The Evolution of Authenticated Encryption

Phillip Rogaway
University of California, Davis, USA

Those who've worked with me on AE:
Mihir Bellare
John Black
Ted Krovetz
Chanathip Namprempre
Tom Shrimpton
David Wagner

Workshop on Real-World Cryptography
Thursday, 10 January 2013
Stanford, California, USA

Traditional View (~2000) of Symmetric Goals

Sender ^{K} → **Receiver** ^{K}

Privacy
(confidentiality)

Authenticity
(data-origin authentication)

Encryption scheme

Authenticated Encryption
Achieve both of these aims

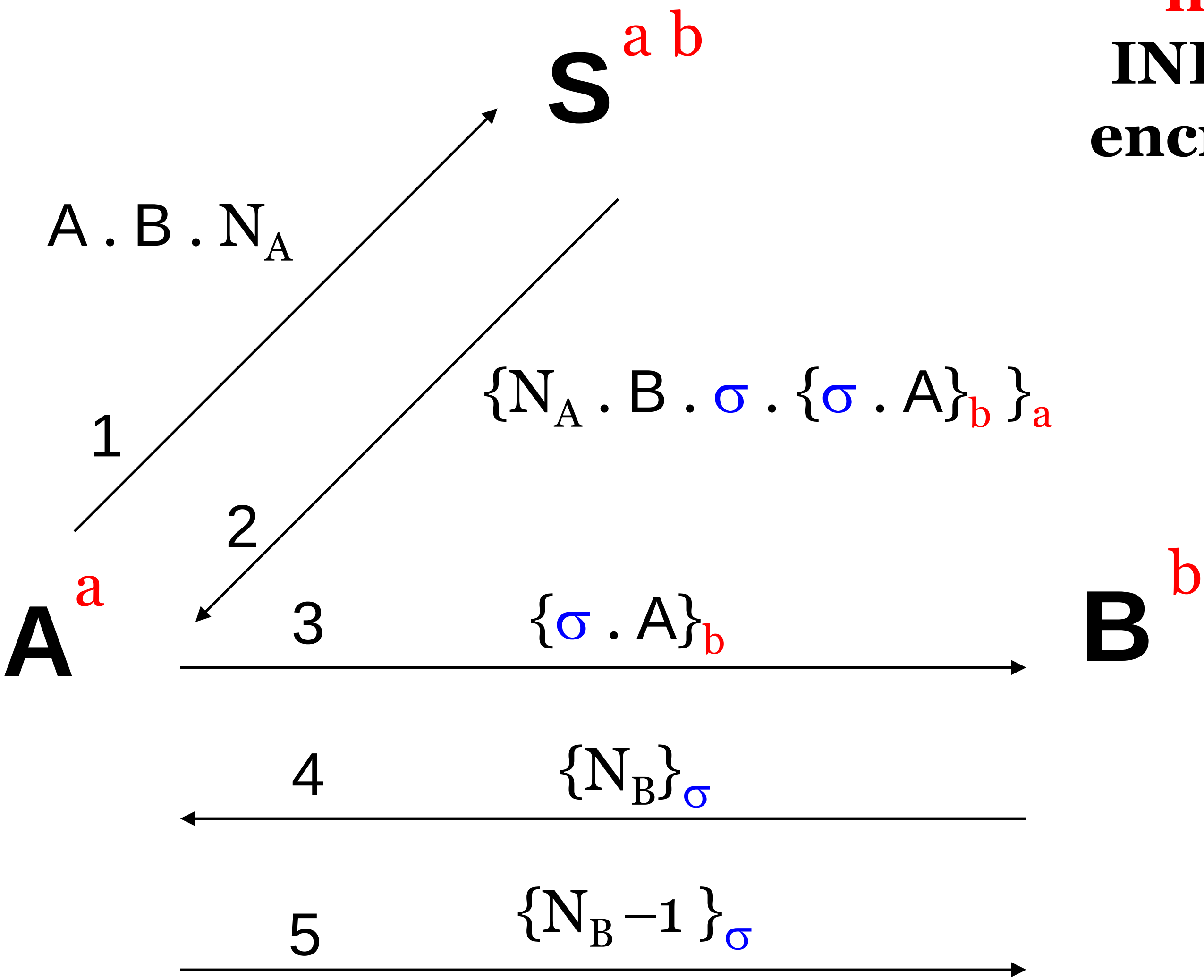
Message Authentication Code (MAC)

IND-CPA
[Goldwasser, Micali 1982]
[Bellare, Desai, Jokipii, R 1997]

Existential-unforgeability under ACMA
[Goldwasser, Micali, Rivest 1984, 1988],
[Bellare, Kilian, R 1994], [Bellare, Guerin, R 1995]

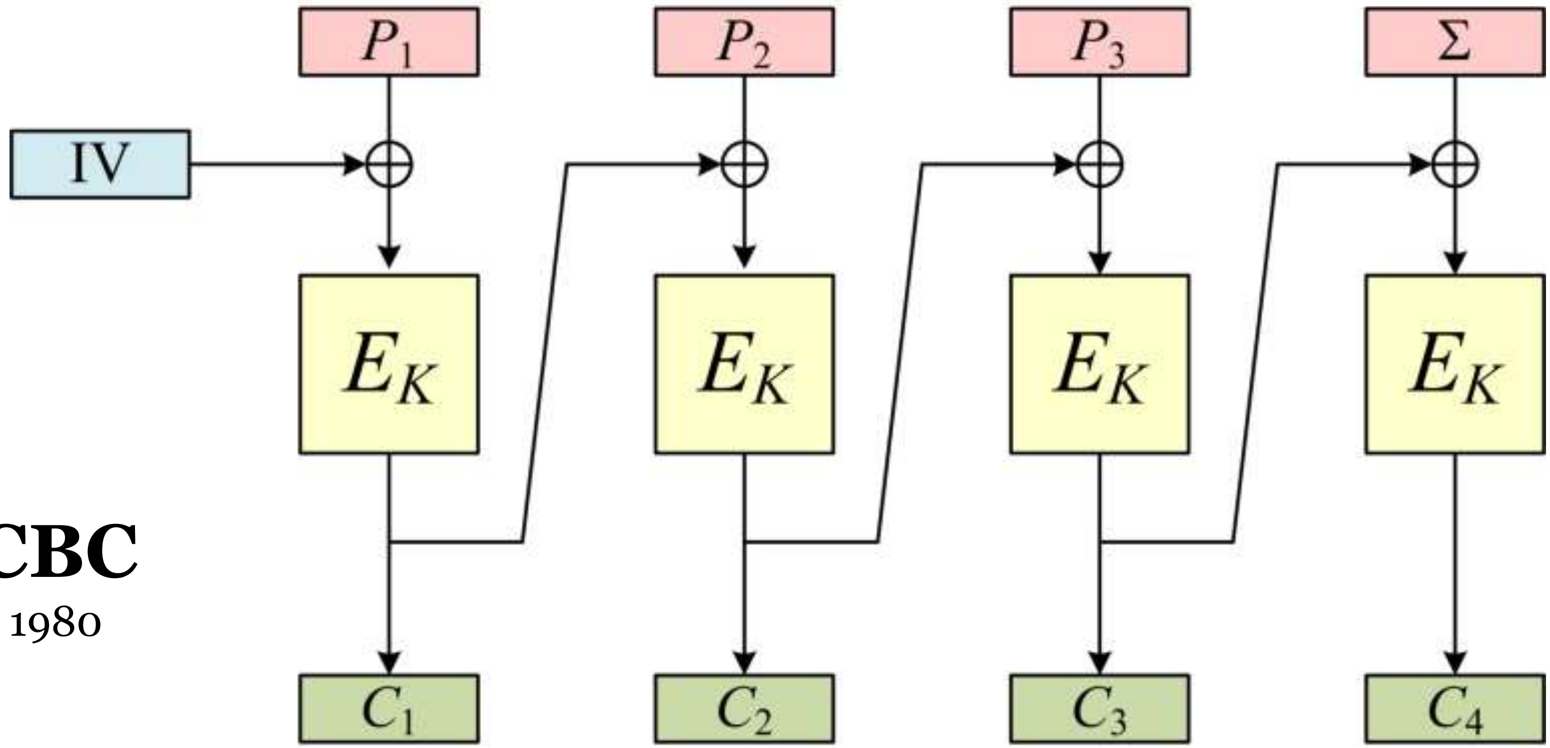
Needham-Schroeder Protocol (1978)

Attacked by Denning-Saco (1981)



Practioners never saw IND-CPA as encryption's goal

Add redundancy

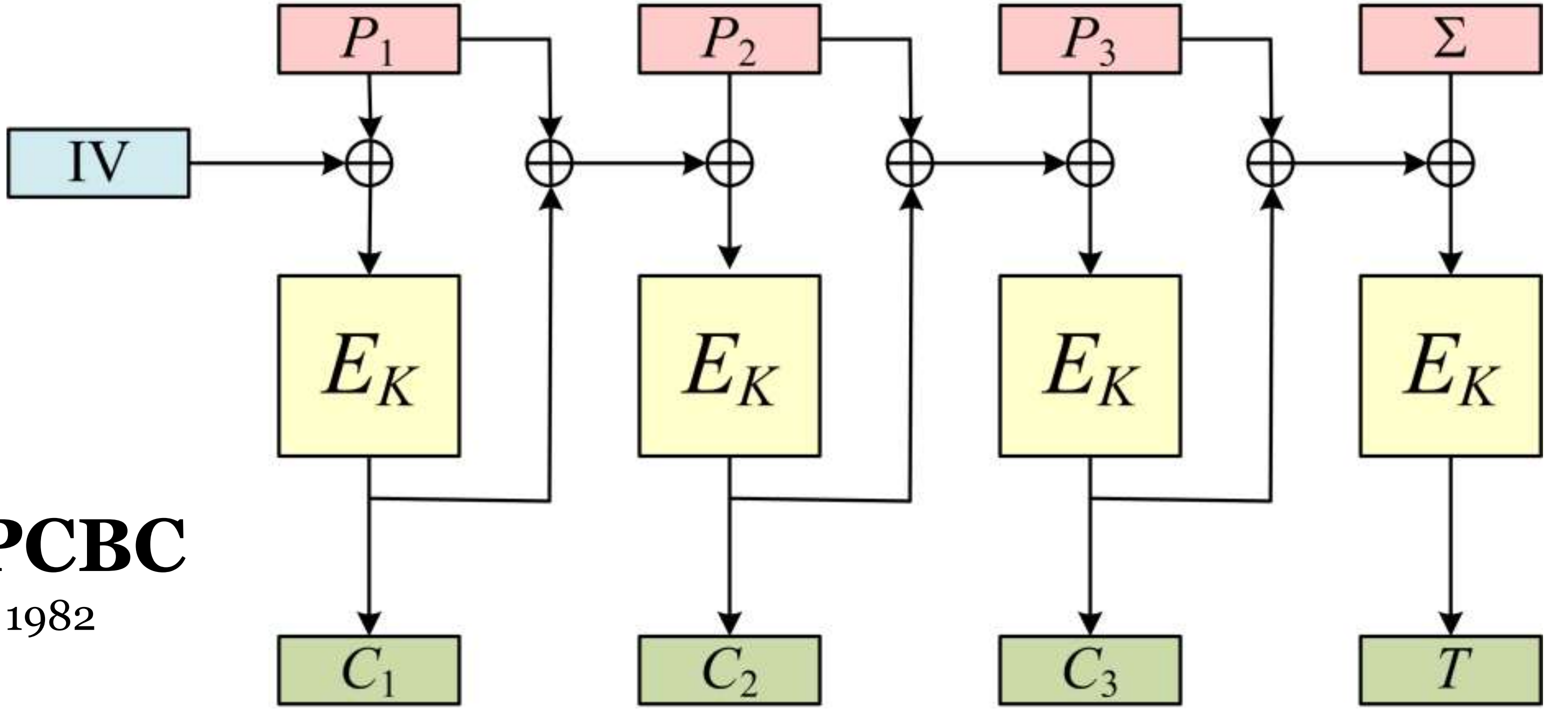


CBC
~ 1980

Doesn't work
regardless of how you compute
the (unkeyed) checksum $\Sigma = R(P_1, \ldots, P_n)$
(Wagner)

Beyond CBC MAC:
unkeyed checksums don't work even
with IND-CCA or NM-CPA schemes
[An, Bellare 2001]

