

AEZ: Anything-but EaZy in Hardware



**Ekawat Homsirikamol,
& Kris Gaj
George Mason University
USA**

Based on work partially supported by NSF under Grant No. 1314540.

Special thanks to the authors of AEZ:
Phillip Rogaway, Ted Krovetz, and Viet Tung Hoang.

First Author



Ekawat Homsirikamol
a.k.a “Ice”

PhD Thesis defense on Nov. 18, 2016;
currently with
Cadence Design Systems
in San Jose, CA

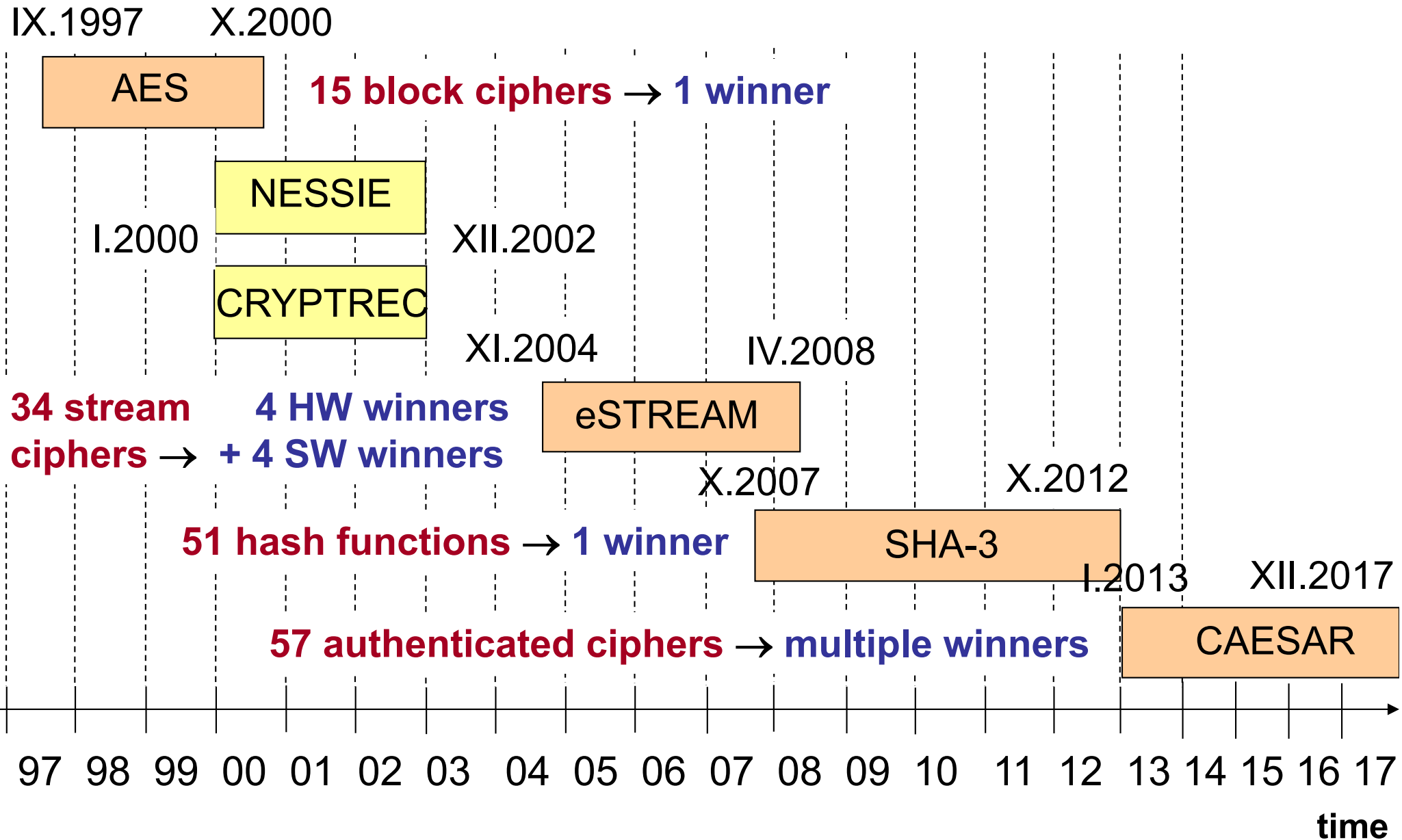
Outline

- **Introduction & Motivation**
- **CAESAR Hardware API**
- **Hardware Architecture of AEZ**
- **Results & Discussions**
- **Conclusions & Future Work**



**Introduction &
Motivation**

Cryptographic Standard Contests



CAESAR Competition

Goal: A portfolio of new-generation authenticated ciphers

- offering **advantages over AES-GCM**
- suitable for **wide-spread adoption**

Period: March 2014 - December 2017 (tentative)

Organizer: An informal committee of leading cryptographic experts

Number of candidates:

57 → 29 → 15 → ? → ?
R1 R2 R3 finalists portfolio

CAESAR Recent and Upcoming Milestones



2016.06.30: Round 2 VHDL/Verilog Code

2016.08.15: Announcement of 15 Round 3 candidates

2016.09.15: Round 3 tweaks

2016.09.25-27: DIAC 2016 - Directions in Authenticated Ciphers

2016.10.15: Round 3 software

2017.04.15 (tentative): Round 3 VHDL/Verilog code

CAESAR Three Use Cases

Use Case 1: Lightweight applications (constrained environments)

Use Case 2: High-performance applications

Use Case 3: Defense in depth

- **critical: authenticity despite nonce misuse**
- **desirable: limited privacy damage from nonce misuse**
- desirable: authenticity despite release of unverified plaintexts
- desirable: limited privacy damage from release of unverified plaintexts
- desirable: robustness in more scenarios; e.g., huge amounts of data

AEZ Strong Security Notions

Misuse Resistant Authenticated Encryption (MRAE)

= authenticity and privacy even if nonce is repeated

Robust Authenticated Encryption (RAE)

= MRAE for any choice of ciphertext expansion
(including no expansion at all)

Advantages: easy to use, less prone to implementation errors

Disadvantages: two-pass, affecting speed and memory requirements

CAESAR Candidates Targeting MRAE

Round 2:

AEZ, Deoxys=, HS1-SIV, Joltik=

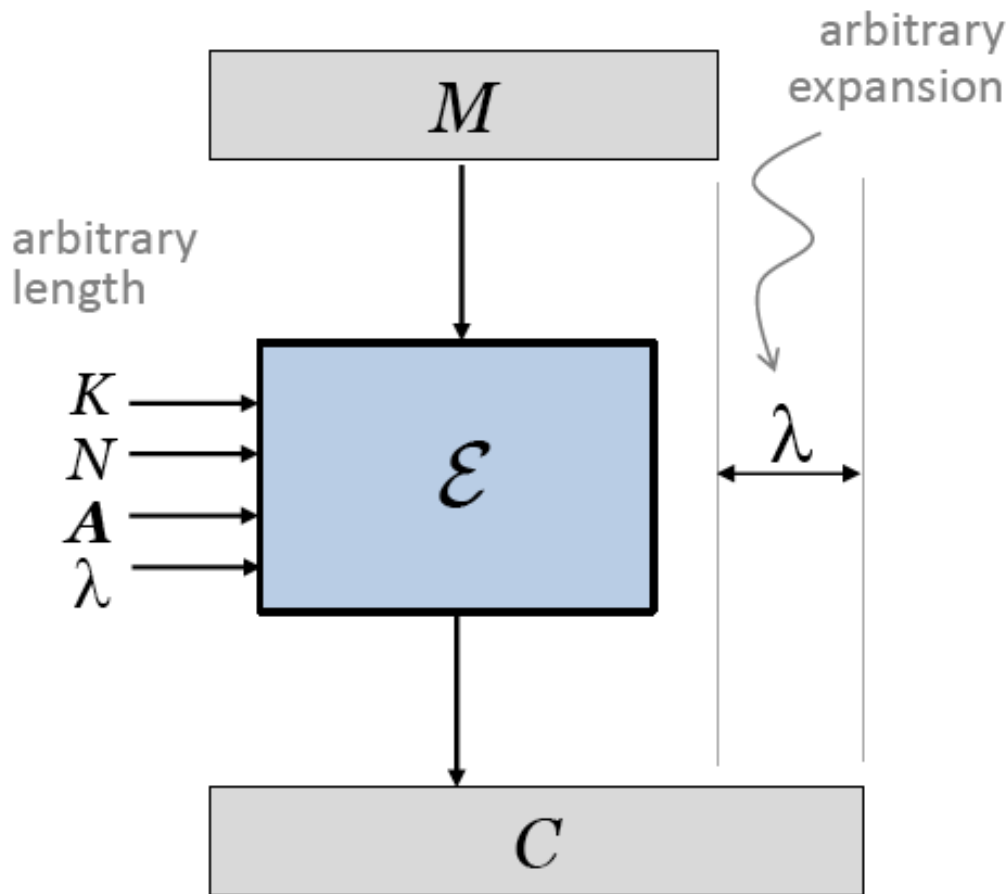
Round 3:

AEZ, Deoxys-II

Robust Authenticated Encryption Scheme

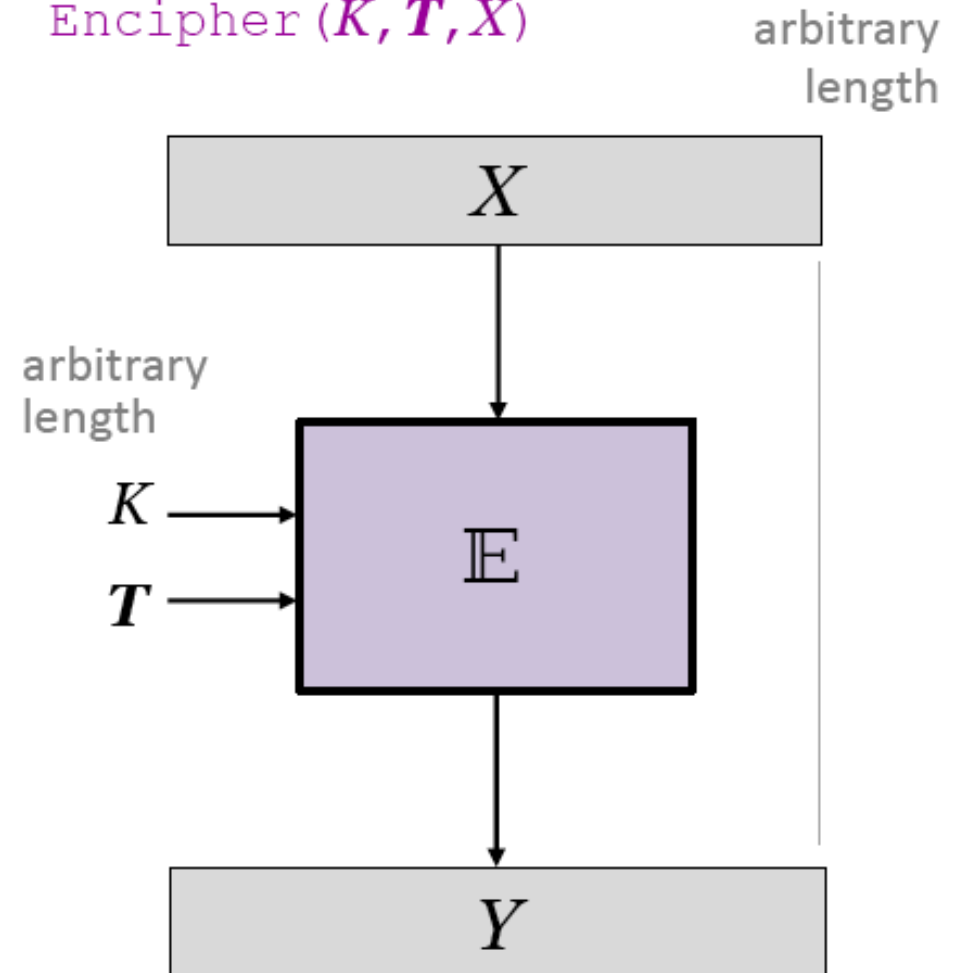
An Robust-AE scheme

Encrypt (K, N, A, M, λ)



