

How Fast Can Higher-Order Masking Be in Software?

Dahmun Goudarzi and Matthieu Rivain

EUROCRYPT 2017, Paris



- 1 ■ Introduction
- 2 ■ Field Multiplications
- 3 ■ Non-Linear Operations
- 4 ■ Generic Polynomial Methods
- 5 ■ Polynomial Methods for AES
- 6 ■ The Bitslice Strategy

Higher-Order Masking

$$x = x_1 + x_2 + \cdots + x_d$$

Higher-Order Masking

$$x = x_1 + x_2 + \cdots + x_d$$

- Linear operations: $O(d)$

Higher-Order Masking

$$x = x_1 + x_2 + \cdots + x_d$$

- Linear operations: $O(d)$
- Non-linear operations: $O(d^2)$

Higher-Order Masking

$$x = x_1 + x_2 + \cdots + x_d$$

- Linear operations: $O(d)$
 - Non-linear operations: $O(d^2)$
- Challenge for blockciphers: S-boxes

Ishai-Sahai-Wagner Multiplication

$$\sum_i c_i = \left(\sum_i a_i \right) \times \left(\sum_i b_i \right) = \sum_{i,j} a_i \times b_j$$

$$\begin{pmatrix} a_1b_1 & a_1b_2 & \dots & a_1b_d \\ 0 & a_2b_2 & \dots & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & a_db_d \end{pmatrix} + \begin{pmatrix} 0 & 0 & \dots & 0 \\ a_2b_1 & 0 & \dots & \vdots \\ \vdots & \vdots & & \vdots \\ a_db_1 & a_db_2 & \dots & 0 \end{pmatrix} + \begin{pmatrix} 0 & r_{1,2} & \dots & r_{1,d} \\ r_{1,2} & 0 & \dots & \vdots \\ & & \ddots & r_{d,d-1} \\ r_{1,d} & r_{d,d-1} & & 0 \end{pmatrix}$$

The Polynomial Methods

- Sbox seen as a polynomial over $GF(2^n)$

$$S(x) = \sum_{i=0}^n a_i x^i$$

The Polynomial Methods

- Sbox seen as a polynomial over $GF(2^n)$

$$S(x) = \sum_{i=0}^n a_i x^i$$


Generic Methods

$$S(x) = \sum_i (p_i \star q_i)(x)$$

- CRV decomposition, $\star = \times$ (CHES 2014)
- Algebraic decomposition, $\star = \circ$ (CRYPTO 2015)

The Polynomial Methods

- Sbox seen as a polynomial over $GF(2^n)$

$$S(x) = \sum_{i=0}^n a_i x^i$$

↙ ↘

Generic Methods

$$S(x) = \sum_i (p_i \star q_i)(x)$$

- CRV decomposition, $\star = \times$ (CHES 2014)
- Algebraic decomposition, $\star = \circ$ (CRYPTO 2015)

AES Specific Methods

$$S_{\text{AES}}(x) = \text{Aff}(x^{254})$$

- RP multiplication chain (CHES 2010)
- KHL multiplication chain (CHES 2011)

Our results

- Optimized implementations of state of the art higher-order masking techniques
- Bottom-up approach:
 - ▶ base field multiplication
 - ▶ ISW/CPRR
 - ▶ polynomial methods
- Finely tuned ARM assembly (parallelization)
- Alternative strategy: bitslice method (new AES and PRESENT speed records)

ARM

- 32-bit architecture with 16 registers (13 user accessible register)
- Barrelshifter: shifts and rotates virtually free
- Example: x -times and add on $\text{GF}(2)[x]$ in 1 cycle

```
EOR      $acc , $var , $acc , LSL #1
```

- 1 ■ Introduction
- 2 ■ Field Multiplications**
- 3 ■ Non-Linear Operations
- 4 ■ Generic Polynomial Methods
- 5 ■ Polynomial Methods for AES
- 6 ■ The Bitslice Strategy