Add more arrows



PCBC

≤ 1982

Doesn't work

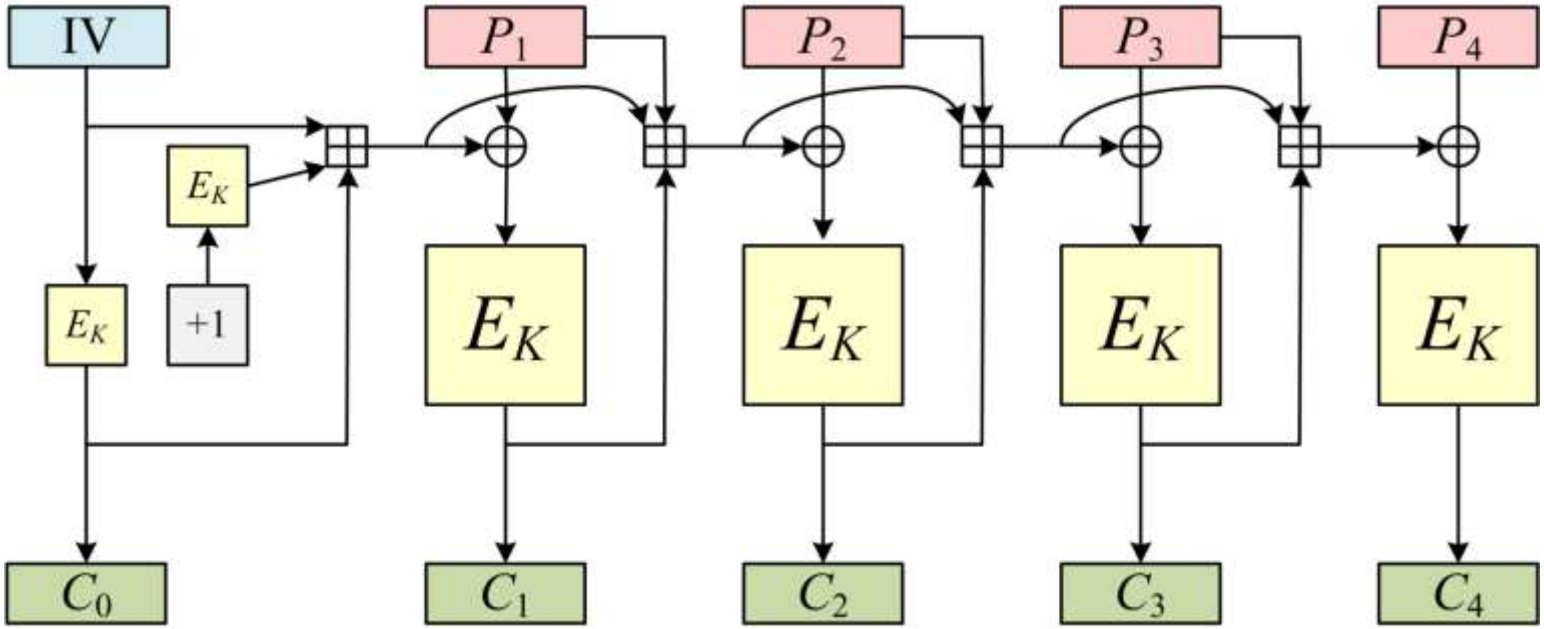
See [Yu, Hartman, Raeburn 2004]

The Perils of Unauthenticated Encryption: Kerberos Version 4

for real-world attacks

Add yet more stuff

iaPCBC
[Gligor, Donescu 1999]



Doesn't work
Promptly broken by Jutla (1999)
& Ferguson, Whiting, Kelsey, Wagner (1999)

Emerging understanding that:

- We'd **like** to get authenticity as an adjunct to privacy
- **Ad hoc** ways to try to get it cheaply **don't work**

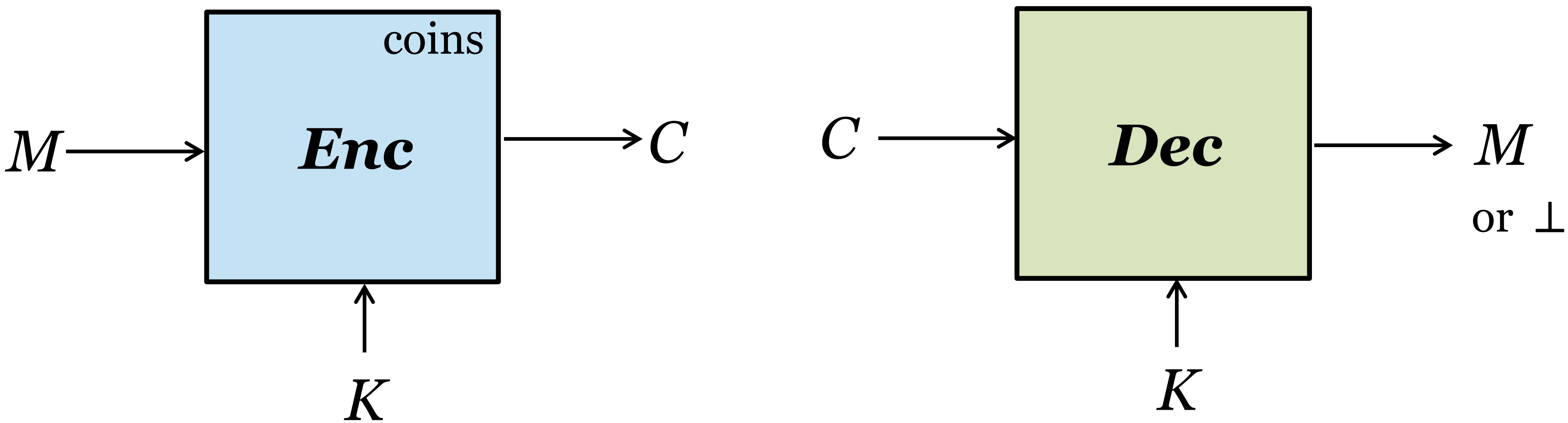
~2000

Similar realization, earlier, in the PK world

- **[Bleichenbacher 1998]** – Attack on PKCS #1
- Reaction: IND-CPA security **not enough**
  - **CCA1** security [Naor-Yung 1990]
  - **CCA2** security [Rackoff-Simon 1991]
  - **Non-malleability** [Dolev-Dwork-Naor 1991]
- **Signcryption** [Zheng 1997] (very different motivation)

AE Defined

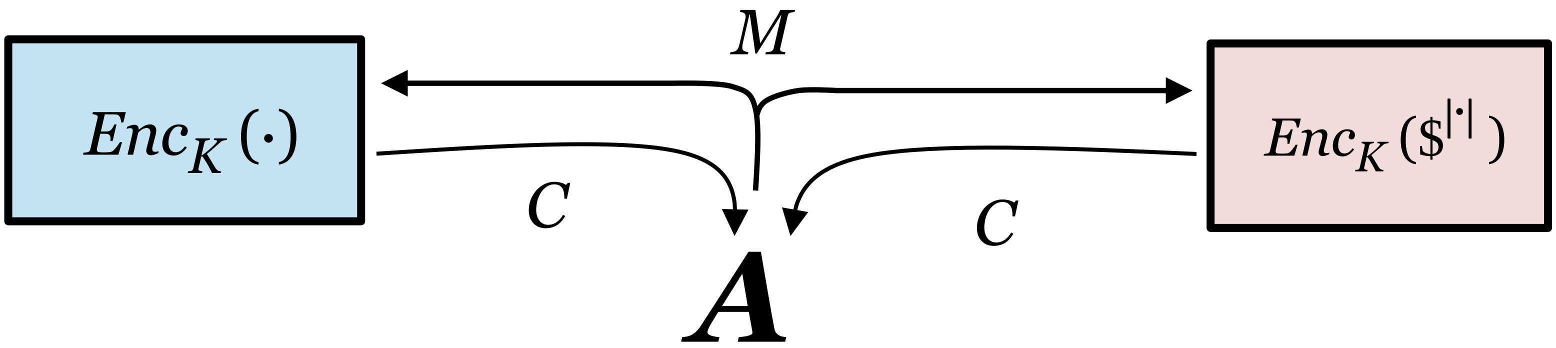
[**Bellare, R 2000**] – "Encode-then-encipher encryption: how to exploit nonces or redundancy in plaintexts for efficient cryptography"
[**Katz, Yung 2000**] – "Unforgeable encryption and chosen ciphertext secure modes of operation"



1. **Privacy** IND-CPA, as defined in [BDJR97]: IND-CPA
2. **Authenticity** The only ciphertexts C an adversary can name that will decrypt to an $M \neq \perp$ are those obtained by an ***Enc***$(\cdot)$ call

Integrity of ciphertexts ← **[Bellare Namprempre 2000]** "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm"

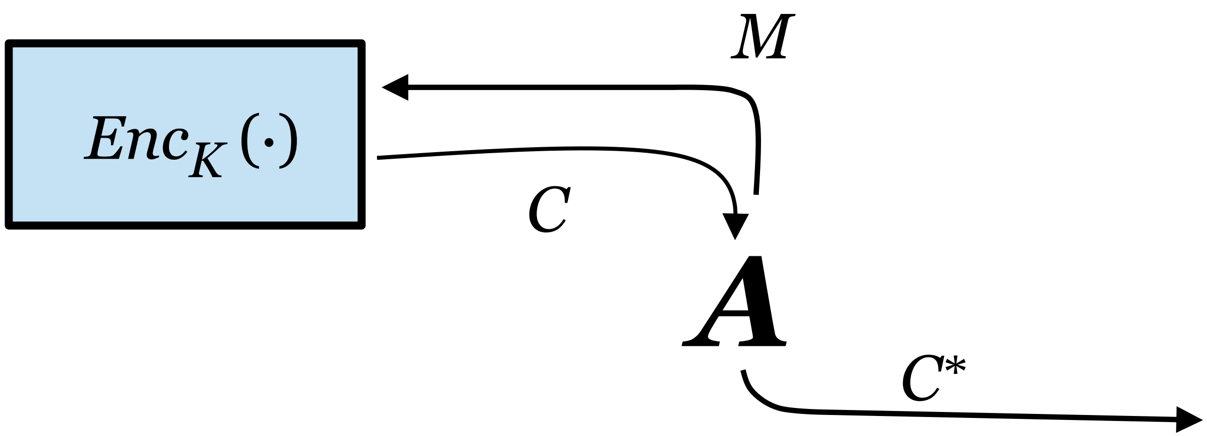
AE Defined



[Bellare, Desai, Jokipii, R 1997]

$$\mathbf{Adv}_{\Pi}^{\text{priv}}(\boldsymbol{A}) = \Pr[\boldsymbol{A}^{Enc_K(\cdot)} \rightarrow 1] - \Pr[\boldsymbol{A}^{Enc_K(\$^{|\cdot|})} \rightarrow 1]$$

AE Defined



[Bellare, Desai, Jokipii, R 1997]

$$\mathbf{Adv}^{\text{priv}}_{\Pi}(\boldsymbol{A}) = \Pr[\boldsymbol{A}^{Enc_K(\cdot)} \rightarrow 1] - \Pr[\boldsymbol{A}^{Enc_K(\$^{|\cdot|})} \rightarrow 1]$$

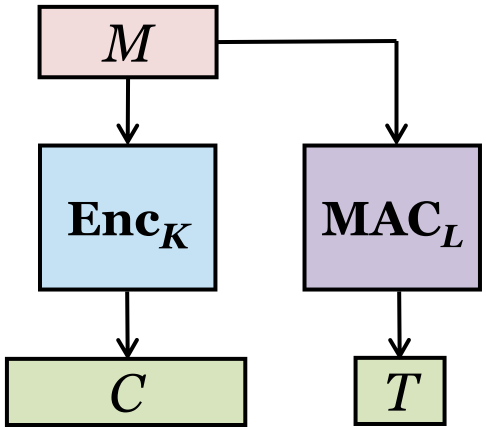
$$\mathbf{Adv}^{\text{auth}}_{\Pi}(\boldsymbol{A}) = \Pr[\boldsymbol{A}^{Enc_K(\cdot)} \rightarrow C^*\text{: no query returned } C^* \text{ and } Dec_K(C^*) \neq \perp]$$

[Bellare, R 2000]
[Katz, Yung 2000]

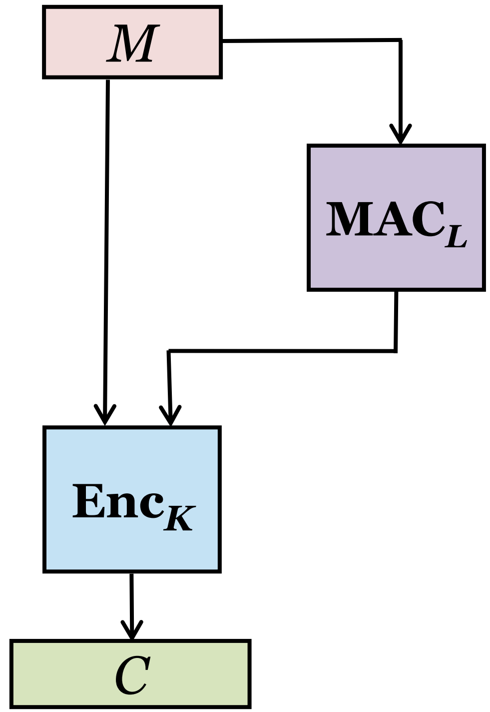
Generic Composition

of an IND-CPA encryption scheme and a PRF

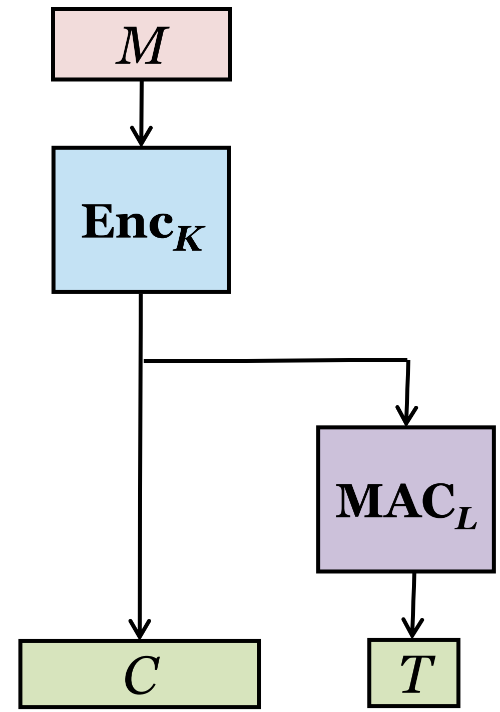
[Bellare, Namprempre 2000]