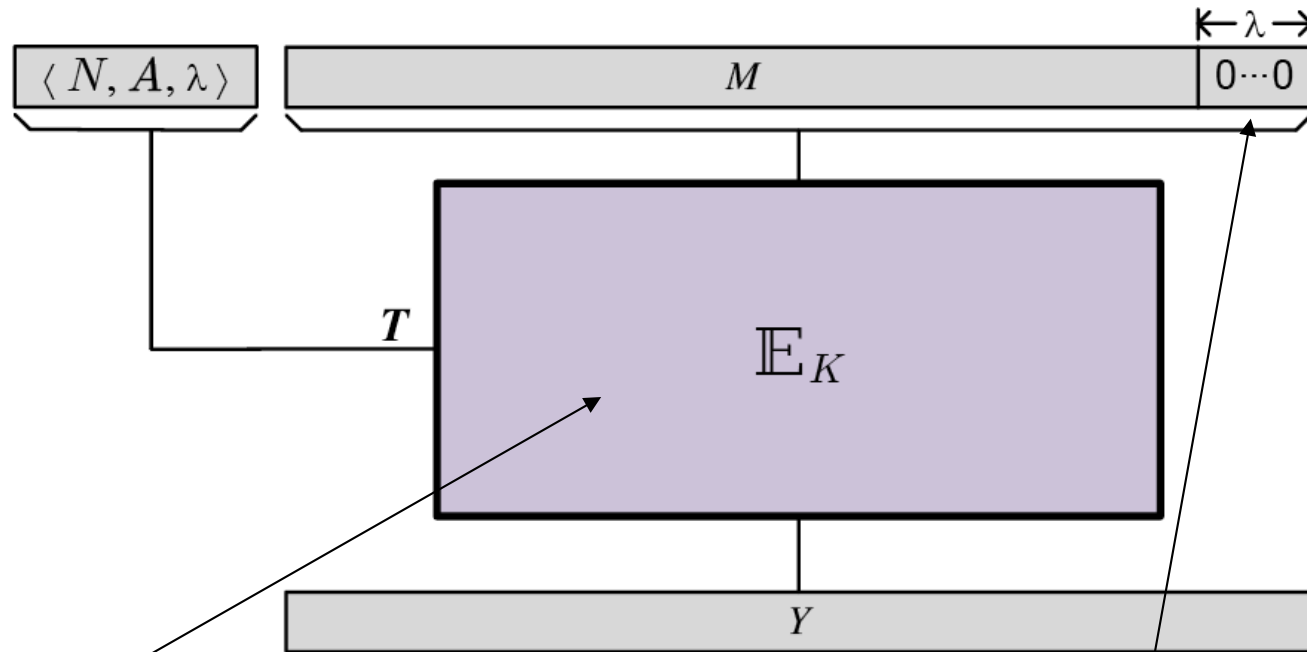
A Generalized Blockcipher

Encipher (K, T, X)



K – Key, N – Nonce, A – Associated Data, λ - Ciphertext Expansion, T - Tweak

Using Generalized Block Cipher to realize Robust Authenticated Encryption (RAE) Scheme



Generalized Block Cipher

- Arbitrary input size in bytes
- Output size = Input size
- Tweak - non-secret value that individualizes the permutation associated with the key

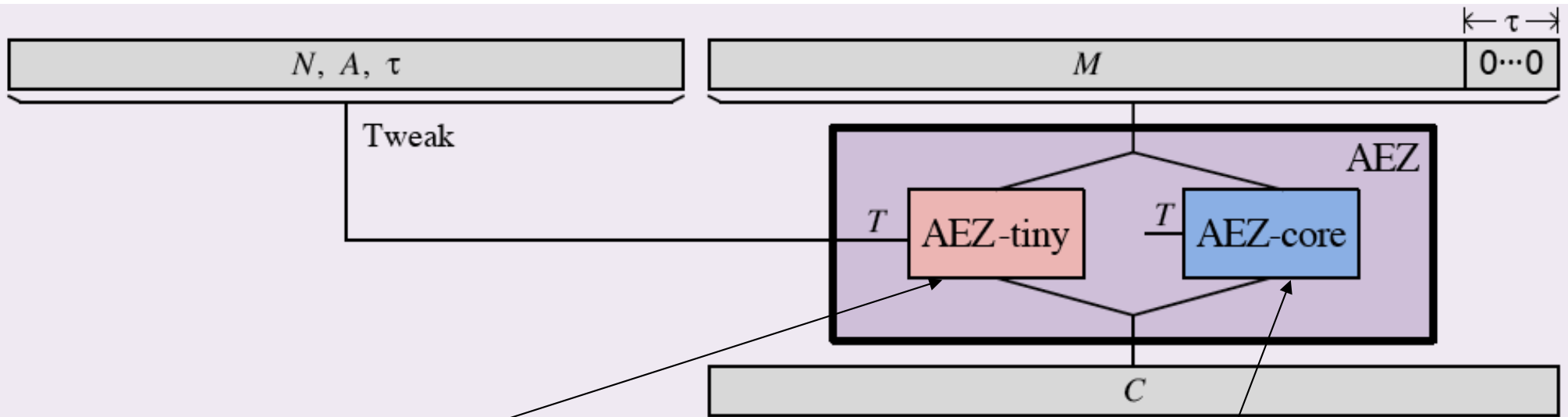
Authenticator

- String of τ zeros
- N – Nonce
 A – Associated Data
 τ – Authenticator length (in bits)
a.k.a. Ciphertext Expansion

Strong Security Properties

- a) If **(M, A) tuples are known not to repeat**, no nonce is needed.
- b) **Nonce repetitions**: privacy loss is limited to revealing repetitions in (N, A, M) tuples, authenticity not damaged at all.
- c) **Any authenticator-length can be selected**, achieving best-possible authenticity for this amount of ciphertext expansion.
- d) If there's **redundancy in plaintexts**, whose presence is verified on decryption, this **augments authenticity**.
- e) By last two properties: one can **minimize length-expansion for bandwidth-constrained apps**.

Structure of AEZ



AEZ-tiny:

For strings < 32 bytes

AES4-based

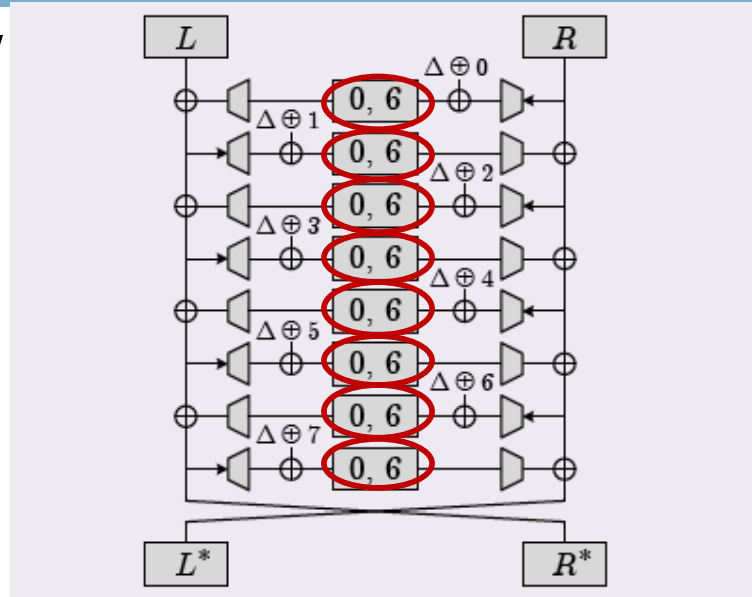
AEZ-core:

For strings ≥ 32 bytes

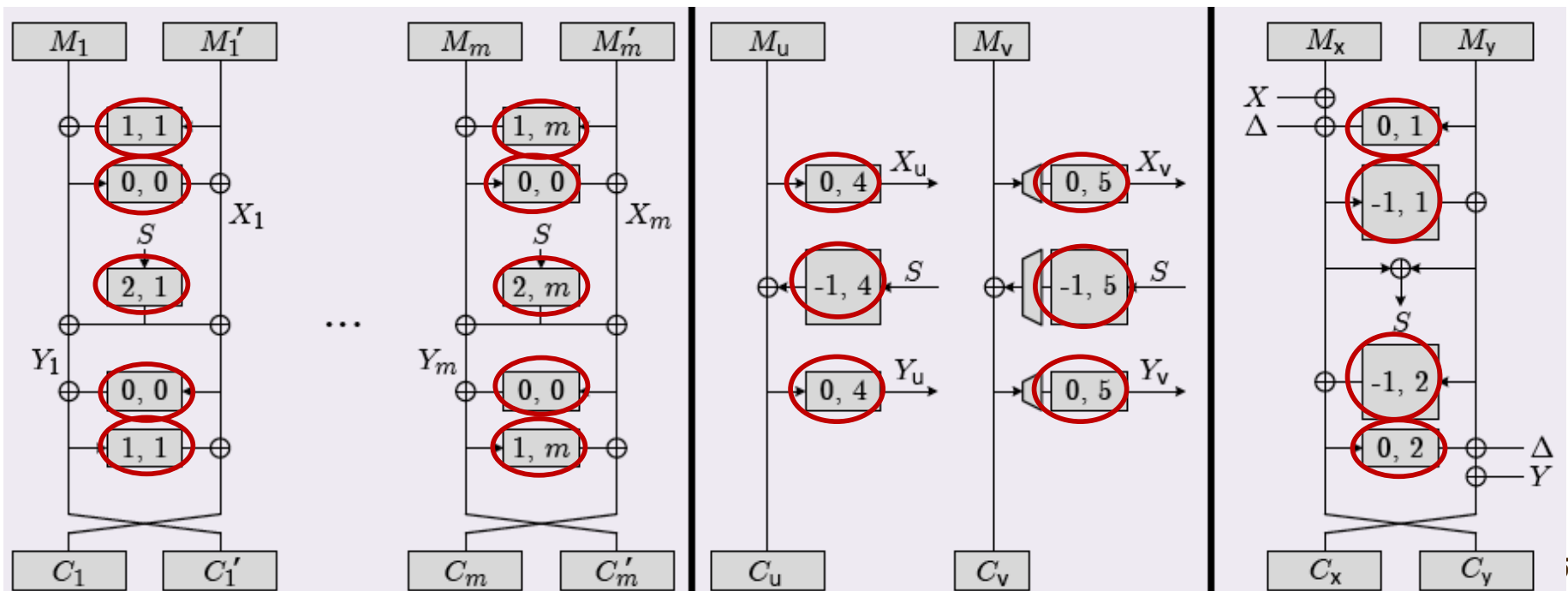
AES4 and AES based

Basic Building Block: Tweakable Block Cipher - TBC

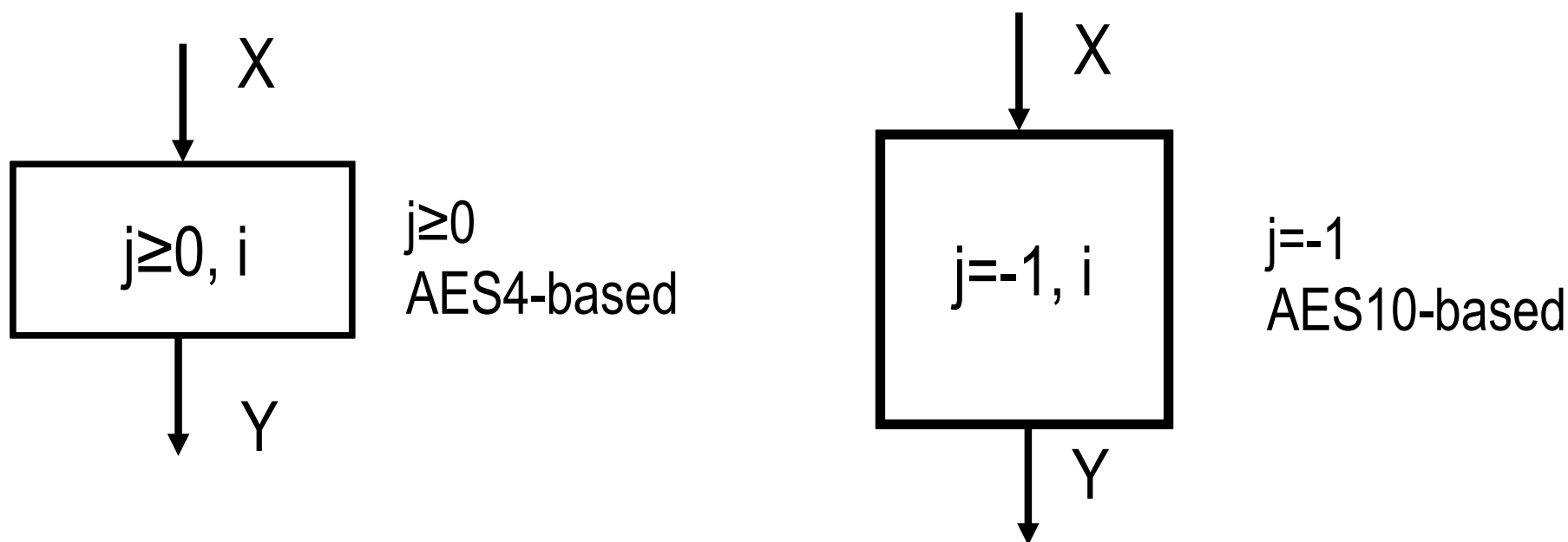
AEZ-tiny



AEZ-core



Tweakable Block Cipher - Notation



```

400 algorithm  $E_K^{j,i}(X)$  // Scaled-down TBC
401  $I \parallel J \parallel L \leftarrow \text{Extract}(K)$  where  $|I| = |J| = |L| = 128$ 
402  $K \leftarrow (\mathbf{0}, I, J, L, I, J, L, I, J, L, I)$ 
403 if  $j = -1$  then return  $\text{AES10}_K(X \oplus iJ)$ 
404  $k \leftarrow k_1 \leftarrow (\mathbf{0}, J, I, L, \mathbf{0})$ ;  $k_2 \leftarrow (\mathbf{0}, L, I, J, L)$ 
405 if  $j = 0$  then  $\Delta \leftarrow iI$ ; return  $\text{AES4}_k(X \oplus \Delta)$  fi
406 if  $1 \leq j \leq 2$  then  $\Delta \leftarrow (2^{3+\lfloor(i-1)/8\rfloor} + ((i-1) \bmod 8))I$ ; return  $\text{AES4}_{k_j}(X \oplus \Delta)$  fi
407 if  $j \geq 3$  and  $i = 0$  then  $\Delta \leftarrow 2^{j-3} \cdot L$ ; return  $\text{AES4}_k(X \oplus \Delta) \oplus \Delta$  fi
408  $\Delta \leftarrow 2^{j-3} \cdot L \oplus (2^{3+\lfloor(i-1)/8\rfloor} + ((i-1) \bmod 8))J$ ; return  $\text{AES4}_k(X \oplus \Delta) \oplus \Delta$ 

