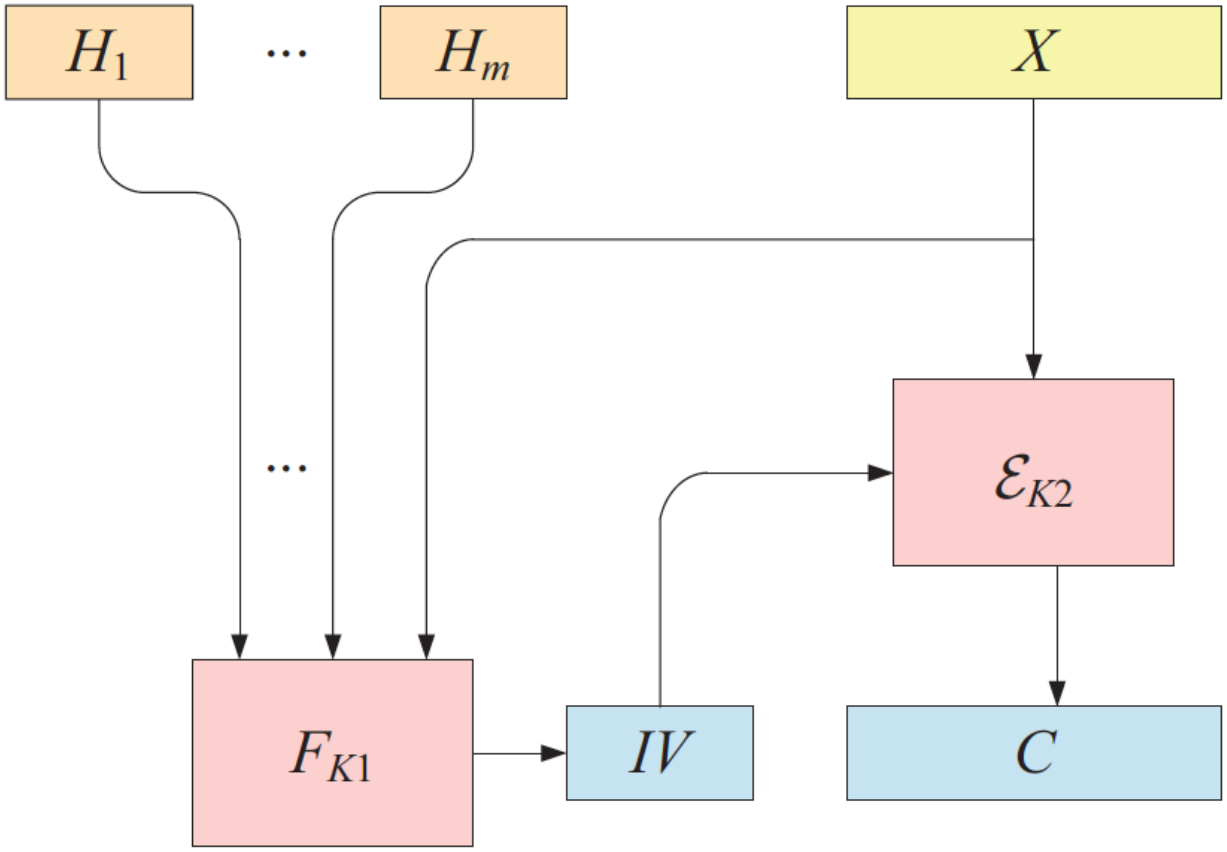
Nonce-free AE(AD)

- Nonce-free
  = deterministic authenticated encryption (DAE)
- There are two flavours
  - Single-pass
  - Double-pass

- Double-pass, SIV [RS06] as a good example
  - Processes data twice
  - Might be inefficient/prohibitive in some applications

- Single-pass, MCoE-G [FFLW12] as a good example
  - Process the data one time both for auth and enc
  - Inherent limitation/flaw: common prefix in P translates to common prefix in C

Double-pass: SIV



H = header, or AD

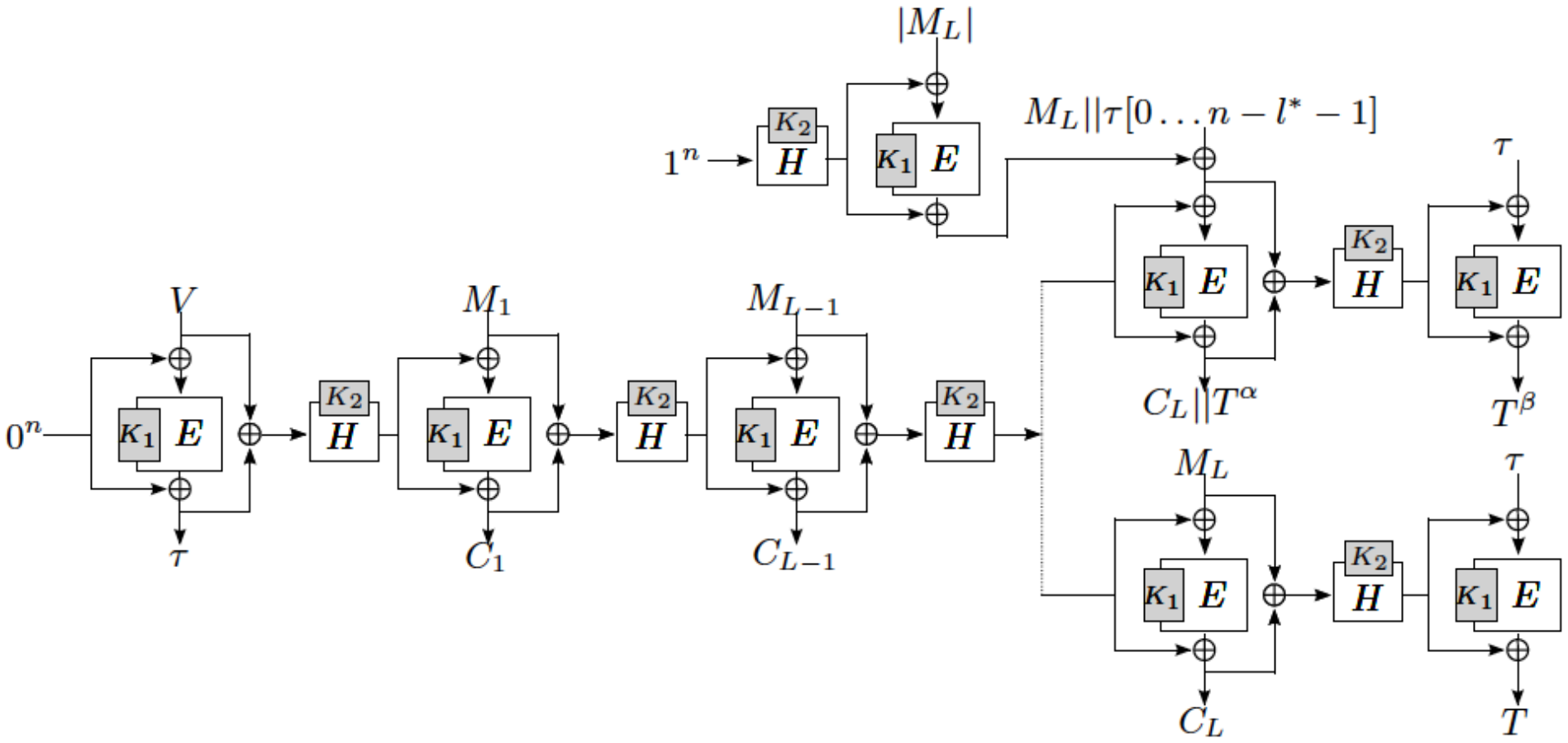
F = MAC

X = P

E = enc, e.g. AES-CTR

[FFLW12]

Single-pass: MCoE-G



[MMRT12]

MCoE-X: a bad example

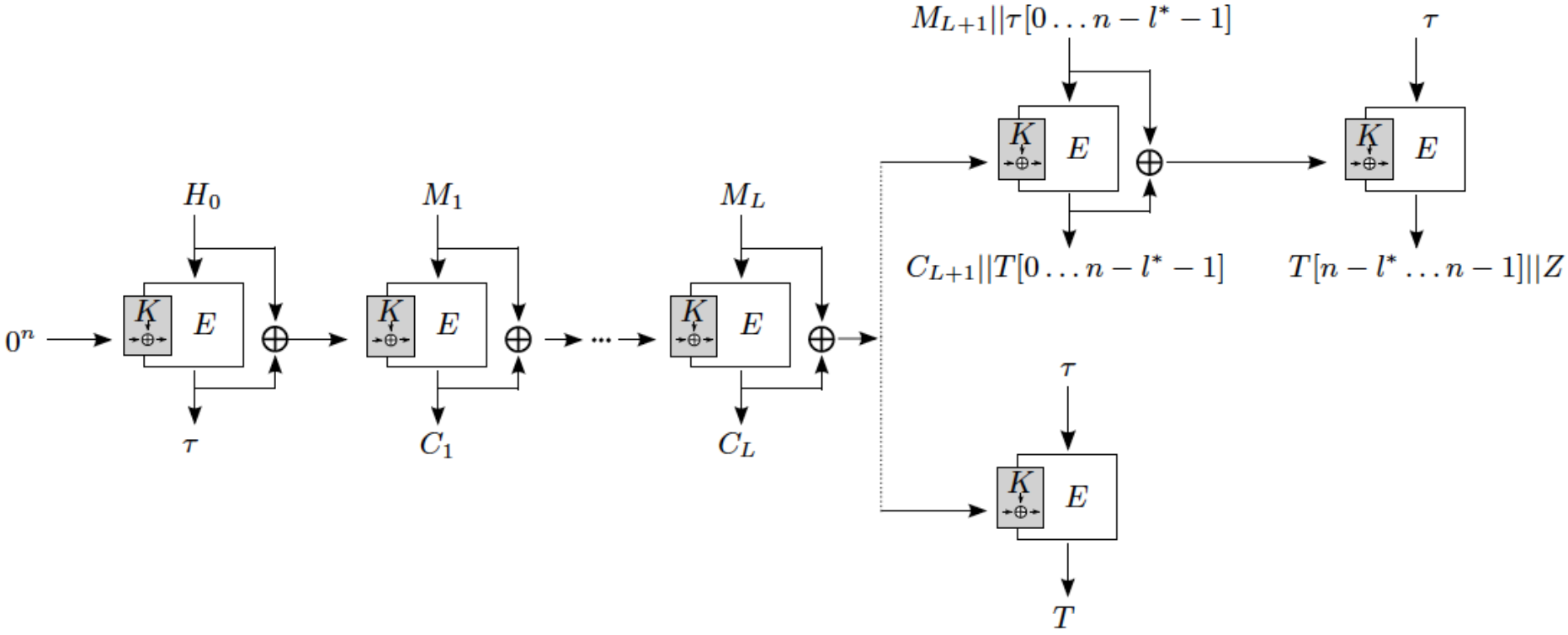


Figure: Structure of McOE-X.

