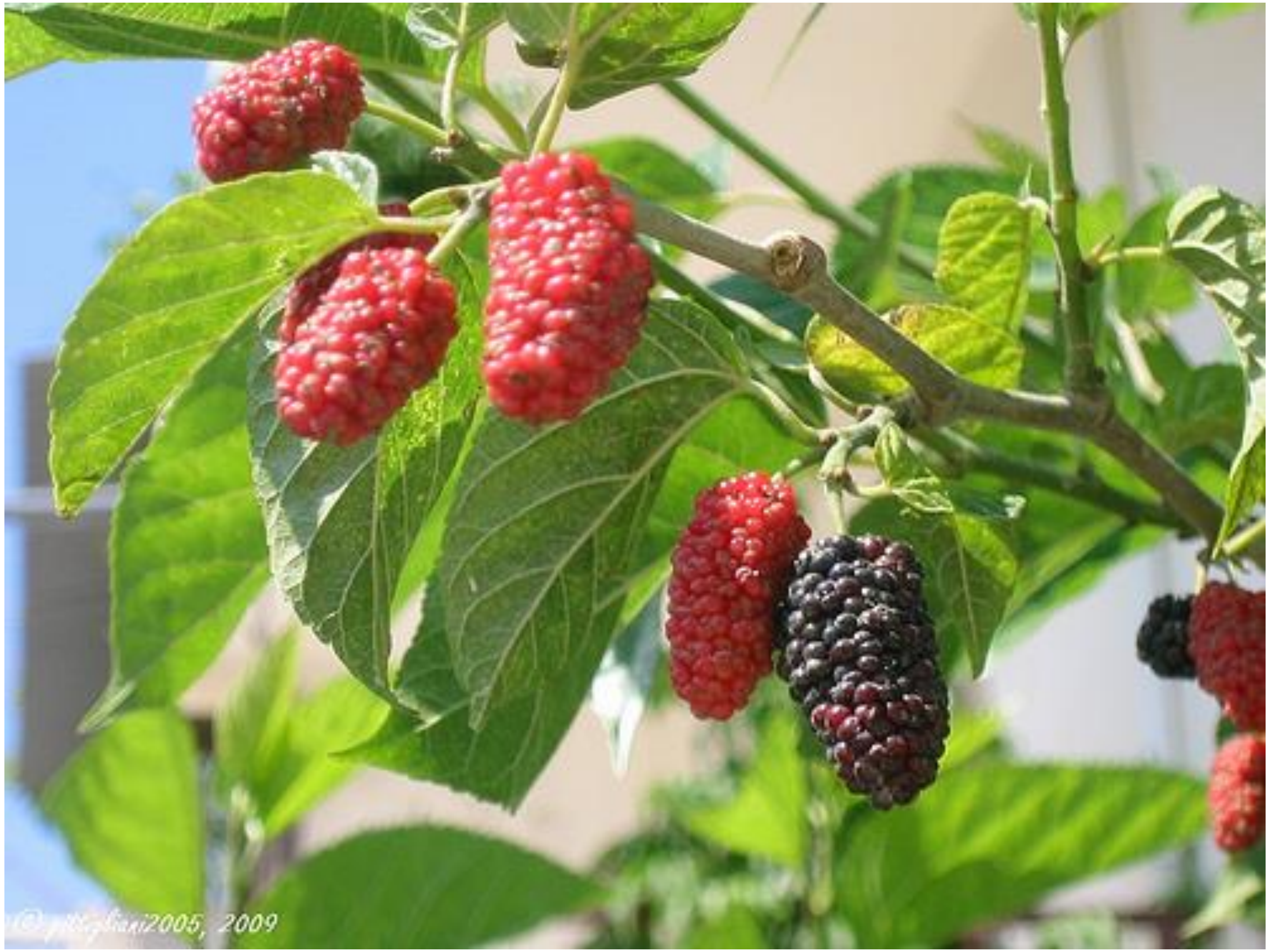
MORUS

A Fast Authenticated Cipher

Hongjun Wu   **Tao Huang**

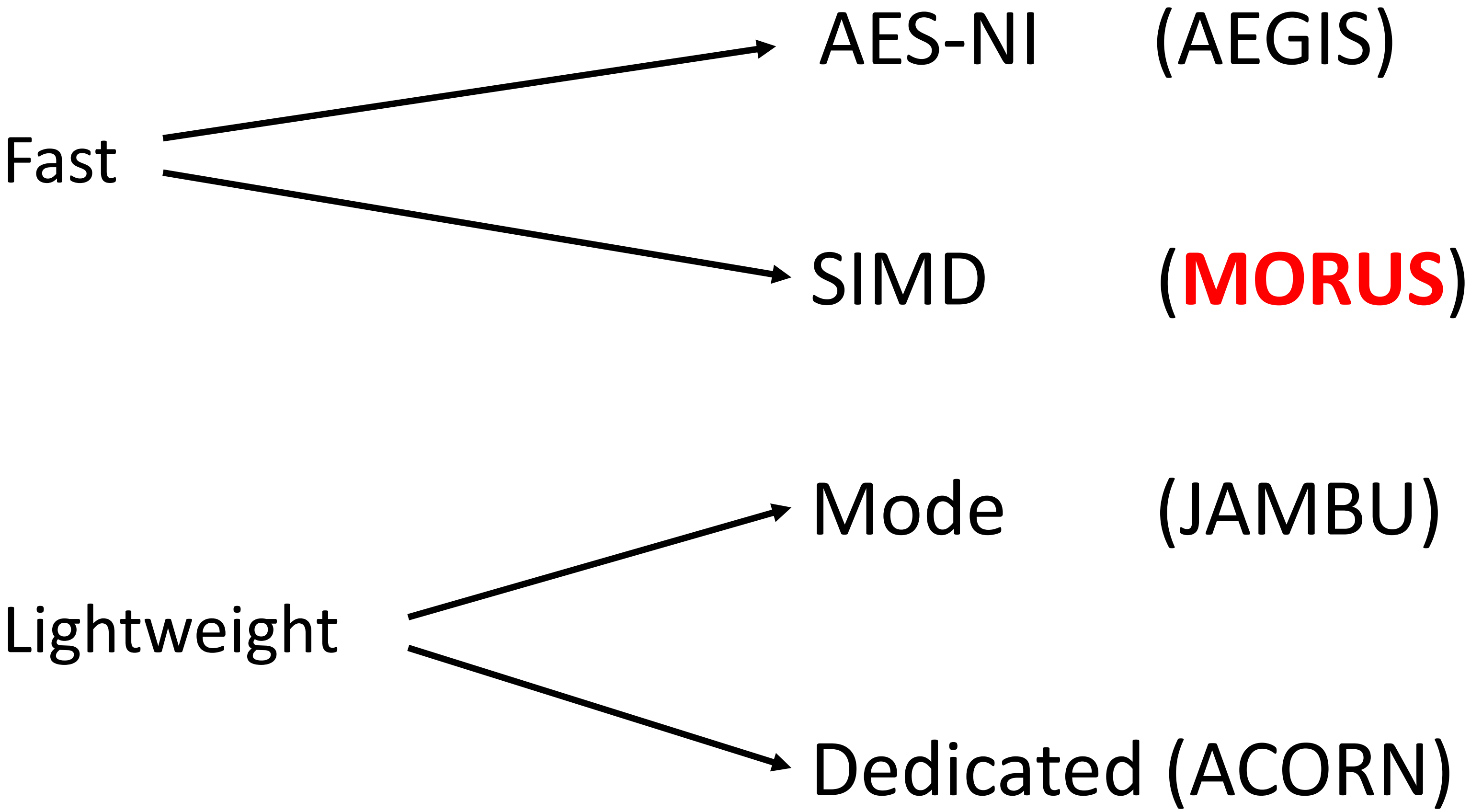
Nanyang Technological University

DIAC 2016, Nagoya
26 Sep 2016



MORUS

Different Design Approaches:



Fast

AES-NI (AEGIS)

SIMD (**MORUS**)

Lightweight

Mode (JAMBU)

Dedicated (ACORN)

Design Motivation and Main Features

- To design a high-speed authenticated cipher:
  - No AES-NI
  - Make use of the SIMD (SSE2, AVX2) instructions

- Features
  - Fast in software: 0.69 cpb on Haswell
  - Fast in hardware: 95.8 Gbps on Xilinx Virtex 7
    250 Gbps on 65 nm ASIC (ETH implementation)
  - Nonce-based