Encrypt-and-MAC ✗



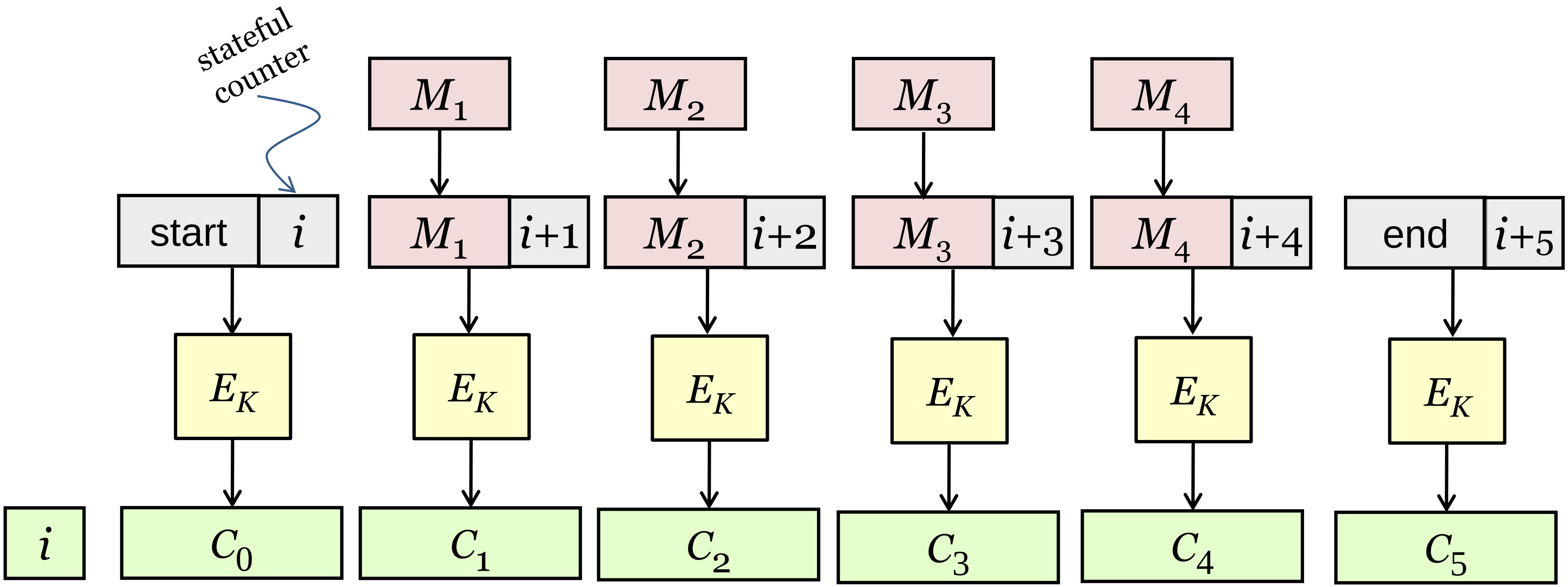
MAC-then-Encrypt ✗



Encrypt-then-MAC ✓

RPC Mode

[Katz, Yung 2000]



- Blockcipher-based AE using ~1.33 m + 2 calls
- Fully parallelizable

IAPM Mode

[Jutla 2001]
Encryption Modes with
Almost Free Message Integrity

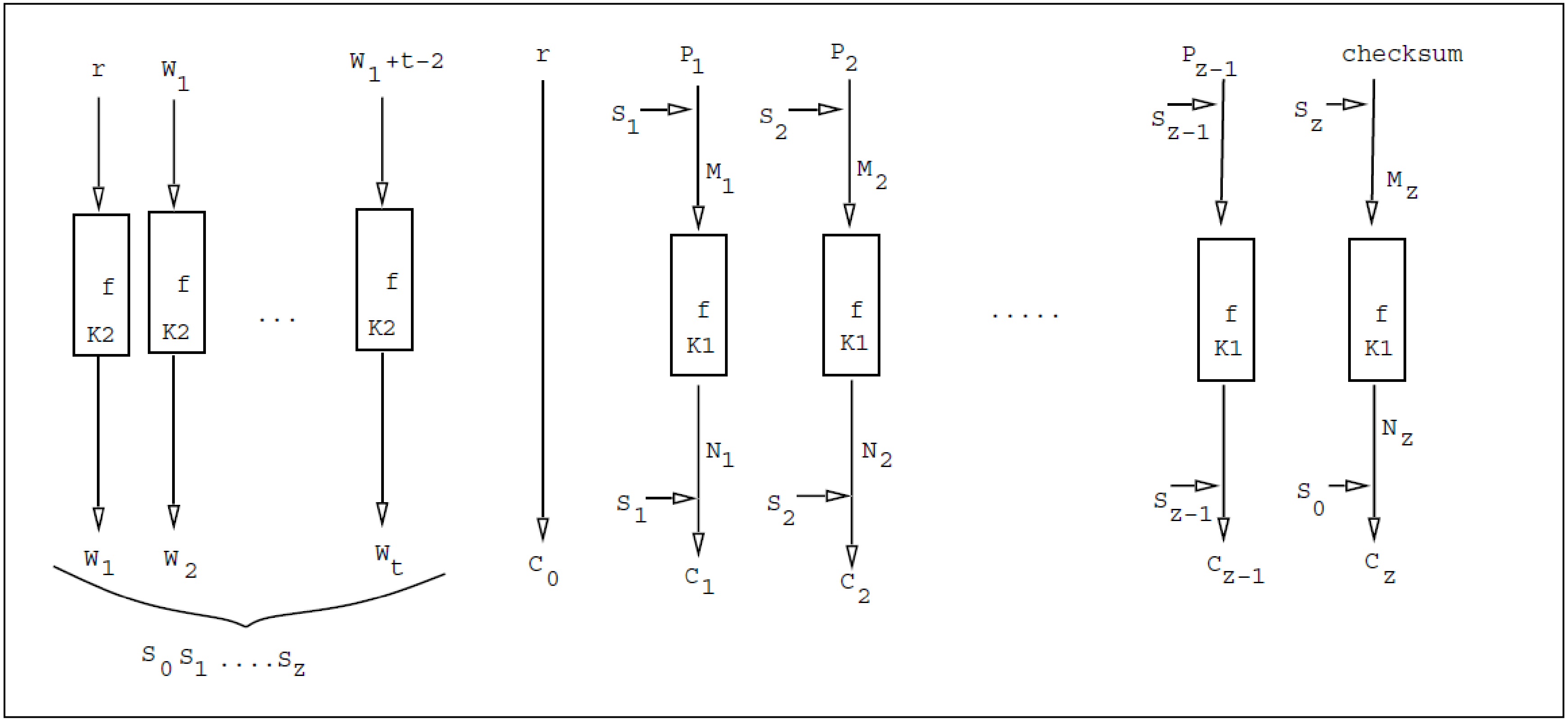


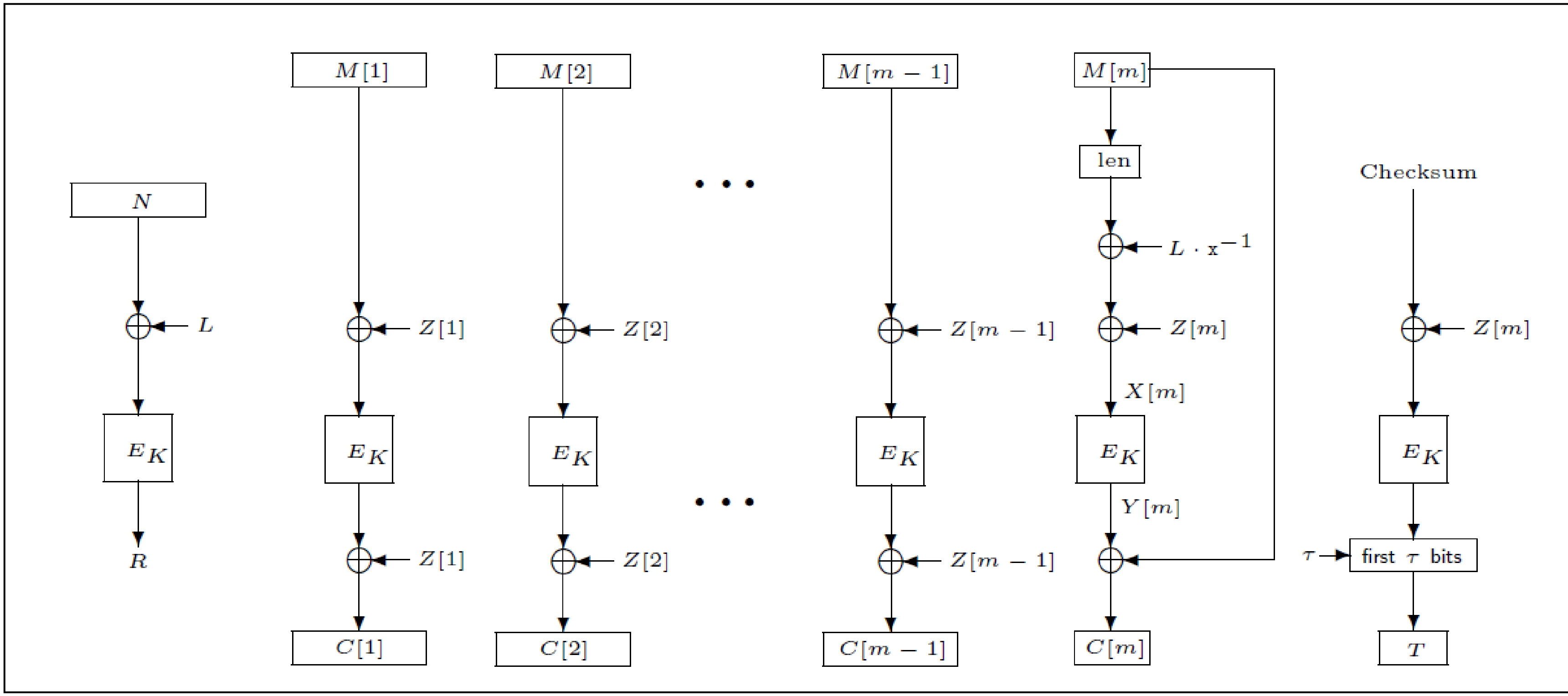
Illustration from
[Jutla 2001]

[Gligor, Donescu 2001]
for many other AE designs

- Blockcipher-based AE using $m + 1$ calls
- Fully parallelizable
- Plaintext a multiple of blocksize. Padding will up $|C|$
- ~ lg m_{max} additional calls for key setup
- Multiple blockcipher keys
- Need for random r

OCB Mode (later "OCB1")

[R, Bellare, Black, Krovetz 2001]



$Z[i] = R \oplus \gamma_i \cdot L$

Checksum = $M[1] \oplus \cdots \oplus M[m\text{-}1] \oplus C[m]0^* \oplus Y[m]$

- Arbitrary-length messages; no padding
- Efficient offset calculations
- Single blockcipher key
- Cheap key setup (one blockcipher call)
- m + 2 blockcipher calls