Field Multiplication

- Goal: efficient implementation of multiplication over $\text{GF}(2^n)$
- Fastest method: precomputed look-up table
- Limitation: constrained memory on embedded system

n	4	5	6	7	8	9	10
Table size	0.25 kiB	1 kiB	4 kiB	16 kiB	64 kiB	512 kiB	2048 kiB

Field Multiplication

	bin mult v1	bin mult v2	exp-log v1	exp-log v2	kara.	half-tab	full-tab
clock cycles	$10n + 3$	$7n + 3$	18	16	19	10	4
registers	5	5	5	5	6	5	5
code size	52	$2^{n-1} + 48$	$2^{n+1} + 48$	$3 \cdot 2^n + 40$	$3 \cdot 2^n + 42$	$2^{\frac{3n}{2}+1} + 24$	$2^{2n} + 12$

Field Multiplication

	bin mult v1	bin mult v2	exp-log v1	exp-log v2	kara.	half-tab	full-tab
clock cycles	$10n + 3$	$7n + 3$	18	16	19	10	4
registers	5	5	5	5	6	5	5
code size	52	$2^{n-1} + 48$	$2^{n+1} + 48$	$3 \cdot 2^n + 40$	$3 \cdot 2^n + 42$	$2^{\frac{3n}{2}+1} + 24$	$2^{2n} + 12$

$$a \times b = (a_h x^{\frac{n}{2}} + a_\ell) \times (b_h x^{\frac{n}{2}} + b_\ell)$$

$$\text{Karatsuba} = T1[a_h \mid b_h] + T2[a_\ell \mid b_\ell] + T3[a_h + a_\ell \mid b_h + b_\ell]$$

Field Multiplication

	bin mult v1	bin mult v2	exp-log v1	exp-log v2	kara.	half-tab	full-tab
clock cycles	$10n + 3$	$7n + 3$	18	16	19	10	4
registers	5	5	5	5	6	5	5
code size	52	$2^{n-1} + 48$	$2^{n+1} + 48$	$3 \cdot 2^n + 40$	$3 \cdot 2^n + 42$	$2^{\frac{3n}{2}+1} + 24$	$2^{2n} + 12$

$$a \times b = (a_h x^{\frac{n}{2}} + a_\ell) \times (b_h x^{\frac{n}{2}} + b_\ell)$$

$$\text{Half table} = \text{T1}[a_h \mid a_\ell \mid b_h] + \text{T2}[a_h \mid a_\ell \mid b_\ell]$$

Field Multiplication

	bin mult v1	bin mult v2	exp-log v1	exp-log v2	kara.	half-tab	full-tab
clock cycles	$10n + 3$	$7n + 3$	18	16	19	10	4
registers	5	5	5	5	6	5	5
code size	52	56 B	80 B	88 B	90 B	152 B	268 B

- For $n = 4$: full table
 - ▶ Fastest multiplication: 4 clock cycles
 - ▶ Low code size: 268 B

Field Multiplication

	bin mult v1	bin mult v2	exp-log v1	exp-log v2	kara.	half-tab	full-tab
clock cycles	$10n + 3$	$7n + 3$	18	16	19	10	4
registers	5	5	5	5	6	5	5
code size	52	176 B	560 B	808 kiB	810 B	8216 B	64 kiB

- For $n = 8$: exp-log or half-tab
 - tradeoff between clock cycles and code size

- 1 ■ Introduction
- 2 ■ Field Multiplications
- 3 ■ Non-Linear Operations**
- 4 ■ Generic Polynomial Methods
- 5 ■ Polynomial Methods for AES
- 6 ■ The Bitslice Strategy

Quadratic Operations

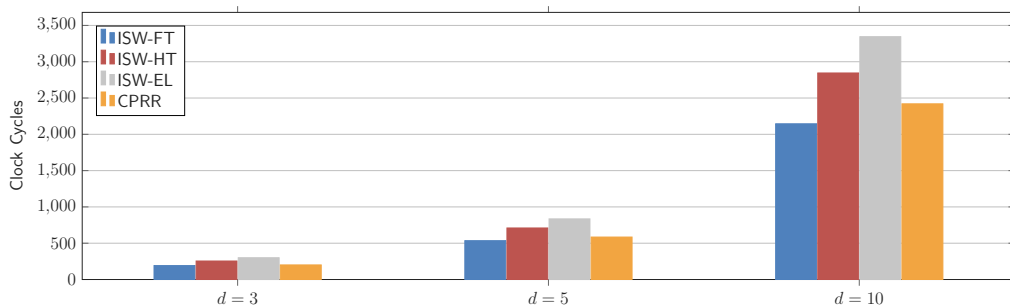
■ ISW

- ▶ Secure GF-mult of 2 operands
- ▶ Might need refreshing (see paper for details)