[MMRT12]

MCoE-X: attack

A trivial attack (key collision):

1. Choose an arbitrary value a.
2. For ℓ values k compute $b = E(k, a)$ and save the pair (b, k) in a list L.
3. Choose an arbitrary x and set $M_1 = x$ and $M_2 = a$ such that $m = x \| a$ and ask for the ciphertext/tag pair (c, T) with $c = C_1 \| C_2$.
4. Check if C_2 is in the list L to get K.
   - If C_2 is in the list L then a candidate for the key is found. Compute $K = k \oplus M_1 \oplus C_1$,
   - Else go back to step 3.

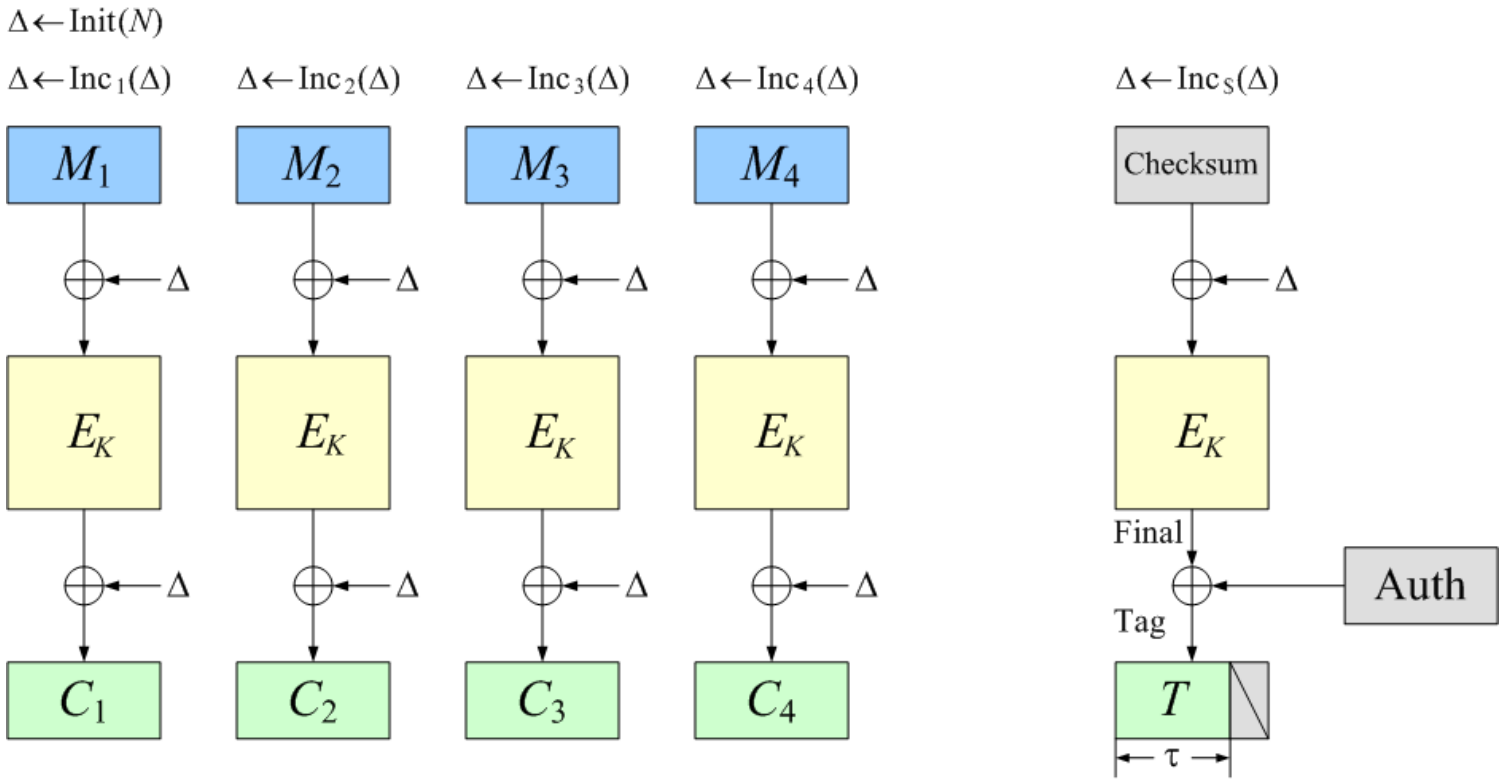
After repeating steps 3-4 about $2^n/\ell$ times one expects to find the correct key with complexity of about $2^n/\ell + \ell$.

Nonce-based AE(AD)

- OCB by Rogaway et al is hard to beat!
  - parallelizable (extremely fast with AES instructions – do we need it faster?)
  - virtually single cipher call per block
  - small overhead (especially with stretching)

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

Nonce-based: AES-OCB



+

- 1 AES-128 call per block
- perfectly parallelizable
- only forgery with nonce reuse
- associated data
- online scheme

-

- enc/dec different
- state 4x128 bits
- (patents pending)

AE: Modes and standards

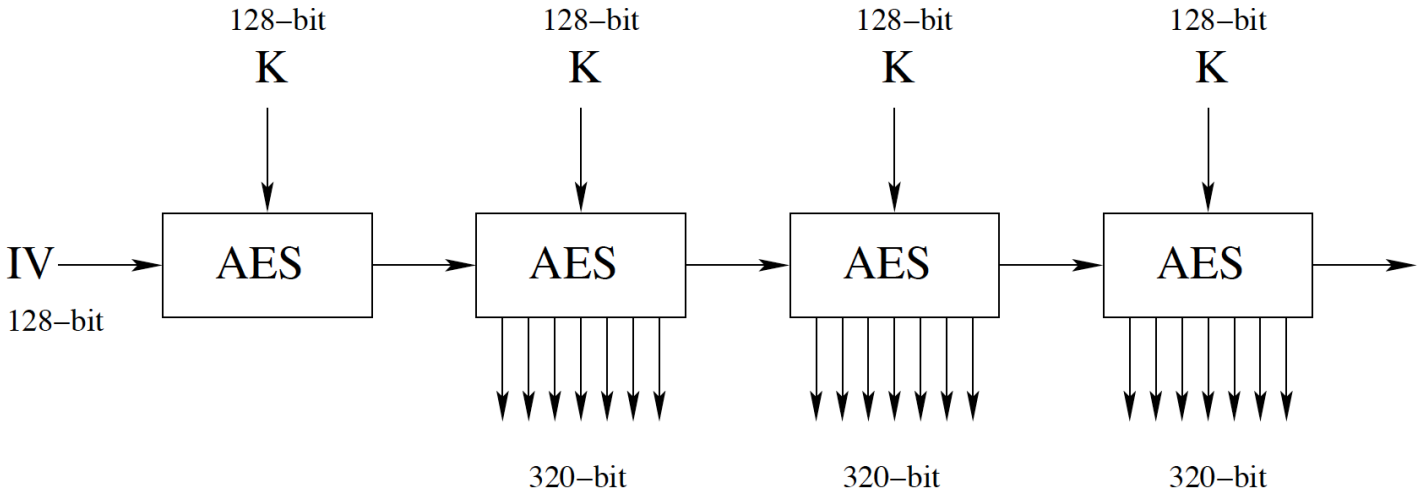
	Algorithms	Standards
1999	IAPCBC	
2000	IACBC, AE	
2001	OCB, AEAD	
2002	CCM	802.11
2003		
2004	GCM	802.1
2005		IPsec
2006		FC-SP, 1619.1, LTO-4
2007		
2008		RFC5116
2009	SIV	TLSv1.2, IKE, XMLsec, SSH
2010		
2011	OCBv3	
2012	CBC+HMAC	SRTP, *JOSE*

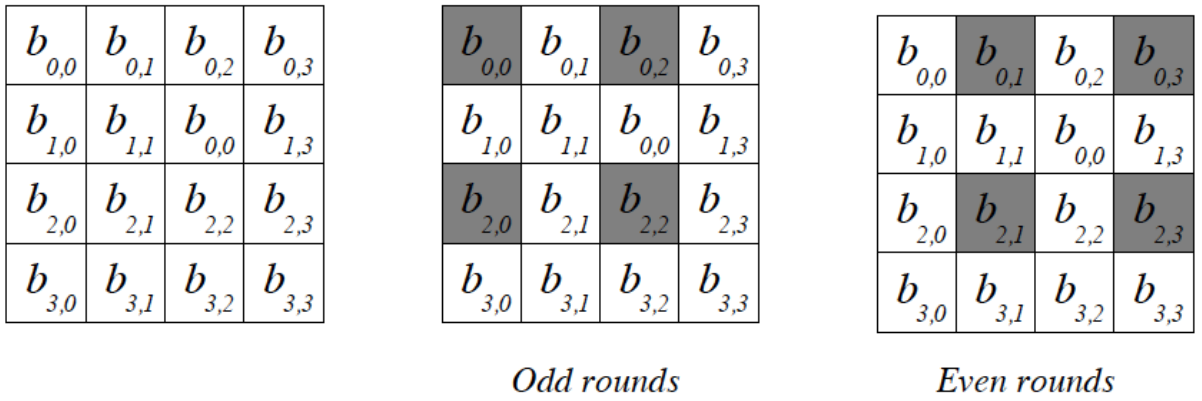
Nonce-based: Improve OCB?