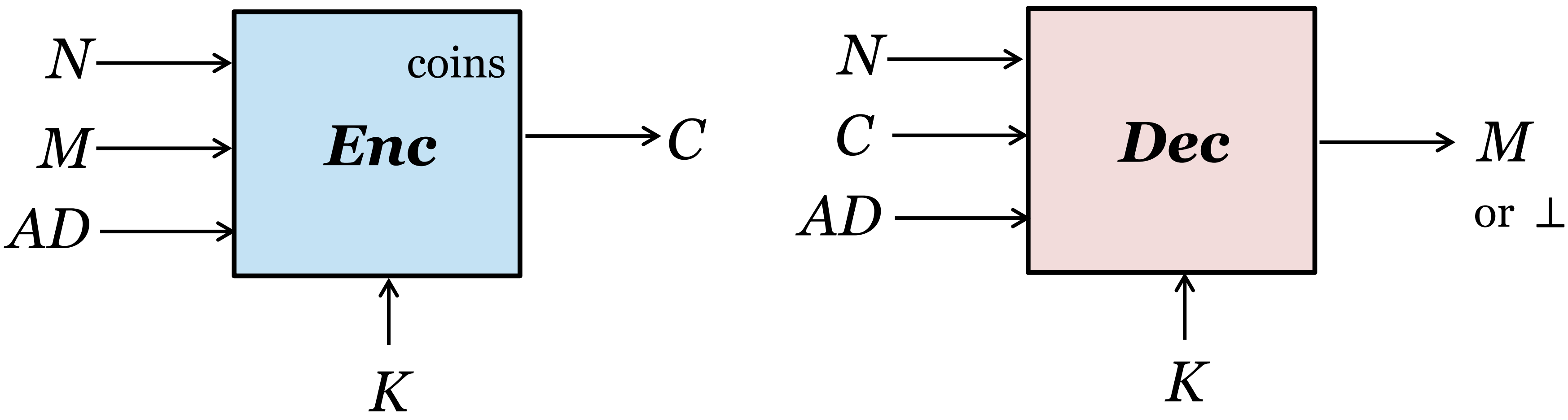
Urgent Real-World Need for AE

- **802.11** standard ratified in 1999
  Uses **WEP** security – RC4 with a CRC-32 checksum for integrity

- **Fatal attacks** soon emerge:
  - [Fluhrer, Mantin, Shamir 2001]
    Weaknesses in the key scheduling algorithm of RC4
  - [Stubblefield, Ioannidis, Rubin 2001]
    Using the Fluhrer, Mantin, Shamir attack to break WEP
  - [Borisov, Goldberg, Wagner 2001]
    Intercepting mobile communications: the insecurity of 802.11
  - [Cam-Winget, Housley, Wagner, Walker 2003]
    Security flaws in 802.11 data links protocols

- **WEP → WPA (uses TKIP) → WPA2 (uses CCM)**
  - Draft solutions based on OCB
  - Politics +patent-avoidance:
    CCM developed **[Whiting, Housley, Ferguson 2002]**
  - Standardized in **IEEE 802.11** – then **NIST**

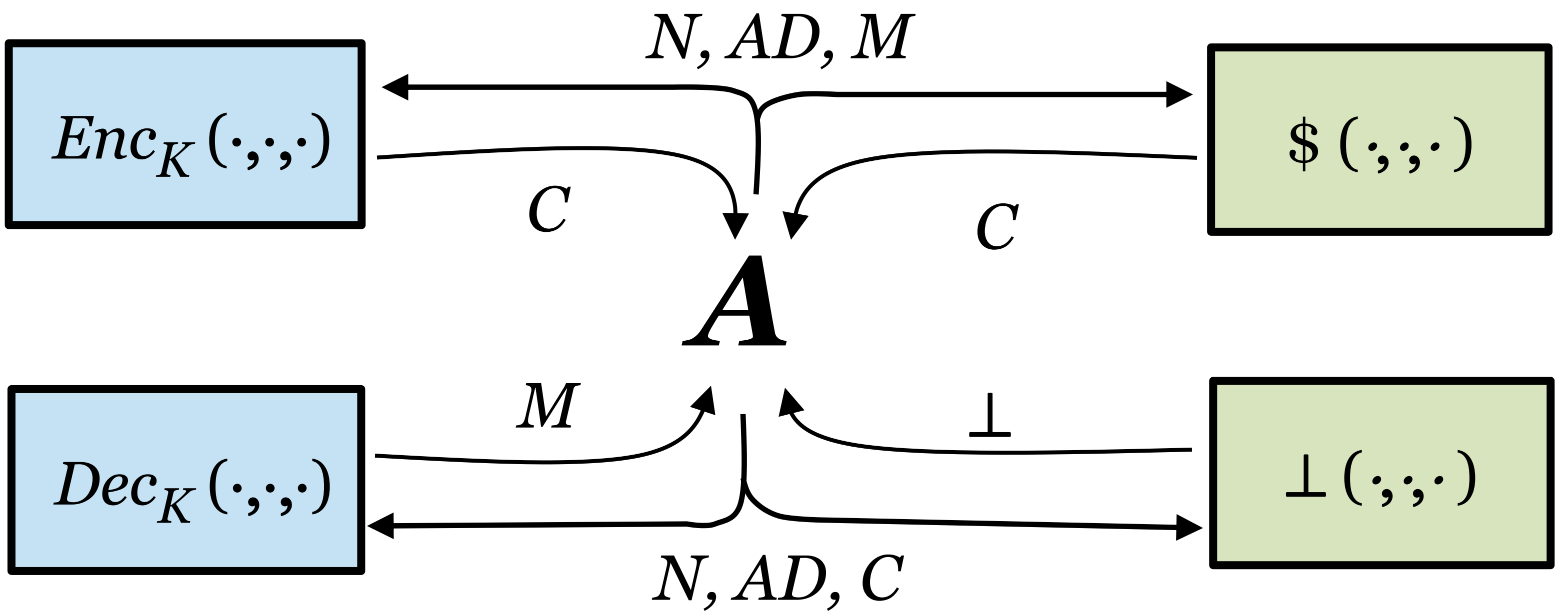
Definitional Issues



1) Move the coins "out" and make *Enc* deterministic **[RBBK01]**

2) Add in "associated data" **[R02]**

AEAD

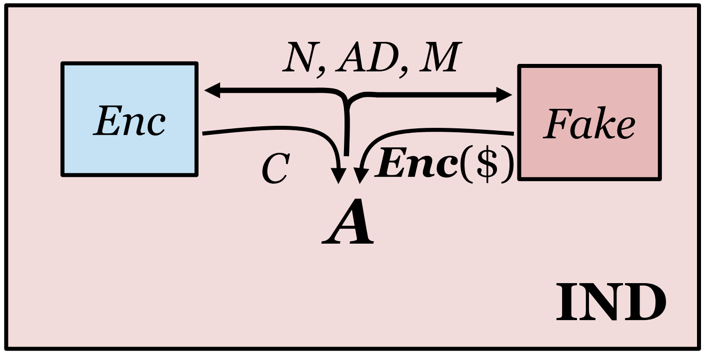


$$\mathbf{Adv}^{\mathrm{aead}}_{\Pi}(A) = \Pr[A^{Enc_K\ Dec_K} \to 1] - \Pr[A^{\$\ \bot} \to 1]$$

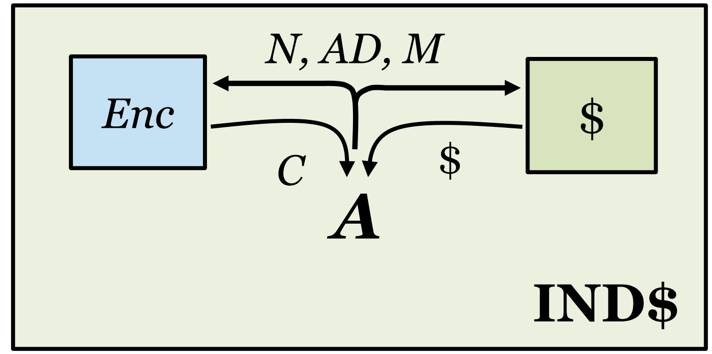
A may **not**

- Repeat an N in an enc query
- Ask a dec query (N, AD, C) after C is returned by an (N, AD, $\cdot$) enc query

IND vs. IND$



vs.



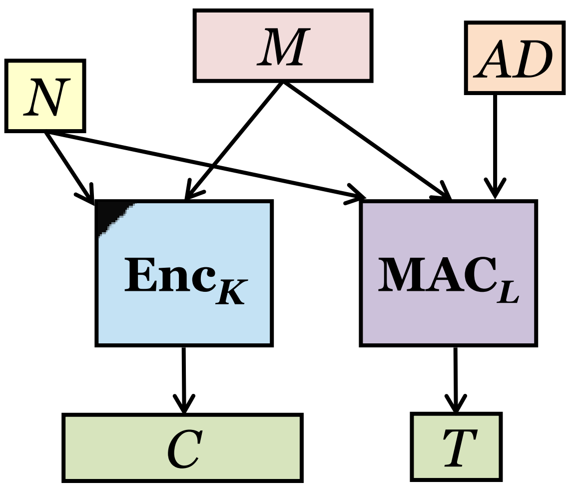
- Overshooting the "right" goal ✗
- *Easier* to prove schemes meet
- Tightly implies other notion
- Conceptually simpler
- Gives you more

Anonymity ←
which-key concealing

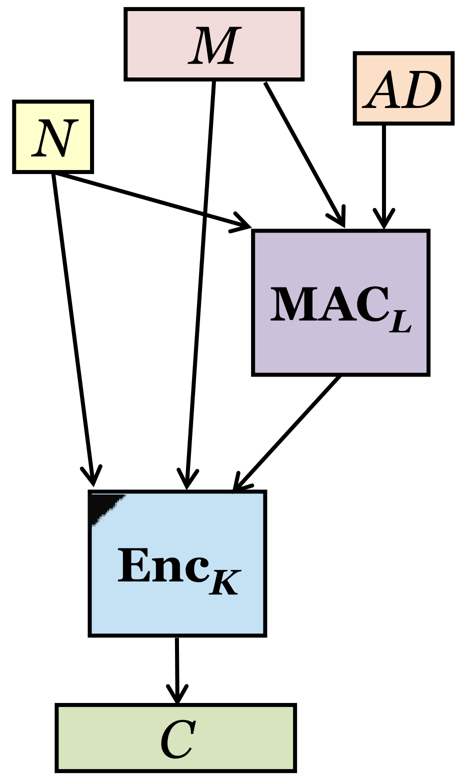
A names i;
- real: use K_i
- fake: use K