■ CPRR

- ▶ Evaluation of quadratic functions in 1 operand
- ▶ Similar to ISW: GF-mult $\rightarrow$ lookup tables
- ▶ Twice more random

Performances Comparisons



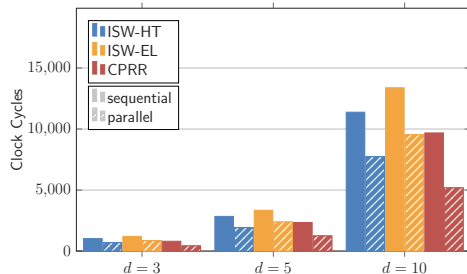
- ISW < CPRR when table too huge
- Asymptotical comp: 1 CPRR $\rightarrow$ 1.16 ISW-FT, 0.88 ISW-HT, 0.75 ISW-EL

Parallelization

- 32-bit register filled with only n -bit elements
- Perform several ISW/CPRR in parallel:
 - ▶ $n = 4 \rightarrow 8$ elements/register
 - ▶ $n = 8 \rightarrow 4$ elements/register
- Consequence:
 - ▶ Parallel: load, store, xor, loops
 - ▶ Sequential: GF mult, CPRR lookups

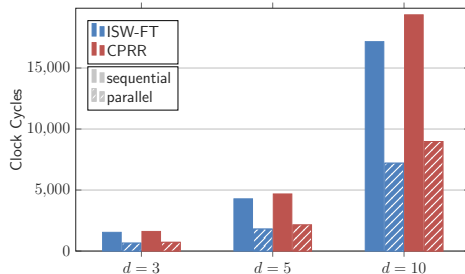
Performances Gain of Parallelization

■ $n = 8$ (4 elements)



■ Asympt. ratio: CPRR 54%.

■ $n = 4$ (8 elements)



■ Asympt. ratio: ISW 42%.

- 1 ■ Introduction
- 2 ■ Field Multiplications
- 3 ■ Non-Linear Operations
- 4 ■ Generic Polynomial Methods**
- 5 ■ Polynomial Methods for AES
- 6 ■ The Bitslice Strategy

Polynomial Decomposition

$$S(x) = \sum_i q_i(x) \star p_i(x)$$

Polynomial Decomposition

$$S(x) = \sum_i q_i(x) \star p_i(x)$$

- q_i : random linear combinations from a basis $\mathcal{B}$

Polynomial Decomposition

$$S(x) = \sum_i q_i(x) \star p_i(x)$$

- q_i : random linear combinations from a basis $\mathcal{B}$
- find p_i by solving a linear system

Polynomial Decomposition

$$S(x) = \sum_i q_i(x) \star p_i(x)$$

- q_i : random linear combinations from a basis $\mathcal{B}$
- find p_i by solving a linear system
- CRV vs AD:
 - ▶ CRV [CRV14]: $\star = \text{GF-multiplication} \rightarrow \text{ISW multiplication}$
 - ▶ AD [CPRR15]: $\star = \text{composition} \rightarrow \text{CPRR evaluation}$

CRV Improvement

- Use CPRR for the basis computation
- Example for $n = 8$:

