AEZ v2

Authenticated Encryption by Enciphering

1. Why we created AEZ
2. Enciphering-based AE
3. Robust-AE
4. Accelerated provable-security
5. Components FF0 and EME4
6. AEZ Extensions

Viet Tung Hoang
Georgetown University, USA
University of Maryland, USA

Ted Krovetz
Sacramento State, USA

Phillip Rogaway
UC Davis, USA
ETH Zürich, Switzerland

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

DIAC 2014
UC Santa Barbara
Aug 23, 2014

AE Thesis

Giving **definitions** that guarantee more.
Giving **schemes** that achieve them.

By **strengthening** symmetric encryption, we can provide a **simpler-to-use** primitive for users, and thereby **minimize misuse**.

(Also: by focusing on the new target, we can **maximize efficiency**.)

Symmetric Encryption

Robust AE

Misuse-Resistant AE (MRAE)

Online AE

Nonce-based AEAD

Nonce-based AE

Probabilistic AE

IND-CCA2 prob encryption

IND-CPA prob encryption

Strength

Isn't MRAE already very strong?

Robust AE
MRAE
Online AE (OAE)
Nonce-based AEAD

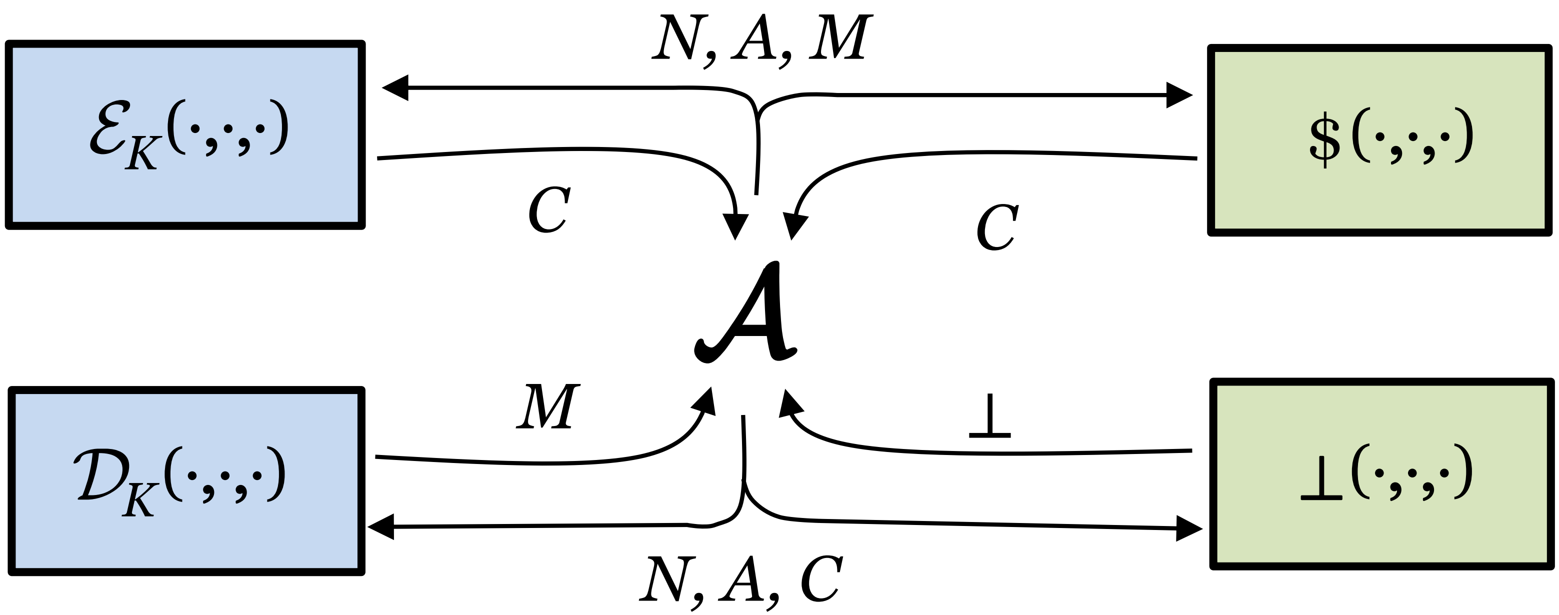
Yes.

Still, there are **important ways** in which MRAE **falls short** of maximizing strength/ease of correct use, in both

- the service it provides (**syntax**)
- what it guarantees (**security**)

MRAE

[Rogaway, Shrimpton 2006]

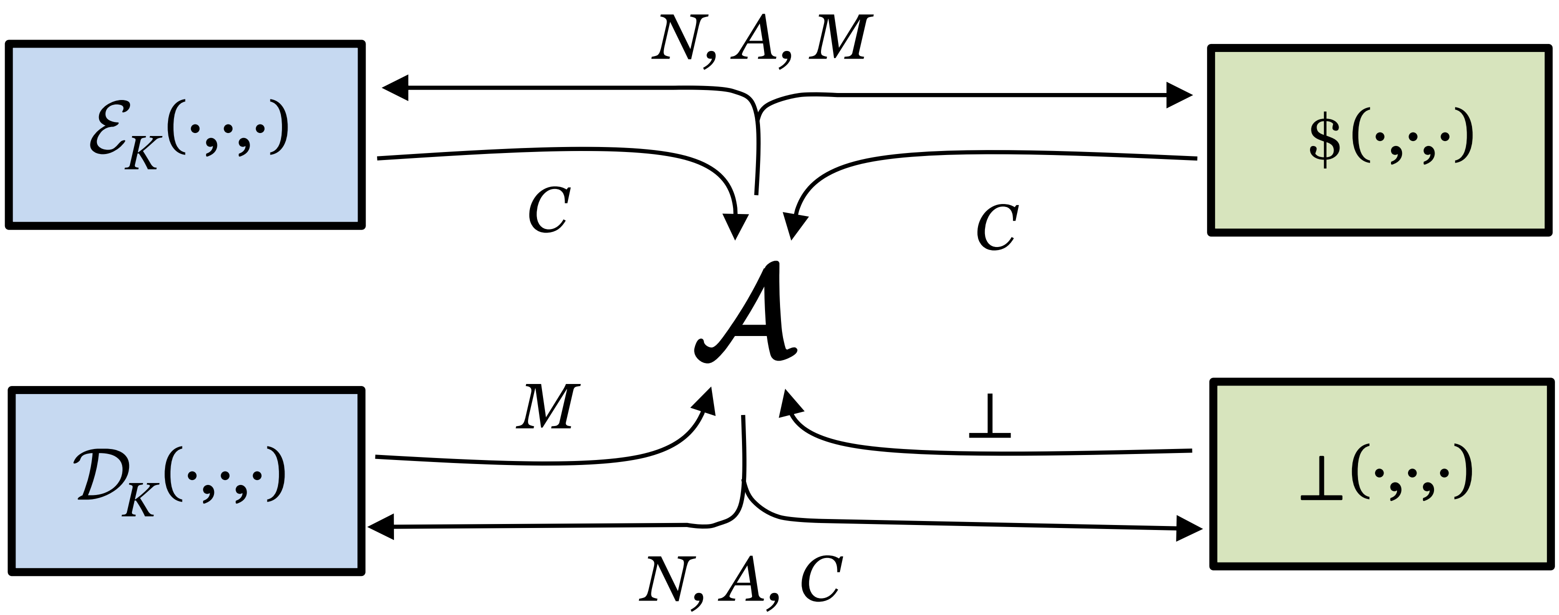


$\mathcal{A}$ may not ask queries that would trivially result in a win

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

MRAE

[Rogaway, Shrimpton 2006]



Effectively **assumes**

$|C| = |M| + 128$

Some *reasonably large* constant τ .
Big enough that, with the "real" scheme, forgeries almost *never* occur.

There <u>are</u> settings where we don’t want to grow plaintexts ~16 bytes

> August 13, 2013 DIAC 2013
>
> **AEAD Ciphers**
> **for**
> **Highly Constrained Networks**
>
> —
>
> René Struik
>
> e-mail: rstruik.ext@gmail.com

Constrained devices: sensor networks, ad hoc networks, “internet of things”:
short tags save energy.

Shaving off 8 octets may justify making symmetric-key crypto 10× more expensive [sl.12]

Crypto cost should *not* ignore cost of data expansion. Authentication tags may be “evil” (authenticity is *not*) [sl.29]

Struik also speaks of the importance of supporting **very short plaintexts** and enabling exploitation of **already-present redundancy**.