- Need to consider other domains to attain improvements
- **Our goal:**

  Design of a dedicated AE scheme which would
  - require less operations on average
  - be compact in hardware, first of all for both enc and dec
  - have low power and low energy
  - be OK in PC software (use AES primitives for AES-NI)
  - be good in embedded software (which is mostly not parallelizable)
  - rely on some previous cryptanalysis

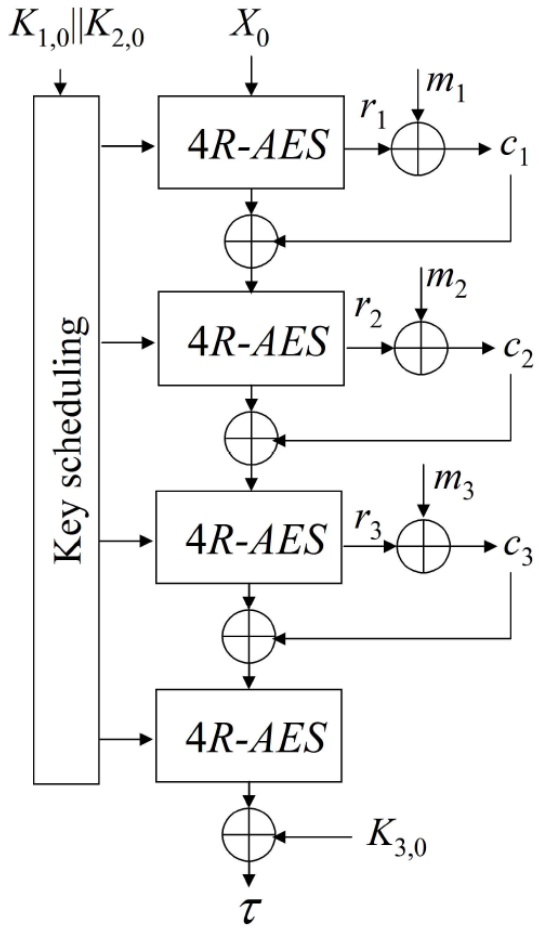
LEX: stream cipher





ASC-1

[JK11]



+

- only 4 AES-128 rounds per block
- enc/dec similar

-

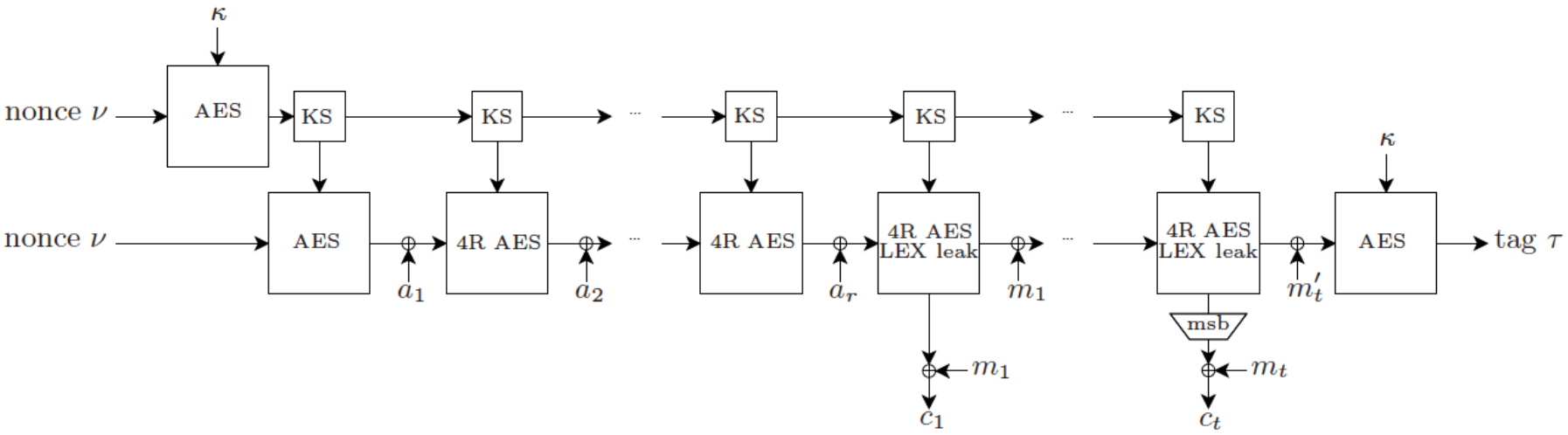
- state 4x128 bits
- serial
- state recovery with nonce reuse
- slow in compact ASIC implementation
- no associated data

$$X_0 = E_K(0^{70}||00||Cntr)$$

$$K_{1,0} = E_K(0^{70}||01||Cntr),\ K_{2,0} = E_K(0^{70}||10||Cntr),\ K_{3,0} = E_K(l(M)||0^6||11||Cntr)$$

ALE – Authenticated Lightweight Encryption

(A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, E. Tischhauser in FSE'13)



a_i = associated data

m_i = message

c_i = ciphertext

AES = AES-128

$\mathcal{K}$ = 128-bit key

$\mathcal{T}$ = 128-bit nonce

+

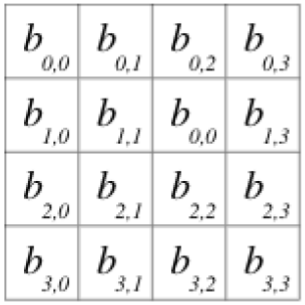
- only 4 AES-128 rounds per block
- enc/dec similar
- state 2x128 bits
- faster in compact ASIC implementation
- associated data
- online scheme

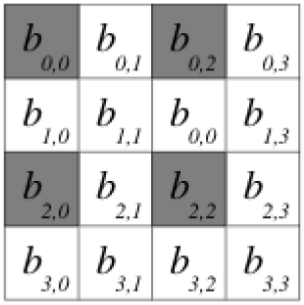
-

- serial
- state recovery with nonce reuse

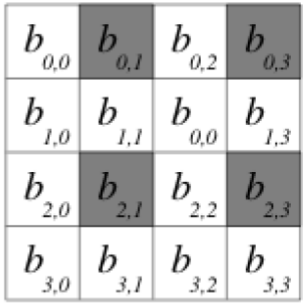
[B06]

LEX leak for ALE encryption





odd rounds



even rounds

Assumptions and claims for ALE

- **Assumption 1. Nonce-respecting adversary:** A nonce is only used once with the same master key for encryption
- **Assumption 2. Abort on verification failure:** No additional information returned if tampering is detected

- **Claim 1. State recovery:** State recovery with complexity = *N* data blocks succeeds with prob at most $N2^{-128}$
- **Claim 2. Key recovery:** State recovery with complexity = *N* data blocks succeeds with prob at most $N2^{-128}$, even if state recovered
- **Claim 3. Forgery w/o state recovery:** forgery not involving key/ state recovery succeeds with prob at most 2^{-128}

Elements of cryptanalysis for ALE

- Hardness of finding internal collisions is essential:
  - for Pelican-MAC type constructions: no internal collisions ➔ forgery is reduced to breaking AES-128
  - all known attacks on LEX use a generic internal collision on the data state

- ALE makes both key and data states dependent on both key and nonce, yielding a 256-bit state ➔ generic collisions on the full state are beyond reach

- Using a nonce-dependent session key stream (like in OCB and ASC-1) complicates differential attacks and makes partial internal collisions difficult to detect

Lightweight ASIC implementation for ALE

Design	Area (GE)	Net per 128-bit block (clock cycles)	Overhead per message (clock cycles)	Power (uW)
AES-ECB	2,435	226	-	87.84
AES-OCB2	4,563	226	452	165.21
AES-OCB2 e/d	5,783	226	452	201.32
ASC-1 A	4,793	370	904	169.11
ASC-1 A e/d	4,964	370	904	193.71
ASC-1 B	5,517	235	904	199.02
ASC-1 B e/d	5,632	235	904	207.13
AES-CCM	3,472	452	-	128.31
AES-CCM e/d	3,765	452	-	162.15
ALE	**2,581**	**84**	678	95.49
ALE e/d	**2,702**	**84**	678	103.11