```


Warnings by the Authors of AEZ

“Writing software for AEZ is not easy, while doing a hardware design for AEZ is far worse”

“From the hardware designer’s perspective, AEZ’s name might seem ironic, the name better suggesting *anti-easy*, the *antithesis of easy*, or *anything-but easy*”

V. T. Hoang, T. Krovetz, and P. Rogaway,
specification of AEZ v4.1, October 2015

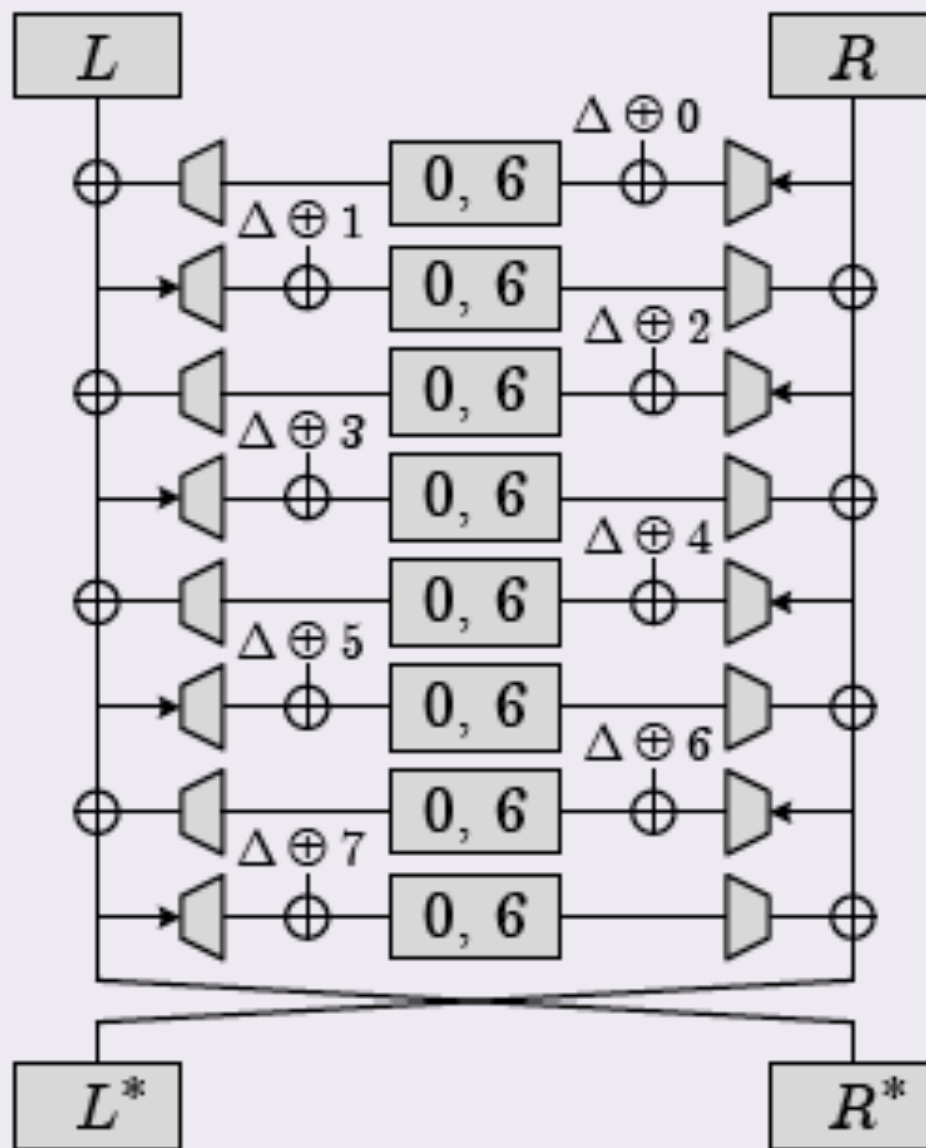
Hardware Implementation Challenges (1)

- **Three algorithms in one**
 - a. AEZ-prf to process empty messages
 - b. AEZ-tiny to process messages of the size smaller than 32 - authenticator length (in bytes)
(= 16 bytes for recommended values of parameters)
 - c. AEZ-core to process all remaining message sizes.

Dilemma:

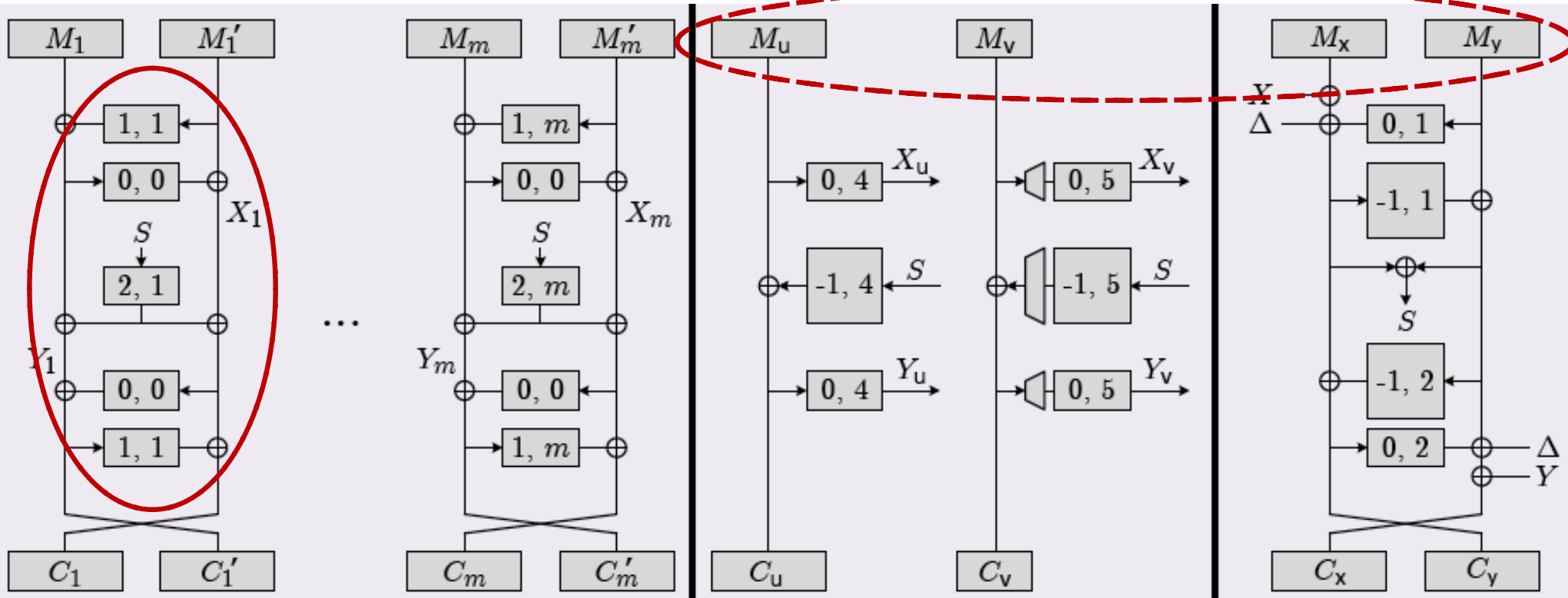
- **Resource sharing: decreases area, complicates scheduling**
- **No resource sharing: increases area, simplifies scheduling**

AEZ-tiny



1 byte: 24 rounds
2 bytes: 16 rounds
3-15 bytes: 10 rounds
16-31 bytes: 8 rounds

AEZ-core



5 · 4 AES rounds per 2 AES blocks
 = 10 AES rounds per 1 AES block

Special treatment of the
 last 4 blocks

Hardware Implementation Challenges (1)

- **Three algorithms in one**
 - a. AEZ-prf to process empty messages
 - b. AEZ-tiny to process messages of the size smaller than 32 - authenticator length (in bytes)
(= 16 bytes for recommended values of parameters)
 - c. AEZ-core to process all remaining message sizes.

Dilemma:

- Resource sharing: decreases area, complicates scheduling
- **No resource sharing: increases area, simplifies scheduling**

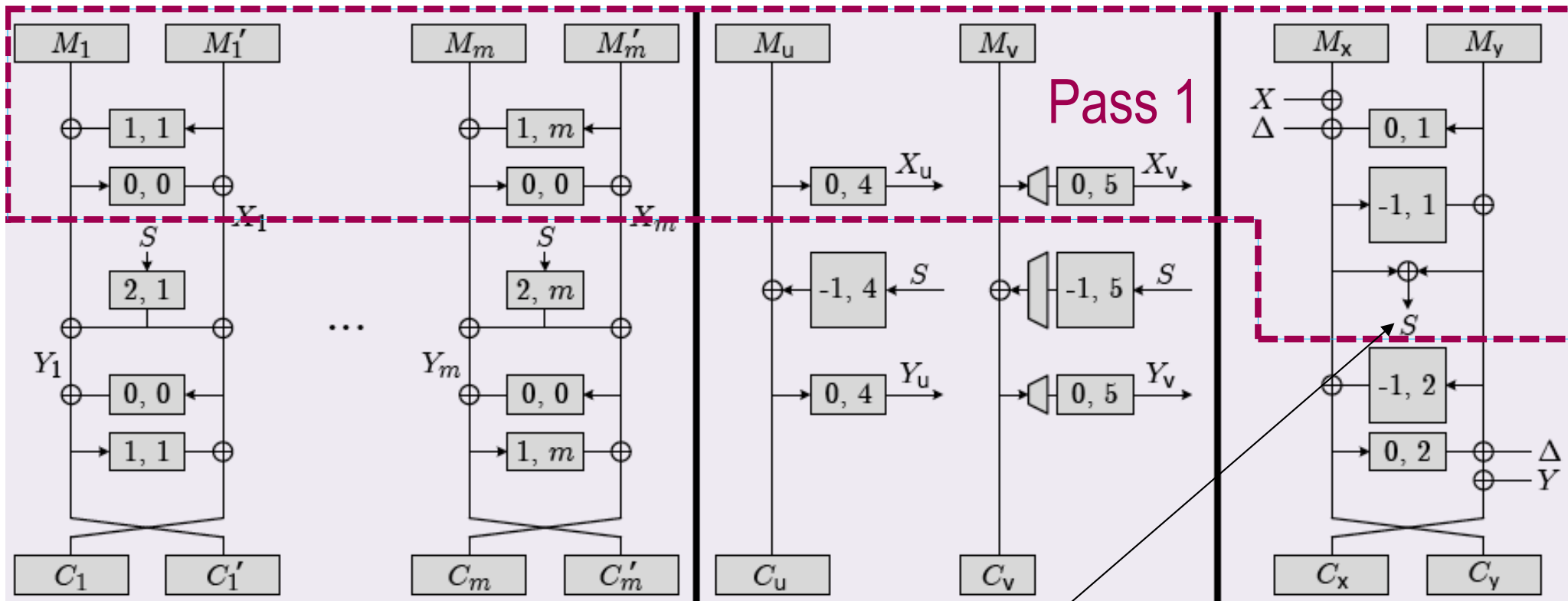
Hardware Implementation Challenges (2)

- **Two-pass algorithm (AES-core)**
 - **Pass 1 – Used to calculate S**
 - **Pass 2 – Used to calculate all output blocks**

Dilemma:

- **repeat ~40% of computations involving all message blocks, already done in the first pass**
- **store intermediate results of the size of the entire message**

AEZ-core



Goal of Pass 1

Hardware Implementation Challenges (2)

- **Two-pass algorithm (AES-core)**
 - **Pass 1 – Used to calculate S**
 - **Pass 2 – Used to calculate all output blocks**

Dilemma:

- **repeat ~40% of computations involving all message blocks, already done in the first pass**
- **store intermediate results of the size of the entire message**

Hardware Implementation Challenges (3)