At some level, we know how to fix this: *Encrypt by Enciphering*

Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography

Mihir Bellare[1] and Phillip Rogaway[2]

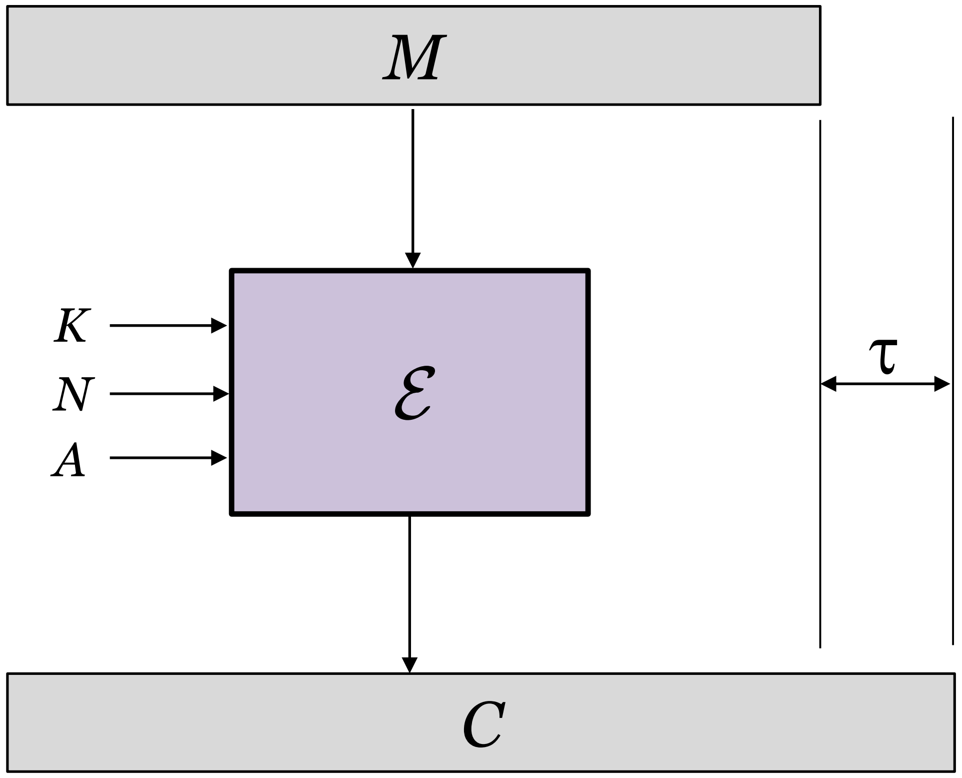
[1] Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA
mihir@cs.ucsd.edu, www-cse.ucsd.edu/users/mihir

[2] Dept. of Computer Science, Engineering II Building, One Shields Avenue, University of California at Davis, Davis, CA 95616, USA, and Dept. of Computer Science, Faculty of Science, Chiang Mai University, Thailand
rogaway@cs.ucdavis.edu, www.cs.ucdavis.edu/~rogaway

Abstract. We investigate the following approach to symmetric encryption: first *encode* the message via some keyless transform, and then *encipher* the encoded message, meaning apply a permutation F_K based on a shared key K. We provide conditions on the encoding functions and the cipher which ensure that the resulting encryption scheme meets strong privacy (eg. semantic security) and/or authenticity goals. The encoding can either be implemented in a simple way (eg. prepend a counter and append a checksum) or viewed as modeling existing redundancy or entropy already present in the messages, whereby encode-then-encipher encryption provides a way to exploit structured message spaces to achieve compact ciphertexts.

Enciphering-Based AE

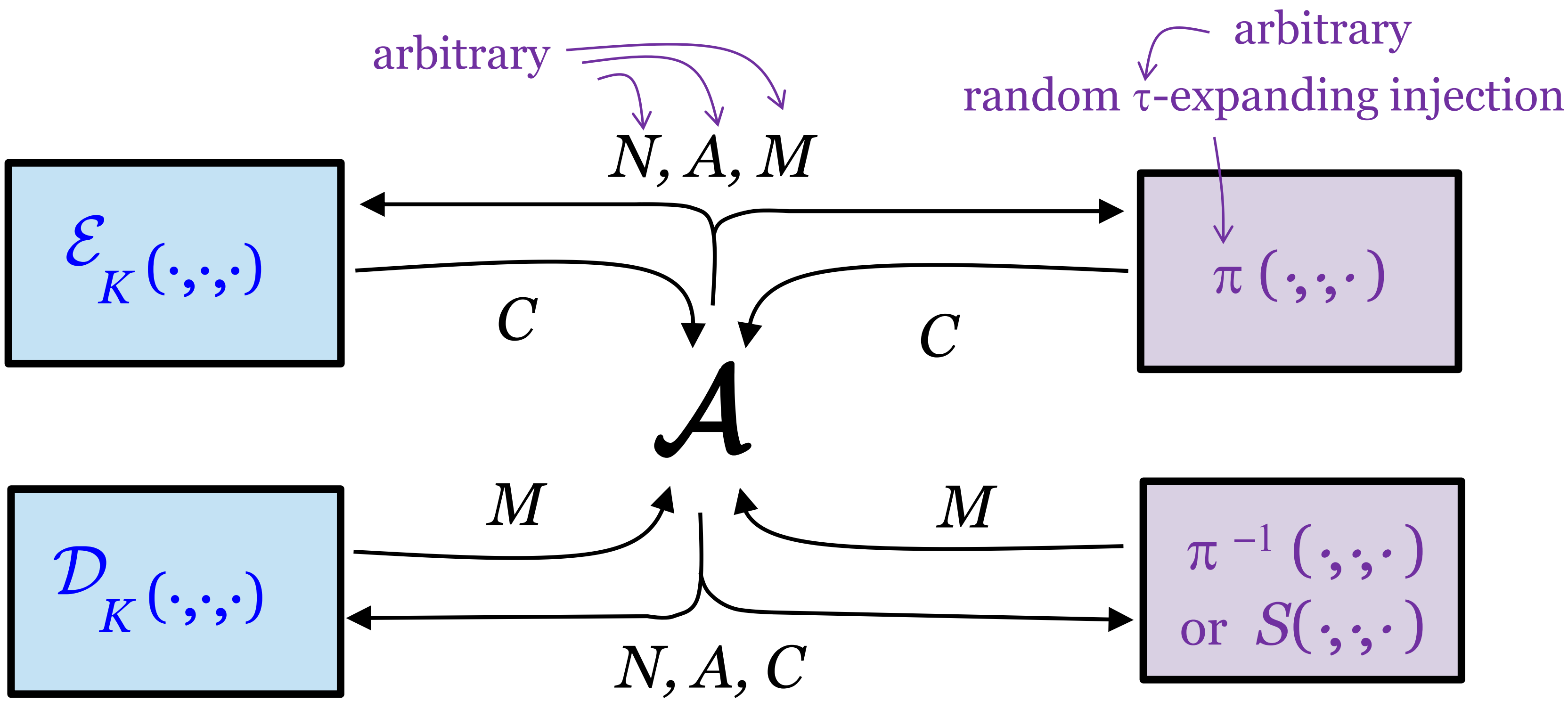
[Bellare, Rogaway 2000]
[Shrimpton, Terashima 2013]



$|K|$, $|N|$, $|A|$, $|M|$, τ
arbitrary

Robust AE: User chooses K, N, A, M, and $\tau \geq 0$.

Scheme is expected to deliver **best-security-possible** for τ.