STMicroelectronics 65 nm CMOS LP-HVT, Synopsis 2009.06, 20 MHz

Software implementation of ALE

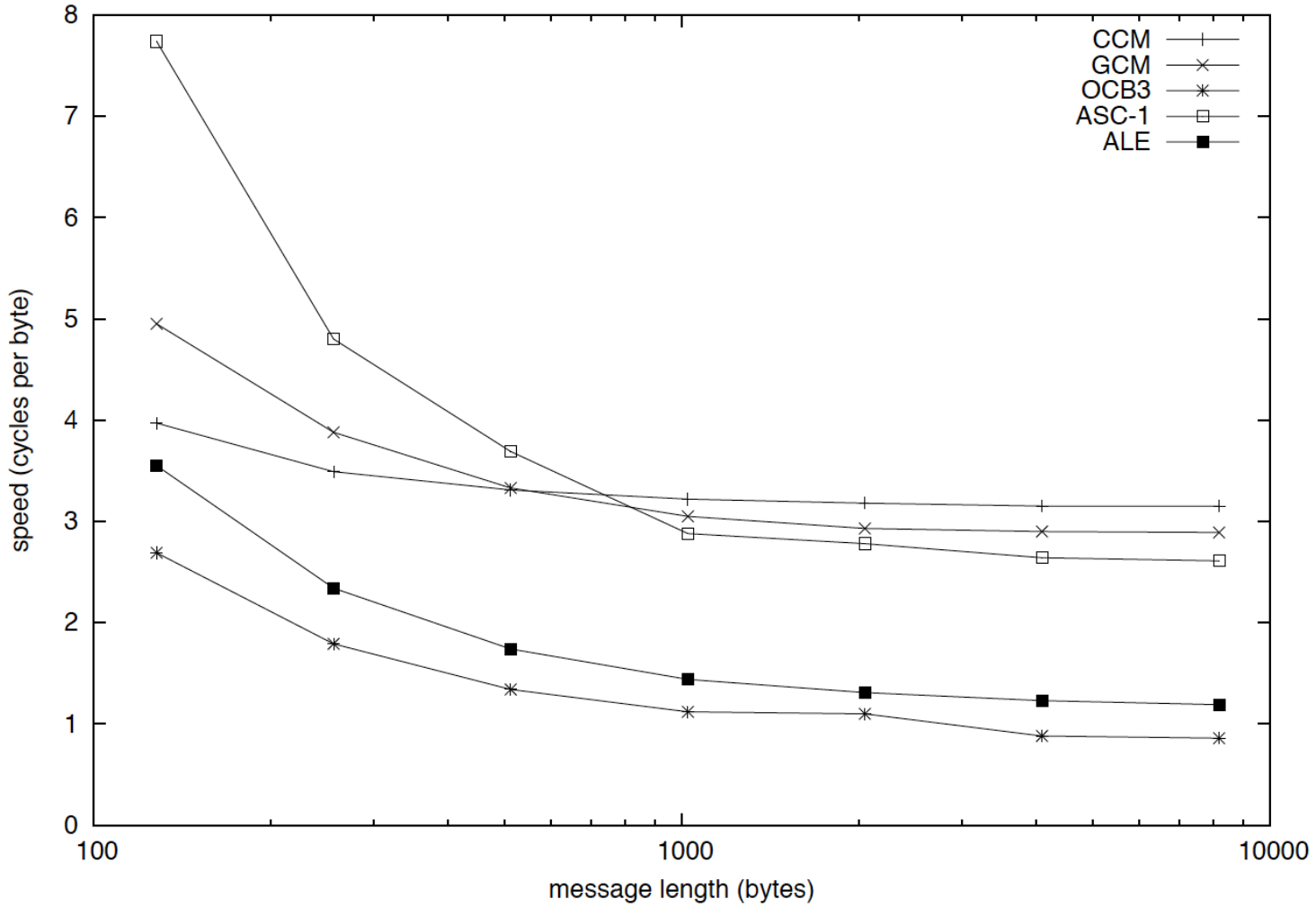
- AES-NI Sandy Bridge, cycles per byte

	message length (bytes)						
Algorithm	128	256	512	1024	2048	4096	8192
ECB	1.53	1.16	0.93	0.81	0.75	0.72	0.71
CTR	1.61	1.22	0.99	0.87	0.80	0.77	0.76
CCM*	3.97	3.49	3.31	3.22	3.18	3.15	3.15
GCM	4.95	3.88	3.33	3.05	2.93	2.90	2.89
OCB3	2.69	1.79	1.34	1.12	1.00	0.88	0.86
ASC-1	7.74	4.80	3.69	2.88	2.78	2.64	2.61
ALE*	3.55	2.34	1.74	1.44	1.31	1.23	1.19

- Embedded software:
  - Serial constructions usually do not cause much overhead
  - 2-2.5 faster than AES-OCB

Software implementation of ALE

- AES-NI Sandy Bridge, cycles per byte



Software implementation of ALE

- High-speed data links: 100 Gbit/s AE needed (McGrew)

- Standard Sandy Bridge desktop chips with 6 cores and AES-NI @ 3.1 GHz available

Thus, for 1KByte as average size of a message

- ALE: 103.3 Gbit/s
- OCB3: 132.8 Gbit/s

Do we need it faster?

Application of ALE: FPGA Bitstream Update

(A. Bogdanov, A. Moradi, T. Yalcin in ReConFig'12)

- FPGA = Field Programmable Gate Arrays
- Key feature: Hardware can be updated in the field!
- Risks:
  - Often no online connection
  - Physical access by adversary -> side-channel attacks
  - Reverse engineering
  - Bitstream manipulation
- Solution so far for Xilinx Virtex-6 FPGAs:
  - AES-256-CBC/HMAC-SHA-256
  - Not protected against side-channels!
  - Physical Limitation for bitstream update: 800 Mpbs @ 100 MHz
- Why not AES-OCB or ALE?

Application of ALE: FPGA Bitstream Update

(A. Bogdanov, A. Moradi, T. Yalcin in ReConFig'12)

Hardware implementations not protected against side-channels

Scheme	Area in GE (130nm)	Area in GE (90nm)	Area in GE (45nm)	Area in slices (V6)
ALE	**7.9K**	**7.1K**	**8.2K**	**594**
OCB	23.1K	18.1K	20.0K	2776
CCM	24.4K	20.4K	21.9K	1947
GCM	24.9K	21.6K	23.2K	2049
CBC/HMAC	34.3K	29.9K	33.5K	2201

Hardware implementations protected against side-channels

Scheme	Area in GE (130nm)	Area in GE (90nm)	Area in GE (45nm)	Area in slices (V6)
ALE	**48.1K**	**45.7K**	**49.0K**	**7122**
OCB	133.1K	118.3K	123.8K	48150
CCM	173.0K	161.1K	165.5K	33500
GCM	138.5K	128.6K	133.4K	23900
CBC/HMAC	165.9K	152.7K	164.7K	18900

Wrap-up for AE

- Authenticated ciphers
- Nonce assumption is crucial
- Efficient AE modes of operation for block ciphers
- Dedicated designs such as ALE tend to be even more efficient

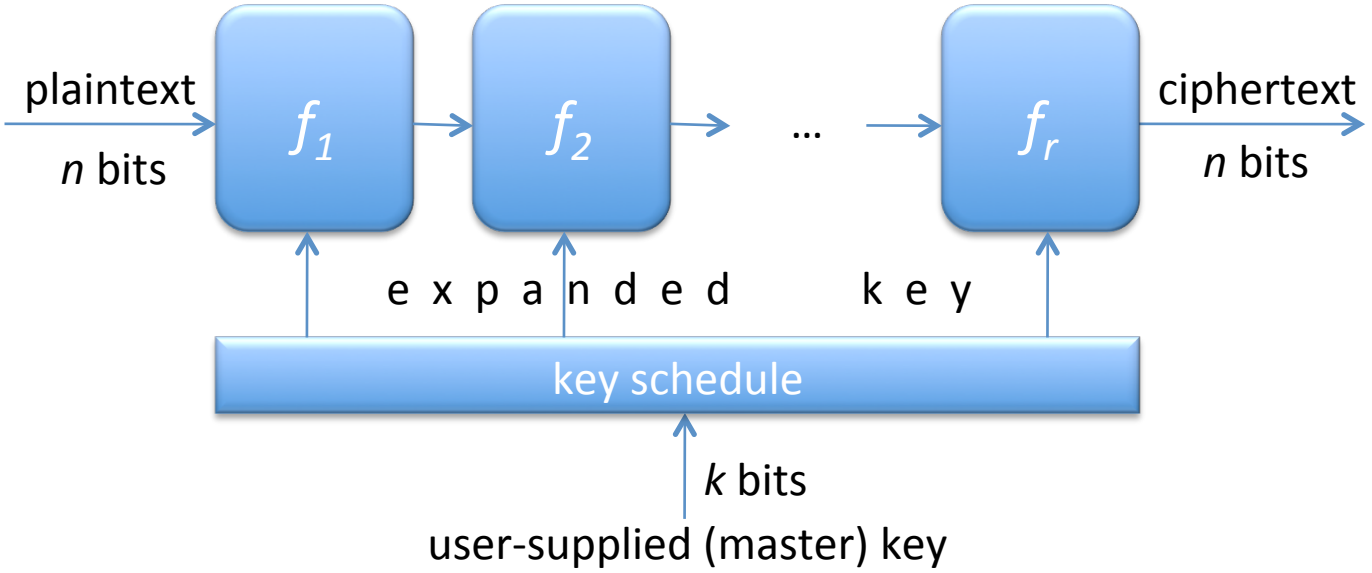
PART 4:
Meet-in-the-middle attacks

Some Headlines

- **“Researchers attack the super encryption”** Der Spiegel
- **“AES crypto broken by ‘groundbreaking’ attack“** The Register
- **“Microsoft finds holes in top-secret encryption key”** NewScientist
- **“AES proved vulnerable by Microsoft researchers”** Techworld
- **“Security fears over encryption after key recovery”** PC Pro
- **“Admins advised to move off Advanced Encryption Standard”** itnews
- **“AES cracked - or is it?”** Voltage Security
- **“Has the advanced encryption standard been broken or weakened?”** SC Magazine