CRV

$$x^3 = x \cdot x^2$$

$$x^7 = x \cdot (x^3)^2$$

$$x^{29} = x \cdot (x^7)^4$$

$$x^{87} = x^3 \cdot x^{29}$$

$$x^{251} = (x^6)^{16} \cdot (x^{87})^{128}$$

5 ISW

This paper

$$x^3 = x^3$$

$$x^9 = (x^3)^3$$

$$x^5 = x^5$$

$$x^{25} = (x^5)^5$$

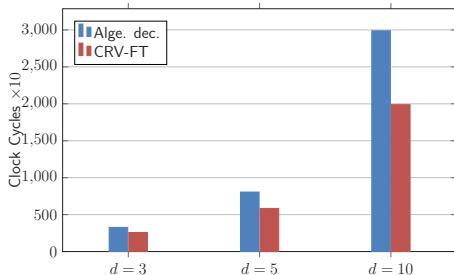
$$x^{125} = (x^{25})^5$$

$$x^{115} = (x^{125})^5$$

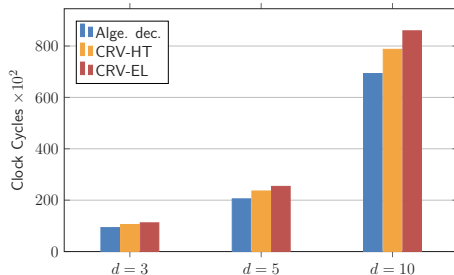
6 CPRR

Implementation Results

■ $n = 4$ (8 s-boxes in $//$)



■ $n = 8$ (4 s-boxes in $//$)



- 1 ■ Introduction
- 2 ■ Field Multiplications
- 3 ■ Non-Linear Operations
- 4 ■ Generic Polynomial Methods
- 5 ■ Polynomial Methods for AES**
- 6 ■ The Bitslice Strategy

Polynomial Methods for AES

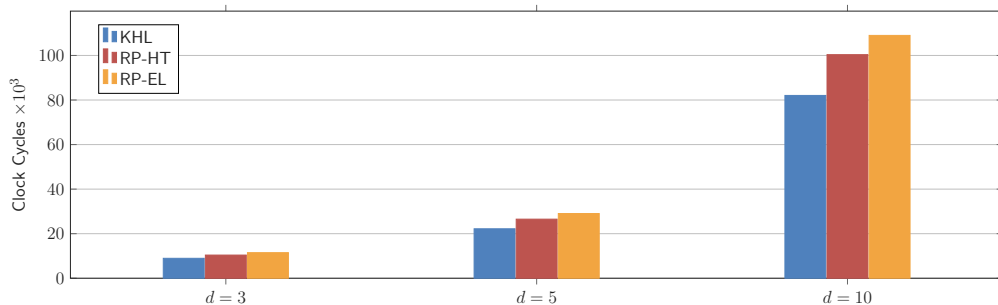
- Based on the specific algebraic structure of the AES:

$$S(x) = \text{Aff}(x^{2^{54}})$$

- RP10 method : 4 ISW mult
 - Security flaw due to refreshing
 - Patch [CPRR13]: 1 CPRR + 3 ISW
 - Improvement [GPS14]: 3 CPRR + 1 ISW
- KHL11 method: 5 ISW mult on $\text{GF}(16)$
 - Patch [this paper]: 1 CPRR + 4 ISW

Implementation Results

- 16 s-boxes in //

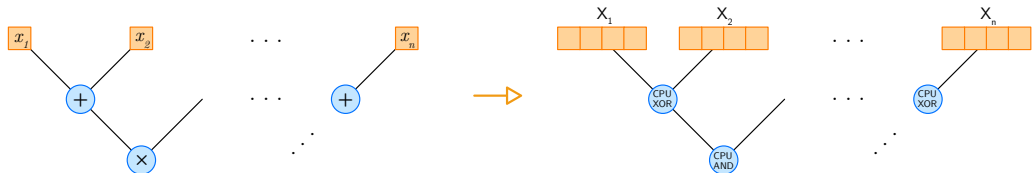


- KHL < RP-*: smaller elements $\rightarrow$ higher parallelization degree

- 1 ■ Introduction
- 2 ■ Field Multiplications
- 3 ■ Non-Linear Operations
- 4 ■ Generic Polynomial Methods
- 5 ■ Polynomial Methods for AES
- 6 ■ The Bitslice Strategy**