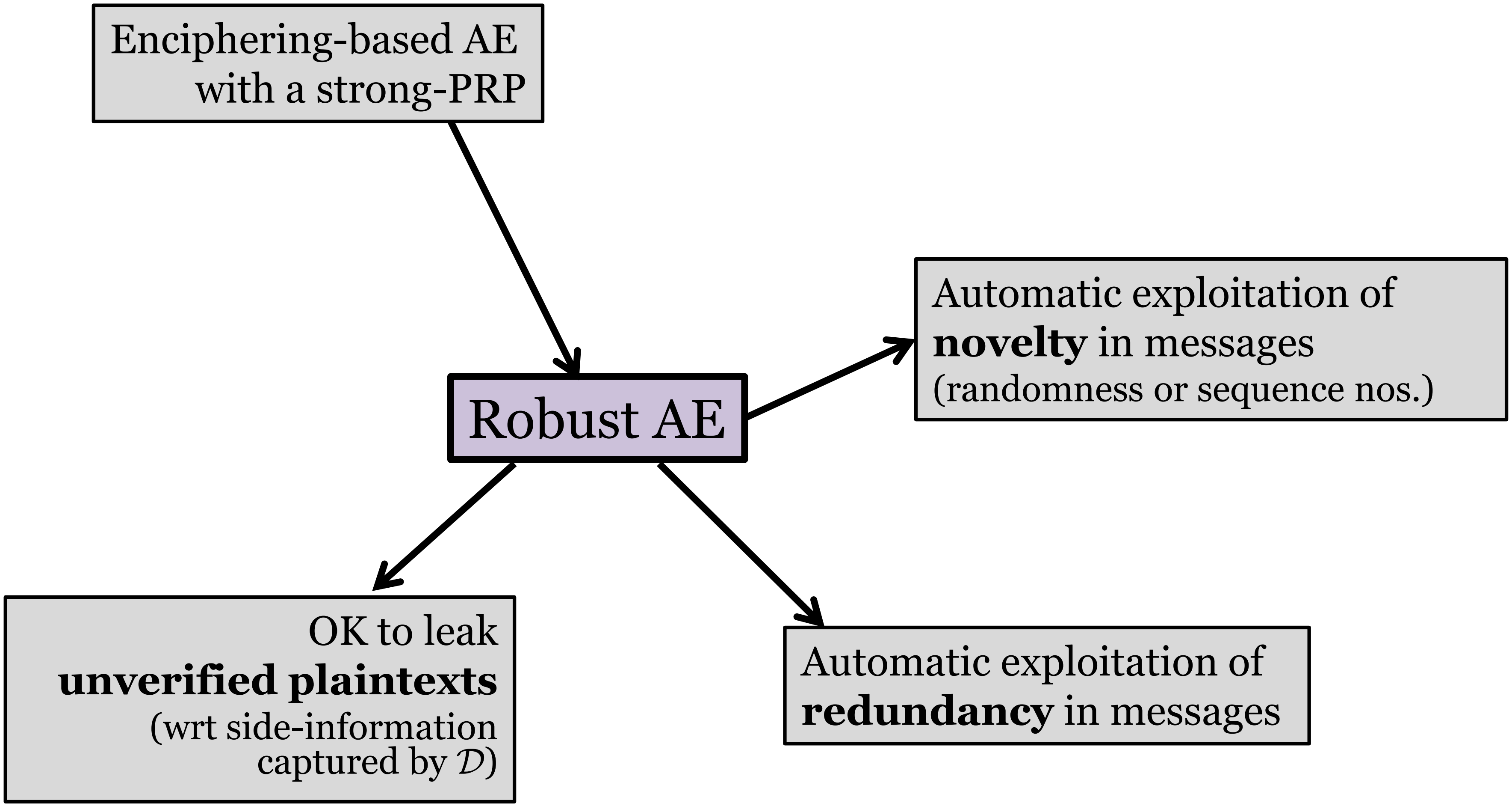
Pseudorandom injection

[R, Shrimpton 2006]

but now understood **prescriptively**, for all τ — not just an alternative characterization of an MRAE scheme

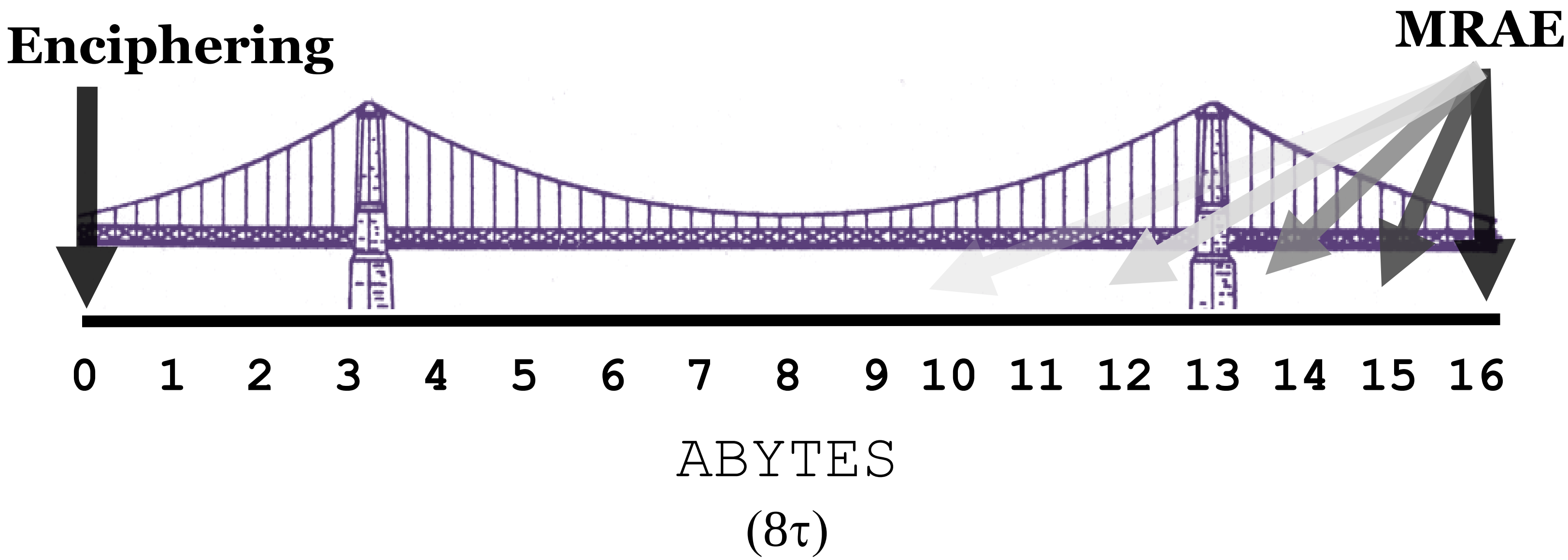
Inclusion of the simulator lets one formalize that release of **unverified plaintext** is not damaging

(cf: [ABLMMY14])

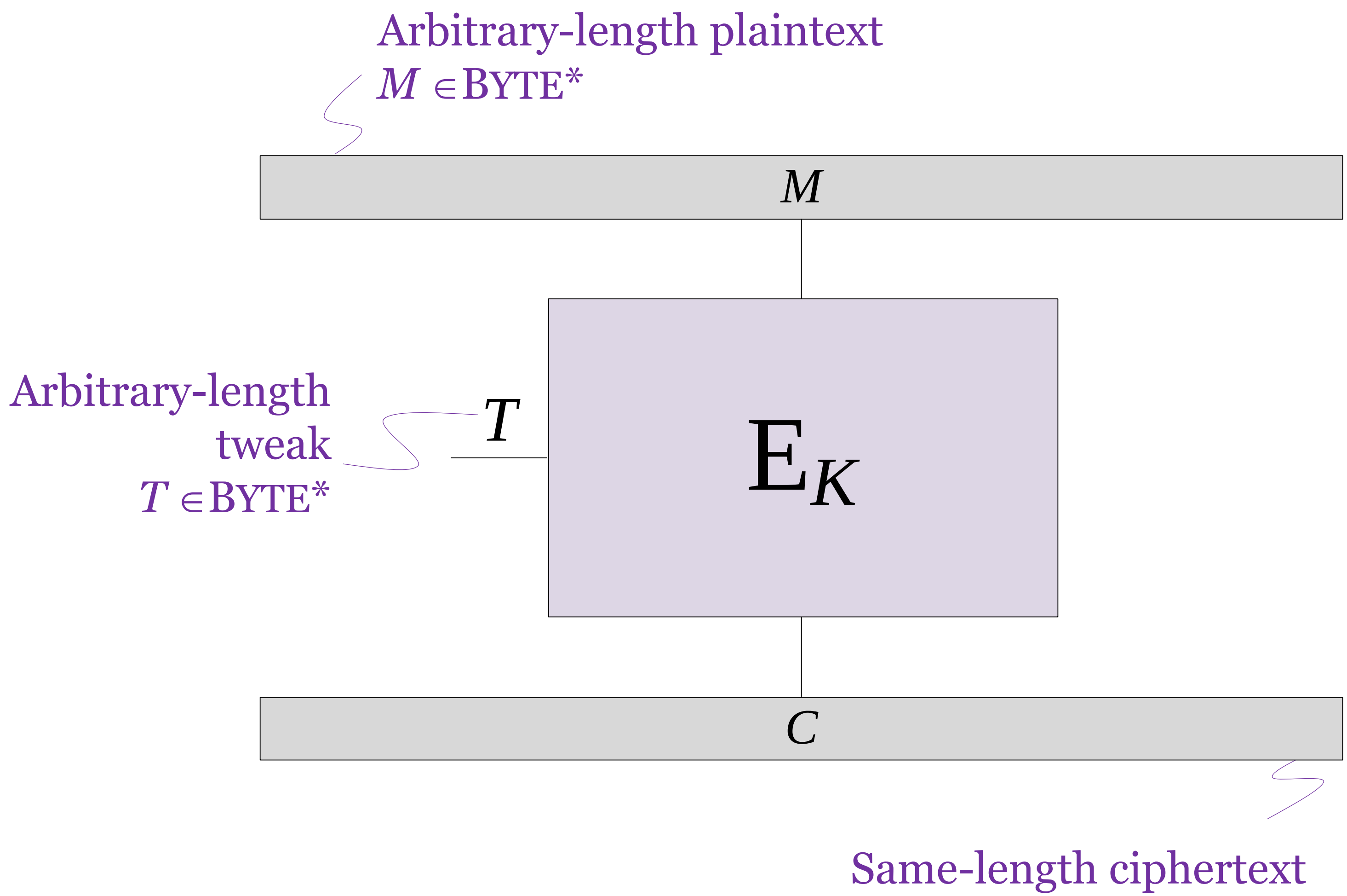

Enciphering-based AE with a strong-PRP
Robust AE
Automatic exploitation of **novelty** in messages (randomness or sequence nos.)
OK to leak **unverified plaintexts** (wrt side-information captured by $\mathcal{D}$)
Automatic exploitation of **redundancy** in messages