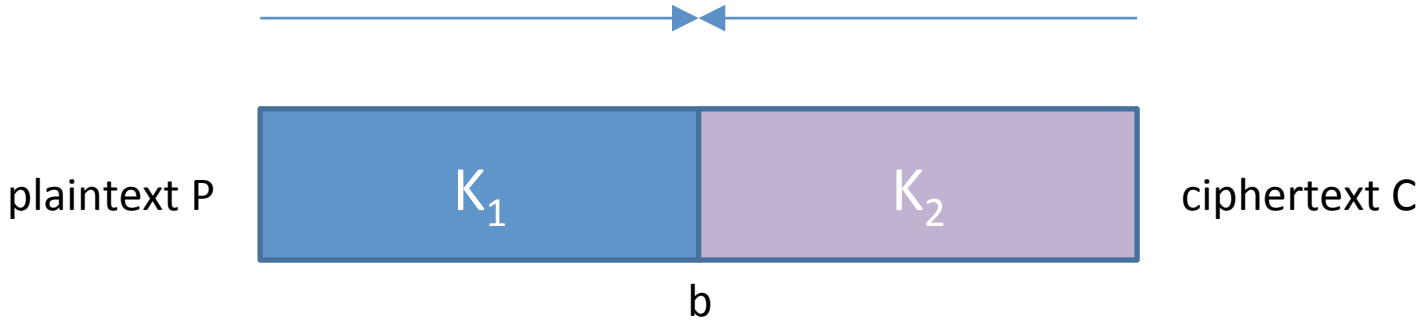
and many more

Iterative block ciphers



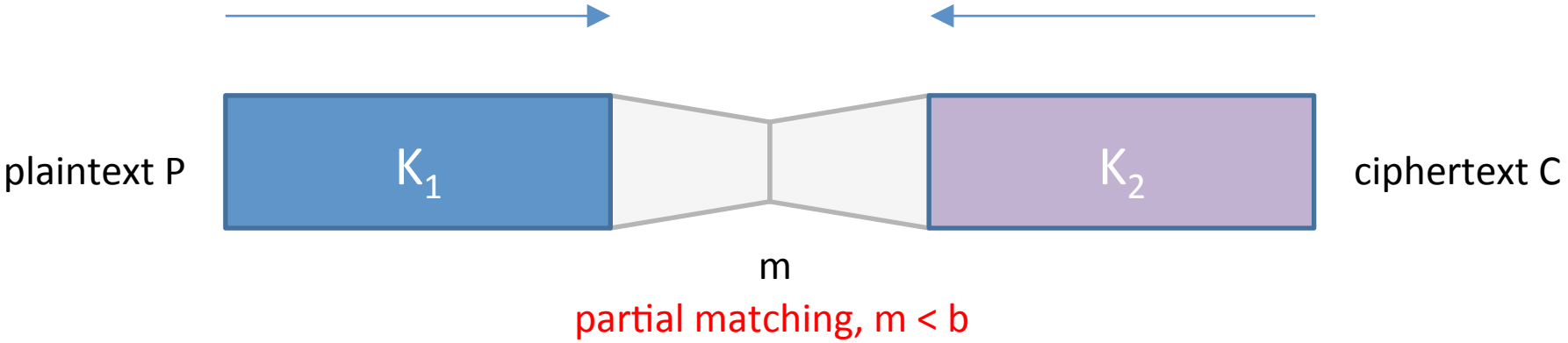
An iterative block cipher consists of r consecutive applications of simpler key-dependent transforms $f = f_r \circ f_{r-1} \circ \ldots f_2 \circ f_1$

Basic MITM



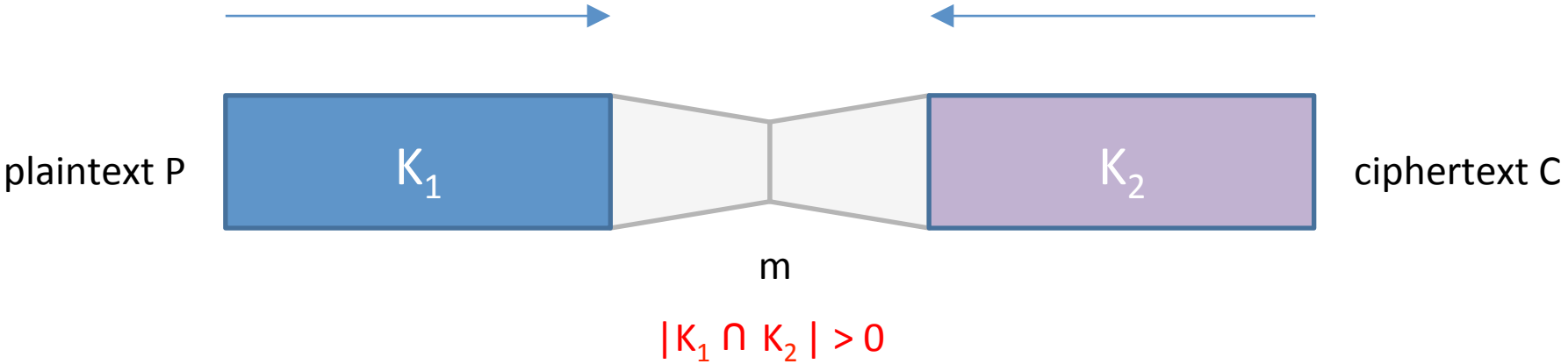
- Guess K_1 and K_2 independently
- Compute forwards from P
- Compute backwards from C
- Matching at state on b bits
- Complexity: $2^{|K1|}+2^{|K2|}+2^{|K|-b}$ computations

MITM with partial matching



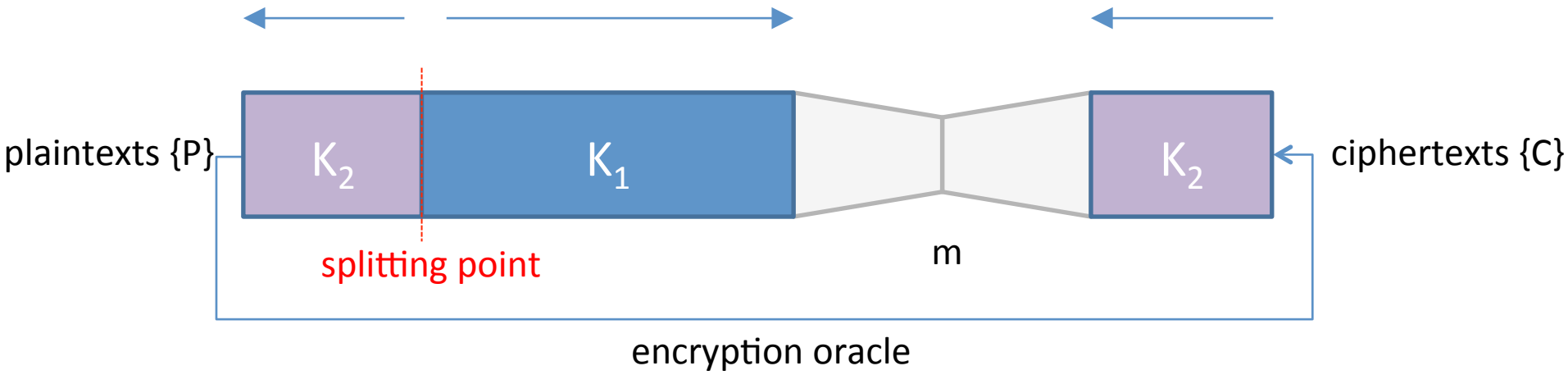
- Guess K_1 and K_2 independently
- Compute forwards from P
- Compute backwards from C
- Matching at part of state on m bits
- Complexity: $2^{|K1|}+2^{|K2|}+2^{|K|-m}$ computations

MITM with 3 subsets



- 3 key spaces:
  - A_0 (both in K_1 and K_2)
  - A_1(K_1 only), A_2 (K_2 only)
- Guess A_0, then A_1 and A_2 independently
- Complexity: $2^{|A0|}(2^{|A1|}+2^{|A2|})+2^{|K|-m}$

MITM with Splice & Cut



- Guess A_0, then A_1 and A_2 independently
- Compute forwards from splitting point
- Compute backwards from splitting point
  - Over encryption/decryption oracle!
- Complexity: $2^{|A0|}(2^{|A1|}+2^{|A2|})+2^{|K|-m}$

Target?

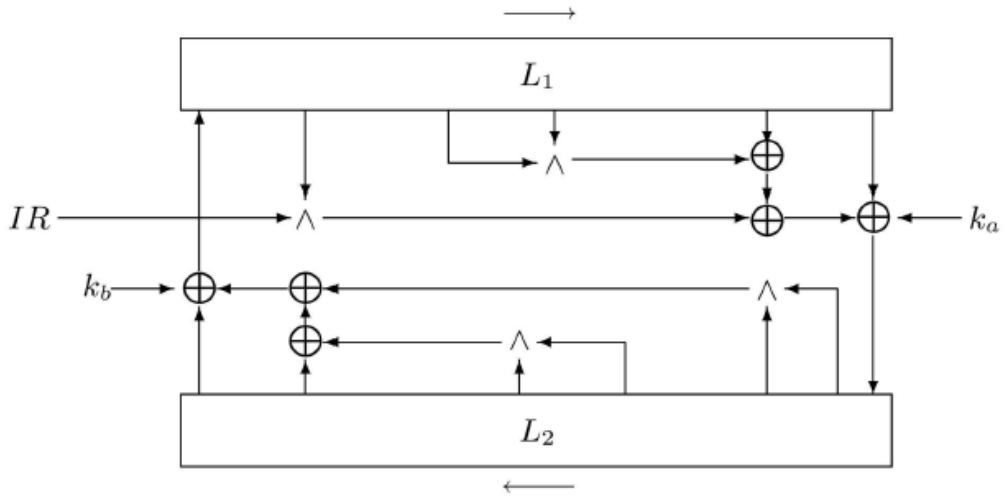
- Bitwise key schedules are more likely to have:
  - $A_0 = K_1 \cap K_2 \neq K$
  - $A_1 = K_1 \backslash K_2 \neq \emptyset$
  - $A_2 = K_2 \backslash K_1 \neq \emptyset$
  - $\Rightarrow$ complexity $<$ brute force
- Bitwise round transforms have as a rule longer matching
  - $\Rightarrow$ more rounds can be covered

$$\Downarrow$$

KTANTAN ciphers are natural targets!