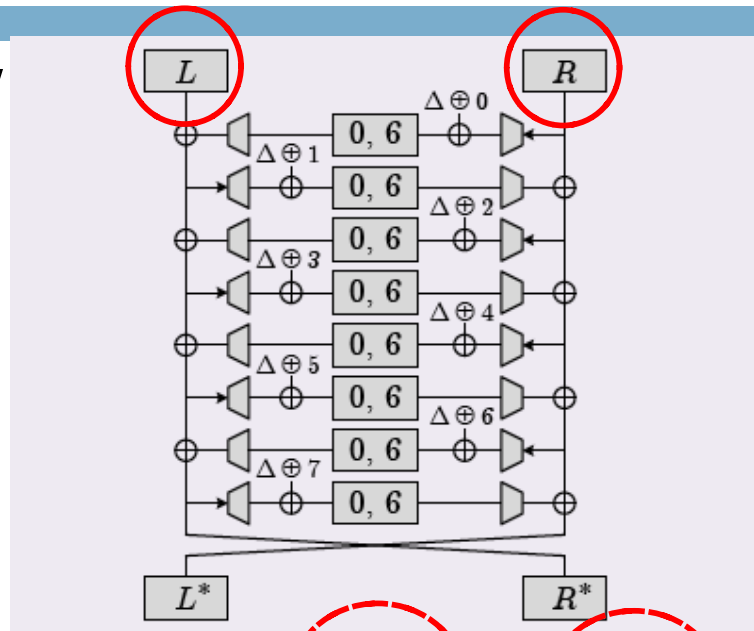
- **Input Reblocking caused by blocks with variable length dependent on the overall message size**
 - **AEZ-tiny: variable-size blocks L and R**
 - **AEZ-core: variable-size blocks M_u and M_v**
 - **Last but one pair of blocks, M_u possibly empty**

Problem:

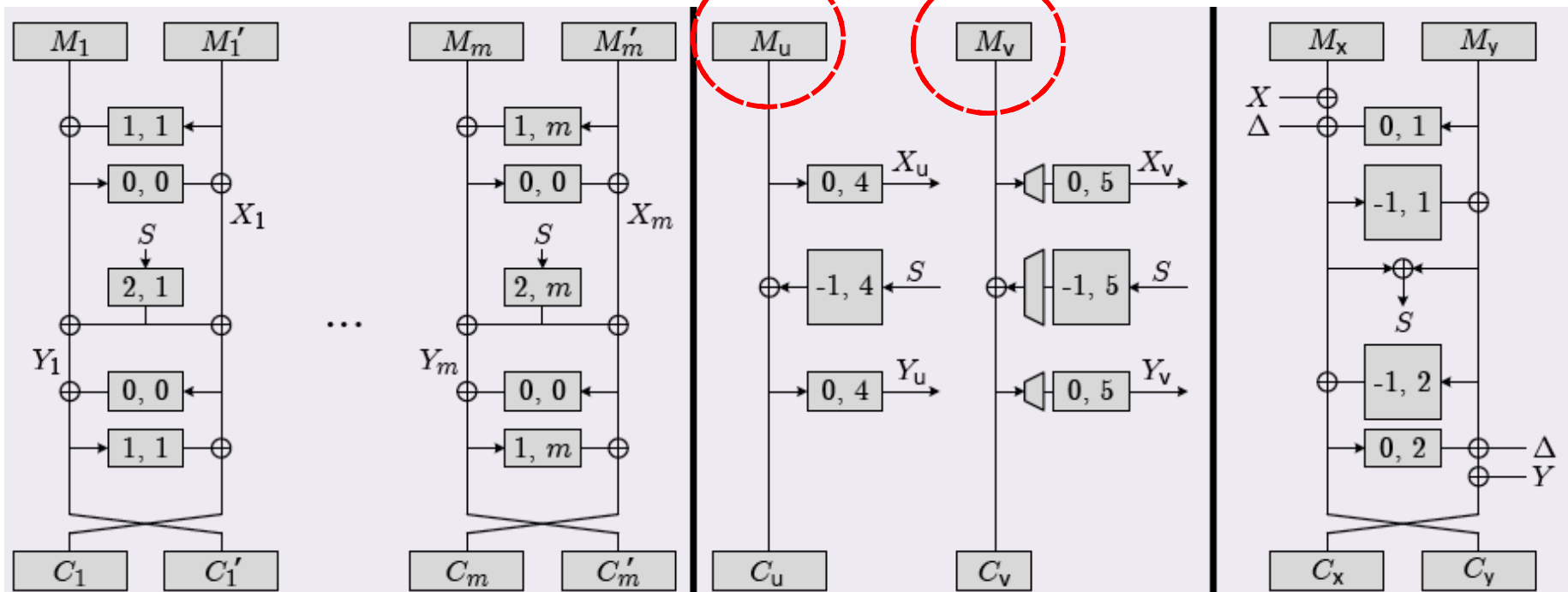
- **must internally create and process blocks of unconventional sizes**
- **requires variable shifts and rotations costly in terms of area**

Variable-size blocks

AEZ-tiny



AEZ-core



Hardware Implementation Challenges (4)

- Treatment of incomplete blocks
 - Complex padding required for blocks other than the last block of the message (M_u and M_v)
- Need for precomputations in TBC
 - Time and storage required depends on the maximum size of message and the maximum size of AD
- Complex scheduling and control

Limitations of Our Implementation

- **No support for**
 - **arbitrary key length**
 - **vector-valued Associated Data**
 - **arbitrary ciphertext expansion**

(features not supported by implementations of any other candidates)

- **Key size fixed at 384 bits**
- **Authenticator length a.k.a. Ciphertext Expansion fixed at 16 bytes = 128 bits**

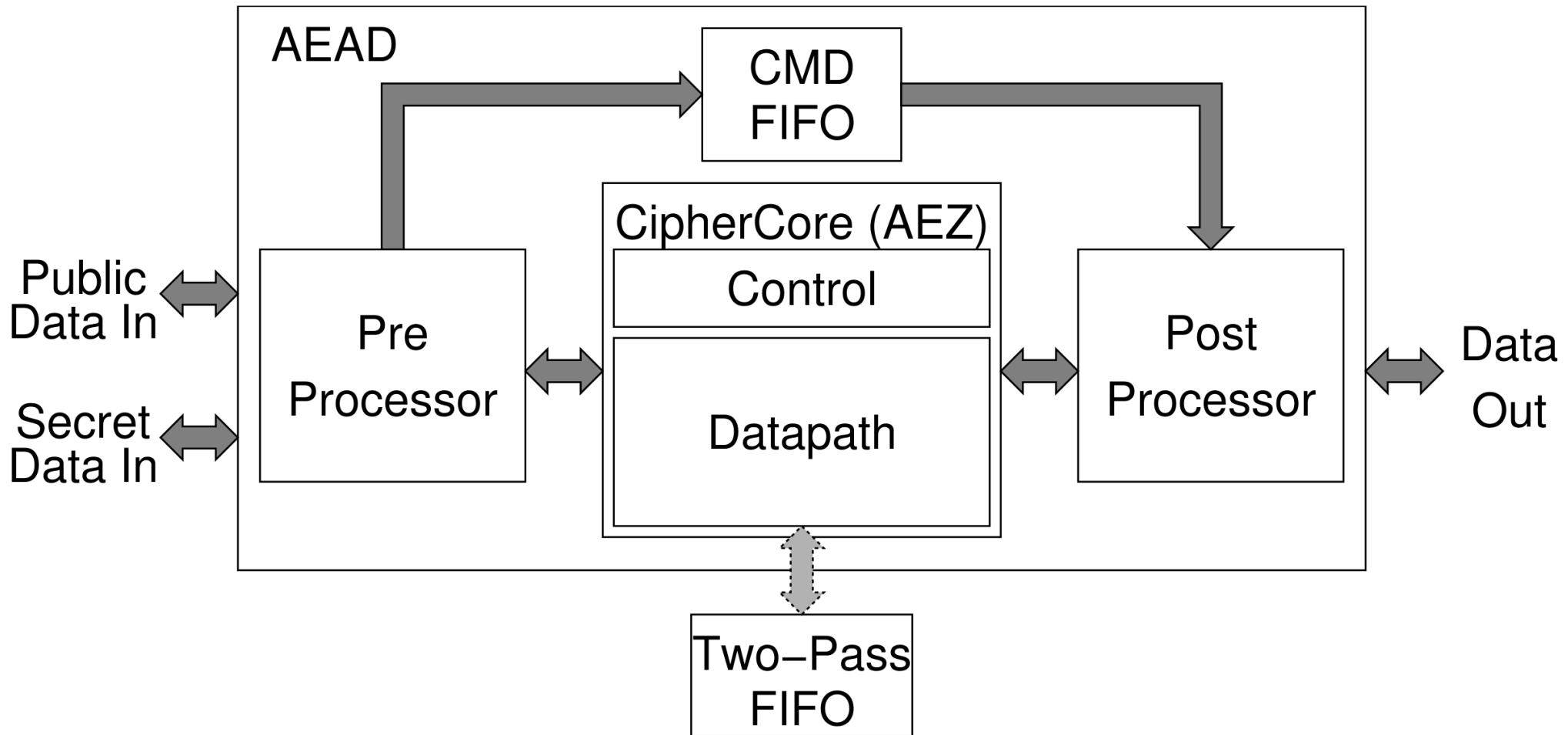


CAESAR
Hardware API

CAESAR Hardware API

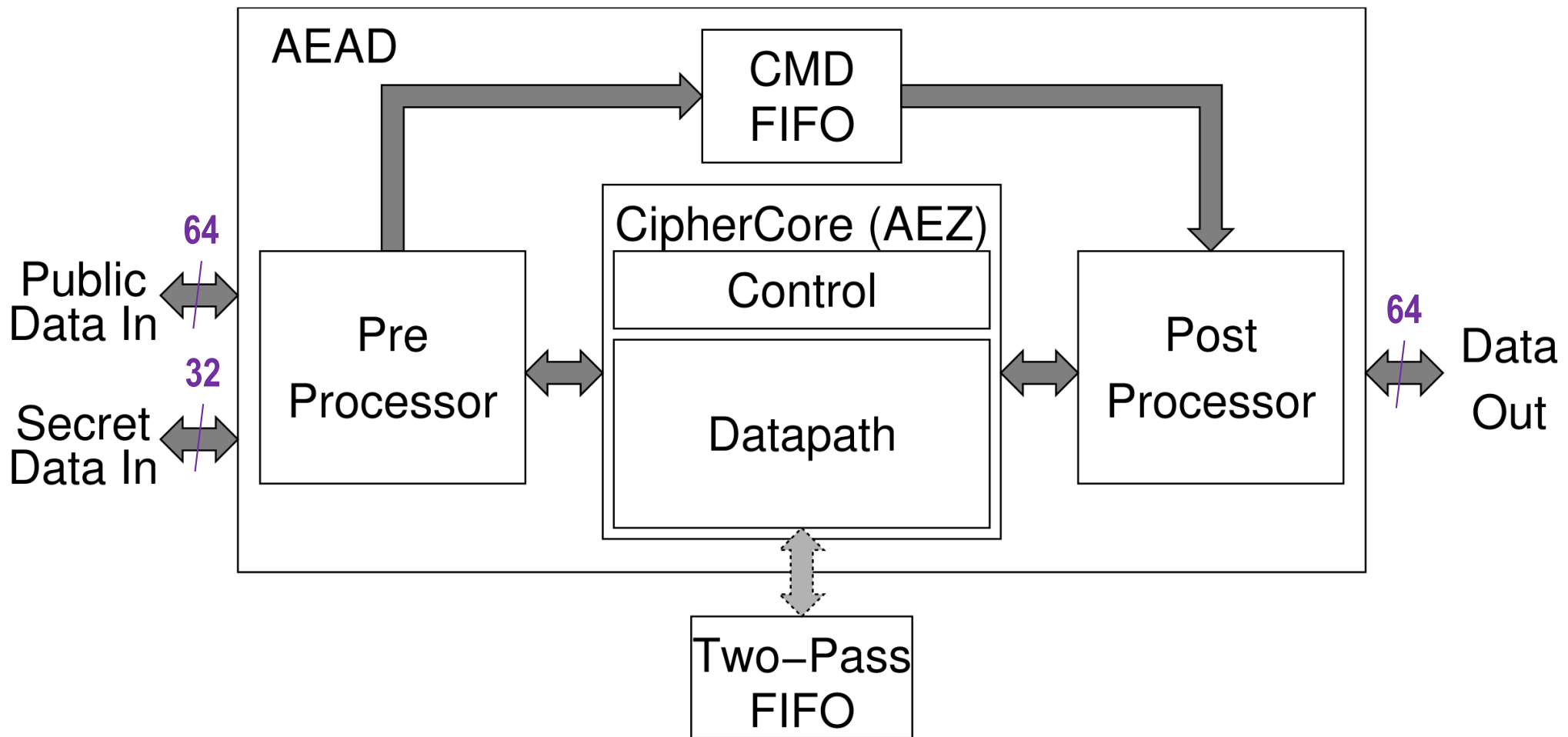
- **Specifies:**
 - Minimum compliance criteria
 - Interface
 - Communication protocol
 - Timing characteristics
- **Assures:**
 - Compatibility
 - Fairness
- **Timeline:**
 - Based on the GMU Hardware API presented at CryptArchi 2015, DIAC 2015, and ReConFig 2015
 - Revised version posted on Feb. 15, 2016
 - Officially approved by the CAESAR Committee on May 6, 2016

Top-Level Block Diagram



Pre-Processor, Post-Processor, CMD FIFO, and Two-pass FIFO
generic, common for all candidates