Changes in MORUS v2

- Tweaks are **only** applied to the ***finalization*** of MORUS
  - Remove register S_3 in the message word of finalization
  - Change the tag generation to the same way as the keystream generation
    - Increase the number of steps from 8 to 10 (compensating the change in tag generation)
- Rationale for tweaks
  - Improve the hardware efficiency of MORUS

MORUS: Parameters

	State size (bits)	Key size (bits)	Tag size (bits)	Plaintext size (bits)	AD size (bits)
MORUS-1280-128	1280	128	128	$<2^{64}$	$<2^{64}$
MORUS-640-128	640	128	128	$<2^{64}$	$<2^{64}$
MORUS-1280-256	1280	256	128	$<2^{64}$	$<2^{64}$

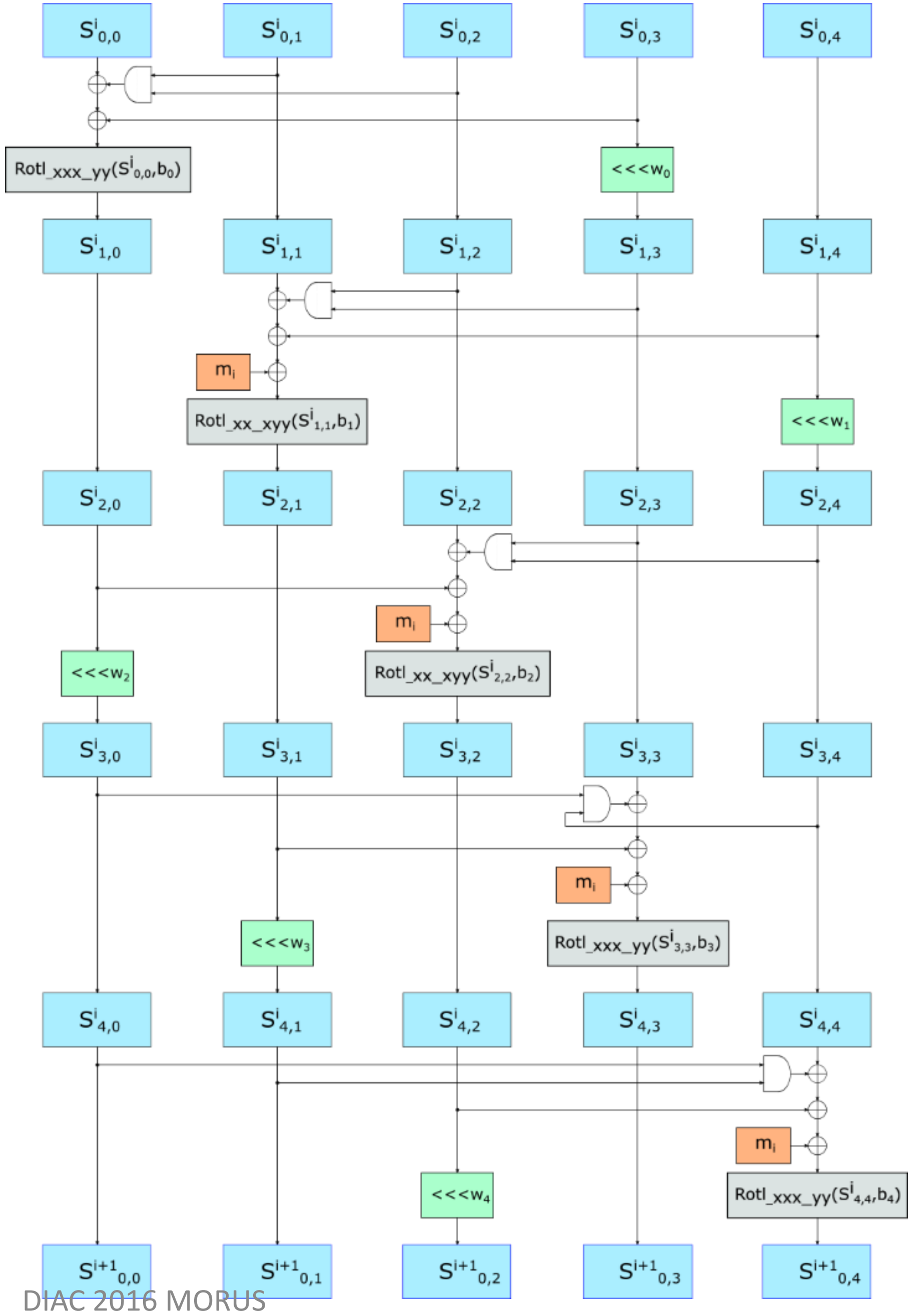
MORUS: State and Operations

- State organization
  - MORUS-1280: five 256-bit words
  - MORUS-640 : five 128-bit words

- Operations:
  - XOR, AND, SHIFT
  - $Rotl\_128\_32(x, n)$: Divide a 128-bit block x into 4 32-bit words, rotate each word left by n bits.
  - $Rotl\_256\_64(x, n)$: Divide a 256-bit block x into 4 64-bit words, rotate each word left by n bits.

