Efficient Design Strategies Based on the AES Round Function

Jérémy Jean[1,2] Ivica Nikolić[2]

[1]ANSSI, Paris, France

[2]Nanyang Technological University, Singapore

FSE 2016 – March 22, 2016

ANSSI Agence nationale de la sécurité des systèmes d'information

NANYANG TECHNOLOGICAL UNIVERSITY

Introduction: Motivations

- **Plenty** of primitives based on the AES round function.
- **Several** provide good efficiency on modern processors with AES support.
- **Only a few** are very efficient (e.g., AEGIS, Tiaoxin):
  - Rely on parallel execution.
  - Have low number of AES round calls per message.

How far can we go?

Goal

Provide design strategy based on the AES round function
that is secure and extremely efficient (0.1-0.3 cycles/byte).

Introduction: Summary of Results

Final result

We show design strategy that achieves our goals.

- Design strategy can be used for hash functions, AE, or MAC.
- Designs extremely efficient on `AES−NI` supported platforms.
  - We benchmark our recent platforms, including Intel Skylake.
  - Fastest design: 0.125 cycles/Bytes.
  - Smallest design: 0.188 cycles/Bytes.
- Other platforms: good efficiency as well, since only 2-3 rounds of `AES`.

AES Instruction Set (AES-NI)

- Proposed by Intel in 2008.
- Provides processors instructions performing AES operations.
  - AESENC: One round of AES round function
  - AESDEC: One round of AES inverse round function.
  - AESENCLAST: Last round of AES encryption.
  - AESDECLAST: Last round of AES decryption.
  - AESKEYGENASSIST: AES key schedule.
  - AESIMC: Apply the inverse MC operation.

$$\mathtt{aesenc}(X, K) = (\mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB})(\mathsf{X}) \oplus \mathsf{K}$$

Latency and Throughput of Instructions

Informal Definitions

- **Latency**: number of clock cycles required to execute an instruction.
- **Throughput**: number of clock cycles required to wait before executing the same instruction.

	Processor	Date	Latency	Throughput
`⋆bridge`	Sandy Bridge	Q1 2011	8	1
	Ivy Bridge	Q2 2012	8	1
`⋆well`	Haswell	Q2 2013	7	1
	Broadwell	Q1 2015	7	1
`⋆lake`	Skylake	Q3 2015	4	1

`aesenc` efficiency.

Our Goals

Main Goal

We want building blocks based on `aesenc` that achieve very high performances, and possibly *optimally efficient*.

Precisions:

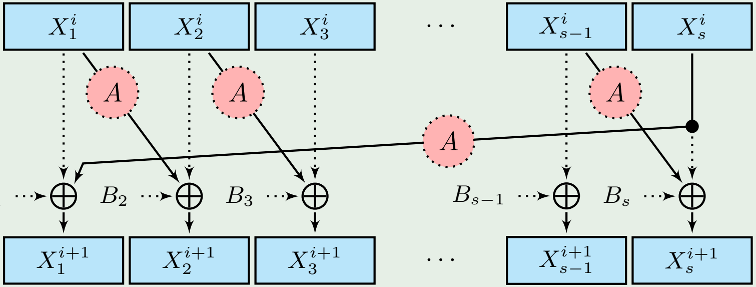
- We want a single primitive, not a mode using a primitive.
- Its (possibly large) internal state consists of several 128-bit words.
- It includes an internal state updated by one (or more) input block.

We investigate two concurrent approaches:

- Low number of `aesenc` calls per input block,
- Parallelization of the `aesenc` calls.

Design Strategies

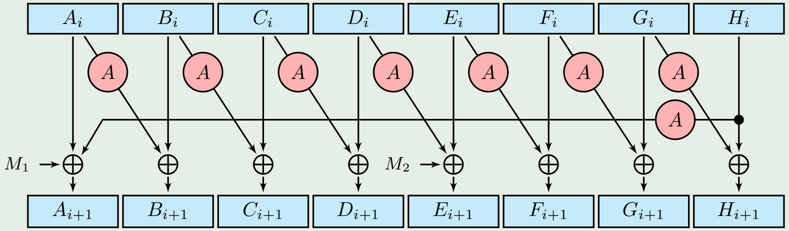
General Structure of the Step Function



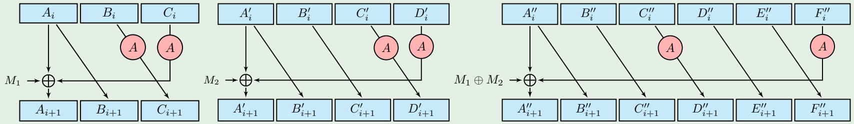
- Internal state: *s* state words.
- Overall right shift with possible application of *A* (`aesenc`).
- Possible feed-forward XOR of previous value.
- Inputs B_i can be any linear combination of $0, M_1, M_2, \ldots$