Top-Level Block Diagram of AEZ





**Hardware
Architecture
of AEZ**

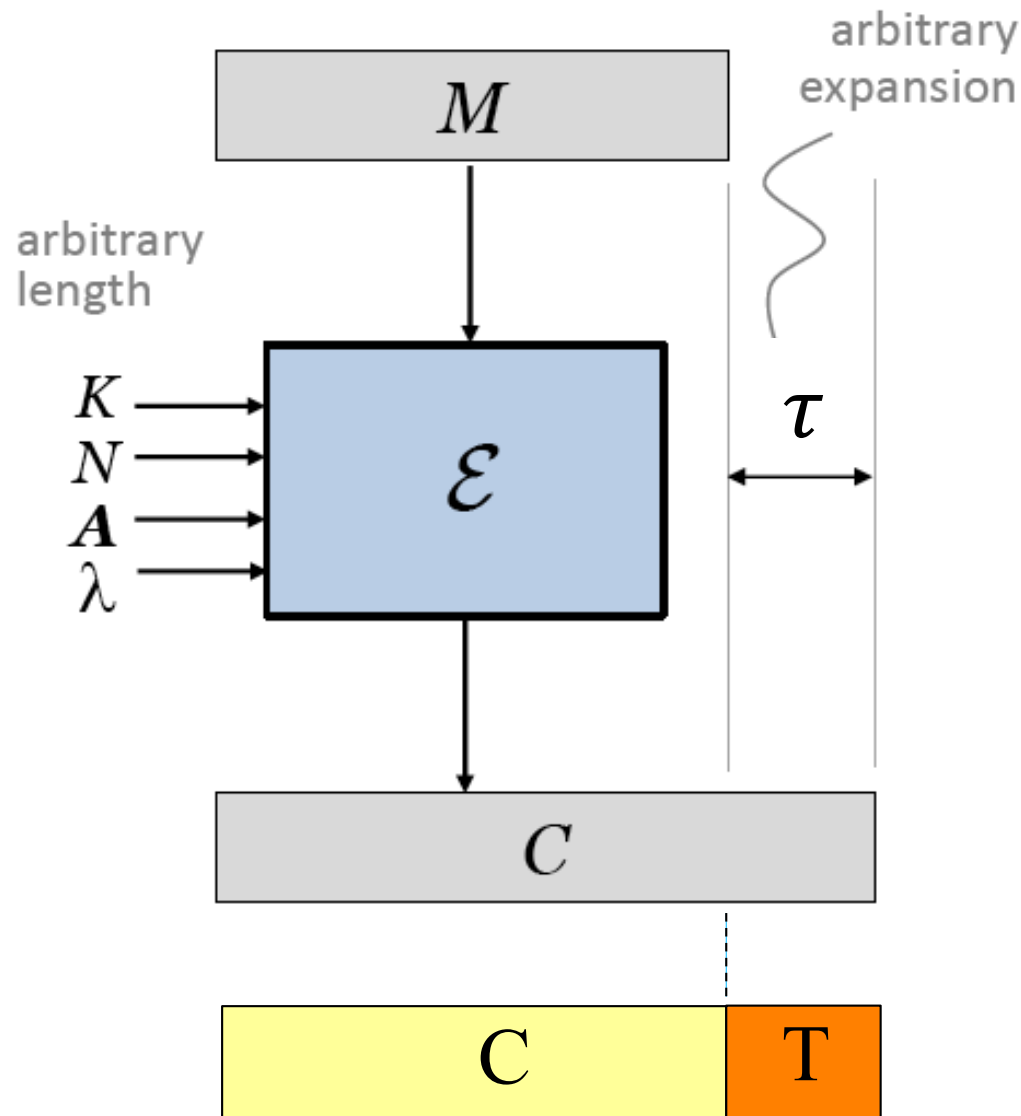
Design Parameters

- **Optimization Target**
 - **Maximum Throughput to Area ratio**
- **Operations**
 - **Encryption and decryption in one module, but only one of them performed at a time (half-duplex)**
 - **Key scheduling, padding and handling of incomplete blocks in hardware**
- **Choice of Parameters**
 - **Key length = 384 bits**
 - **Nonce length = 96 bits**
 - **Authenticator length = 16 bytes = 128 bits**

Maximum Message/AD Length

- **Selection**
 - **Maximum Message Length = $2^{11}-1$ bytes**
 - **Maximum AD Length = $2^{10}-1$ bytes**
- **Maximum Message Length (2047 bytes)**
 - **Greater than the maximum length of the Ethernet v2 packets (1500 bytes)**
 - **Limits the amount of memory required for the Two-Pass FIFO**
 - **Approved by the CAESAR Committee as a recommended maximum length for all two pass-algorithms and an optional maximum length for single-pass algorithms**

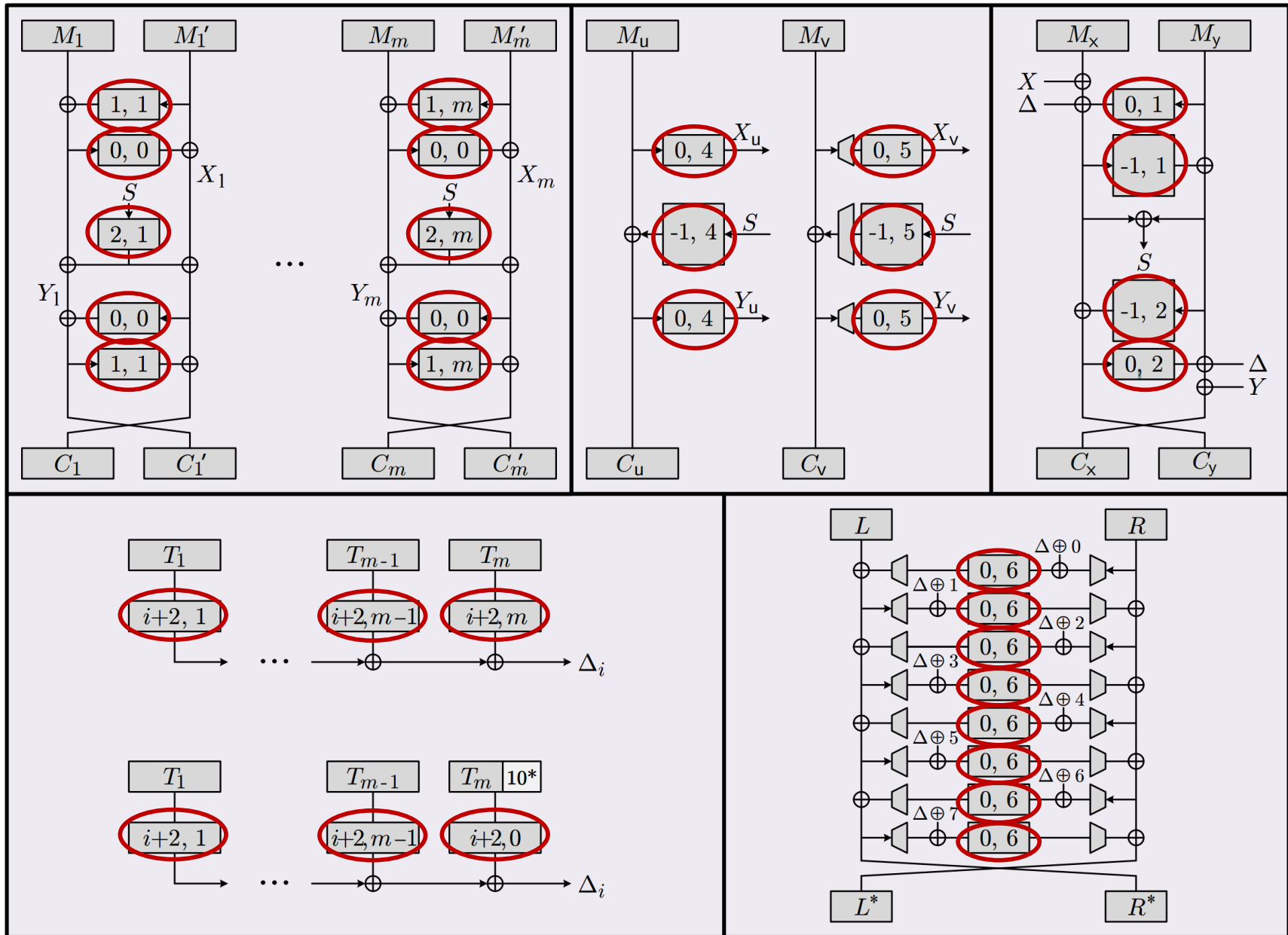
Difference Compared to Software



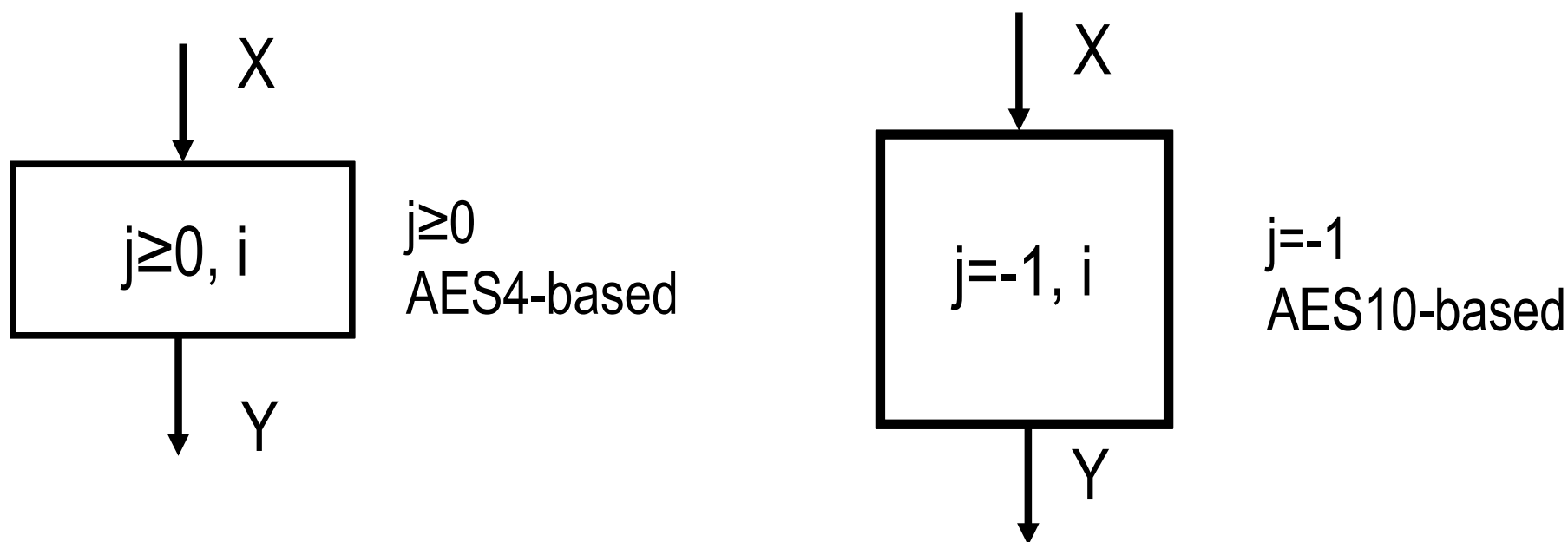
Ciphertext after expansion
divided into the Ciphertext C
and the Tag T

Low-Level Block – Tweakable Block Cipher

TBC



Tweakable Block Cipher – $E_{\mathbf{K}}^{j,i}$



```

400 algorithm  $E_{\mathbf{K}}^{j,i}(X)$  // Scaled-down TBC
401  $I \parallel J \parallel L \leftarrow \text{Extract}(\mathbf{K})$  where  $|I| = |J| = |L| = 128$ 
402  $\mathbf{K} \leftarrow (\mathbf{0}, I, J, L, I, J, L, I, J, L, I)$ 
403 if  $j = -1$  then return  $\text{AES10}_{\mathbf{K}}(X \oplus iJ)$ 
404  $\mathbf{k} \leftarrow \mathbf{k}_1 \leftarrow (\mathbf{0}, J, I, L, \mathbf{0}); \mathbf{k}_2 \leftarrow (\mathbf{0}, L, I, J, L)$ 
405 if  $j = 0$  then  $\Delta \leftarrow iI$ ; return  $\text{AES4}_{\mathbf{k}}(X \oplus \Delta)$  fi
406 if  $1 \leq j \leq 2$  then  $\Delta \leftarrow (2^{3+\lfloor (i-1)/8 \rfloor} + ((i-1) \bmod 8))I$ ; return  $\text{AES4}_{\mathbf{k}_j}(X \oplus \Delta)$  fi
407 if  $j \geq 3$  and  $i = 0$  then  $\Delta \leftarrow 2^{j-3} \cdot L$ ; return  $\text{AES4}_{\mathbf{k}}(X \oplus \Delta) \oplus \Delta$  fi
408  $\Delta \leftarrow 2^{j-3} \cdot L \oplus (2^{3+\lfloor (i-1)/8 \rfloor} + ((i-1) \bmod 8))J$ ; return  $\text{AES4}_{\mathbf{k}}(X \oplus \Delta) \oplus \Delta$ 