MORUS: State Update (Overview)

One step: 5 rounds


$S^i_{0,0}$ $S^i_{0,1}$ $S^i_{0,2}$ $S^i_{0,3}$ $S^i_{0,4}$
Rotl_xxx_yy($S^i_{0,0}$,b_0)
<<<w_0
$S^i_{1,0}$ $S^i_{1,1}$ $S^i_{1,2}$ $S^i_{1,3}$ $S^i_{1,4}$
m_i
Rotl_xx_xyy($S^i_{1,1}$,b_1)
<<<w_1
$S^i_{2,0}$ $S^i_{2,1}$ $S^i_{2,2}$ $S^i_{2,3}$ $S^i_{2,4}$
m_i
<<<w_2
Rotl_xx_xyy($S^i_{2,2}$,b_2)
$S^i_{3,0}$ $S^i_{3,1}$ $S^i_{3,2}$ $S^i_{3,3}$ $S^i_{3,4}$
m_i
<<<w_3
Rotl_xxx_yy($S^i_{3,3}$,b_3)
$S^i_{4,0}$ $S^i_{4,1}$ $S^i_{4,2}$ $S^i_{4,3}$ $S^i_{4,4}$
m_i
<<<w_4
Rotl_xxx_yy($S^i_{4,4}$,b_4)
$S^{i+1}_{0,0}$ $S^{i+1}_{0,1}$ $S^{i+1}_{0,2}$ $S^{i+1}_{0,3}$ $S^{i+1}_{0,4}$

MORUS: Initialization

- Load IV, key and constants into the initial state
- Update state: 16 steps
- Key is XORed to the state at the end of the initialization

MORUS: Keystream Generation

- State $S \ = \ \{S_0, S_1, S_2, S_3, S_4\}$
- For MORUS-640:
  - $keystream = S_0 \oplus (S_1 <<< 96) \oplus (S_2 \,\&\, S_3)$
- For MORUS-1280
  - $keystream = S_0 \oplus (S_1 <<< 192) \oplus (S_2 \,\&\, S_3)$

MORUS: Finalization (Tweaked!)

MORUS v1

- State update: 8 steps
- Message
  $S_3 \oplus (adlen||msglen)$
- Tag generation
  $S_1 \oplus S_2 \oplus S_3 \oplus S_4$

MORUS v2

- State update: 10 steps
- Message
  $(adlen||msglen)$
- Tag generation
  $S_0 \oplus (S_1 <<< C^{*}) \oplus (S_2 \,\&\, S_3)$

* $C = 96$ for MORUS-640;
$C = 192$ for MORUS-1280

MORUS: Security Goal

	Confidentiality (bits)	Integrity (bits)
MORUS-640-128	128	128
MORUS-1280-128	128	128
MORUS-1280-256	256	128

Security of MORUS: Initialization

- Algebraic degree
  - After 10 steps, the algebraic degree exceeds 256
- Differential cryptanalysis
  - differential probability $< 2^{-256}$

Security of MORUS: Encryption

- Guess-and-determine attack
  - state size of MORUS is at least five times of key size
  - keystream generation function
    - state bits are not directly known to the adversary

Security of MORUS: Finalization

- Internal state collision
  - Probability < 2^{-128}
- Differential forgery attack on the finalization
  - 10 steps, differential probability < 2^{-256}

Security of MORUS

- **Remark** on the analysis by **Mileva et al.** in BalkanCryptSec 2015
  - Not that relevant to the security of MORUS
    - Collison on the state update function: assuming special difference in the state – unrealistic
    - Distinguisher in nonce-reuse scenarios – excluded in our security claim
    - “differential bias” – becomes invalid when a different key is used

MORUS: Hardware Performance

- State update function of MORUS is designed to be fast in hardware
    - AND and XOR gates are used
    - Short critical path