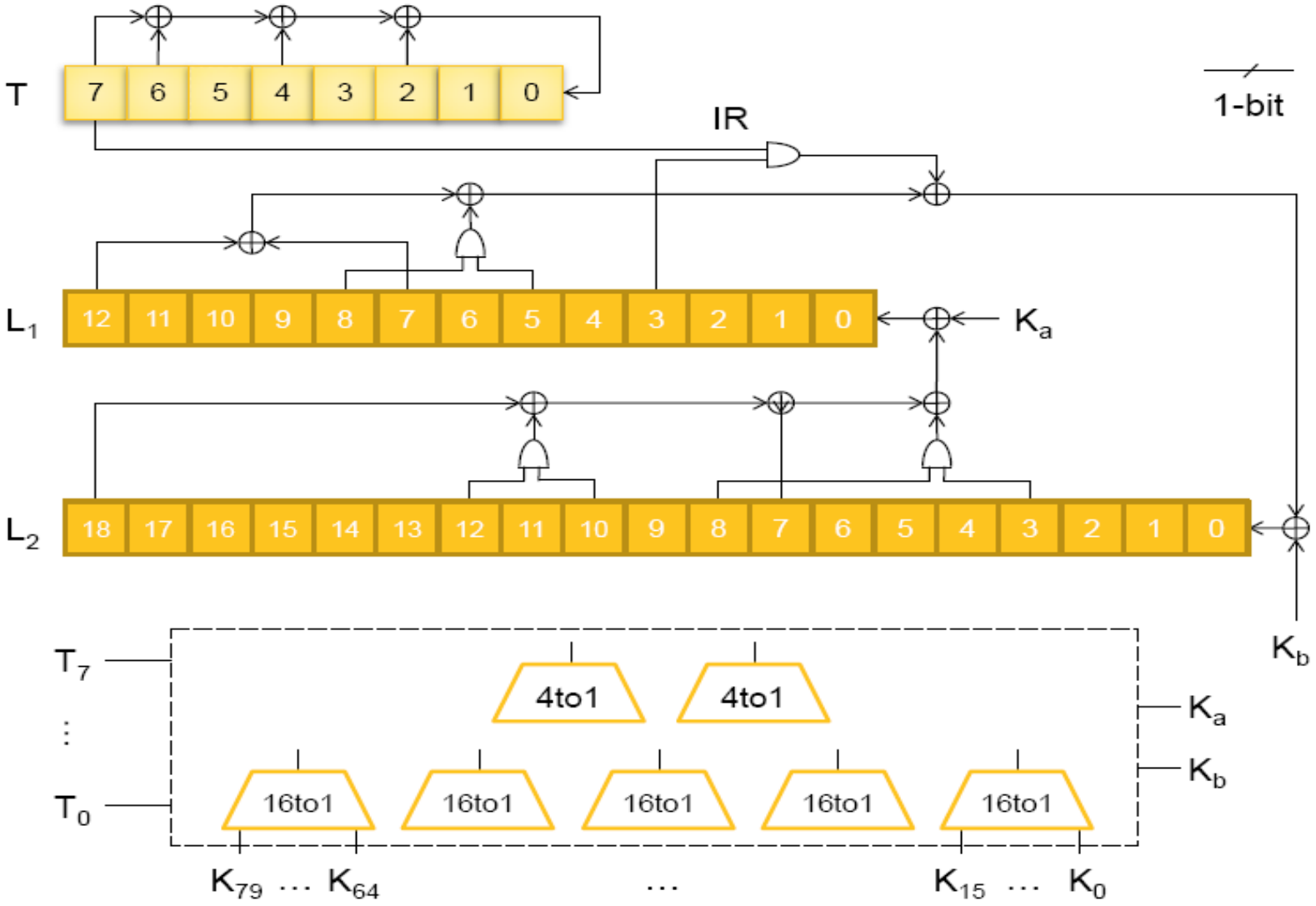
KTANTAN

Data transform of 254 rounds



- IR = a constant string of bits ($\approx$ round constants)
- bits k_a and k_b from a key schedule
- 3 versions: 32-, 48- and 64-bit blocks
- different clocking in different versions to make a round
  - 1 clock = round of KTANTAN32
  - 2 clocks = round of KTANTAN48
  - 3 clocks = round of KTANTAN64

KTANTAN


T
7 6 5 4 3 2 1 0
IR
1-bit
L1
12 11 10 9 8 7 6 5 4 3 2 1 0
Ka
L2
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Kb
T7
T0
4to1
4to1
16to1
16to1
16to1
16to1
16to1
Ka
Kb
K79 ... K64
...
K15 ... K0

KTANTAN: Key Schedule

Key schedule

- key schedule is the same for all KTANTAN versions
- user-supplied key of 80 bits $\{k_0, \ldots, k_{79}\}$
- in each round, 2 of them are chosen as k_a and k_b
  - round 1: $k_a = k_{31}, k_b = k_{63}$
  - round 2: $k_a = k_{47}, k_b = k_{47}$
  - round 3: $k_a = k_{79}, k_b = k_{15}$
  - round 4: $k_a = k_{78}, k_b = k_{14}$
  - ...
  - round 253: $k_a = k_{71}, k_b = k_7$
  - round 254: $k_a = k_{63}, k_b = k_{31}$

$\Downarrow$

- long spans not using all key bits are likely to exist

$\Downarrow$

apply generalized MITM!

KTANTAN: Key Schedule

$$K = W_4||W_3||W_2||W_1||W_0$$

$$\omega_{i,r} = \text{MUX16to1}(W_i, l_{7,r}l_{6,r}l_{5,r}l_{4,r}), i = 0, \ldots, 4,$$

$$\kappa_{1,r} = \overline{l_{3,r}} \cdot \overline{l_{2,r}} \cdot \omega_{0,r} \oplus (l_{3,r} \vee l_{2,r}) \cdot \text{MUX4to1}(\omega_{4,r}\omega_{3,r}\omega_{2,r}\omega_{1,r}, l_{1,r}l_{0,r})$$

$$\kappa_{2,r} = \overline{l_{3,r}} \cdot l_{2,r} \cdot \omega_{4,r} \oplus (l_{3,r} \vee \overline{l_{2,r}}) \cdot \text{MUX4to1}(\omega_{3,r}\omega_{2,r}\omega_{1,r}\omega_{0,r}, \overline{l_{1,r}l_{0,r}})$$

KTANTAN: Properties

Related-key differentials for KTANTAN*b*

covered rounds	#rounds	differential	probability
$\varphi_{1,218}$	218	$(0, 00000000800000000000) \mapsto 0$	1
$\varphi^{-1}_{81,254}$	174	$(0, 00000000000000010000) \mapsto 0$	1

KTANTAN: Properties

Fact 1 *$\varphi_{1,\alpha}$ does not use key bits $\{k_{32}, k_{39}, k_{44}, k_{61}, k_{66}, k_{75}\}$ for $1 \leq \alpha \leq 111$.*

Fact 2 *$\varphi_{254-\beta+1,254}$ does not use key bits $\{k_3, k_{20}, k_{41}, k_{47}, k_{63}, k_{74}\}$ for $1 \leq \beta \leq 131$.*

b	R	α	A_2	$R-\beta$	A_1	matching bits m	complexity C_{comp} MITM	key test	total
32	254	111	32,39,44,61,66,75	131	3,20,41,47,63,74	8	75.000	72	75.170
48	254	111	32,39,44,61,66,75	131	3,20,41,47,63,74	10	75.000	70	75.044
64	254	123	32,44,61,66,75	131	3,20,41,47,63,74	47	75.584	33	75.584

True complexity

$$C_{\text{comp}} = \underbrace{2^{|A_0|}(2^{|A_1|} + 2^{|A_2|})}_{\text{MITM stage}} + \underbrace{(2^{\ell-m} + 2^{\ell-m-b} + 2^{\ell-m-2b} + \dots)}_{\text{key testing stage}}$$