Robust AE

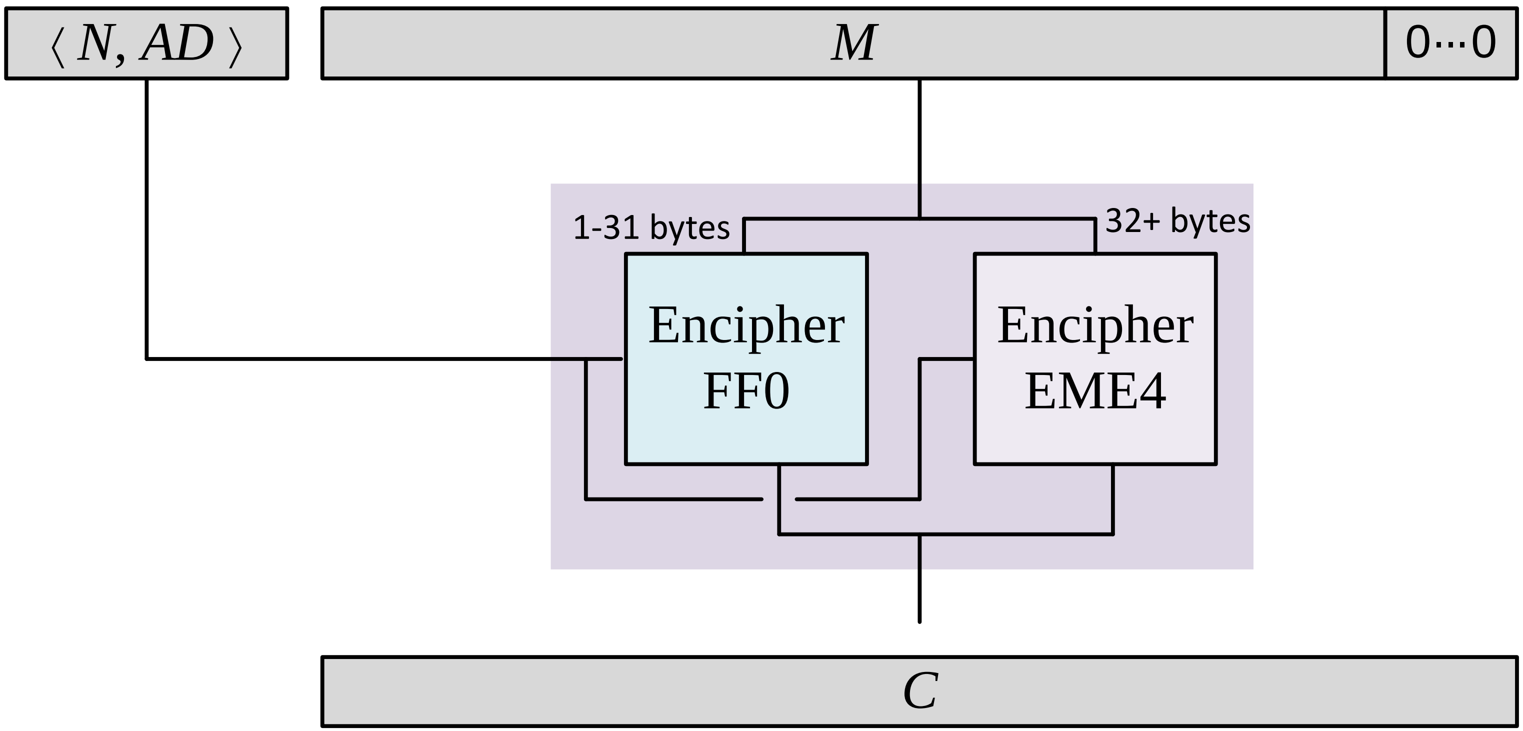
Generalizes strong-PRP and MRAE definitions



What to use for the enciphering scheme?



Length-Dependent Dispatch



FF0
FFX-like (Feistel)
[NIST SP 800-38G]
AES4-Based

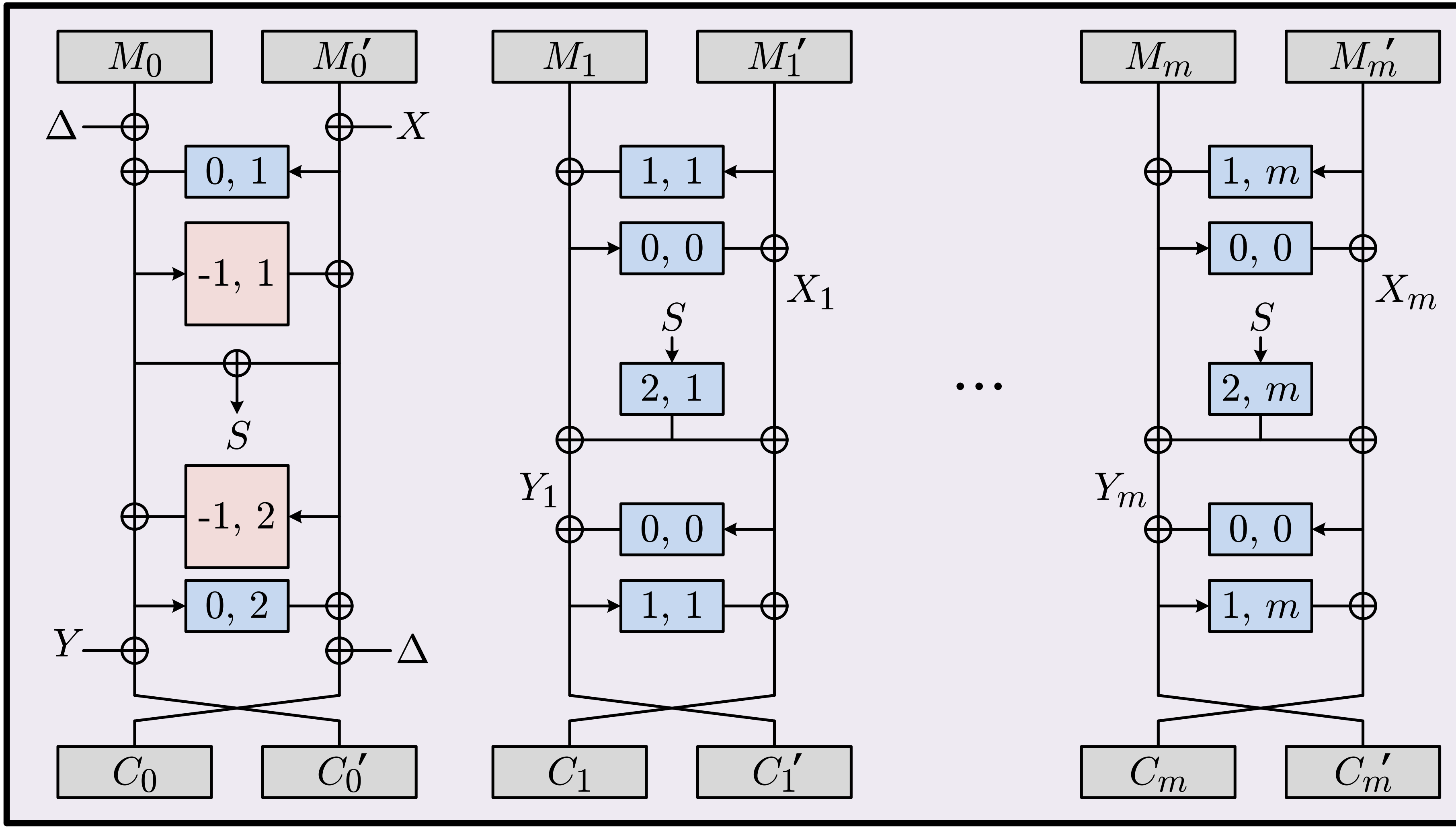
EME4
Builds on EME [Halevi, Rogaway]
and OTR [Minematsu 2014]
AES4 & AES based.

Designing FF0 and EME4

Accelerated Provable-Security Paradigm

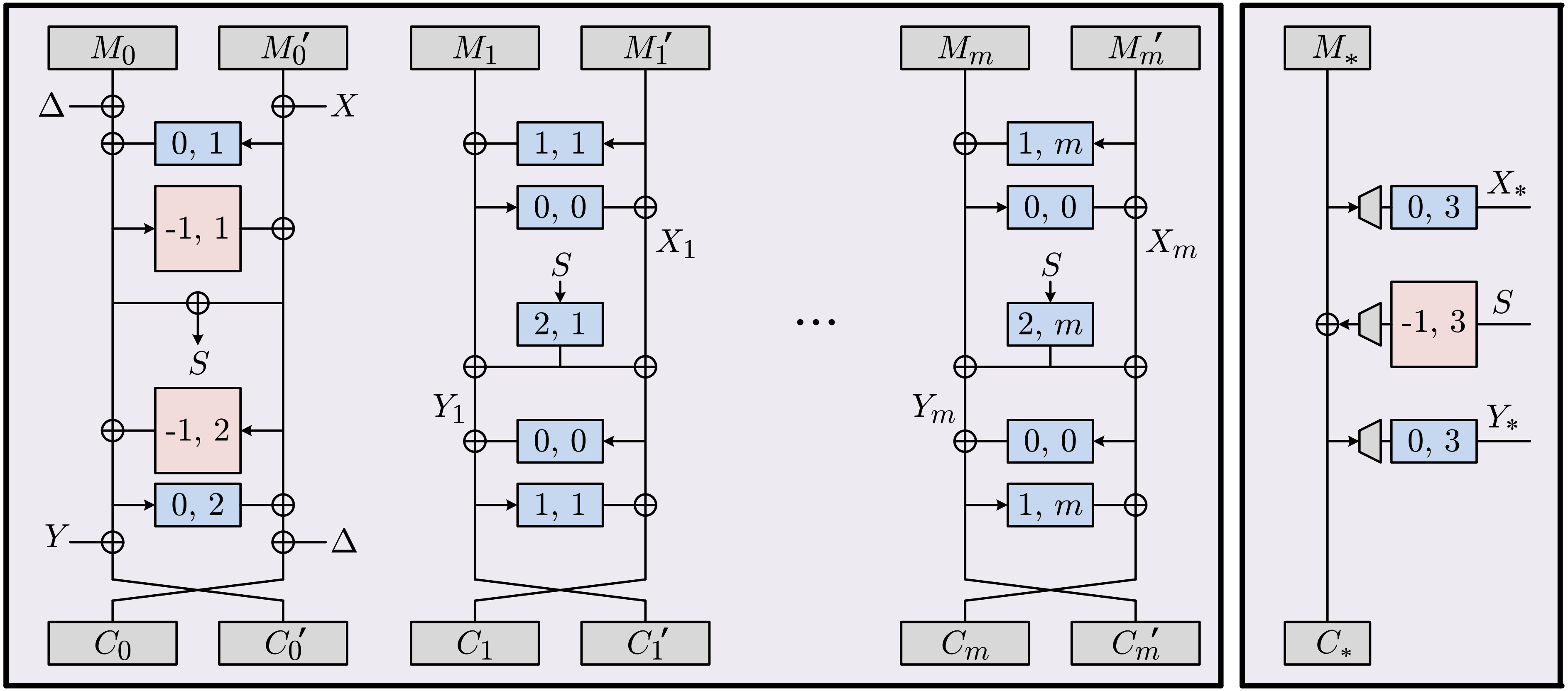
In general	**In our case**
Assume some primitive	[Liskov, Rivest, Wagner 2002] A tweakable blockcipher (TBC) (tweak space $\{0,1,2,3\} \times \mathbb{N}$)
Design assuming the primitive meets some standard assumption	The TBC is good as a tweakable PRP
Instantiate with "standard" primitive: the **scaled-up** design	Realize the TBC with AES / XE. **Not** what we submitted
Selectively instantiate with a mix of standard and reduced-round primitives: the **scaled-down** design	Use AES + AES4