IND $\not\Rightarrow$ anonymity $\Leftarrow$ IND$

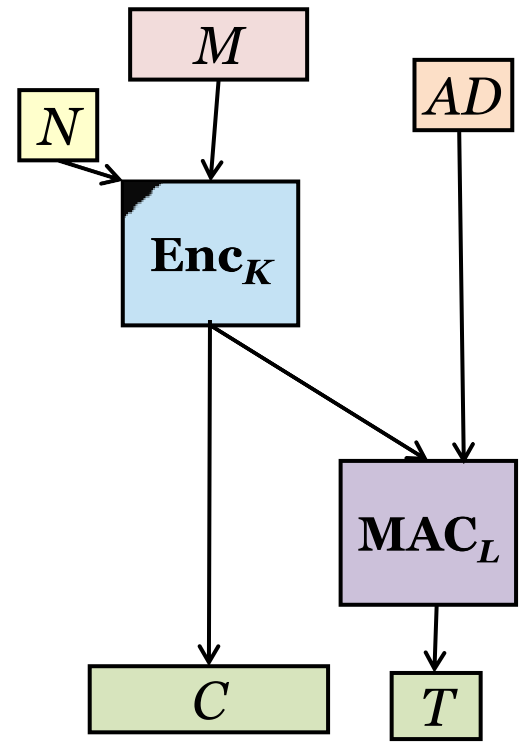
Nonce-Based Generic Composition



Encrypt-and-MAC ✓

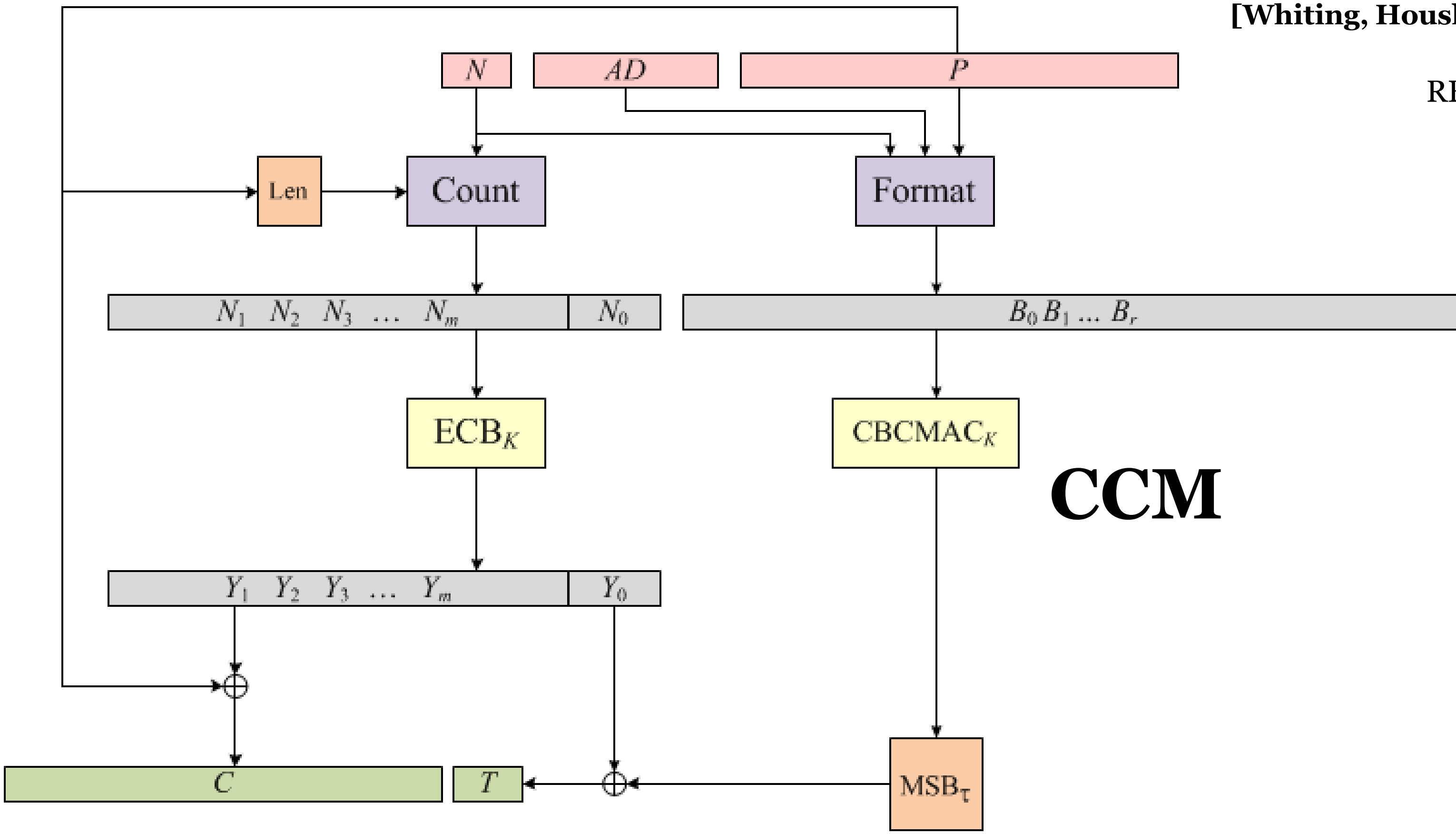


MAC-then-Encrypt ✓



Encrypt-then-MAC ✓

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



CCM

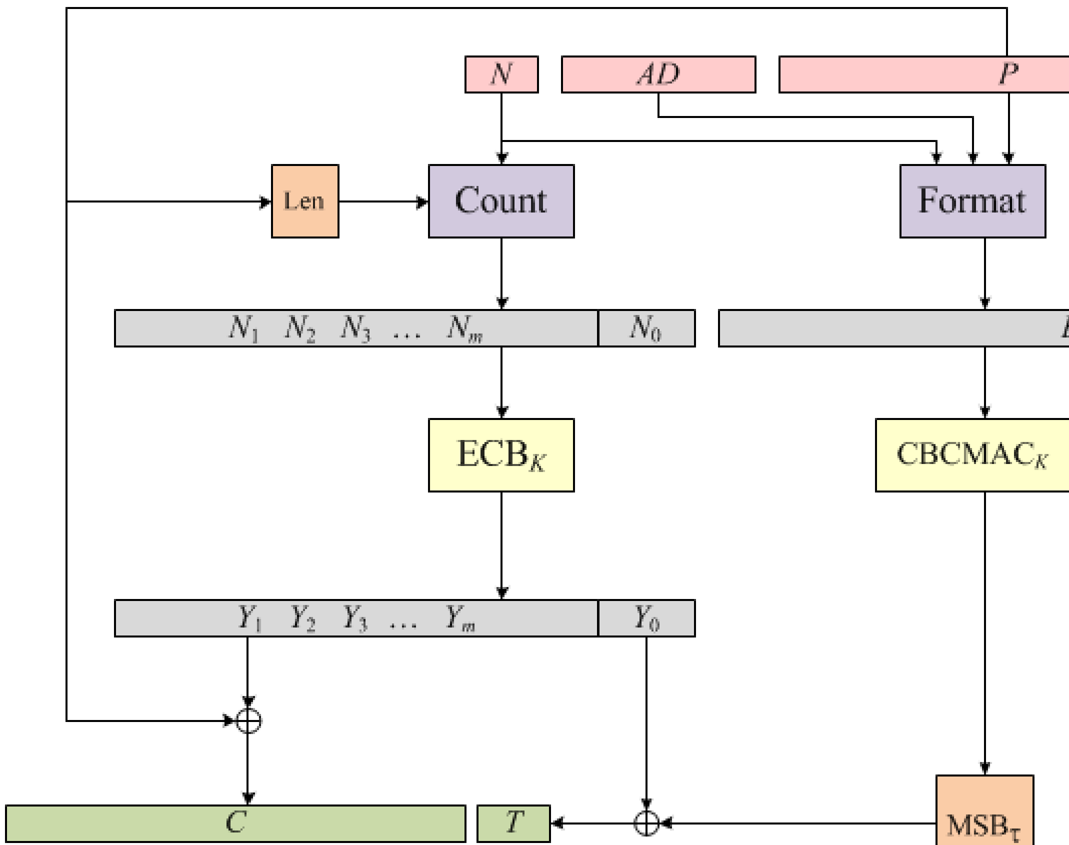
Functions $\textsc{Count}$ and $\textsc{Format}$

$$\textsc{Count}_q(N, m) = N_1 \parallel N_2 \parallel \cdots \parallel N_m$$
$$N_i = 0^5 \parallel [\,q-1\,]_3 \parallel N \parallel [\,i\,]_{8q}$$

$$\textsc{Format}_{q,t}(N, A, P) =$$
$$0 \parallel \textbf{if } A = \varepsilon \textbf{ then } 0 \textbf{ else } 1 \textbf{ endif} \parallel [\,t/2-1\,]_3 \parallel [\,q-1\,]_3 \parallel$$
$$N \parallel [\,|P|_8\,]_{8q} \parallel$$
$$\textbf{if } A = \varepsilon \textbf{ then } \varepsilon \textbf{ elseif}$$
$$|A|_8 < 2^{16} - 2^8 \textbf{ then } [\,|A|_8\,]_{16}$$
$$\textbf{elseif } |A|_8 < 2^{32} \textbf{ then } \texttt{0xFFFE} \parallel [\,|A|_8\,]_{32} \textbf{ else } \texttt{0xFFFF} \parallel [\,|A|_8\,]_{64} \textbf{ endif} \parallel$$
$$A \parallel$$
$$\textbf{if } A = \varepsilon \textbf{ then } \varepsilon \textbf{ elseif } |A|_8 < 2^{16} - 2^8 \textbf{ then } (\texttt{0x00})^{(14-|A|_8) \bmod 16}$$
$$\textbf{elseif } |A|_8 < 2^{32} \textbf{ then } (\texttt{0x00})^{(10-|A|_8) \bmod 16} \textbf{ else } (\texttt{0x00})^{(6-|A|_8) \bmod 16} \textbf{ endif} \parallel$$
$$P \parallel$$
$$(\texttt{0x00})^{(-|M|_8) \bmod 16}$$

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

[R, Wagner 2003]
"A Critique of CCM"



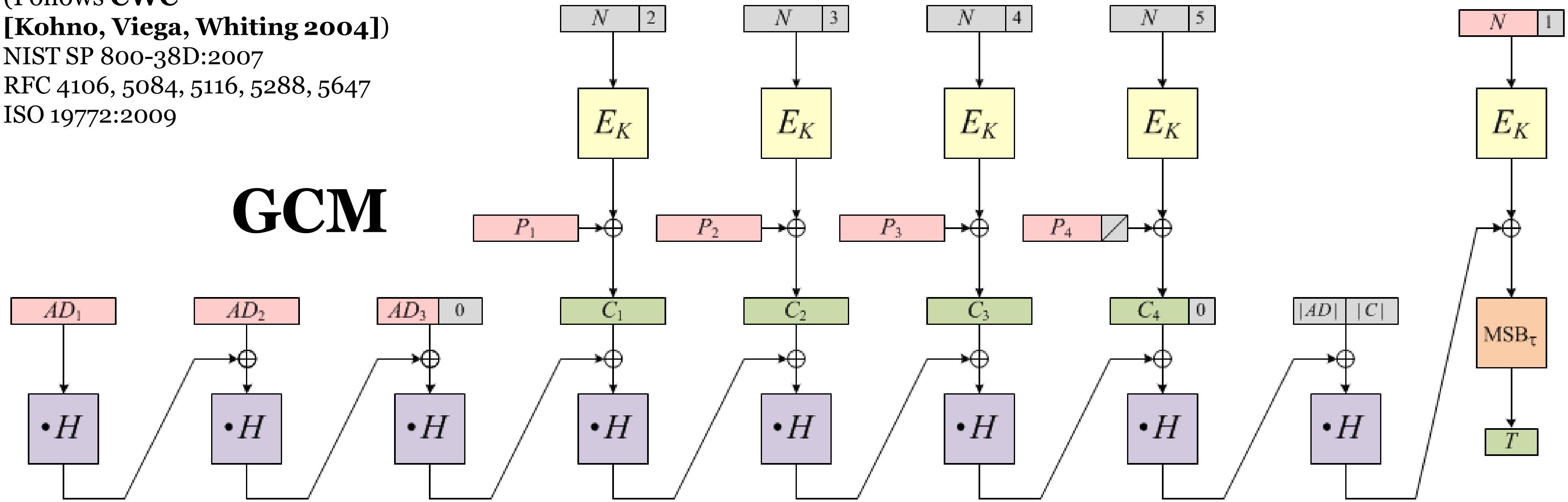
CCM

- About $2m+2$ blockcipher calls
- Half non-parallelizable
- Word alignment disrupted
- Can't preprocess static AD
- Not online
- Parameter $q \in \{2,3,4,5,6,7,8\}$, byte length of byte length of longest message, determines nonce length of $\tau = 15 - q$

- Provably secure **[Jonsson 2002]**
- Widely standardized & used
- Simple to implement
- Only forward direction of cipher used

[McGrew, Viega 2004]
(Follows **CWC**
[Kohno, Viega, Whiting 2004])
NIST SP 800-38D:2007
RFC 4106, 5084, 5116, 5288, 5647
ISO 19772:2009

GCM



- Provably secure
- Widely standardized & used
- Parallelizable, online
- About m+1 blockcipher calls

- Efficient in HW
- Good in SW with AES-NI, PCMULDQ, or tables
- Static *AD* can be preprocessed
- Only forward direction of blockcipher used

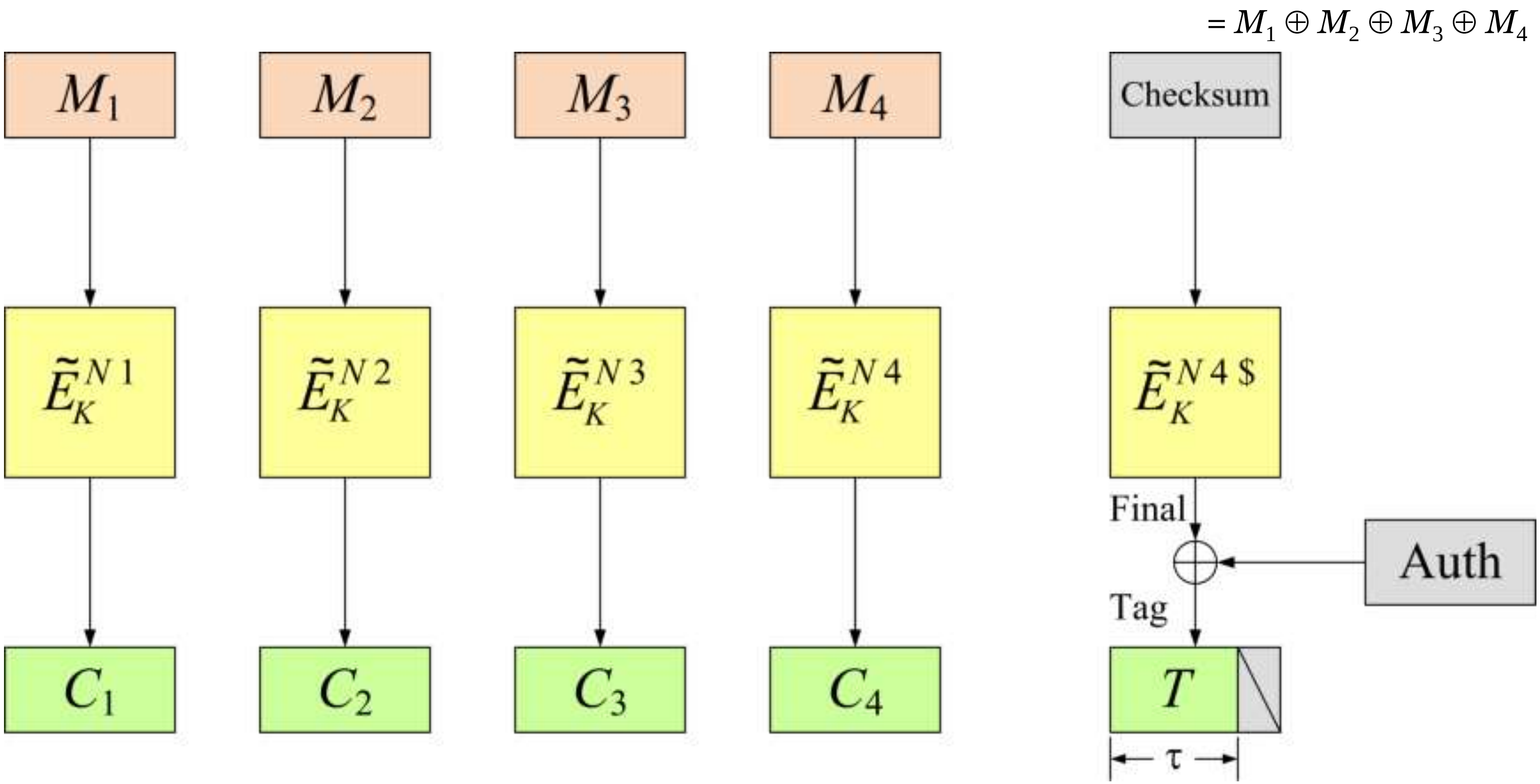
- Poor key agility (table-based implementation)
- Can't use short tags **[Ferguson 05]**
- Not so good in SW
- Timing attacks? (if table-based)