Partial matching (KTANTAN32)

```
10 bits match as follows

forward part:
forward R=111: k1=1, k2=1  0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=112: k1=0, k2=0  1 0 0 0 0 0 0 0 0 0 0 0 0 0   1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=113: k1=0, k2=0  0 1 0 0 0 0 0 0 0 0 0 0 0 0   0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=114: k1=0, k2=0  0 0 1 0 0 0 0 0 0 0 0 0 0 0   0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=115: k1=0, k2=0  0 0 0 1 0 0 0 0 0 0 0 0 0 0   0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=116: k1=0, k2=0  1 0 0 0 1 0 0 0 0 0 0 0 0 0   1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
forward R=117: k1=0, k2=0  0 1 0 0 0 1 0 0 0 0 0 0 0 0   0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
forward R=118: k1=0, k2=0  0 0 1 0 0 0 1 0 0 0 0 0 0 0   1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
forward R=119: k1=0, k2=0  0 0 0 1 0 0 0 1 0 0 0 0 0 0   0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
forward R=120: k1=0, k2=0  1 0 0 0 1 0 0 0 1 0 0 0 0 0   1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
forward R=121: k1=0, k2=0  1 1 0 0 0 1 0 0 0 1 0 0 0 0   1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0
forward R=122: k1=0, k2=0  1 1 1 0 0 0 1 0 0 0 1 0 0 0   1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0
forward R=123: k1=1, k2=0  1 1 1 1 0 0 0 1 0 0 0 1 0 0   0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0
forward R=124: k1=0, k2=0  1 1 1 1 1 0 0 0 1 0 0 0 1 0   1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0
forward R=125: k1=0, k2=0  1 1 1 1 1 1 0 0 0 1 0 0 0 0   1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0
forward R=126: k1=0, k2=0  1 1 1 1 1 1 1 0 0 0 1 0 0 0   1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0
forward R=127: k1=0, k2=1  1 1 1 1 1 1 1 1 0 0 0 1 0 0   1 1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0

backward part:
backward R=131: k1=0, k2=1  0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
backward R=130: k1=1, k2=1  0 0 0 0 0 0 0 0 0 0 0 0 0 0   0 0 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1
backward R=129: k1=0, k2=0  0 0 0 1 0 1 0 1 1 0 0 0 1 0   0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1
backward R=128: k1=0, k2=0  0 0 1 0 1 0 1 1 0 0 0 1 0 0   0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0
backward R=127: k1=0, k2=0  0 1 0 1 0 1 1 0 0 0 1 0 0 0   1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0

The highest number of matching bits is obtained at round 127:
 forward R=127: k1=0, k2=1  1 1 1 1 1 1 1 1 0 0 0 1 0 0   1 1 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0
backward R=127: k1=0, k2=0  0 1 0 1 0 1 1 0 0 0 1 0 0 0   1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0
```

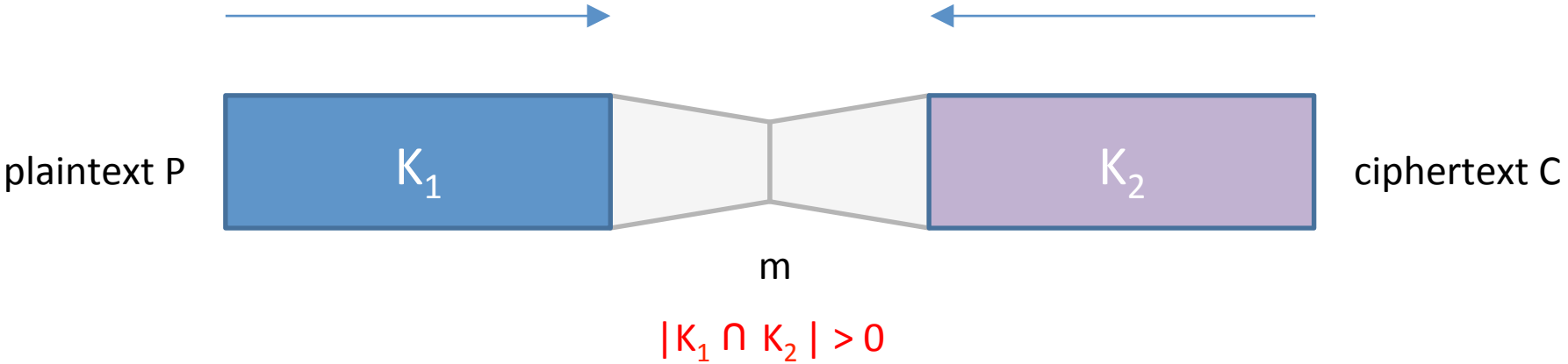
3-Subset MITM Cryptanalysis of KTANTAN

(A. Bogdanov, C. Rechberger in SAC'10)

Table 1. Results on MITM cryptanalysis for KTANTAN

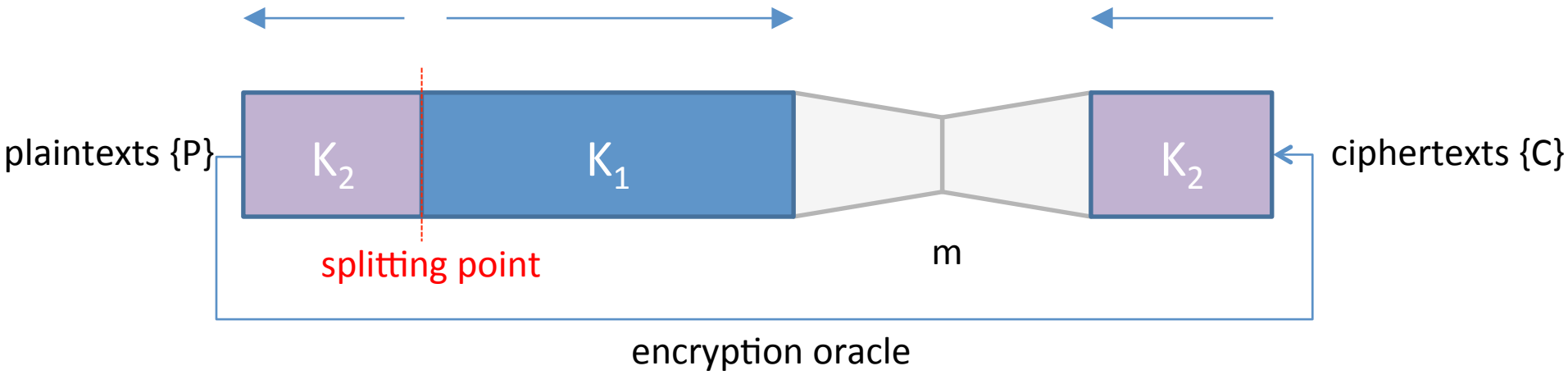
attack/bound	cipher $b \in \{32, 48, 64\}$	#rounds (of 254)	time [encryptions]	data compl. [PT/CT pars]
[9], RK diff. bound	KTANTANb	150	$\mathcal{O}(2^b)$	$\mathcal{O}(2^b)$
[9], DC and LC bound	KTANTANb	128	$\mathcal{O}(2^b)$	$\mathcal{O}(2^b)$
this paper, MITM attack	KTANTAN32	254	$2^{75.170}$	3
this paper, MITM attack	KTANTAN48	254	$2^{75.044}$	2
this paper, MITM attack	KTANTAN64	254	$2^{75.584}$	2

MITM with 3 subsets



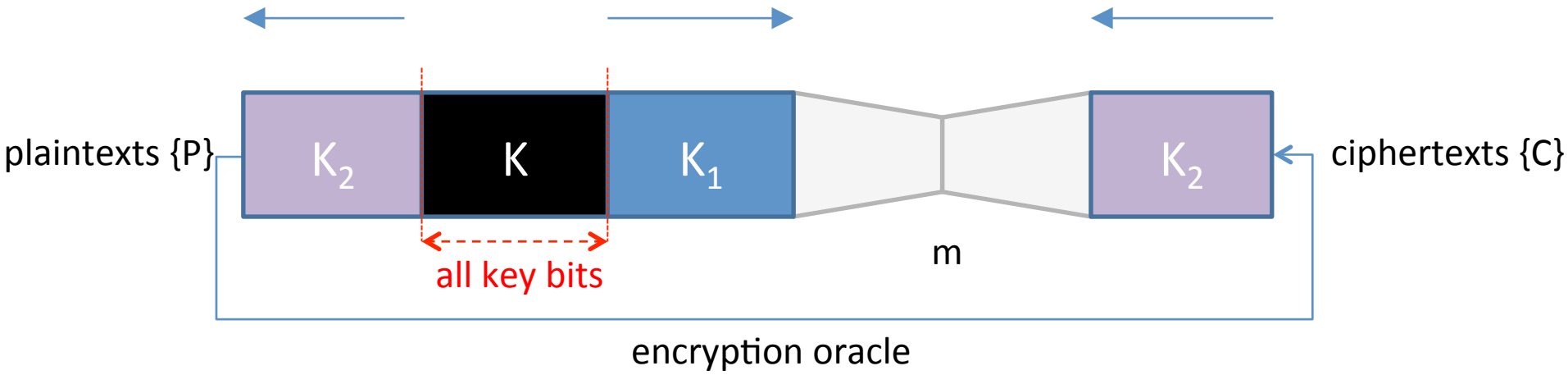
- 3 key spaces:
  - A_0 (both in K_1 and K_2)
  - A_1(K_1 only), A_2 (K_2 only)
- Guess A_0, then A_1 and A_2 independently
- Complexity: $2^{|A0|}(2^{|A1|}+2^{|A2|})+2^{|K|-m}$