EME4



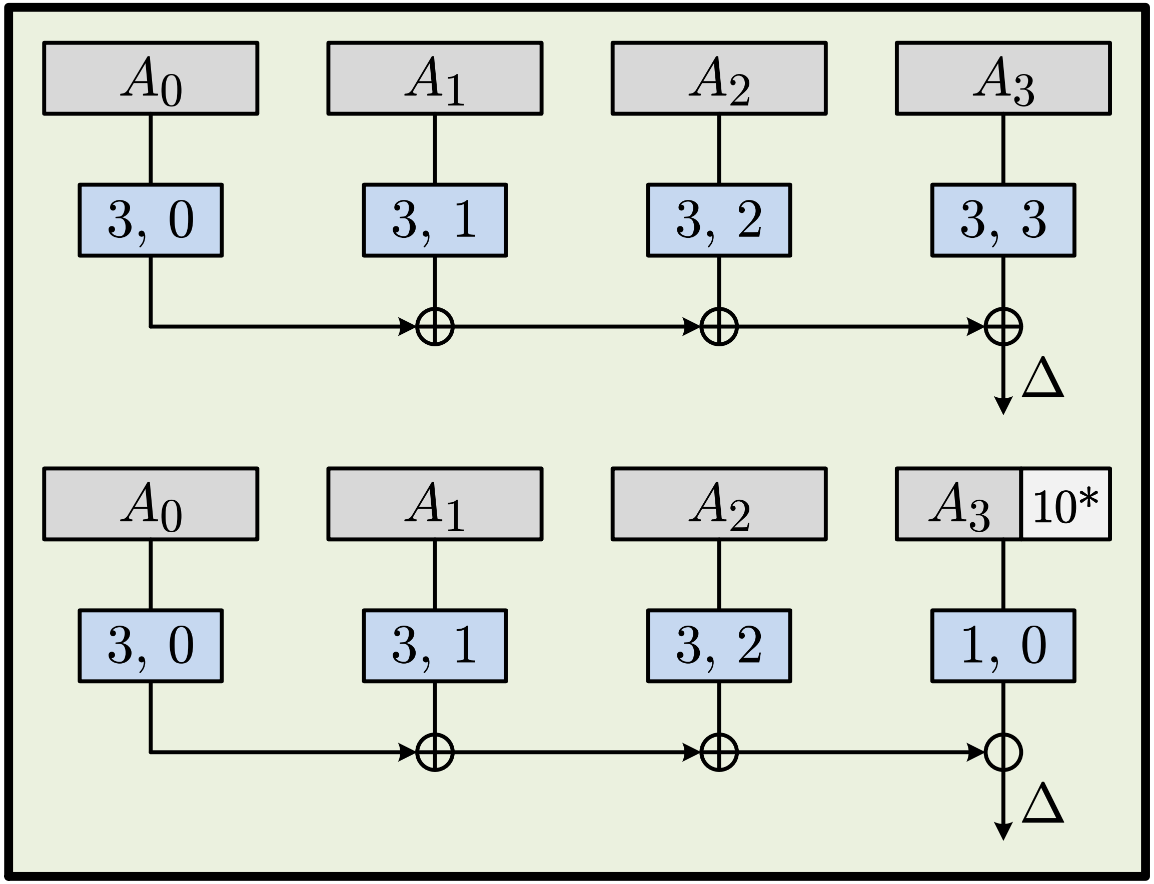
Message with an even number of blocks, no fragment at the end