- "Reflected-bit" convention
- $|N|\neq 96$ not handled well
- Published proof buggy **[Iwata, 2012]**

In terms of tweakable blockcipher **[LRW02]**

OCB

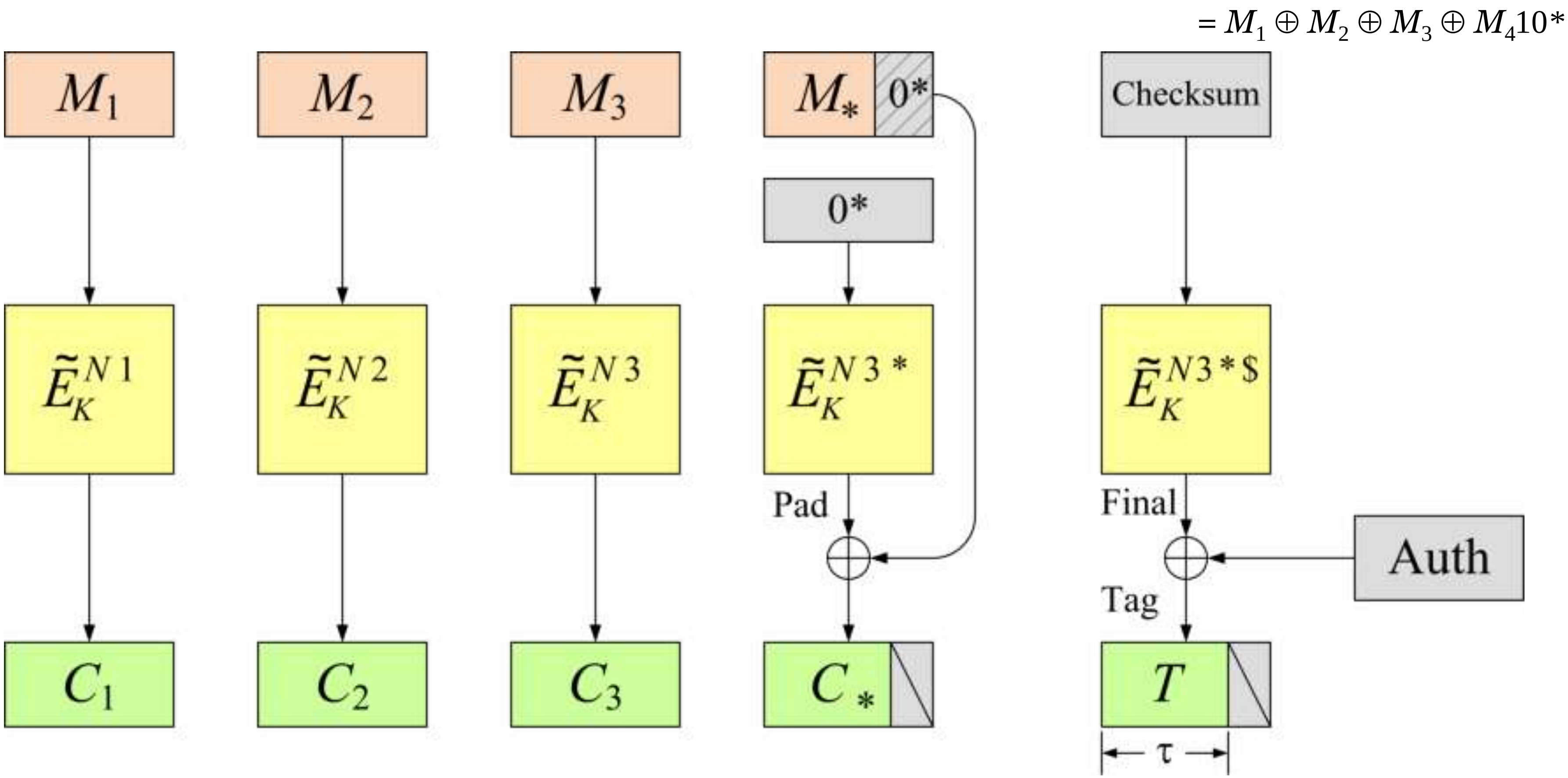
[KR11], following **[RBBK01,LRW02,R04]**



In terms of tweakable blockcipher **[LRW02]**

OCB

[KR11], following **[RBBK01,LRW02,R04]**



Making the Tweakable Blockcipher

$$\widetilde{E}_K^{N\,i}(X) = E_K(X \oplus \Delta) \oplus \Delta \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i L$$

$$\widetilde{E}_K^{N\,i\,*}(X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i^{*} L$$

$$\widetilde{E}_K^{N\,i\,\$}(X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i^{\$} L$$

$$\widetilde{E}_K^{N\,i\,*\,\$}(X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \text{Initial} + \lambda_i^{*\$} L$$

$$\widetilde{E}_K^{i}(X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \lambda_i L$$

$$\widetilde{E}_K^{i\,*}(X) = E_K(X \oplus \Delta) \quad \text{with} \quad \Delta = \lambda_i^{*} L$$

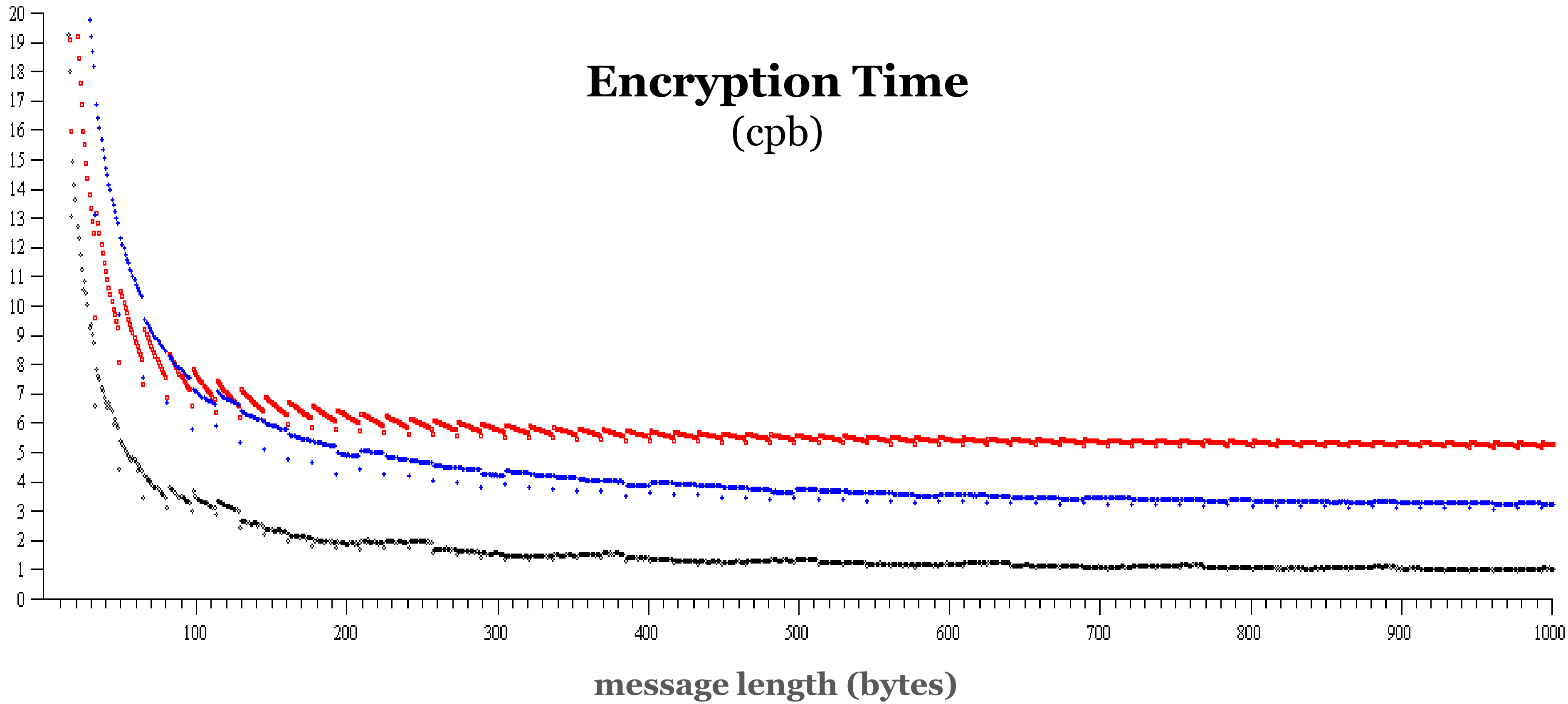
$$\text{Nonce} = 0^{127-|N|}\,1\,N$$
$$\text{Top} = \text{Nonce}\ \&\ 1^{122}\,0^{6}$$
$$\text{Bottom} = \text{Nonce}\ \&\ 1^{122}\,1^{6}$$
$$\text{Ktop} = E_K(\text{Top})$$
$$\text{Stretch} = \text{Ktop} \parallel (\text{Ktop} \oplus (\text{Ktop} \ll 8))$$
$$\text{Initial} = (\text{Stretch} \ll \text{Bottom})\,[1..128]$$

$$L = E_K(0^{128})$$
$$\lambda_i = 4\,a(i)$$
$$\lambda_i^{*} = 4\,a(i)+1$$
$$\lambda_i^{\$} = 4\,a(i)+2$$
$$\lambda_i^{*\$} = 4\,a(i)+3$$

$$a(0) = 0$$
$$a(i) = a(i-1) \oplus 2^{\mathbf{ntz}\,(i)}$$

Software Performance
Intel Core x86 i7 – "Sandy Bridge"
64-bit OS, using AES/GCM NIs

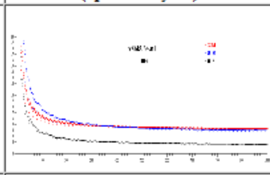
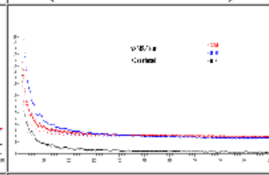
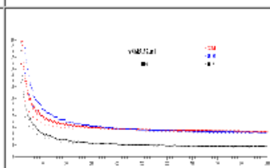
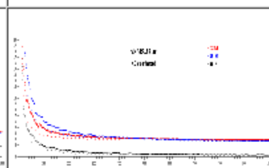
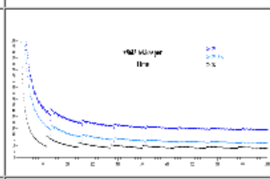
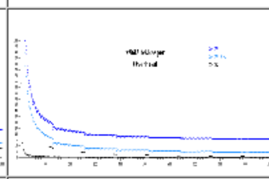
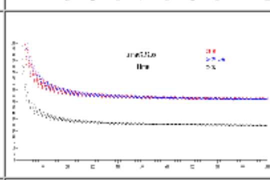
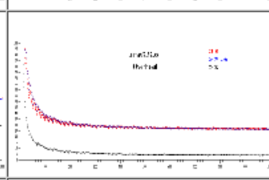
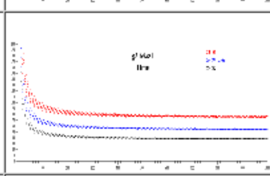
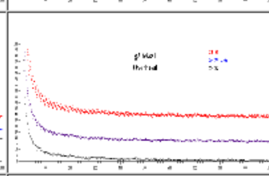
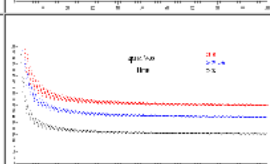
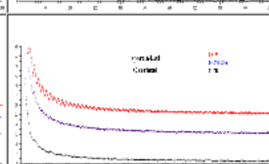
Mode	4KB cpb
CCM	5.14
GCM	2.95
OCB	0.87



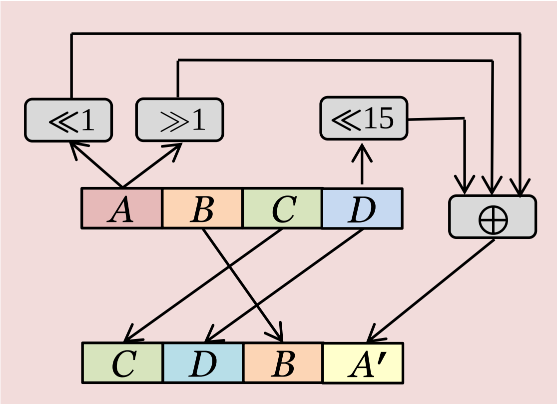
Authenticated-Encryption Software Performance: Comparison of CCM, GCM, and OCB

See the OCB homepage www.cs.ucdavis.edu/~rogaway/ocb for more platforms and data, +reference code

- Click on a **Time** or **Overhead** plot to see a larger version of it.
- Click on a **Mode** (CCM, GCM, OCB, etc) to retrieve the **raw data**.
- Here OCB means OCB3. A companion webpage compares the performance of OCB variants.
- Further notes can be found on the bottom of this page.