Examples: AEGIS-128L and Tiaoxin-346

AEGIS-128L



Tiaoxin-346



Both designs inject two 128-bit inputs in the state.

First Approach: Low Number of AES Rounds

Definition: Rate

We define the rate ρ of a design as the number of calls to `aesenc` used to process a 16-byte input.

Primitive	**State Size**	**#Inputs**	**Rate**
AES-128	1	1	10
AES-256	1	1	14
AEGIS-128L	8	2	4
Tiaoxin-346	13	2	3

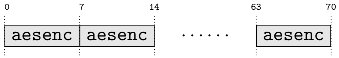
We want to achieve rates as low as possible.

Second Approach: Parallelization of `aesenc` Calls

Parallelization of `aesenc` instructions may allow significant efficiency improvements.

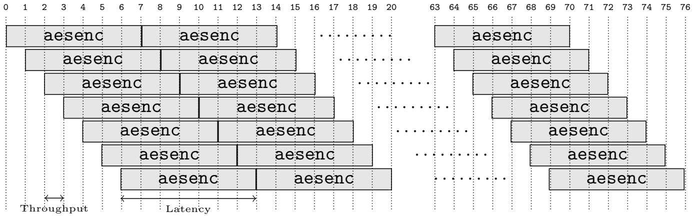
Example: `AES−CBC` on Haswell (`aesenc` latency: 7).

$\Longrightarrow$ Efficiency: 70 cycles per 16 bytes = 70/16 = 4.375 cpb.



Example: `AES−CTR` on Haswell (`aesenc` latency: 7, throughput: 1).

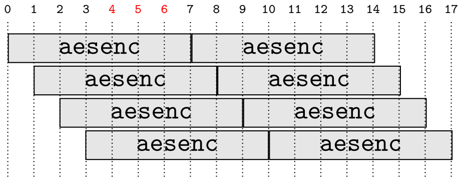
$\Longrightarrow$ Efficiency: 70 cycles per 7×16 bytes = $70/(7 \times 16)$ = 0.625 cpb.



On Efficiency Optimality

- Full parallelization can only be achieved if the state is large enough:
  - `AES-CBC`: single 128-bit word, but no parallelization
  - `AES-CTR`: arbitrary number of 128-bit words, and full parallelization.
- Consequently, if we use *n* calls to `aesenc`, the primitive must have at least *n* words in the internal state.

- The optimal number of calls depends on the latency/throughput ratio.
  - Consider for instance a rate-4 design on Haswell (ratio: 7/1).
  - Cycles 4, 5, 6 are wasted: no call to `aesenc` possible → effective speed: 7/16 cpb.
  - On Skylake (ratio: 4/1), then no empty cycle → effective speed: 4/16 cpb is optimal.



Summary of Design Choices

Design strategies followed to achieve high efficiency:

- Low rate.
- At least as many 128-bit state words as `aesenc` calls.
- Independent calls to `aesenc`.
  - Allows parallel execution.

Security

Security notions

- We provide building blocks, not fully defined instantiations.
- Security claims reduced to resistance to internal collisions.
  - Capture many possible applications (MAC, Hashing, AE).
  - Well-understood problem.

Security evaluation

- Find diff. char. $0 \rightarrow \Delta_1 \rightarrow \ldots \rightarrow \Delta_s \rightarrow 0$ with maximal probability p.
  - Differences introduced in the input blocks.
- Task easier for designs based on the AES round function.
  - Count the number N_b of active Sboxes in the characteristic.
  - $N_b \geq 22 \implies p \leq 2^{-128}$.
  - Classical example: 4-round AES (SK): $N_b \geq 25 \implies p \leq 2^{-6 \cdot 25} \ll 2^{-128}$.
  - We target a minimum of 22 active Sboxes.