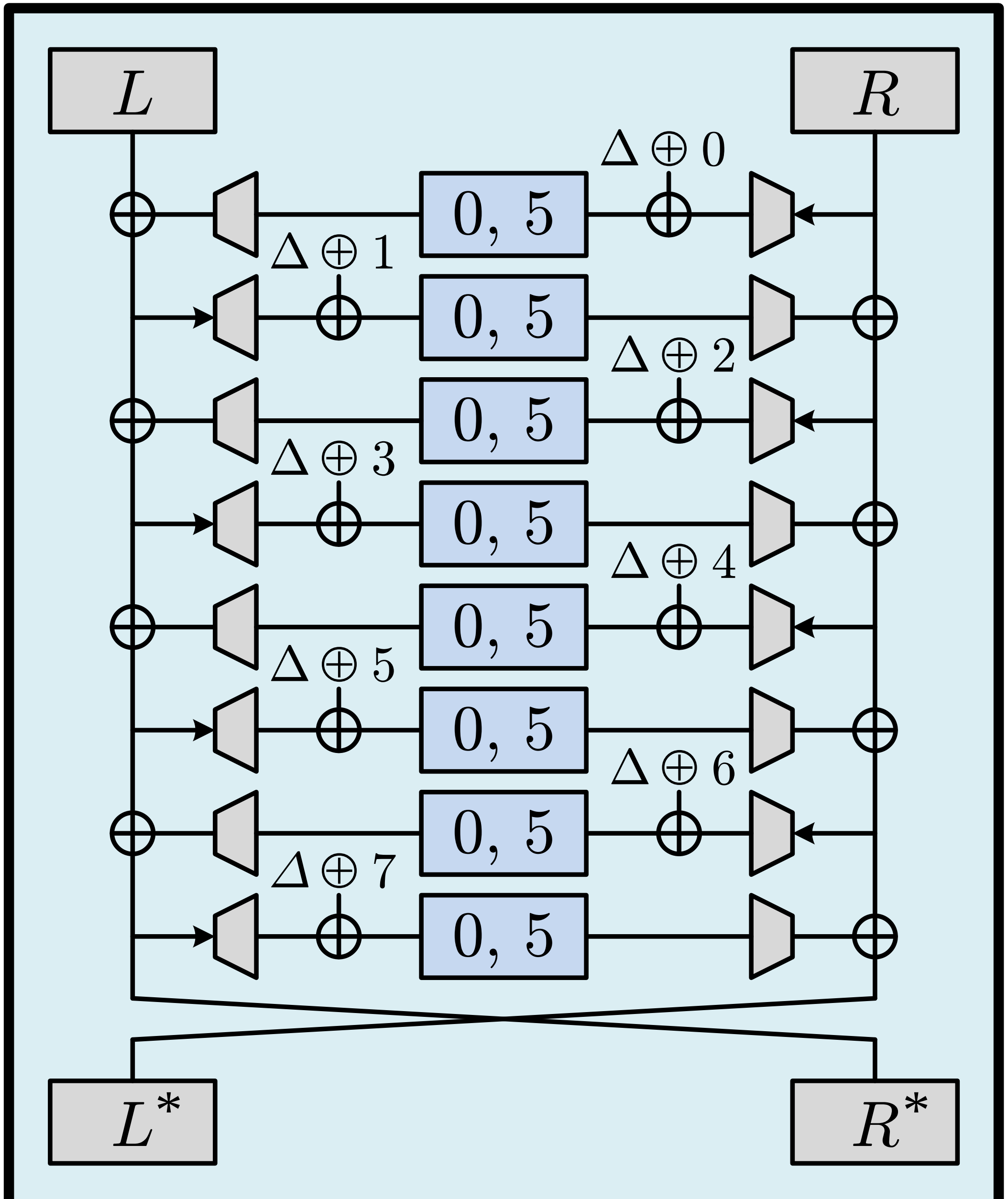
EME4



Message with an odd number of blocks, the last possibly a fragment

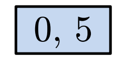
AHash



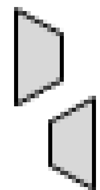


FF0

Δ is a universal-hash of A

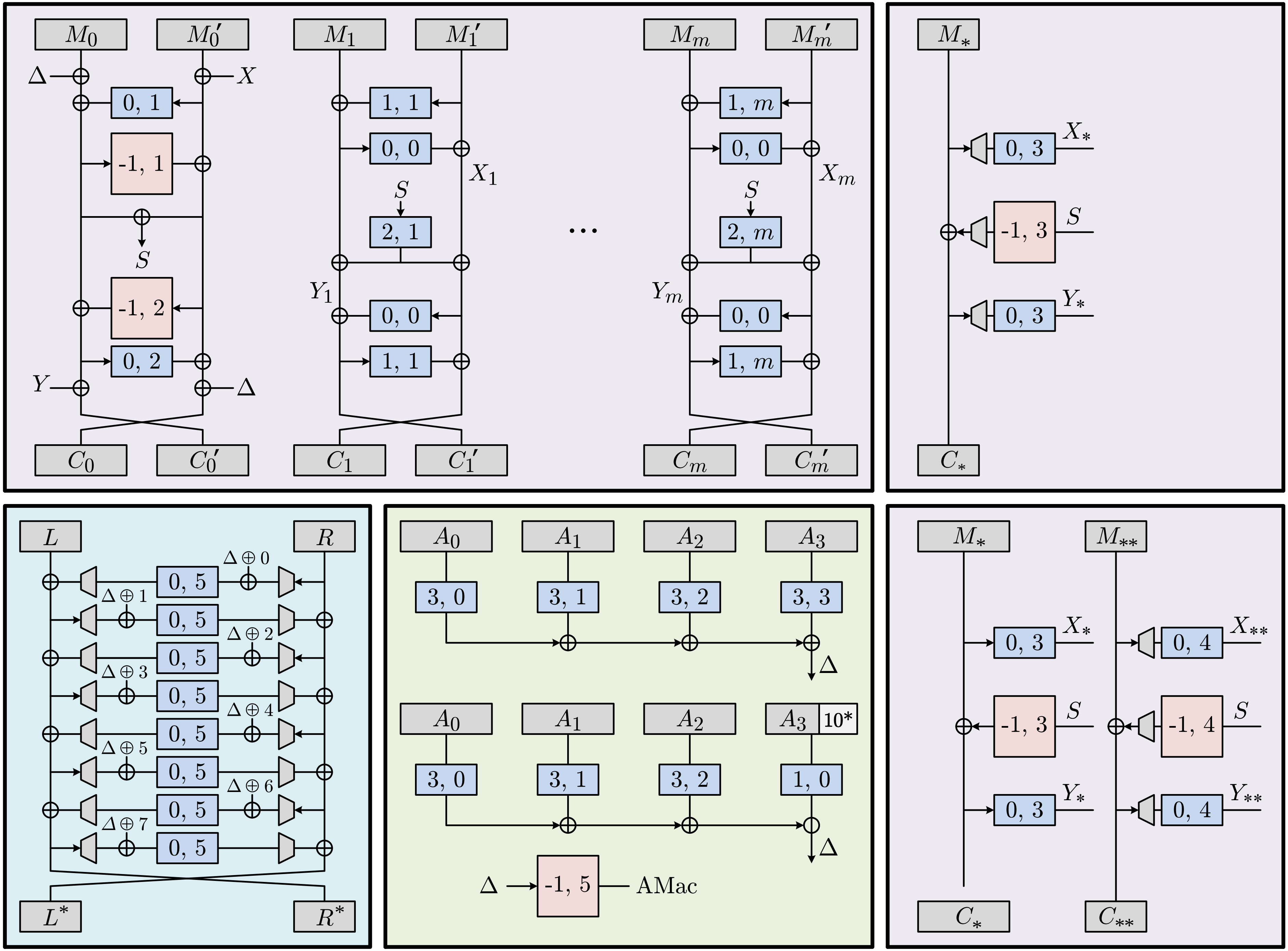


is our TBC


is truncation or $X\,0^*$ padding
(depending on orientation)

← 16-31 bytes

1-15 bytes: more rounds
(up to 24) and correct the
“even permutation” issue



M0
M0′
M1
M1′
Mm
Mm′
Δ
X
0, 1
-1, 1
S
-1, 2
0, 2
Y
Δ
1, 1
0, 0
X1
S
2, 1
Y1
0, 0
1, 1
1, m
0, 0
Xm
S
2, m
Ym
0, 0
1, m
…
C0
C0′
C1
C1′
Cm
Cm′
M*
X*
0, 3
S
-1, 3
Y*
0, 3
C*
L
R
Δ ⊕ 0
Δ ⊕ 1
Δ ⊕ 2
Δ ⊕ 3
Δ ⊕ 4
Δ ⊕ 5
Δ ⊕ 6
Δ ⊕ 7
0, 5
L*
R*
A0
A1
A2
A3
3, 0
3, 1
3, 2
3, 3
Δ
A0
A1
A2
A3
10*
3, 0
3, 1
3, 2
1, 0
Δ
Δ
-1, 5
AMac
M*
M**
X*
0, 3
X**
0, 4
S
-1, 3
S
-1, 4
Y*
0, 3
Y**
0, 4
C*
C**

Security property

The user chooses the ciphertext-expansion $\tau \geq 0$ and the scheme delivers *best-possible-security* for τ.

- Robust AE (Robust AE > MRAE $\gg$ Online-AE)
- Automatic novelty & redundancy exploitation
- Unverified-plaintext-release OK

Basic approach

- Enciphering-based AE
- FF0 and EME4
- Accelerated provable security (AES+AES4; ~~AES key schedule)~~

Additional features

- Blockcipher calls: 1× AES enc; **0.4**× AES for AD and fast-reject
- Inverse-free
- Parameter-free (well, ABYTES)
- Highly symmetric: encipher ≈ decipher
- Good key-agility
- Arbitrary-length keys (extract 256 bits; then expand) & nonces
- Small context size (≈ 144 bytes for speed-optimized)
- AEZ Extensions (coming soon)

AEZ Efficiency

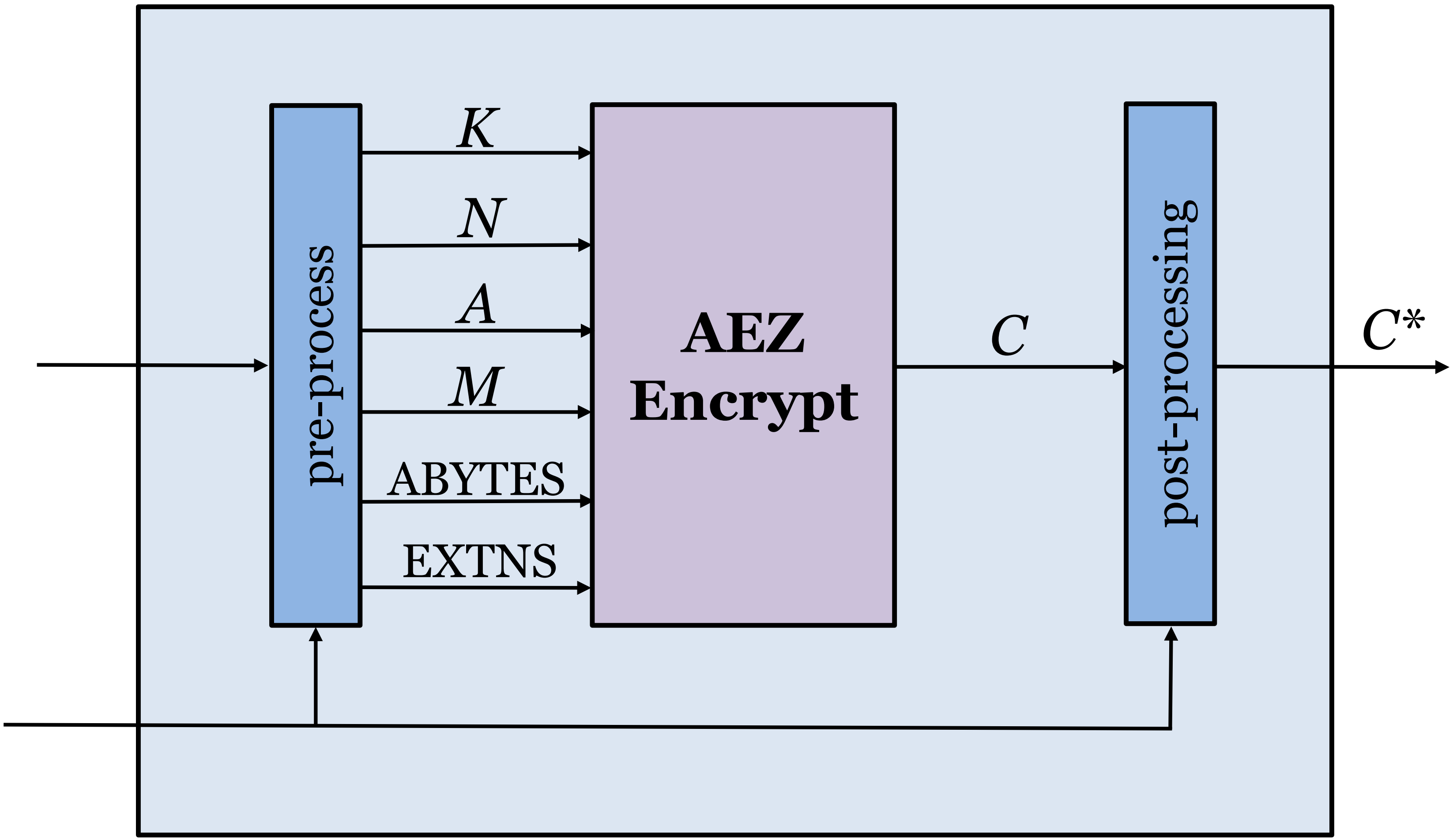
Message of $m \geq 2$ blocks	in "AES equivalents" (10 AES rounds) computation $\leq$	latency $\leq$
Encipher/Decipher	$m + 2.4$	**3.6**
Encrypt/Decrypt	$m + 3.8$	**3.6**
Reject invalid ciphertext	$0.4\,m + 3$	**3.2**
Process AD	$0.4\,m$	**0.4**
Setup 128-bit key	**2.4**	**0.8**