Environment (details)	Time (cpb *vs.* bytes)	Overhead (subtract time for CTR)	Mode (clickable)	Over 4096	Time 4096	IPI (cpb)	Size (bytes)	Init (cycles)
Intel x86 i5-650 "Clarkdale" 64-bit NI			CCM	2.90	4.17	4.57	512	265
			GCM	2.46	3.73	4.53	656	337
			OCB	0.21	1.48	1.87	624	295
			CTR		1.27	1.37	244	115
Intel x86 i5-650 "Clarkdale" 32-bit NI			CCM	2.79	4.18	4.70	512	274
			GCM	2.49	3.88	4.79	656	365
			OCB	0.20	1.59	2.04	624	318
			CTR		1.39	1.52	244	130
Intel x86 i5-650 "Clarkdale" 64-bit Käsper-Schwabe			GCM	14.7	22.4	26.7	1456	3780
			GCM-8K	3.19	10.9	15.2	9648	2560
			OCB	0.31	8.05	9.24	3216	3430
			CTR		7.74	8.98	1424	1180
ARM Cortex-A8 32-bit OpenSSL			CCM	25.9	51.3	53.7	512	1390
			GCM-256	26.7	50.8	53.9	656	3440
			OCB	3.49	28.9	30.9	784	2050
			CTR		25.4	25.9	244	236
PowerPC 970 64-bit OpenSSL			CCM	38.2	75.7	77.8	512	1510
			GCM-256	16.0	53.5	56.2	656	1030
			OCB	0.0	37.5	39.6	784	2300
			CTR		37.5	37.8	244	309
UltraSPARC III 64-bit OpenSSL			CCM	25.3	49.4	51.7	512	1280
			GCM-256	15.2	39.3	41.5	656	904
			OCB	0.9	25.0	26.5	784	1770
			CTR		24.1	24.4	244	213

Utility of Implementations for Understanding What's Fast / Desirable

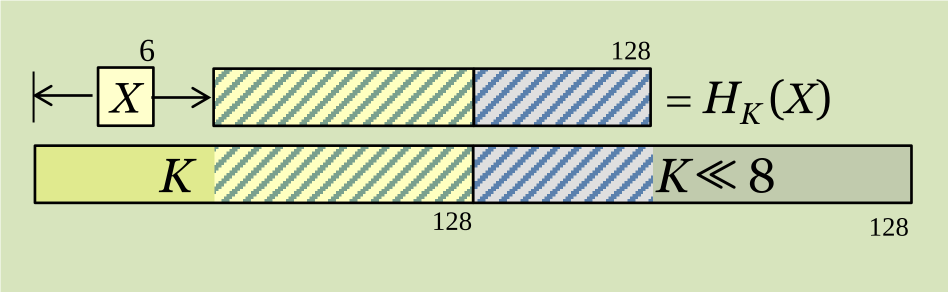


Word-Oriented LFSRs
[Chakraborty, Sarkar 2008]
don't help

```
int ae_encrypt(
  ae_ctx       *ctx,
  const void *nonce,
  const void *pt,
  int           pt_len,
  const void *ad,
  int           ad_len,
  void         *ct,
  void         *tag,
  int           final);
```

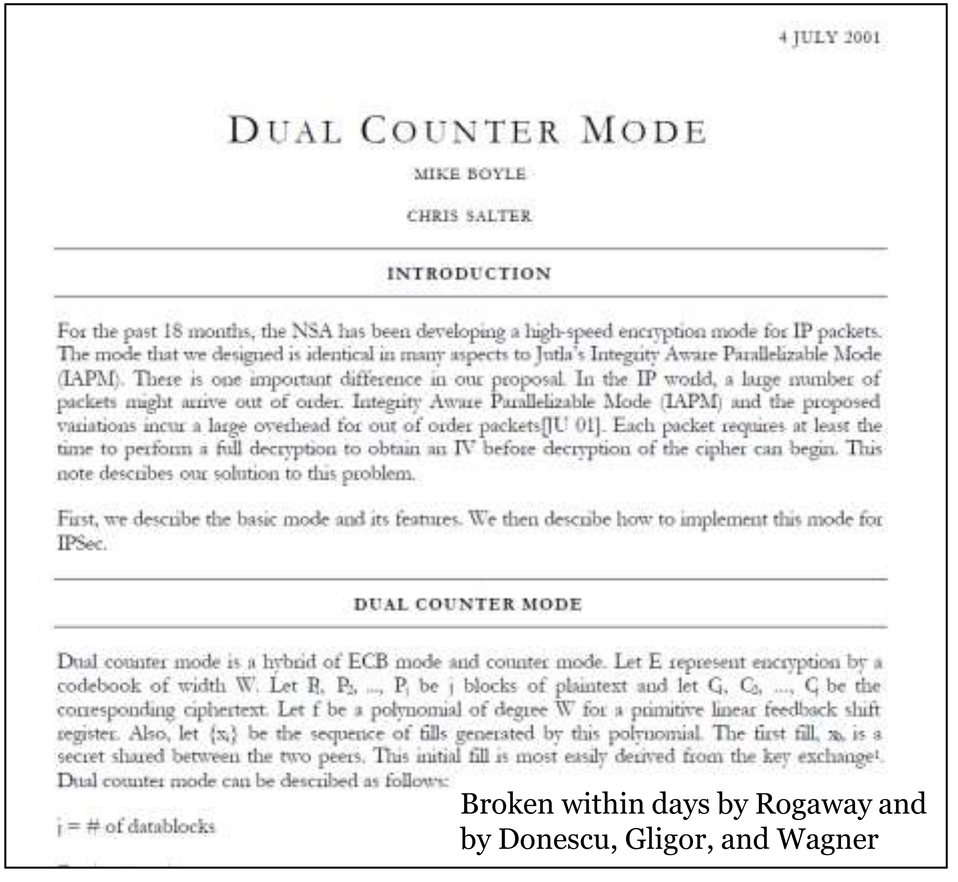
Incremental API impacts processing of final chunks

Stretch-then-Shift hash **does help**



Utility of Theory for Designing Fast / Correct Schemes

- Modes as efficient as OCB **can't be designed** without a supporting theory
- Errors are **expected** without a supporting theory



4 JULY 2001

DUAL COUNTER MODE

MIKE BOYLE

CHRIS SALTER

INTRODUCTION

For the past 18 months, the NSA has been developing a high-speed encryption mode for IP packets. The mode that we designed is identical in many aspects to Jutla's Integrity Aware Parallelizable Mode (IAPM). There is one important difference in our proposal. In the IP world, a large number of packets might arrive out of order. Integrity Aware Parallelizable Mode (IAPM) and the proposed variations incur a large overhead for out of order packets[JU 01]. Each packet requires at least the time to perform a full decryption to obtain an IV before decryption of the cipher can begin. This note describes our solution to this problem.

First, we describe the basic mode and its features. We then describe how to implement this mode for IPSec.

DUAL COUNTER MODE

Dual counter mode is a hybrid of ECB mode and counter mode. Let E represent encryption by a codebook of width W. Let $P_1, P_2, \ldots, P_j$ be j blocks of plaintext and let $C_1, C_2, \ldots, C_j$ be the corresponding ciphertext. Let f be a polynomial of degree W for a primitive linear feedback shift register. Also, let $\{x_i\}$ be the sequence of fills generated by this polynomial. The first fill, x_0, is a secret shared between the two peers. This initial fill is most easily derived from the key exchange[1]. Dual counter mode can be described as follows:

j = # of datablocks

Broken within days by Rogaway and by Donescu, Gligor, and Wagner

http://www.nsa.gov/research/tech_transfer/fact_sheets/dual_counter_mode.shtml

OCB

- Fastest provably-secure blockcipher-based construction for SW
- Parallelizable, online, ~ m+1.02 blockcipher calls

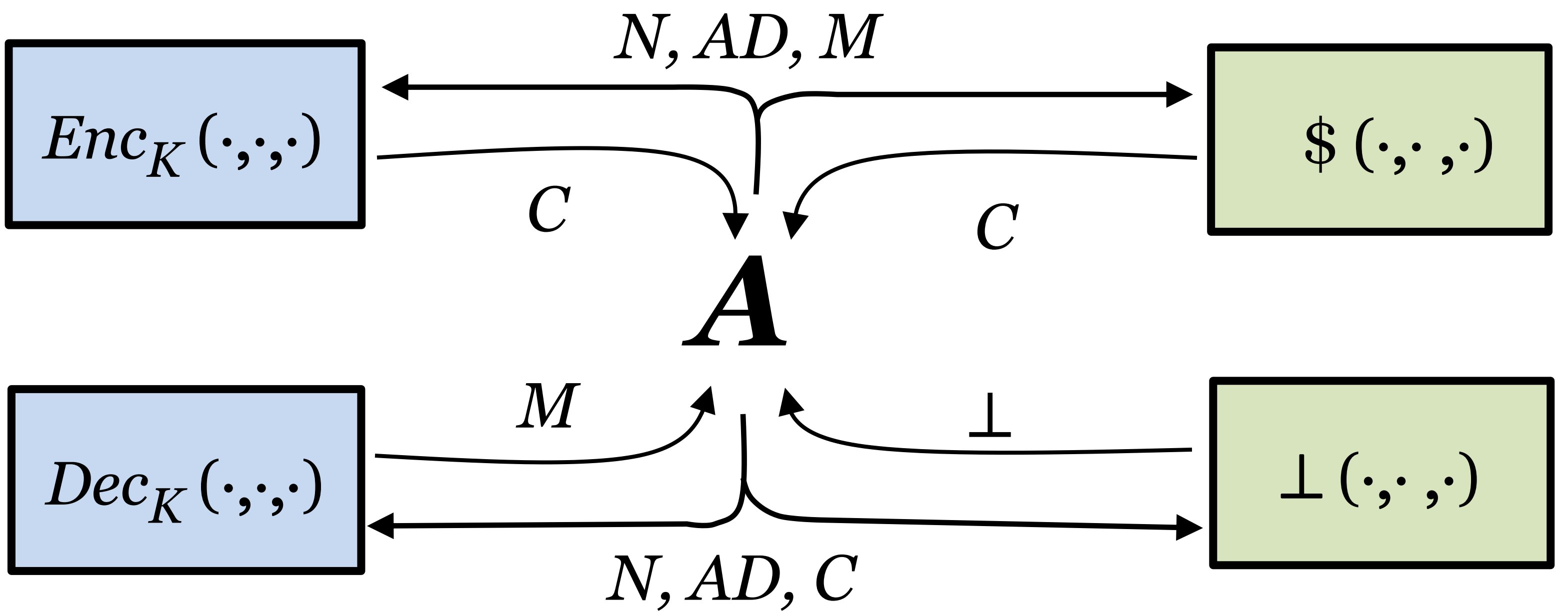
- Blockcipher used in the forward *and backward* direction
- There are faster *de novo* approaches
- Security only to the birthday bound
- Patents
- Limited **misuse resistance**
    - Nonce reuse
    - Tag truncation
    - Incremental-decrypt exploit