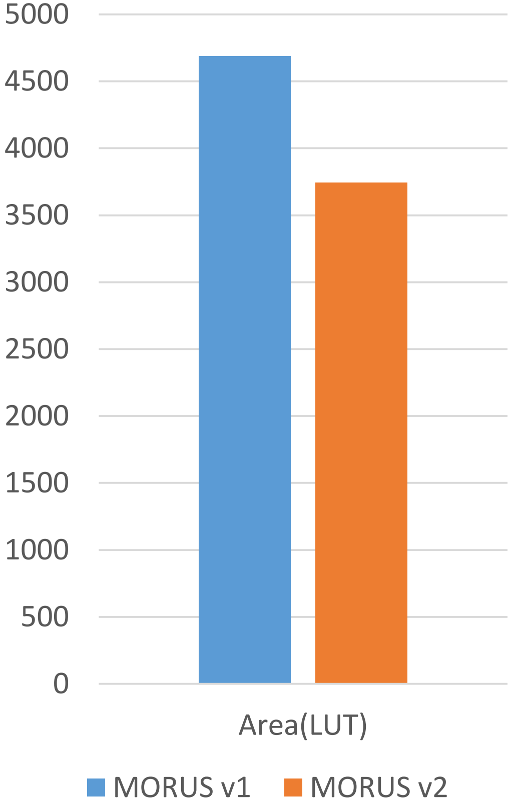
MORUS: Hardware Performance

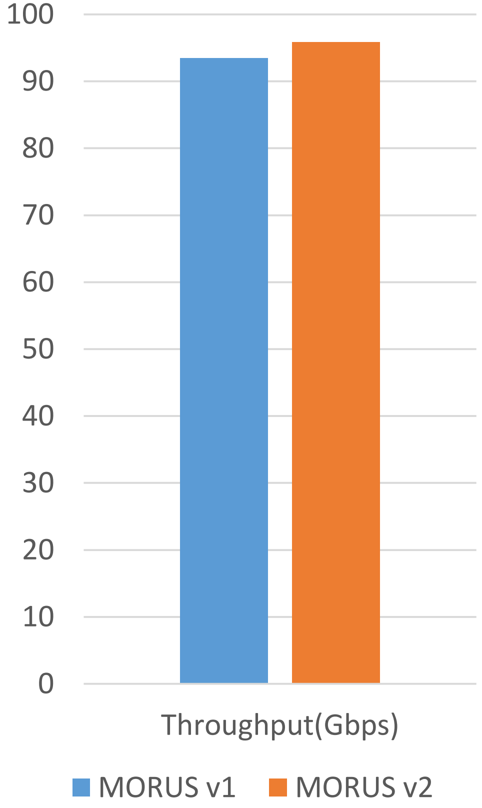
- Current implementation on FPGA using CAESAR API
  - Virtex 7, Xilinx Vivado 2016.2

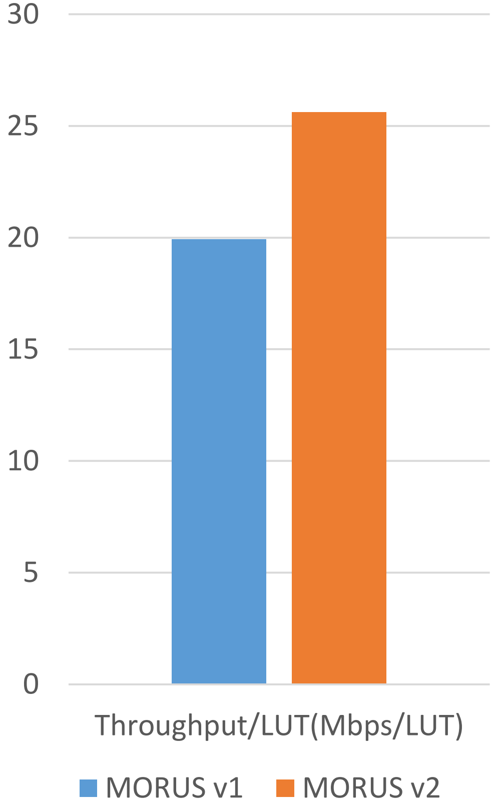
	Area (Slice)	Area (LUT)	Frequency (MHz)	TP (Gbps)	TP/LUT (Mbps/LUT)
MORUS-640	681	2129	342.4	43.8	20.6
MORUS-1280	1045	3746	370.4	**95.8**	**25.6**