Experimental implementation:

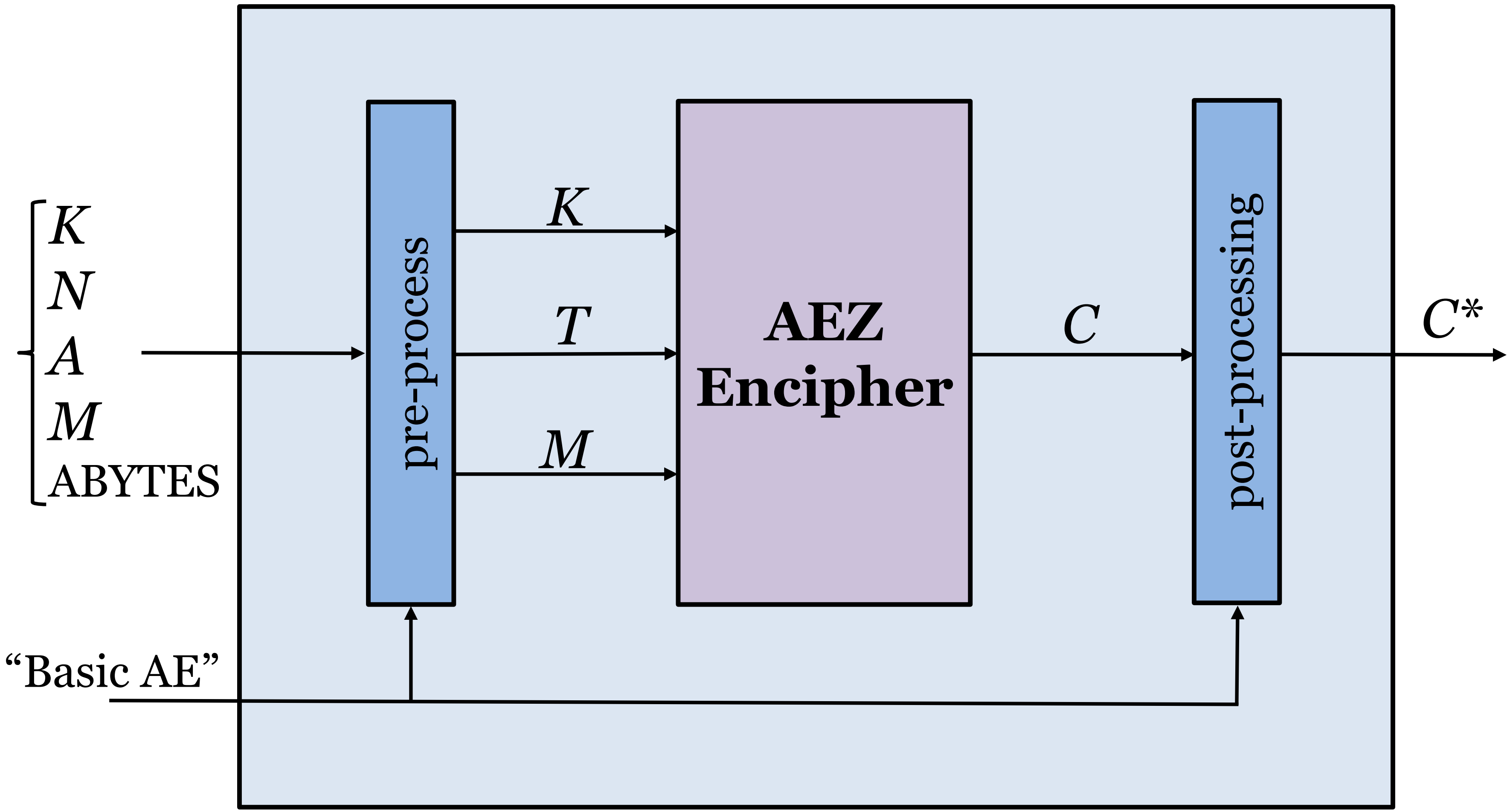
0.75 cpb (4Kb, Haswell)
0.69 cpb (marginal cost, Haswell)
(cf. the CTR, OCB: **0.64 cpb**)

AEZ Extensions

A wrapper to realize additional functionality



AEZ-Encrypt is Already an Extension
of its underlying enciphering scheme



Functionality Deliverable via AEZ Extensions

1. Secret Message Numbers — By encoding the SMN into the plaintext
2. Plaintext length-obfuscation — By padding (eg, to 2^n blocks)
3. Salting passwords — By encoding the salt in with the key
4. Slow PW-processing — By iterating a permutation
5. Convenient ciphertext alphabet — By, eg, base64url [RFC 4648] encoding
6. Vector-valued plaintexts and AD — By argument-encoding

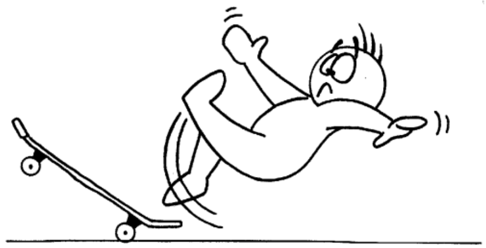
Arbitrary-length keys could have been delivered by an AEZ Extension, but were put into AEZ itself.

AEZ Conclusions

- Getting the **strongest** security & versatility guarantee is **not** expensive
  - **Cost**(Robust AE) ≈ **Cost** (AES-CTR)

- Properly done, deterministic encryption can be **good**:
  - Eliminates need for coins and state
  - Shortens ciphertexts
  - Main security concern – equality leakage – is often irrelevant
  - Frustrates one line of mass-surveillance [Bellare, Paterson, Rogaway 14]

[Bellare, Boldyreva, Knudsen, Namprempre 2001]
[Boldyreva, Taesombut 2004], [Rogaway, Zhang 2011]
[Fleischmann, Forler, Lucks, Wenzel 2012]



Online AE

- Requires a parameter — OAE[n] — to be meaningful.
- With fixed n: makes an implementation characteristic a security goal.
- Does not approximate best-possible security for an online scheme.
- Far weaker than MRAE — no exploitation of novelty or redundancy
- Notion will not be understandable by users. Attacks likely.
- Name: ~~*Online-MR*~~ ~~*max-online MR*~~

Paper on this in the coming months.

```
algorithm Encrypt(K, N, A, M)                                  // AEZ authenticated encryption
X ← M ∥ [0]^ABYTES
T ← Format(N, A)
if M = ε then return AMac(K, T)[1..ABYTES]
C ← Encipher(K, T, X)
return C
```

```
algorithm Decrypt(K, N, A, C)                                  // AEZ authenticated decryption
T ← Format(N, A)
if ∥C∥ ≤ ABYTES then if (C = AMac(K, T)[1..ABYTES]) then return ε else return ⊥
X ← Decipher(K, T, C)
M ∥ Z ← X where ∥Z∥ = ABYTES
if (Z = [0]^ABYTES) then return M else return ⊥
```

```
algorithm Format(N, A)                                         // Encode inputs and parameters
if ∥N∥ ≤ 11 then return 00 ∥ (ABYTES)_6 ∥ EXTNS ∥ N ∥ 10* ∥ A
if ∥N∥ = 12 then return 01 ∥ (ABYTES)_6 ∥ EXTNS ∥ N ∥ A
if ∥N∥ ≥ 13 then return 10 ∥ (ABYTES)_6 ∥ EXTNS ∥ N[1..12] ∥ A ∥ 10* ∥ N[13..∥N∥] ∥ [∥N∥]_8
```

```
algorithm Encipher(K, T, X)                                    // AEZ enciphering
if ∥X∥ < 32 then return EncipherFF0(K, T, X)
if ∥X∥ ≥ 32 then return EncipherEME4(K, T, X)
```