Design Strategies: Two Classes

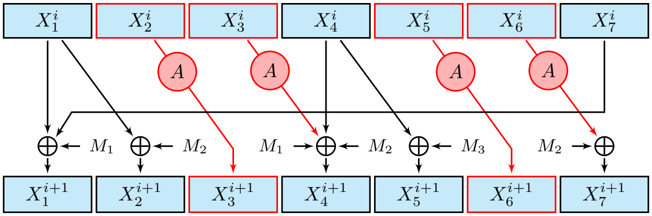
We now present our two main design strategies, only using the AES round function A and the XOR operation $\oplus$.

Simple class

- Class $\mathcal{A}_{\oplus}^{r}(r > 1)$.
- Uses r cascaded iterations of the AES round function.
- Equivalent to single-key model.
- Direct lower bounds from the wide-trail strategy.

General class

- Class $\mathcal{A}_{\oplus}(r = 1)$.
- Allows intermediate XORs between consecutive AES rounds.
- Equivalent to related-key model.
- Security analysis more complex $\Rightarrow$ we rely on MILP.



Example from the class $\mathcal{A}_{\oplus}^{2}$ (insecure).

Simple Class: Limited Optimal Efficiency

Theorem

The rate ρ of a secure design in $\mathcal{A}^r_\oplus$ cannot be less than r: $\rho(\mathcal{A}^r_\oplus) \geq r$.

Proof intuition: Enough freedom to introduce differences and cancel them *before* they enter A^r $\Rightarrow$ $0 \rightarrow 0$ characteristic with 0 active Sboxes.

Corollary

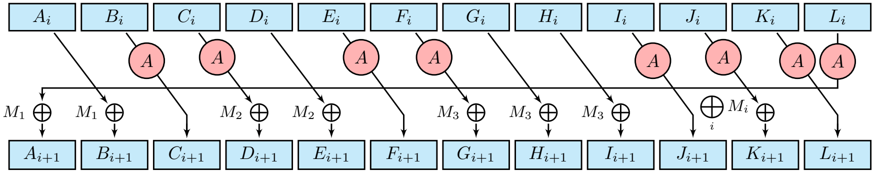
Designs in $\mathcal{A}^r_\oplus$ cannot run faster than

0.250 cpb for $r = 4$, 0.188 cpb for $r = 3$, 0.125 cpb for $r = 2$.

Simple Class: Results

Results for $\mathcal{A}^r_\oplus, r > 1$

- Complete search of $\mathcal{A}^3_\oplus$:
  - From theorem: smallest rate achievable is 3.
  - We consider at most 12 state words
  - Need to prove that a difference enters (at least) three times the cascaded AES.
  - Indeed, $N_b \geq 9 \Rightarrow 3 \times 9 = 27 \geq 22$.
  - No schemes are secure.
- Partial search of $\mathcal{A}^2_\oplus$
  - Space too large: exhaustive search impossible.
  - There exists secure designs: smallest rate we achieve is 2.66.
  - Figure below: secure design from $\mathcal{A}^2_\oplus$, with rate $\rho = 8/3$ and 25+ active Sboxes.



General Case: Finding Designs

We consider the more general class to find more efficient designs.

General method

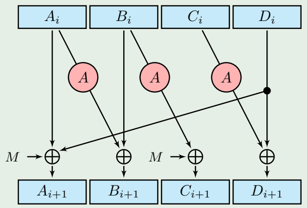
- Search space with several dimensions:
  - Rate: ρ.
  - Number of `aesenc`: a.
  - State size: s.
  - Number of 128-bit input blocks: m.
- We successively look at:
  - $\rho = 3$.
  - $2 < \rho < 3$.
  - $\rho = 2$.
- Then, for each rate, we try to minimize the state size s.

General Case: Results with $\rho = 3$ ($s \leq 5$)

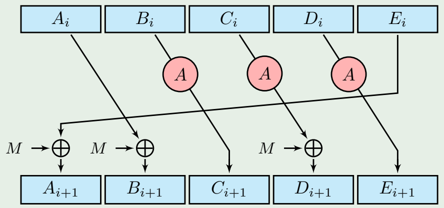
Preliminary results for $\rho = 3$

- Minimal state size: $s = 3$.
  - Input space can be completely exhausted.
  - No secure design exist.
- For state sizes $s = 4$ and $s = 5$.
  - There exists secure designs (24+ and 26+ active Sboxes).
  - However: not optimal designs.
  - Indeed: $a = 3 \Rightarrow$ empty cycles on all current processors.