MORUS: Hardware Performance

Comparison between MORUS-1280 v1 and MORUS-1280 v2

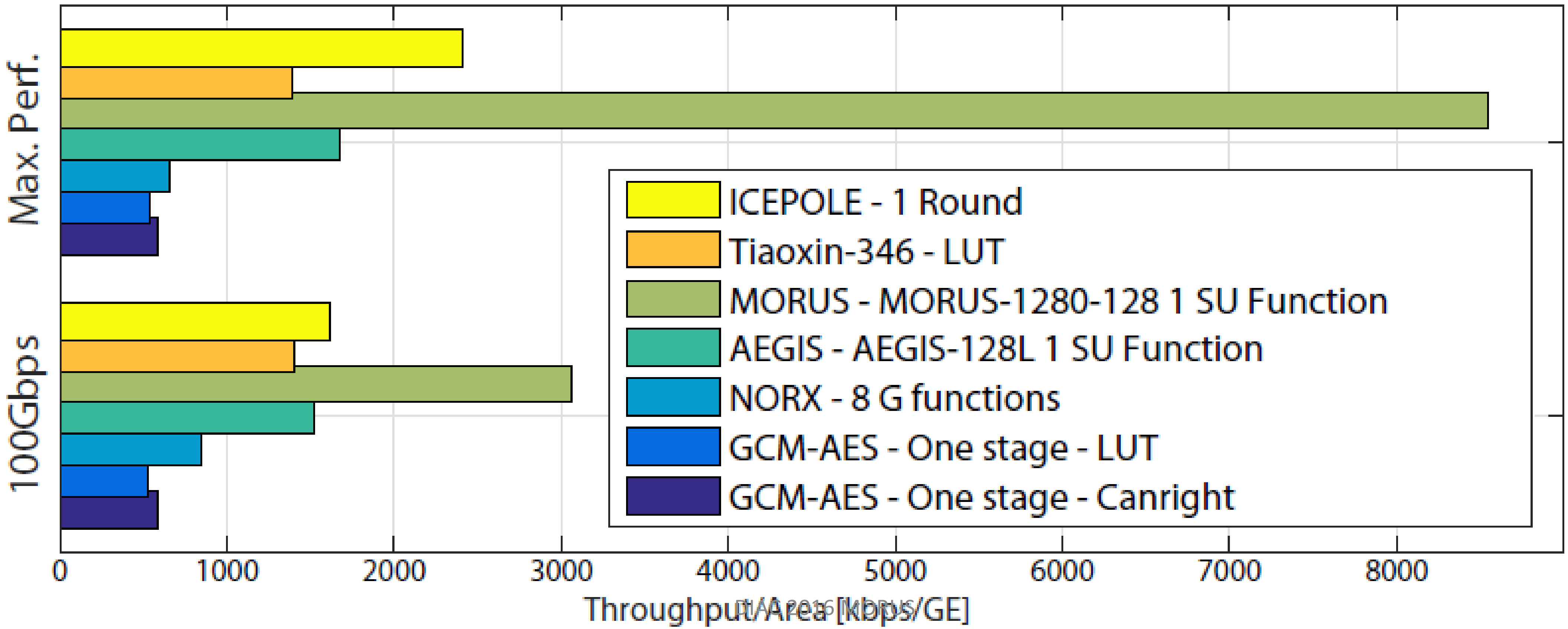




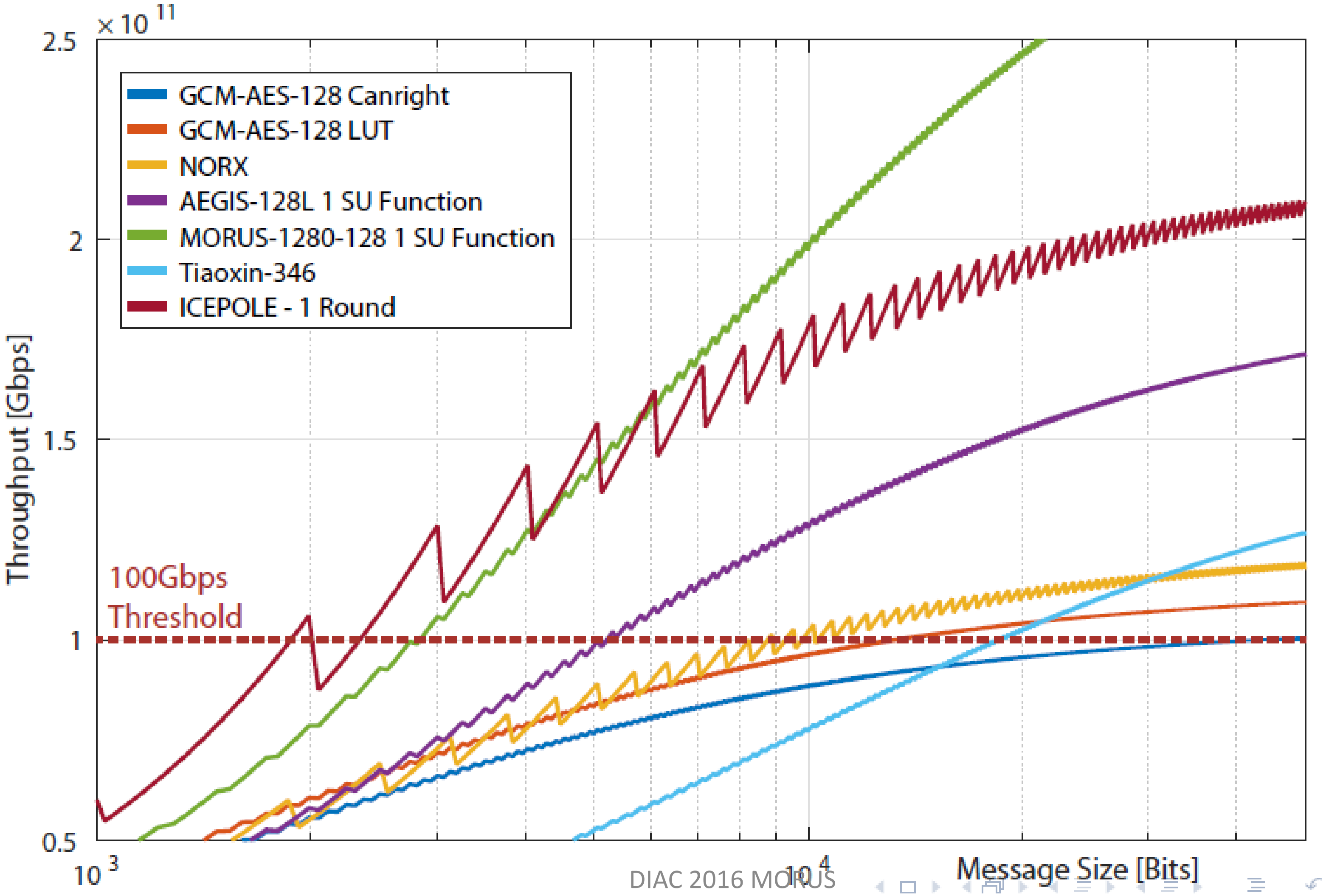


MORUS: Hardware Performance

- Performance on ASIC: high throughtput/area
  (Michael Muehlberghuber and Frank K. Gürkaynak, DIAC 2015)



- Performance on ASIC: high throughput (250Gbps) (Michael Muehlberghuber and Frank K. Gürkaynak, DIAC 2015)



MORUS: Software Performance

- Speed on Haswell, AVX2 is used in MORUS-1280

	16B	64B	512B	1024B	4096B	16384B
MORUS-640(EA)	40.64	10.35	2.30	1.72	1.30	1.19
MORUS-640(DV)	38.47	10.13	2.30	1.72	1.29	1.18
MORUS-1280(EA)	45.32	10.38	1.85	1.24	0.80	0.69
MORUS-1280(DV)	45.74	10.66	1.91	1.28	0.81	0.70

MORUS: Software Performance

- Faster than AES-GCM on Haswell (1.03 cpb)
- Almost the same as MORUS v1 for long message
- Reasons:
    - Benefits from SIMD
    - Removed the redundant operations in the cipher

Conclusion

- **MORUS**
  - The fastest candidate on the platforms with SIMD but with no AES-NI (0.69 cpb with AVX2)
  - The most efficient candidate in hardware
    MORUS-1280: 95.88 Gbps, 3764 LUTs, 25.6 Mbps/LUT
- **MORUS v2**
  - Tweaked finalization to reduce hardware area.
    Throughput/Area is increased by 28%

Thanks for your attention!