algorithm EncipherFF0(K, T, M) // FF0 enciphering
$m \leftarrow |M|$; $n \leftarrow m/2$; $\Delta \leftarrow \mathrm{AHash}(K,T)$
if $m = 8$ **then** $k \leftarrow 24$ **else if** $m = 16$ **then** $k \leftarrow 16$ **else if** $m < 128$ **then** $k \leftarrow 10$ **else** $k \leftarrow 8$
$L \leftarrow M(1\,..\,n)$; $R \leftarrow M(n+1\,..\,m)$; **if** $m \geq 128$ **then** $j \leftarrow 5$ **else** $j \leftarrow 6$
for $i \leftarrow 0$ **to** $k-1$ **do** $R' \leftarrow L \oplus ((\mathrm{E}_K^{0,j}(\Delta \oplus R10^* \oplus [i]_{16}))(1..n))$; $L \leftarrow R$; $R \leftarrow R'$ **od**; $C \leftarrow R \parallel L$
if $m < 128$ **then** $C \leftarrow C \oplus (\mathrm{E}_K^{0,7}(\Delta \oplus (C \vee 10^*)) \wedge 10^*)$
return C

algorithm EncipherEME4(K, T, M) // EME4 enciphering
$\Delta \leftarrow \mathrm{AHash}(K,T)$; $(M_0, M_0', \ldots, M_m, M_m', M_*, M_{**}) \leftarrow M$; $d \leftarrow |M| \bmod 256$
for $i \leftarrow 1$ **to** m **do** $X_i' \leftarrow M_i \oplus \mathrm{E}_K^{1,i}(M_i')$; $X_i \leftarrow M_i' \oplus \mathrm{E}_K^{0,0}(X_i')$ **od**
if $d = 0$ **then** $X \leftarrow X_1 \oplus \cdots \oplus X_m \oplus \mathbf{0}$ **else if** $d \leq 127$ **then** $X \leftarrow X_1 \oplus \cdots \oplus X_m \oplus \mathrm{E}_K^{0,3}(M_*10^*)$
else $X \leftarrow X_1 \oplus \cdots \oplus X_m \oplus \mathrm{E}_K^{0,3}(M_*) \oplus \mathrm{E}_K^{0,4}(M_{**}10^*)$ **fi**
$R \leftarrow M_0 \oplus \mathrm{E}_K^{0,1}(M_0' \oplus X) \oplus \Delta$; $R' \leftarrow M_0' \oplus \mathrm{E}_K^{-1,1}(R) \oplus X$; $S \leftarrow R \oplus R'$
for $i \leftarrow 1$ **to** m **do** $Z \leftarrow \mathrm{E}_K^{2,i}(S)$; $Y_i \leftarrow X_i' \oplus Z$; $Y_i' \leftarrow X_i \oplus Z$; $C_i' \leftarrow Y_i \oplus \mathrm{E}_K^{0,0}(Y_i')$; $C_i \leftarrow Y_i' \oplus \mathrm{E}_K^{1,i}(C_i')$ **od**
if $d = 0$ **then** $C_* \leftarrow C_{**} \leftarrow \varepsilon$; $Y \leftarrow Y_1 \oplus \cdots \oplus Y_m \oplus \mathbf{0}$
else if $d \leq 127$ **then** $C_* \leftarrow M_* \oplus \mathrm{E}_K^{-1,3}(S)$; $C_{**} \leftarrow \varepsilon$; $Y \leftarrow Y_1 \oplus \cdots \oplus Y_m \oplus \mathrm{E}_K^{0,3}(C_*10^*)$
else $C_* \leftarrow M_* \oplus \mathrm{E}_K^{-1,3}(S)$; $C_{**} \leftarrow M_{**} \oplus \mathrm{E}_K^{-1,4}(S)$; $Y \leftarrow Y_1 \oplus \cdots \oplus Y_m \oplus \mathrm{E}_K^{0,3}(C_*) \oplus \mathrm{E}_K^{0,4}(C_{**}10^*)$ **fi**
$C_0'' \leftarrow R \oplus \mathrm{E}_K^{-1,2}(R')$; $C_0 \leftarrow R' \oplus \mathrm{E}_K^{0,2}(C_0'') \oplus \Delta$; $C_0' \leftarrow C_0'' \oplus Y$
return $C_0 C_0' \cdots C_m C_m'\, C_* C_{**}$

algorithm AHash(K, A) // AXU hash
$(A_0, \ldots, A_m) \leftarrow A$
if $|A_m| \bmod 128 = 0$ **then return** $\mathrm{E}_K^{3,0}(A_0) \oplus \mathrm{E}_K^{3,1}(A_1) \oplus \cdots \oplus \mathrm{E}_K^{3,m}(A_m)$
if $|A_m| \bmod 128 \neq 0$ **then return** $\mathrm{E}_K^{3,0}(A_0) \oplus \mathrm{E}_K^{3,1}(A_1) \oplus \cdots \oplus \mathrm{E}_K^{3,m-1}(A_{m-1}) \oplus \mathrm{E}_K^{1,0}(A_m 10^*)$

---

algorithm AMac(K, A) // PRF
return $\mathrm{E}_K^{-1,5}(\mathrm{AHash}_K(A))$

---

algorithm $\mathrm{E}_K^{i,j}(X)$ // TBC on $\mathcal{T} = \{0\} \times [0..7] \cup \{1,2,3\} \times \mathbb{N}$
$(J, L, K_0, K_1, K_2, K_3) \leftarrow \mathrm{Expand}(\mathrm{Extract}(K))$
$\boldsymbol{k}_0 \leftarrow (K_0, K_1, K_2, K_3, \mathbf{0})$; $\boldsymbol{k}_2 \leftarrow (K_2, K_3, K_0, K_1, \mathbf{0})$
$\boldsymbol{k}_1 \leftarrow (K_1, K_2, K_3, K_0, \mathbf{0})$; $\boldsymbol{k}_3 \leftarrow (K_3, K_0, K_1, K_2, \mathbf{0})$
$\boldsymbol{K} \leftarrow (L, J, 2J, 4J, K_0, K_1, K_2, K_3, K_0, K_1, K_2)$
if $i = -1$ **then return** $\mathrm{AES}_{\boldsymbol{K}}(X \oplus jJ)$
if $i = 0$ **or** $j = 0$ **then return** $\mathrm{AES4}_{\boldsymbol{k}_i}(X \oplus jJ)$
return $\mathrm{AES4}_{\boldsymbol{k}_i}(X \oplus (j \bmod 8)J \oplus 2^{\lfloor (j-1)/8 \rfloor} L)$

---

algorithm Extract(K) // Convert key to 256 bits
$\boldsymbol{z} \leftarrow [0][1][2]\cdots[15]$; **for** $i \leftarrow 1$ **to** 7 **do** $C_i \leftarrow \mathrm{AES4}_{(\boldsymbol{z},\boldsymbol{z},\boldsymbol{z},\boldsymbol{z},\boldsymbol{z})}([i]^{16})$
$\boldsymbol{a} \leftarrow (\mathbf{0}, C_1, C_2, C_3, \mathbf{0})$; $\boldsymbol{b} \leftarrow (\mathbf{0}, C_4, C_5, C_6, \mathbf{0})$; $C \leftarrow C_7$
$(I_0, \ldots, I_m) \leftarrow K$; **if** $\|I_m\| = 16$
then $J \leftarrow \mathrm{AES4}_{\boldsymbol{a}}(I_0 \oplus C) \oplus \mathrm{AES4}_{\boldsymbol{a}}(I_1 \oplus 2C) \oplus \mathrm{AES4}_{\boldsymbol{a}}(I_2 \oplus 2^2\,C) \oplus \cdots \oplus \mathrm{AES4}_{\boldsymbol{a}}(I_m \oplus 2^m\,C)$
$\quad L \leftarrow \mathrm{AES4}_{\boldsymbol{b}}(I_0 \oplus C) \oplus \mathrm{AES4}_{\boldsymbol{b}}(I_1 \oplus 2C) \oplus \mathrm{AES4}_{\boldsymbol{b}}(I_2 \oplus 2^2\,C) \oplus \cdots \oplus \mathrm{AES4}_{\boldsymbol{b}}(I_m \oplus 2^m\,C)$
else $J \leftarrow \mathrm{AES4}_{\boldsymbol{a}}(I_0 \oplus C) \oplus \mathrm{AES4}_{\boldsymbol{a}}(I_1 \oplus 2C) \oplus \mathrm{AES4}_{\boldsymbol{a}}(I_2 \oplus 2^2\,C) \oplus \cdots \oplus \mathrm{AES4}_{\boldsymbol{a}}(I_m 10^* \oplus 3C)$
$\quad L \leftarrow \mathrm{AES4}_{\boldsymbol{b}}(I_0 \oplus C) \oplus \mathrm{AES4}_{\boldsymbol{b}}(I_1 \oplus 2C) \oplus \mathrm{AES4}_{\boldsymbol{b}}(I_2 \oplus 2^2\,C) \oplus \cdots \oplus \mathrm{AES4}_{\boldsymbol{b}}(I_m 10^* \oplus 3C)$
return $J \,\|\, L$

---

algorithm Expand(K) // Map 256-bit string to vector of 128-bit subkeys
$(J, L) \leftarrow K$; $\boldsymbol{k} \leftarrow (J, L, 2J, L, 4J)$
for $i \leftarrow 0$ **to** 3 **do** $K_i \leftarrow \mathrm{AES4}_{\boldsymbol{k}}([i]^{16})$
return $(J, L, K_0, K_1, K_2, K_3)$