$s = 4$



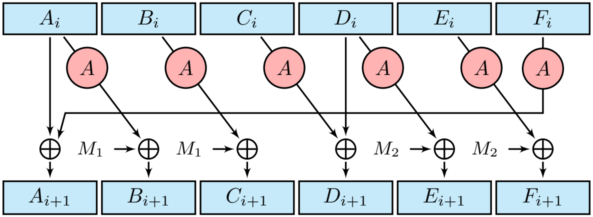
$s = 5$



General Case: Results with $\rho = 3$ ($s \geq 6$)

		$\rho = 3$			
		$s = 6$	$s = 7$	$s = 8$	$s = 9$
Details	a	6	6	6	9
	m	2	2	2	3
	x	0	3	4	0
	#SB	22	25	34	25
Performances (cpb)	⋆bridge	0.250	0.250	0.250	0.222
	⋆well	0.219	0.219	0.219	0.188
	Skylake	0.188	0.188	0.188	0.188

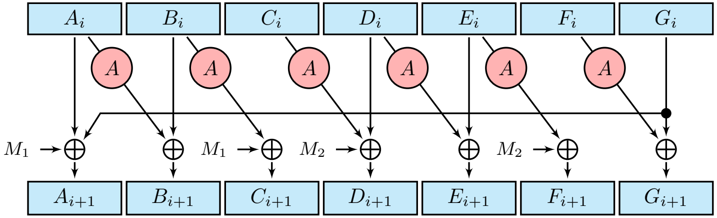
x: number of XOR not included in aesenc.



General Case: Results with $\rho = 3$ ($s \geq 6$)

		$\rho = 3$			
		$s = 6$	$s = 7$	$s = 8$	$s = 9$
Details	a	6	6	6	9
	m	2	2	2	3
	x	0	3	4	0
	#SB	22	25	34	25
Performances (cpb)	`*bridge`	0.250	0.250	0.250	0.222
	`*well`	0.219	0.219	0.219	0.188
	`Skylake`	0.188	0.188	0.188	0.188

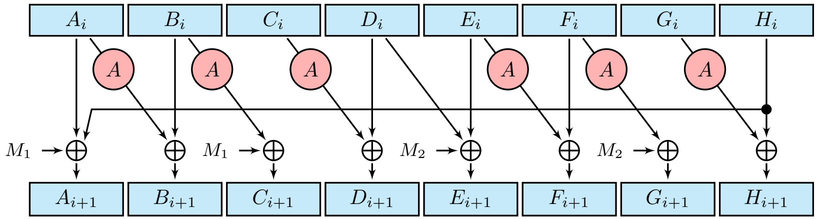
x: number of XOR not included in `aesenc`.



General Case: Results with $\rho = 3$ $(s \geq 6)$

		$\rho = 3$			
		$s = 6$	$s = 7$	$s = 8$	$s = 9$
Details	a	6	6	6	9
	m	2	2	2	3
	x	0	3	4	0
	#SB	22	25	34	25
Performances (cpb)	`⋆bridge`	0.250	0.250	0.250	0.222
	`⋆well`	0.219	0.219	0.219	0.188
	`Skylake`	0.188	0.188	0.188	0.188

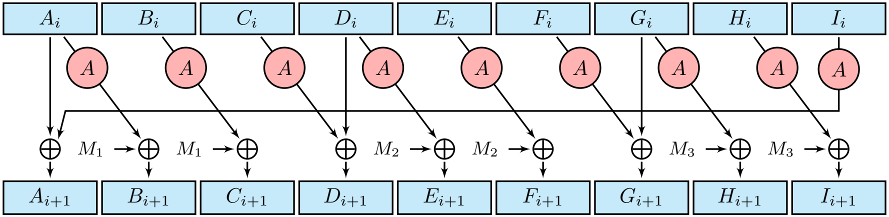
x: number of XOR not included in `aesenc`.