Nonce Repetitions
One form of misuse

- If N **is** a nonce, you get what an AE delivers

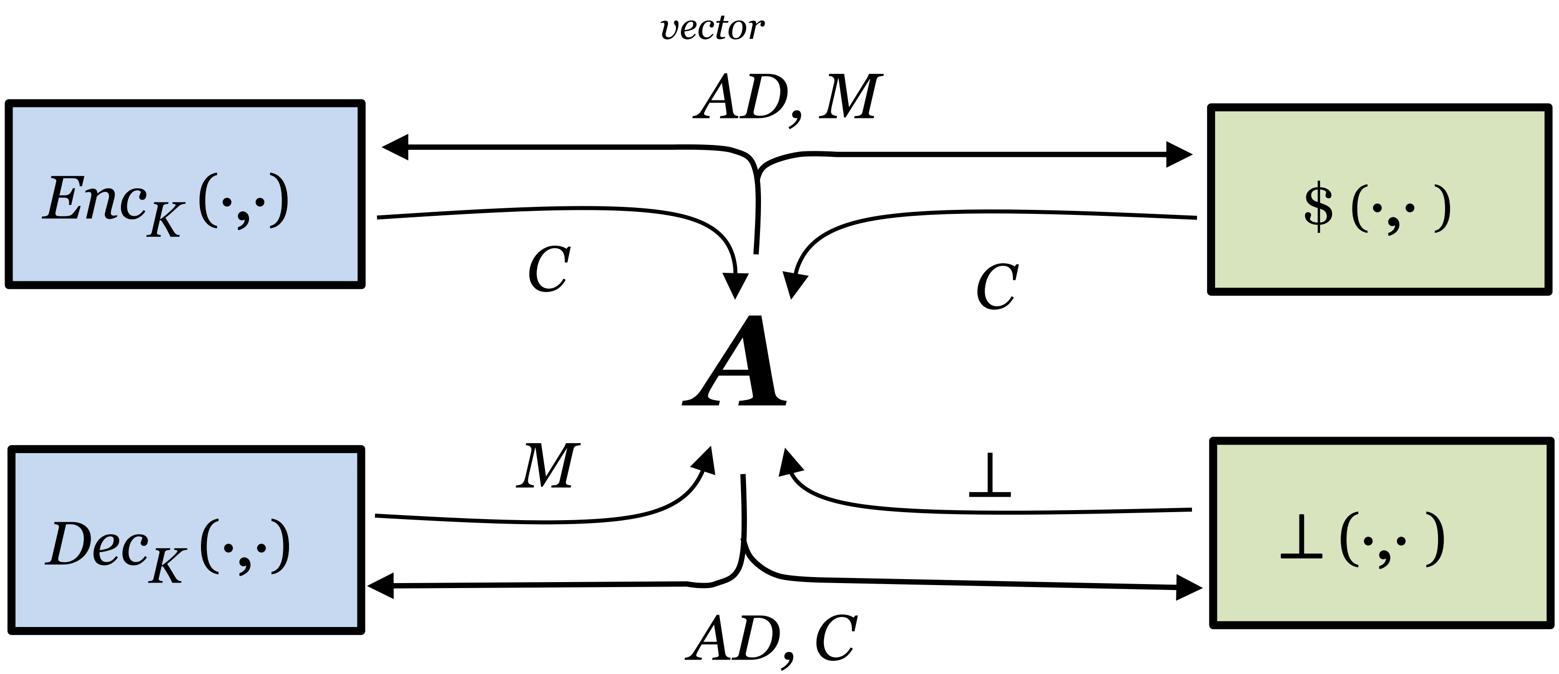
- If N gets **reused**, all that leaks is **repetitions:**
  - authenticity is undamaged
  - privacy damaged to the extent unavoidable—repetitions of (N, AD, M) revealed

Nonce-Reuse-Resistant AE



A may not ask queries that would trivially result in a win

Deterministic AE



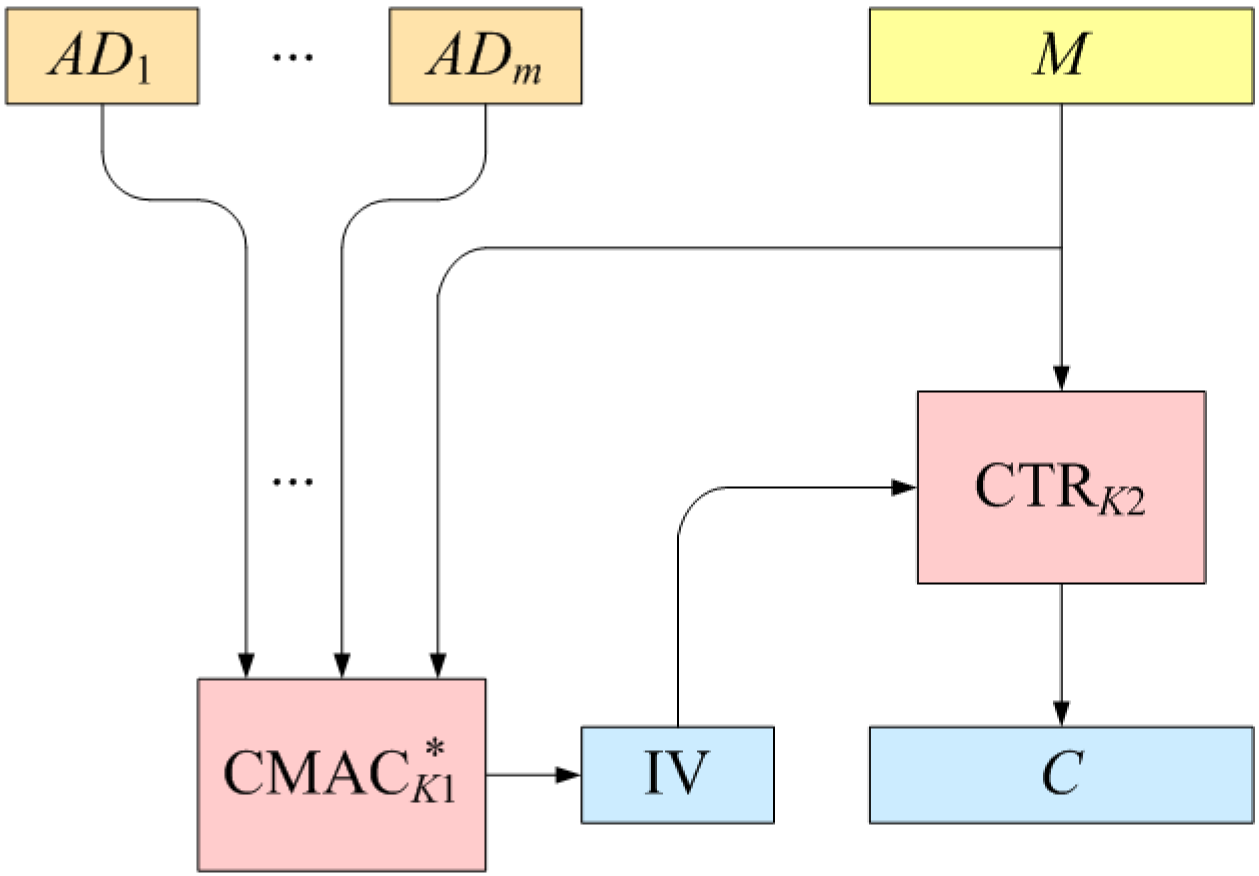
A may not ask queries that would trivially result in a win

Deterministic AE ➔ Nonce-Reuse AE
Regard a component of the AD as the nonce

[R, Shrimpton 2006]

ISO/IEC 19772:2009
RFC 5297

SIV



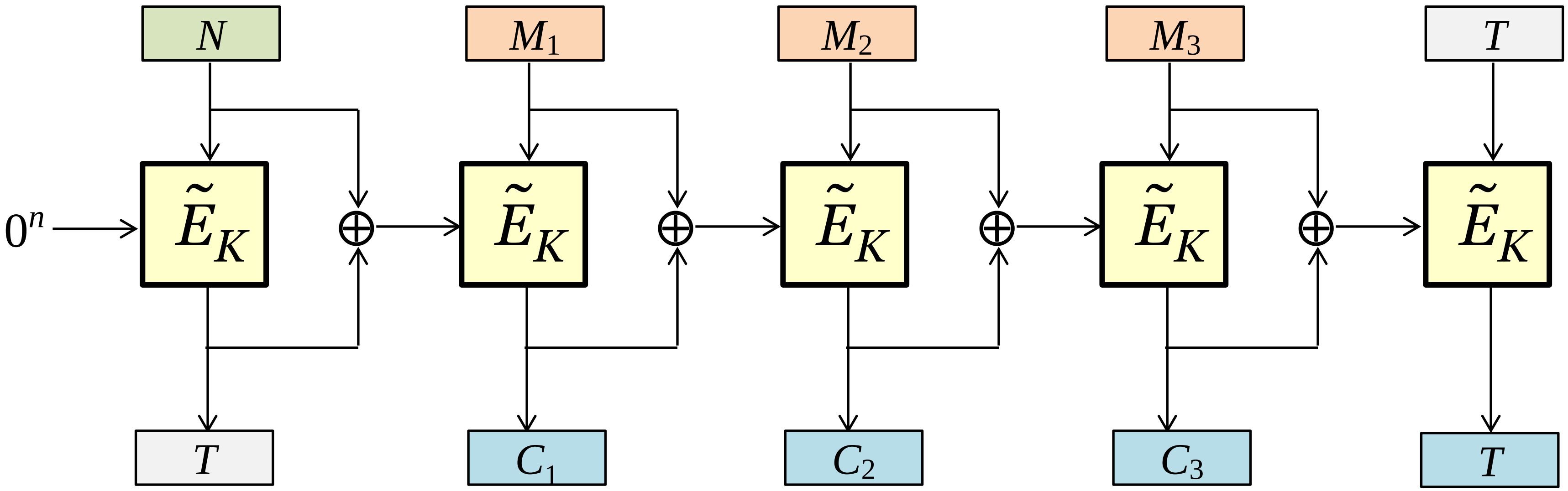
The Last Definitions are Impossible for Online Schemes

The **first bit of ciphertext**
must depend on the **last bit of plaintext**

- Need **unbounded memory**
- Long message: **performance hit**

Online AE **[Fleischmann, Forler, Lucks, Wenzel 2012]**
following [R, Zhang 2011] and
[Bellare, Boldyreva, Knudsen, Namprempre 2001]

An Online AE Scheme



Security: when nonces repeat,
leak equality of longest blockwise-prefixes

128-bit blocks

What does the **goal** have to do with
the **blocksize** of the blockcipher?!

Patents

Patent-related FUD
(+ some politics)
killed OCB in 802.11,
limit its adoption
now, and gave us
CCM and GCM

6,963,976
Jutla (IBM)

6,973,187
Gligor and Donescu (VDG)

7,046,802
Rogaway

7,093,126
Jutla (IBM)

7,200,227
Rogaway

7,840,003
Kim, Han, Yoo, and Kwon
High-speed GCM-AES
block cipher apparatus and method

7,853,801
Kim, Kwon, and Kim
System and method for providing
authenticated encryption in
GPON network [sic]

7,949,129
Rogaway

7,970,130
Yen. Low-latency method and apparatus
of GHASH operation for authenticated encryption
Galois Counter Mode [sic]

8,107,620
Jutla (IBM)

8,340,280
Gueron and Kounavis
Using a single instruction
multiple data (SIMD)
instruction to speed up
Galois Counter Mode (GCM)
Computations
Dec 25, 2012

8,321,675
Rogaway

8,190,894
Sandberg and Schaffer
Method and system for generating
ciphertext and message
authentication codes using
shared hardware

ANNOUNCEMENT

FREELY LICENSED!

www.cs.ucdavis.edu/~rogaway/ocb

Thanks to Harvard's Cyberlaw Clinic at the Berkman Center for Internet & Society

This is a non-binding summary of a legal document. The parameters of the license are specified in the license document and that document is controlling.

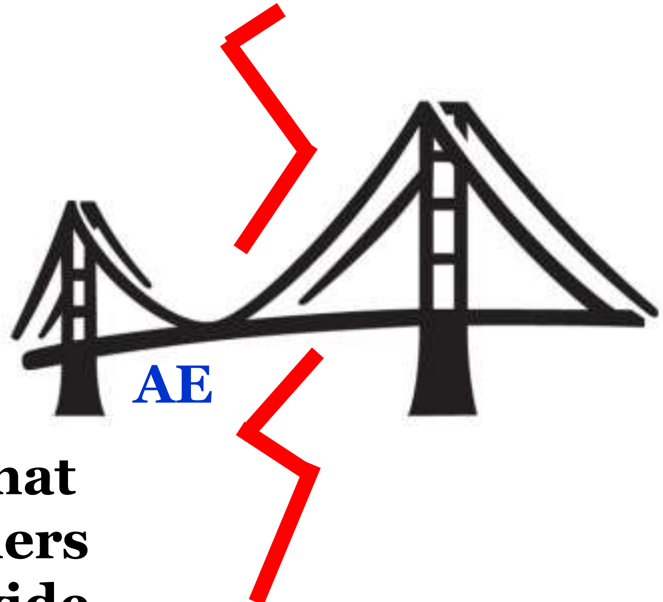
License for Open-Source Software Implementations of OCB

Under this license, you are authorized to make, use, and distribute open source software implementations of OCB. This license terminates for you if you sue someone over their open source software implementation of OCB claiming that you have a patent covering their implementation.

General License for Non-Military Software Implementations OCB

This license does not authorize any military use of OCB. Aside from military uses, you are authorized to make, use, and distribute (1) any software implementation of OCB and (2) non-software implementations of OCB for noncommercial or research purposes. You are required to include notice of this license to users of your work so that they are aware of the prohibition against military use. This license terminates for you if you sue someone over an implementation of OCB authorized by this license claiming that you have a patent covering their implementation.

Conclusions



AE represent a **triumph** of practice-oriented provable security

Better Security **&** **Better Efficiency**

than anything *ad hoc* design could deliver

At the same time, **disappointing** that what is used,
CCM and GCM, are so far removed from how well we can do.