```


TBC Output Y Calculations

$$Y = \text{AES-round}_{\text{Key}}(X + \Delta) \quad \text{or} \quad Y = \text{AES-round}_{\text{Key}}(X + \Delta) + \Delta$$


mode	(j, i)	Second Stage (main round)		
		round	Key	Finalization
0	(0, x)	4	k_1	No
1	(1, x)	4	k_1	No
2	(2, x)	4	k_2	No
3	(3, 1)	4	k_1	Yes
4	(4, 0)	4	k_1	Yes
5	(5, 0)	4	k_1	Yes
6	(5, x)	4	k_1	Yes
7	(-1, x)	10	K	No

Δ Calculations as a Function of (K, j, i)

$$\Delta \leftarrow \text{Init} \oplus (bn[0])A \oplus (2 \cdot bn[1])A \oplus (4 \cdot bn[2])A \oplus (2^{3+bn[6:3]})A$$

$$I \mid J \mid L \leftarrow \text{Extract}(K)$$

$$\text{Init} = 0 \text{ or } L \text{ or } 2L \text{ or } 4L$$

$$A = I \text{ or } J$$

$2^{3+bn[6:3]}A$ term present
only if $\alpha = \text{Yes}$

$$bn = i - 1$$

mode	(j, i)	First Stage (pre-computation)		
		Init	I or J	α
0	(0, x)	0	I	No
1	(1, x)	0	I	Yes
2	(2, x)	0	I	Yes
3	(3, 1)	L	J	Yes
4	(4, 0)	2L	J	No
5	(5, 0)	4L	J	No
6	(5, x)	4L	J	Yes
7	(-1, x)	0	J	No

x represents any value

Precomputations Required

$$\Delta \leftarrow Init \oplus (bn[0])A \oplus (2 \cdot bn[1])A \oplus (4 \cdot bn[2])A \oplus (2^{3+bn[6:3]})A$$

$$A = I \text{ or } J$$

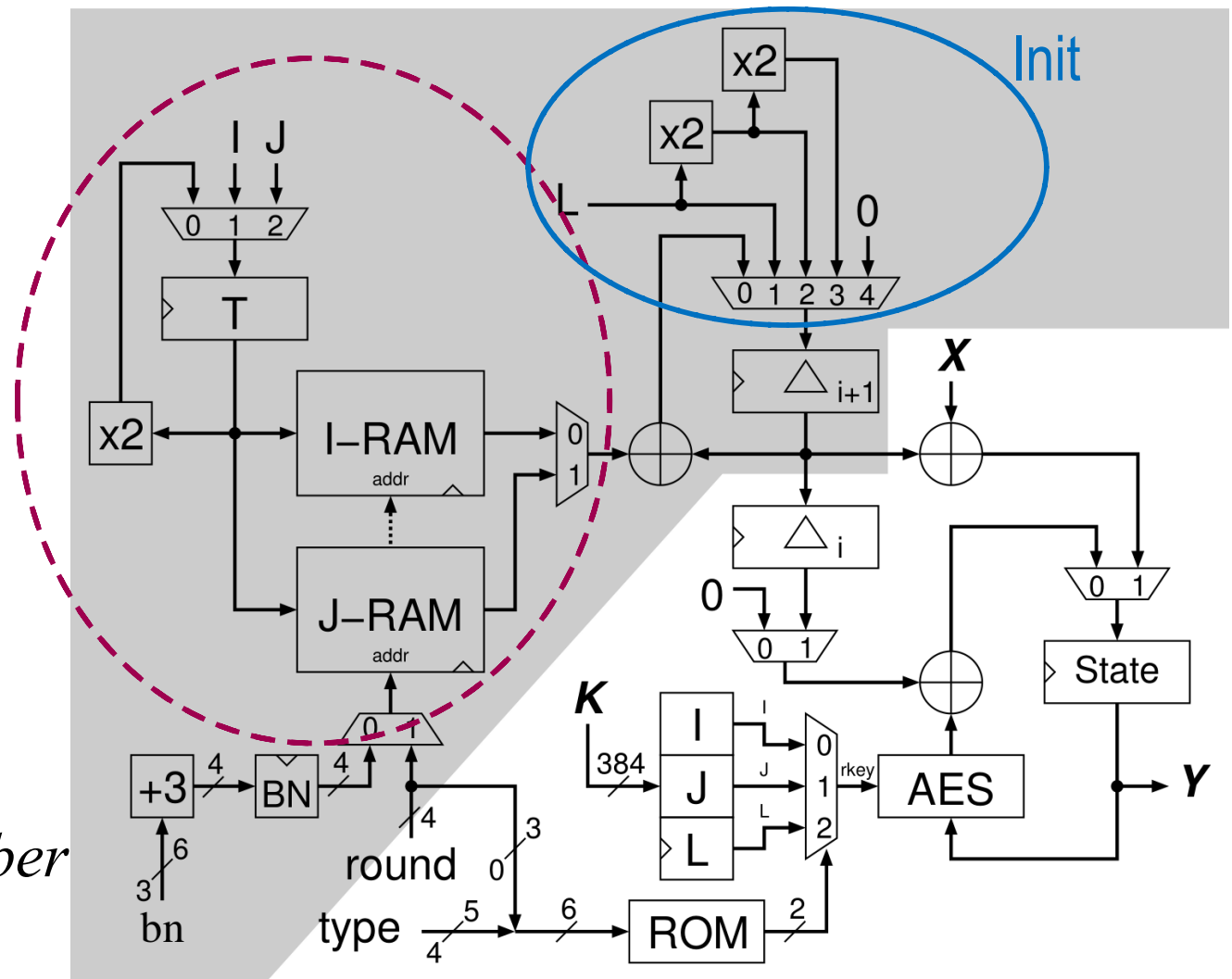
$$2A, 4A, 8A, \dots, 2^{10}A$$

because

$$\max(bn) = \max(i-1) = 2^6 - 1$$

$$\max(3 + bn/8) = 10$$

value of i limited
by the maximum number
of AD blocks

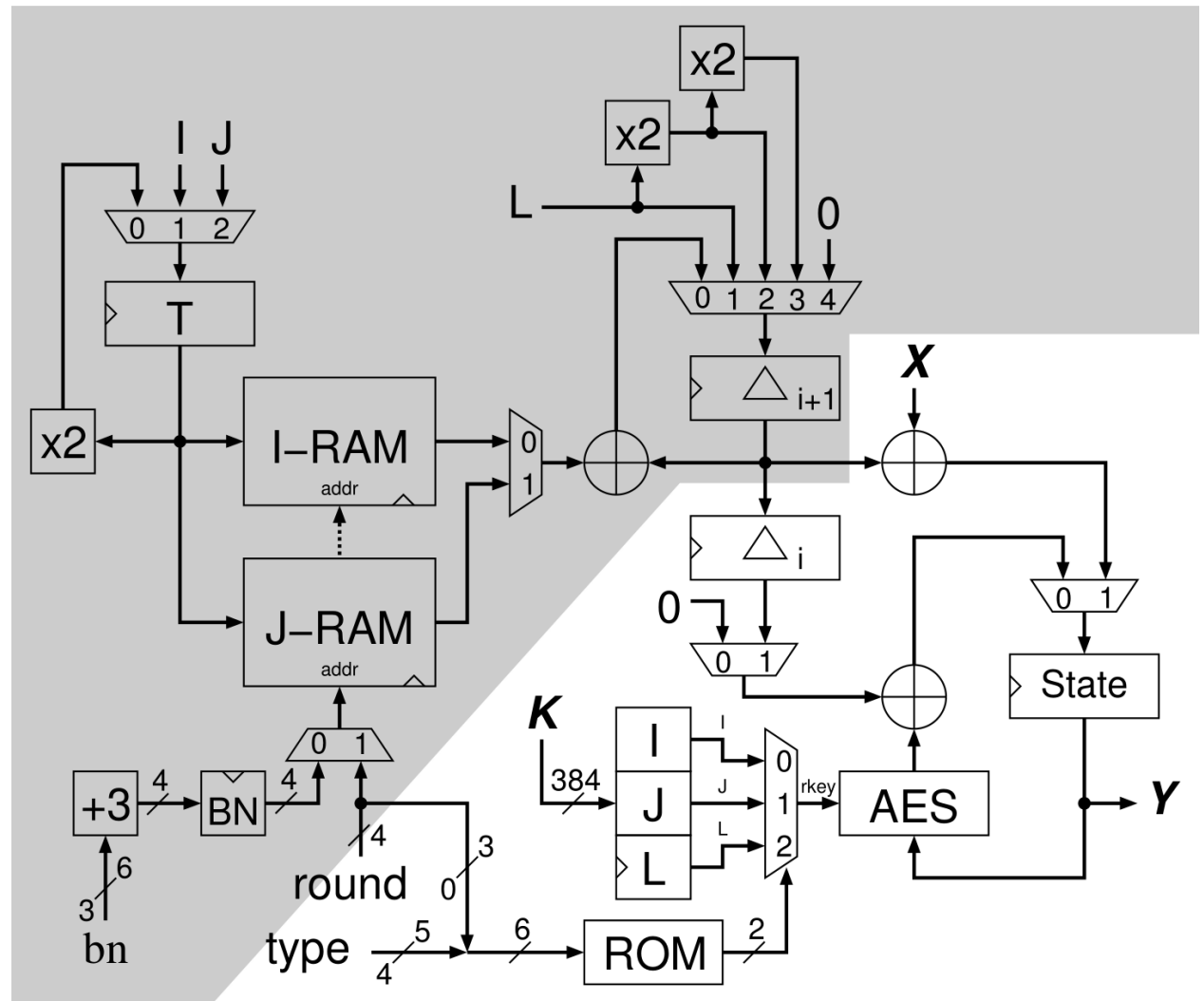


Computations of Δ

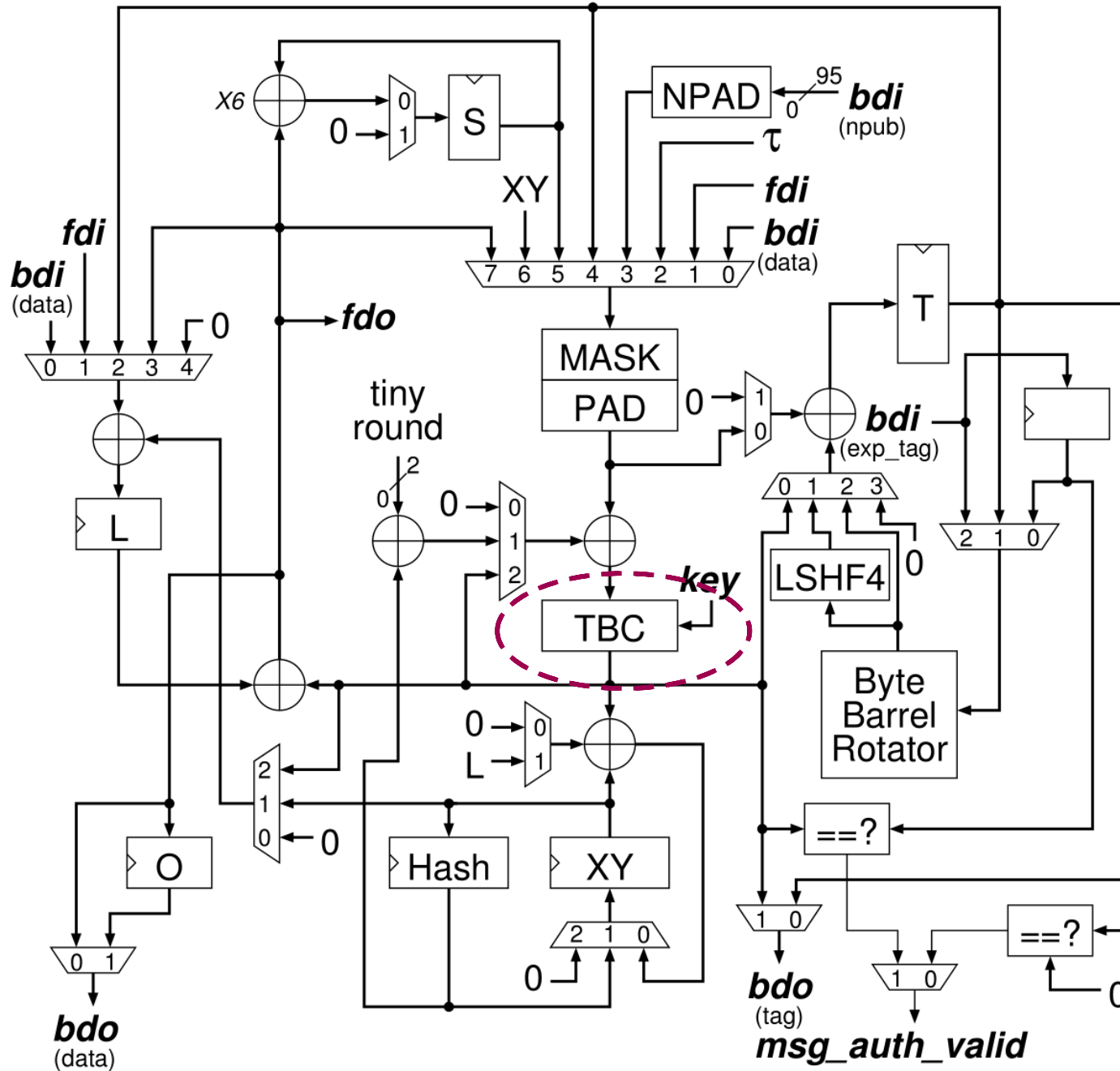
$$\Delta \leftarrow Init \oplus (bn[0])A \oplus (2 \cdot bn[1])A \oplus (4 \cdot bn[2])A \oplus (2^{3+bn[6:3]})A$$