MITM with splice and cut



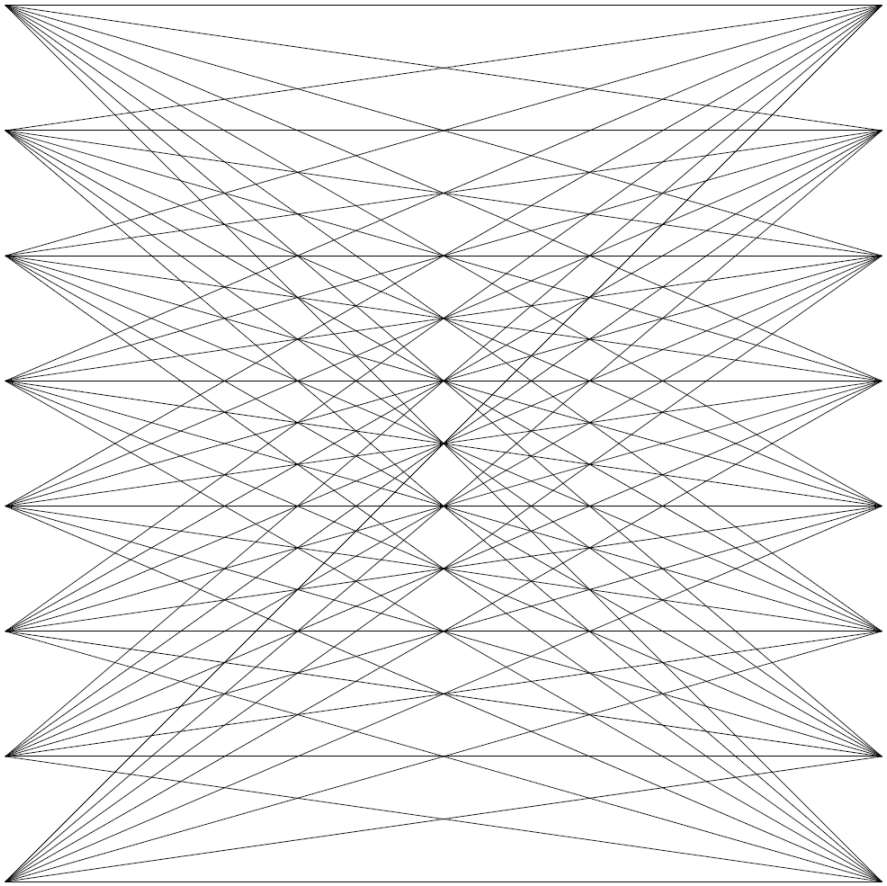
- Guess A_0, then A_1 and A_2 independently
- Compute forwards from splitting point
- Compute backwards from splitting point
  - Over encryption/decryption oracle!

MITM with bicliques



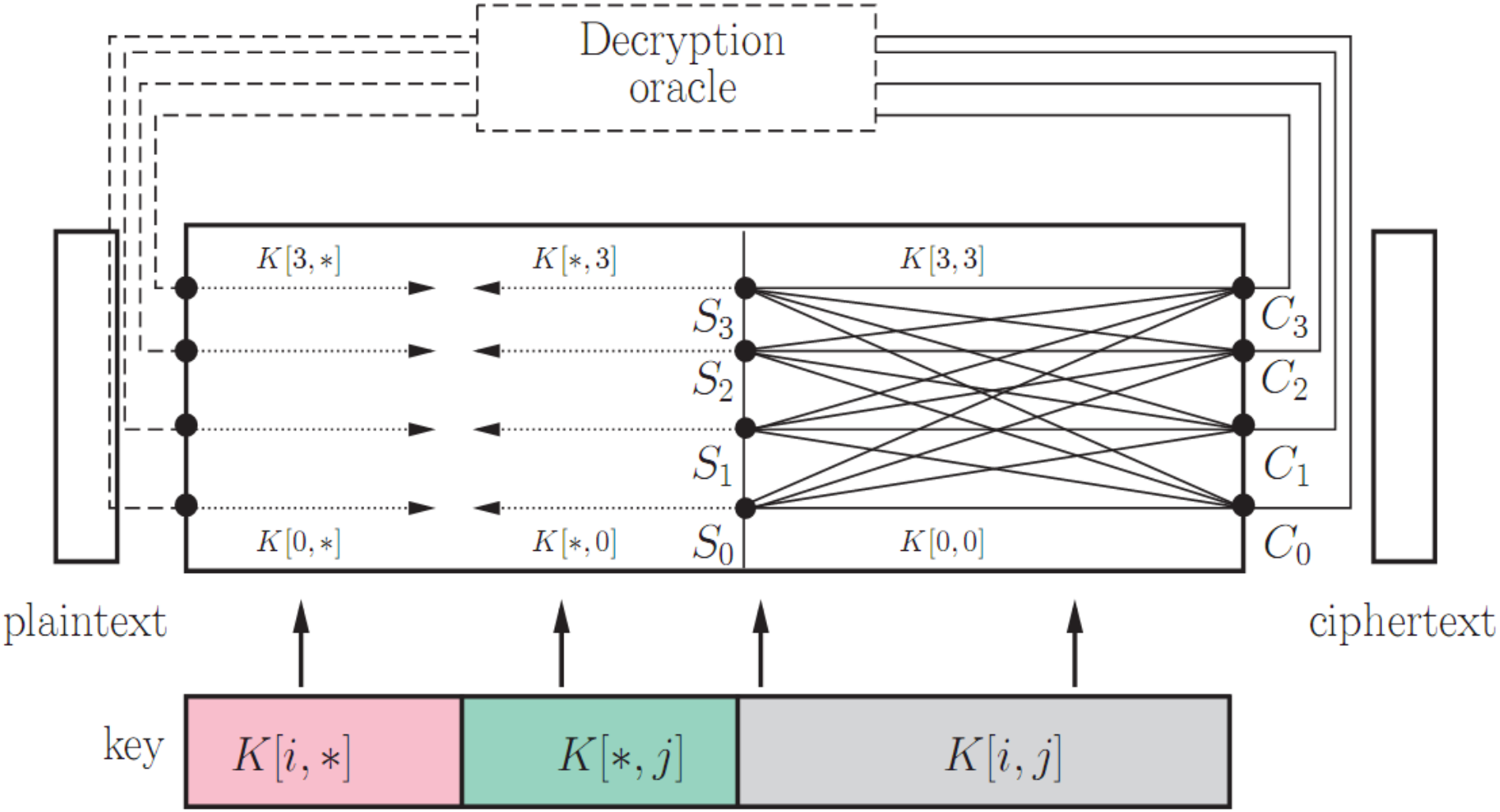
- Allow all key bits affect a part of the cipher
- Stick to a structure to enable efficient enumeration of keys and states in this part
- Structure = **biclique!**

Biclique:
Complete bipartite graph



Biclique of dimension 3:
$2 \cdot 2^3$ vertices and 2^6 edges

MITM with bicliques



10+ years of AES-128 cryptanalysis: Key recovery in single-key setting

FSE'00: Ferguson et al

10 years

ASIACRYPT'10: Dunkelman et al

rounds	data	workload	memory	method
7	$2^{127.997}$	2^{120}	2^{64}	Square
7	2^{32}	$2^{128-\epsilon}$	2^{100}	Square-functional
7	$2^{117.5}$	2^{123}	2^{109}	impossible
7	$2^{115.5}$	2^{119}	2^{45}	impossible
7	$2^{115.5}$	2^{119}	2^{109}	impossible
7	$2^{112.2}$	$2^{112} + 2^{117.2}$MA	2^{109}?	impossible
7	2^{80}	$2^{113}+2^{123}$ precomp.	2^{122}	MitM
7	$2^{106.2}$	$2^{107.1} + 2^{117.2}$MA	$2^{94.2}$	impossible
7	2^{103}	2^{116}	2^{116}	Square-multiset

After 10 years of cryptanalysis, still 7 rounds broken only!

MITM key recovery for AES

(A. Bogdanov, D. Khovratovich, C. Rechberger in Asiacrypt'11)

rounds	data	computations/succ.rate	memory	biclique length in rounds
AES-128 secret key recovery				
8	$2^{126.33}$	$2^{124.97}$	2^{102}	5
8	2^{127}	$2^{125.64}$	2^{32}	5
8	2^{88}	$2^{125.34}$	2^{8}	3
10	2^{88}	$2^{126.18}$	2^{8}	3
AES-192 secret key recovery				
9	2^{80}	$2^{188.8}$	2^{8}	4
12	2^{80}	$2^{189.74}$	2^{8}	4
AES-256 secret key recovery				
9	2^{120}	$2^{253.1}$	2^{8}	6
9	2^{120}	$2^{251.92}$	2^{8}	4
14	2^{40}	$2^{254.42}$	2^{8}	4

Preimage finding for AES hash modes

(A. Bogdanov, D. Khovratovich, C. Rechberger in Asiacrypt'11)

rounds	computations	succ.rate	memory	biclique length in rounds
AES-128 compression function preimage, Miyaguchi-Preneel				
10	$2^{125.83}$	0.632	2^8	3
AES-192 compression function preimage, Davies-Meyer				
12	$2^{125.71}$	0.632	2^8	4
AES-256 compression function preimage, Davies-Meyer				
14	$2^{126.35}$	0.632	2^8	4

Biclique MITM in the prospective

- Since its introduction and application to the AES, the technique of biclique MITM has been used by many others to attack further ciphers:
  - IDEA (KLR'12)
  - Piccolo (WWY'12, JKLSH'12)
  - TWINE (CKB'12)
  - HIGHT (HKK'11, KDH'13)
  - ARIA (CX'12)
  - SQUARE (M'11)
  - PRESENT (JKLSH'12)
  - LED (JKLSH'12)
  - LBlock (WWYZ'12, KDH'13)
  - …

Wrap-up for MITM

- Apparatus of meet-in-the-middle:
  - Partial matching, 3-subset, cut-and-splice, bicliques, …
- Bicliques yield some interesting (but still purely theoretical) results on AES
- Still some room for new techniques and improvements

Some references

- Andrey Bogdanov, Florian Mendel, Francesco Regazzoni, Vincent Rijmen and Elmar Tischhauser. *ALE: AES-Based Lightweight Authenticated Encryption.* FSE 2013
- Andrey Bogdanov, Amir Moradi, Tolga Yalcin. *Efficient and Side-Channel Resistant Authenticated Encryption of FPGA bitstreams.* ReConFig 2012
- Andrey Bogdanov, Dmitry Khovratovich, Christian Rechberger. *Biclique Cryptanalysis of the Full AES.* ASIACRYPT 2011
- Andrey Bogdanov, Christian Rechberger. *A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN.* SAC 2010.