Bitslice for the AES

- Sbox seen as boolean circuit



- 16 S-boxes in //

Application for AES S-boxes

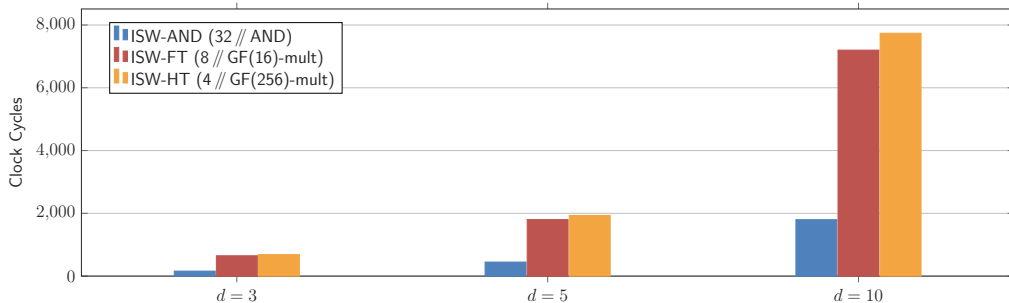
- Circuit for the AES S-box [BMP13]
 - ▶ 83 XOR gates
 - ▶ 32 AND gates
- Bitslice (16 s-boxes)
 - ▶ 83 XOR instructions
 - ▶ 32 AND instructions
- Masking at the order d :
 - ▶ $83 \times d$ XOR instructions
 - ▶ 32 ISW-AND

Improvement

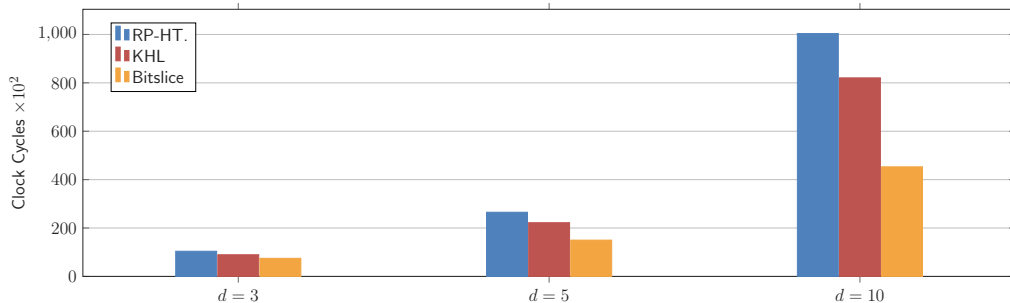
2 16-bit ISW-AND $\rightarrow$ 1 32-bit ISW-AND

- Goal: grouping AND gates per pairs
- Validation on BMP circuit
- 16 s-boxes = 16 ISW-AND $\rightarrow$ 1 ISW-AND per s-box

Performance Comparison of ISW

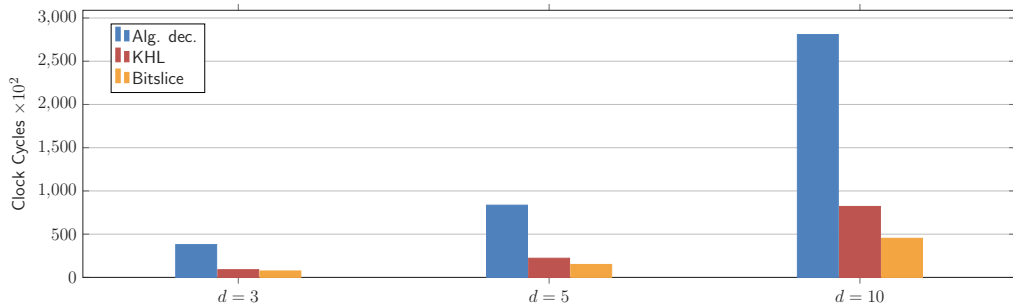


Performances for AES S-box



- RP-HT: 1 ISW-HT/CPRR per s-box
- KHL: 0.83 ISW-FT/CPRR per s-box
- Bitslice: 1 ISW-AND per s-box

AES vs Generic



- KHL $3.1\times$ faster than AD (for $n = 8$)
- Bitslice $2.3\times$ faster than KHL

Timing for AES and PRESENT Block-Cipher

	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 10$
Bitslice AES	0.89 ms	1.39 ms	1.99 ms	2.7 ms	8.01 ms
Bitslice PRESENT	0.62 ms	0.96 ms	1.35 ms	1.82 ms	5.13 ms

Conclusion

- Case study of state of the art polynomial techniques
- Optimization at each layer of the bottom-up approach
 - ▶ Selection of best field multiplication algorithm
 - ▶ Parallelization of non-linear operations
 - ▶ New optimal parameters for existing methods
- Alternative approach with bitslice and speed records for AES and PRESENT

Questions?