*Each term determined
in one clock cycle*

*Maximum 5 clock
cycles required*



Datapath: Top





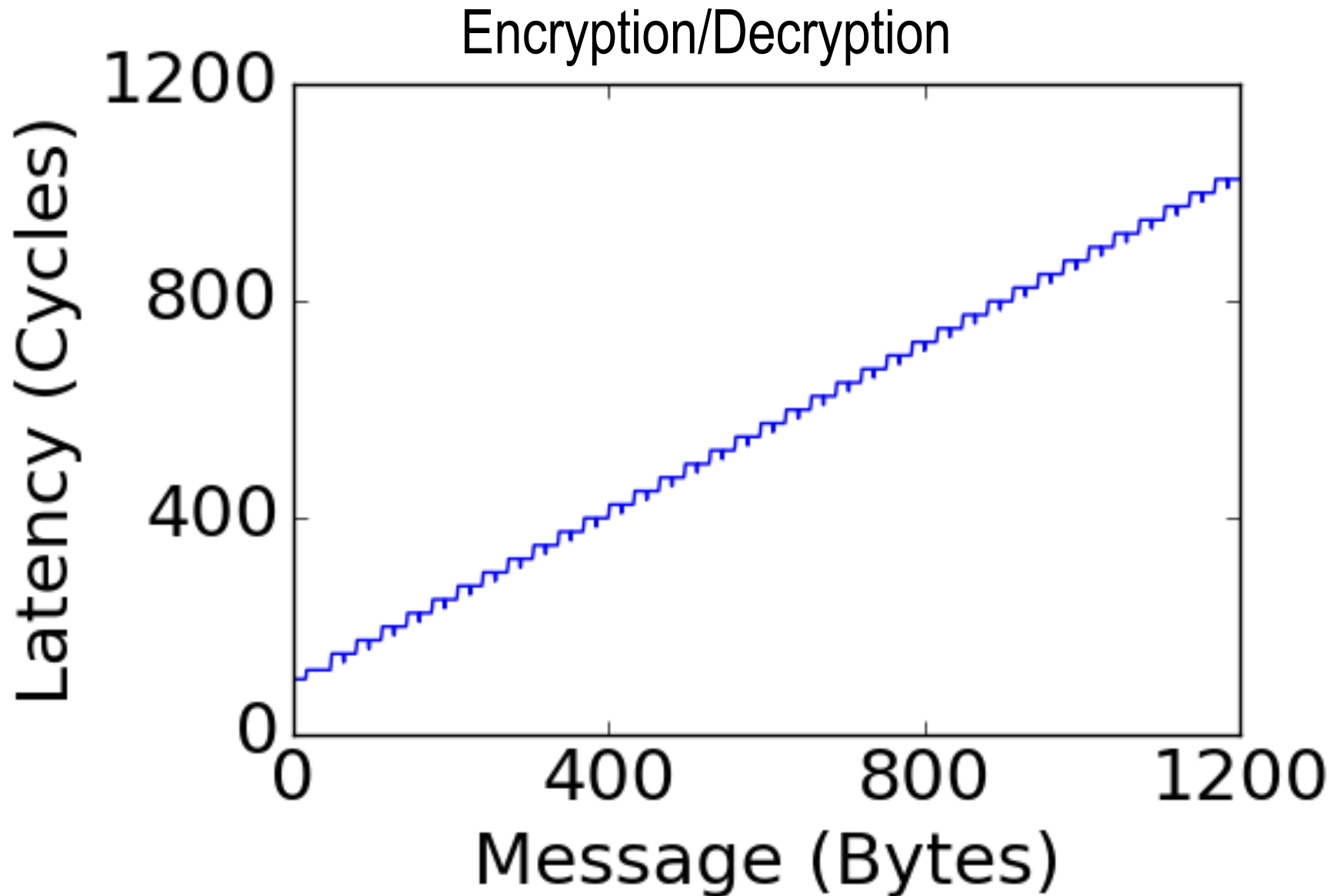
**Results &
Discussion**

Resource Utilization & Maximum Clock Frequency

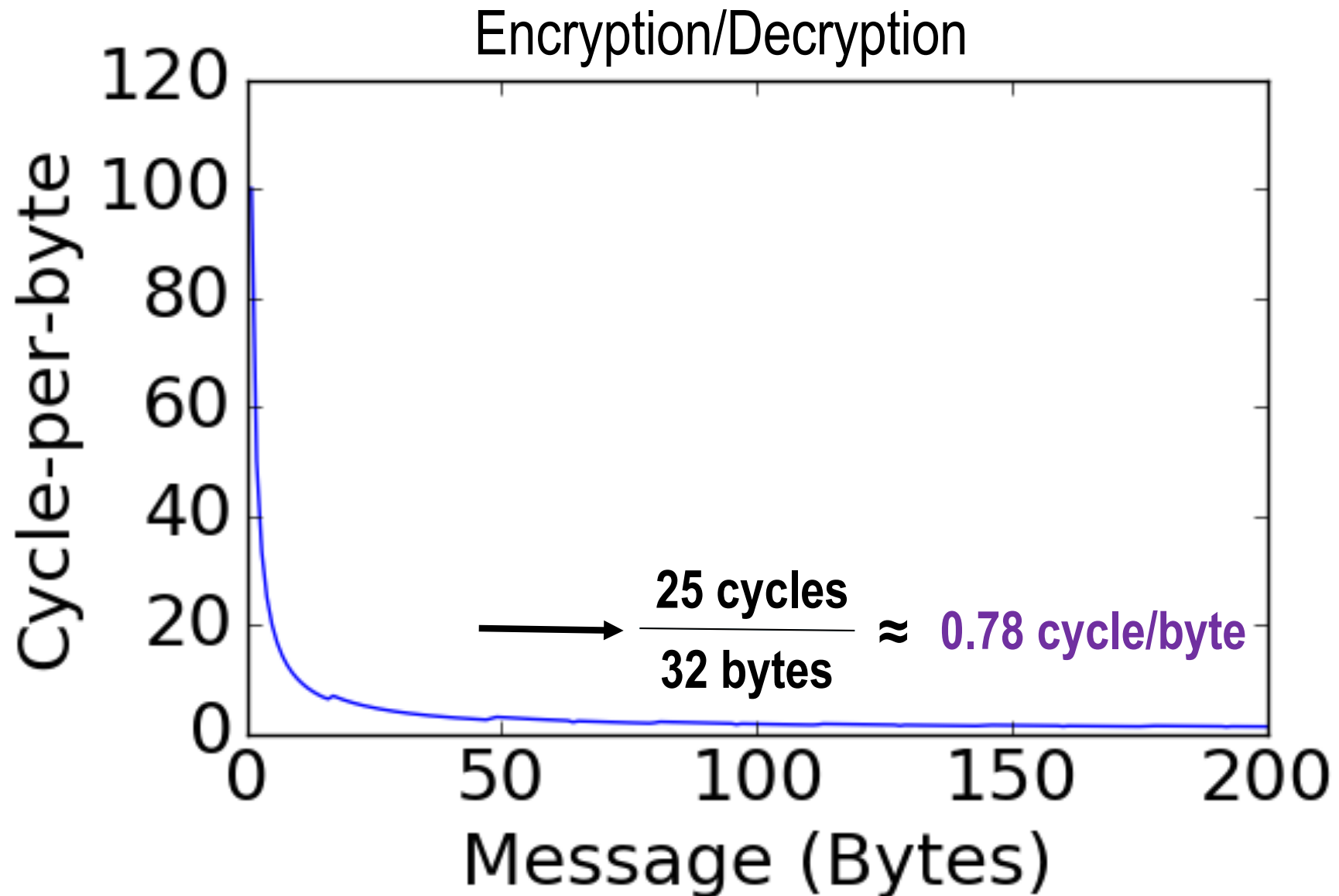
	FF	FF%	LUTs	LUTs%	Slices	Slices%
TBC	927	39%	1527	33%	480	39%
CipherCore	1983	84%	4166	91%	1259	101%
AEAD	2347	100%	4597	100%	1246	100%

	Clk Freq	Clk Freq%
TBC	362	100%
CipherCore	335	93%
AEAD	335	93%

Latency (Clock Cycles) vs. Message Size (Bytes)



Cycles per Byte vs. Message Size



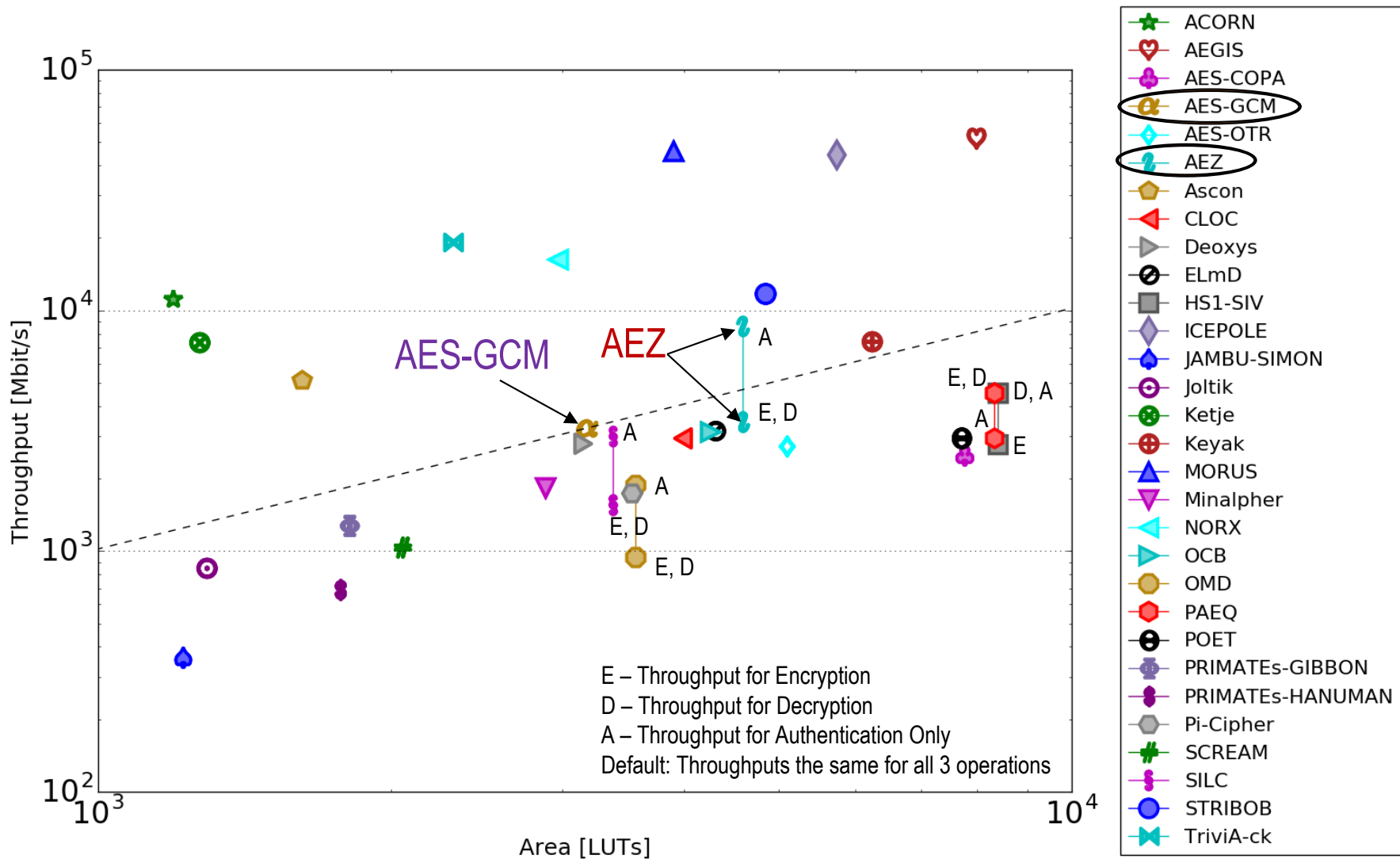
Throughput for Long Messages

$$\text{Throughput}_{\text{Encryption/Decryption}} \approx \frac{256}{25} \cdot \text{Clock Frequency}$$

$$\text{Throughput}_{\text{Authentication}} \approx \frac{128}{5} \cdot \text{Clock Frequency}$$