General Case: Results with $\rho = 3$ ($s \geq 6$)

		$\rho = 3$			
		$s = 6$	$s = 7$	$s = 8$	$s = 9$
Details	a	6	6	6	9
	m	2	2	2	3
	x	0	3	4	0
	#SB	22	25	34	25
Performances (cpb)	`⋆bridge`	0.250	0.250	0.250	0.222
	`⋆well`	0.219	0.219	0.219	0.188
	`Skylake`	0.188	0.188	0.188	0.188

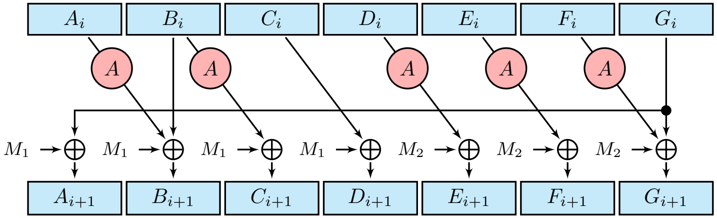
x: number of XOR not included in `aesenc`.



General Case: Results with $2 < \rho < 3$

		$\rho = 2.5$	
		$s = 7$	$s = 8$
Details	a	5	5
	m	2	2
	x	4	5
	#SB	22	23
Performances (cpb)	`*bridge`	0.250	0.250
	`*well`	0.219	0.219
	`Skylake`	0.188	0.188

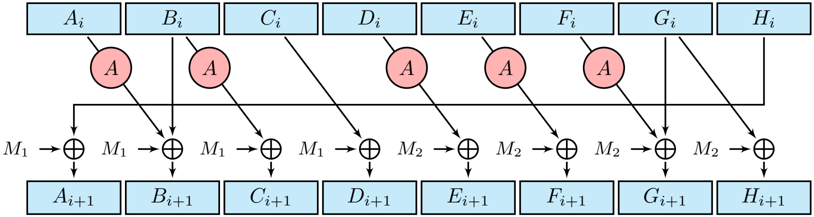
x: number of XOR not included in `aesenc`.



General Case: Results with $2 < \rho < 3$

		$\rho = 2.5$	
		$s = 7$	$s = 8$
Details	a	5	5
	m	2	2
	x	4	5
	#SB	22	23
Performances (cpb)	`*bridge`	0.250	0.250
	`*well`	0.219	0.219
	`Skylake`	0.188	0.188

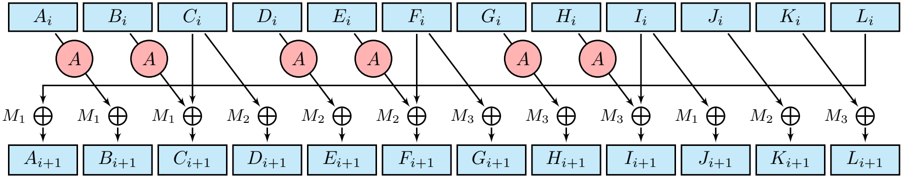
x: number of XOR not included in `aesenc`.



General Case: Results with $\rho = 2$

		$\rho = 2$
		$s = 12$
Details	a	6
	m	3
	x	9
	#SB	28
Performances (cpb)	`*bridge`	0.190
	`*well`	0.136
	`Skylake`	0.125

x: number of XOR not included in `aesenc`.


A_i B_i C_i D_i E_i F_i G_i H_i I_i J_i K_i L_i
A
M_1 M_1 M_1 M_2 M_2 M_2 M_3 M_3 M_3 M_1 M_2 M_3
A_{i+1} B_{i+1} C_{i+1} D_{i+1} E_{i+1} F_{i+1} G_{i+1} H_{i+1} I_{i+1} J_{i+1} K_{i+1} L_{i+1}

Conclusions

Summary:

- New building blocks based on the AES round function.
  - Extremely efficient using AES−NI instructions.
  - Still efficient on older platforms (only 2-3 rounds of AES).
- Several designs with different sizes and security margins:
  - **Smallest**: state of $6 \times 128 = 768$ bits at 0.188 c/B on Skylake.
  - **Fastest**: state of $12 \times 128 = 1536$ bits at 0.125 c/B on Skylake.

Open problem:

- Are rates smaller than 2 achievable?

Conclusions

Summary:

- New building blocks based on the AES round function.
  - Extremely efficient using AES−NI instructions.
  - Still efficient on older platforms (only 2-3 rounds of AES).
- Several designs with different sizes and security margins:
  - **Smallest**: state of $6 \times 128 = 768$ bits at 0.188 c/B on Skylake.
  - **Fastest**: state of $12 \times 128 = 1536$ bits at 0.125 c/B on Skylake.

Open problem:

- Are rates smaller than 2 achievable?

Thank you!