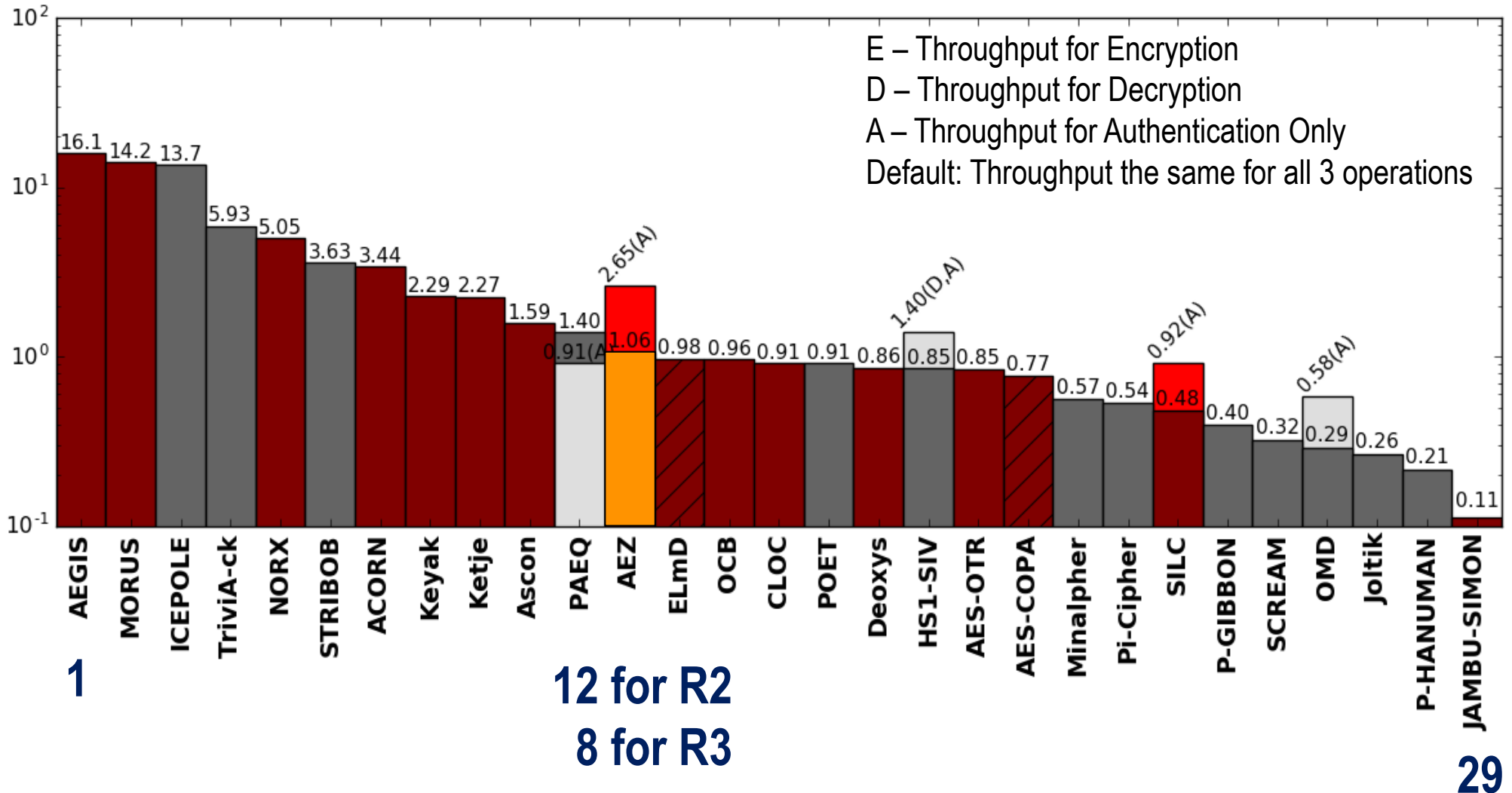
Results for Virtex 6 – Throughput vs. Area

Logarithmic Scale



Relative Throughput in Virtex 6

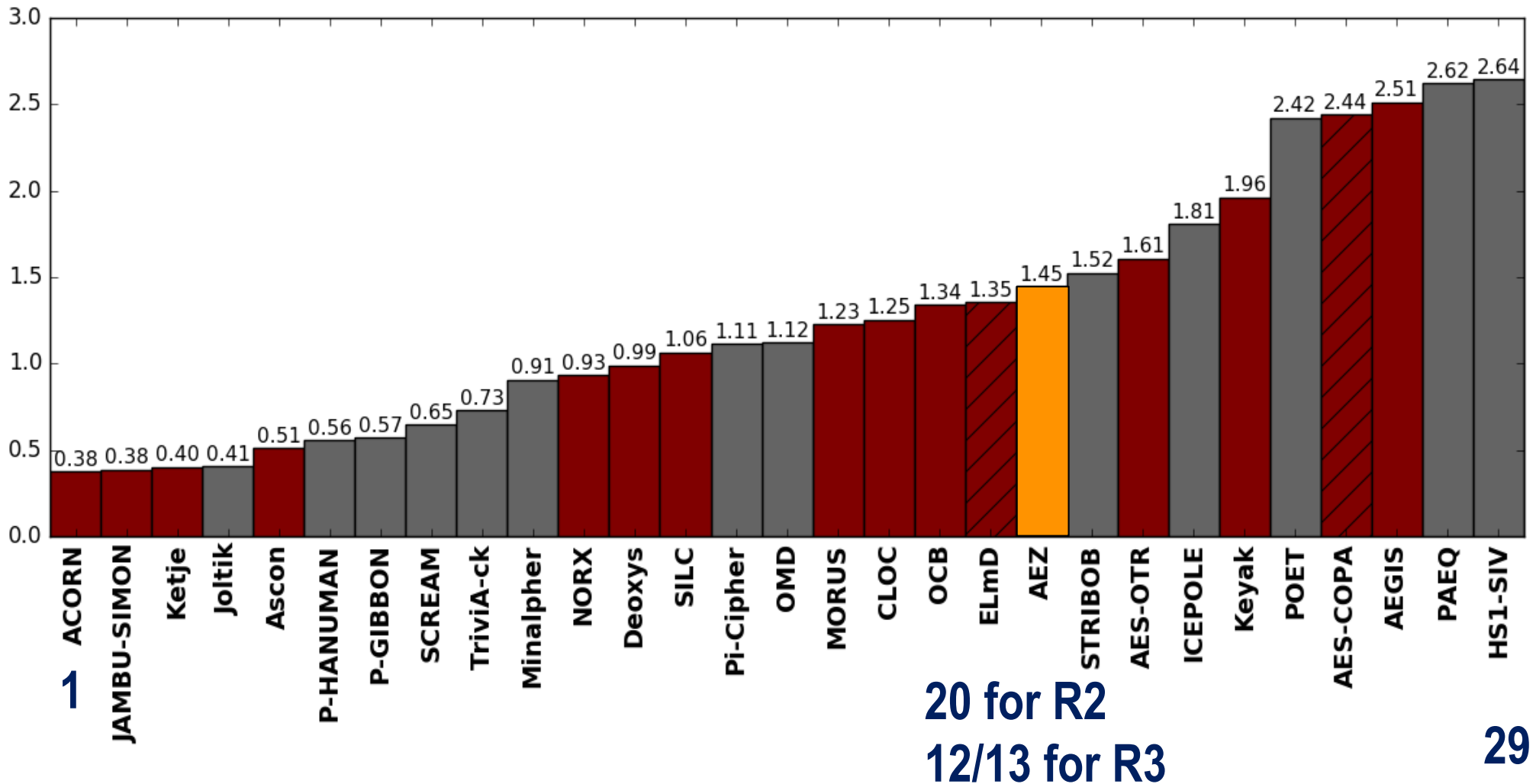
Ratio of a given Cipher Throughput/Throughput of AES-GCM



Throughput of AES-GCM = 3239 Mbit/s

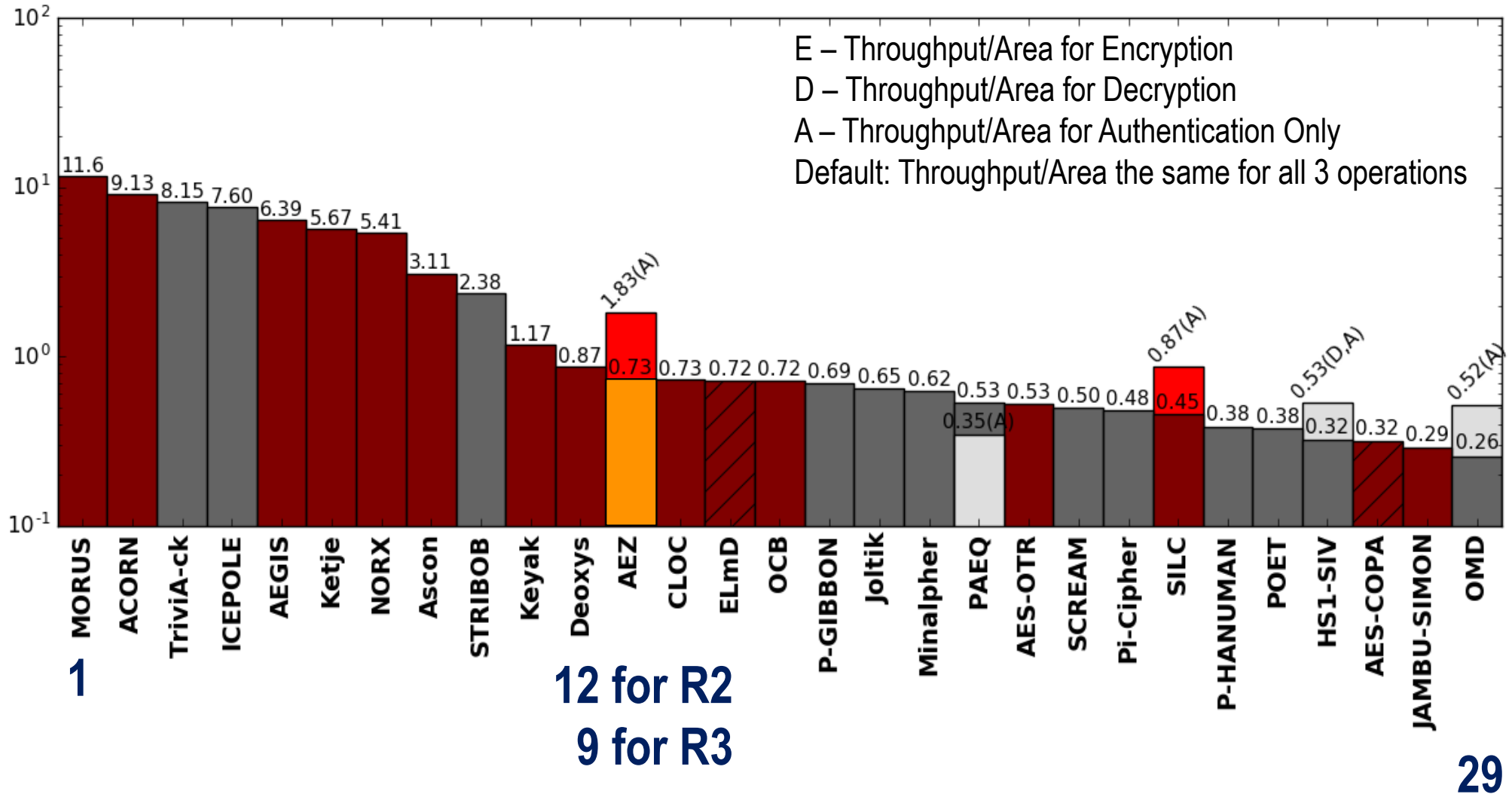
Relative Area (#LUTs) in Virtex 6

Ratio of a given Cipher Area/Area of AES-GCM



Area of AES-GCM = 3175 LUTs

Relative Throughput/Area in Virtex 6 vs. AES-GCM



Throughput/Area of AES-GCM = 1.020 (Mbit/s)/LUTs



**Conclusions &
Future Work**

Conclusions

- First hardware implementation of AEZ
 - Compliant with the CAESAR HW API
 - Optimized for the Throughput/Area ratio
 - Efficient
 - Practical
- Places AEZ 12th in terms of the Throughput/Area ratio among 28 Round 2 CAESAR candidates benchmarked to date (assuming the maximum message length of $2^{11}-1$)
- Almost matches the performance of AES-GCM in hardware, while at the same time offering an unprecedented level of security.

Possible Future Work

- Ability to increase the **maximum message length at the time of synthesis** using a generic
- Ability to modify the **authenticator length at the time of synthesis** using a generic

- Ability to modify the **authenticator length at the run time** using the Reserved field of the API instruction
- Implementation with **inner-round pipelining**
- **Lightweight** implementation

More Details & Code

- **Detailed description of the circuit operation**

Proceedings + ePrint version of the paper
(under development)

- **VHDL Source Code**

<https://cryptography.gmu.edu/athena>

Under: **CAESAR**

GMU Implementations of Authenticated Ciphers and Their Building Blocks

Thank you!

Comments?



Questions?

Suggestions?

ATHENa: <http://cryptography.gmu.edu/athena>

CERG: <http://cryptography.gmu.edu>