Protected Hardware Implementation of WAGE

Yunsi Fei[1] and Guang Gong[2] and Cheng Gongye[1] and
Kalikinkar Mandal[3] and Raghvendra Rohit[4] and Tianhong Xu[1] and
Yunjie Yi[2] and Nusa Zidaric[2]

[1]Department of Electrical and Computer Engineering, Northeastern University, USA
[2]Department of Electrical and Computer Engineering, University of Waterloo, Canada
[3]Faculty of Computer Science, University of New Brunswick, Canada
[4]University of Rennes, CNRS, IRISA, France

NIST LWC Workshop,
October 21, 2020

Outline

- Introduction to WAGE
- The masking scheme for WAGE
- Hardware implementation
- Conclusion and future work

Introduction to WAGE

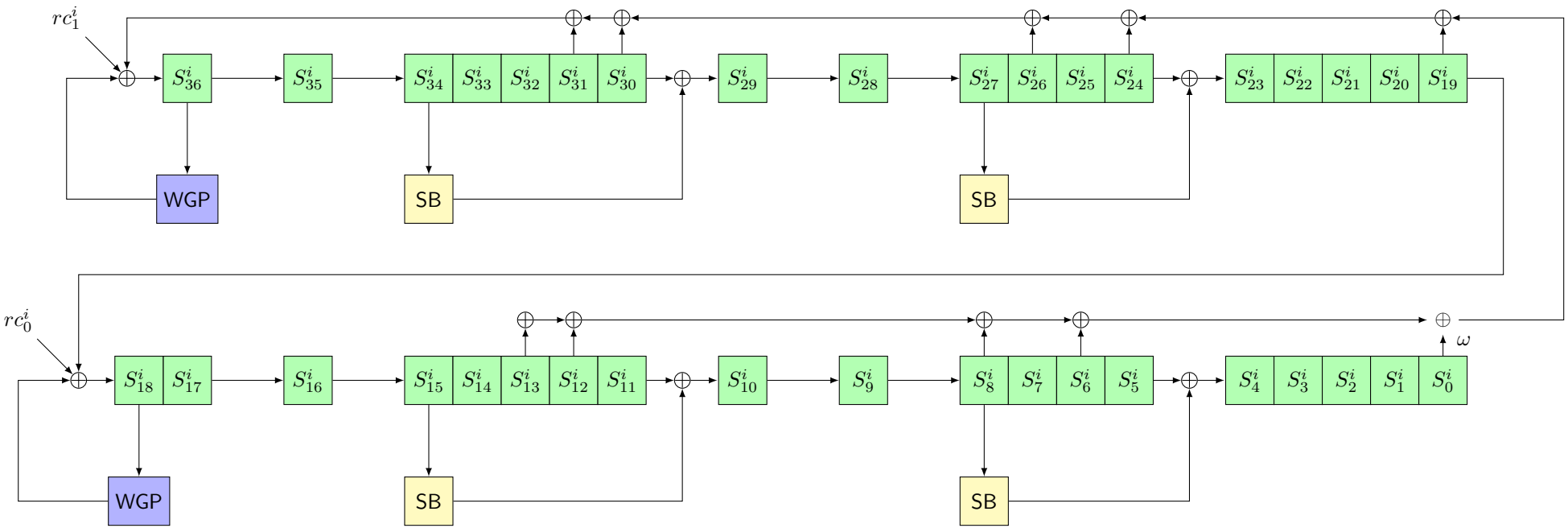
- WAGE is a hardware oriented authenticated encryption scheme (128-bit security)
- (unprotected) hardware implementations of WAGE have a small footprint[1]
  - minimal interface: **2900 GE** (STMicro 65 nm), **3290 GE** (TSMC 65 nm)
  - LWC hw API[2] : **332 slices** (Xilinx Artix-7)
    - among 10 smallest Xilinx Artix-7 candidates
    - the datapath (CryptoCore) takes less than 30% area
- WAGE is built on top of the initialization phase of the Welch-Gong stream cipher
  - theoretical foundations from the 90's
  - WG-29 proceeded to Phase 2 of the eSTREAM competition
- WAGE can be configured as a psuedorandom bit generator WG-PRBG

[1] comparison with other candidates is difficult due to differences in ASIC libraries and optimization levels for synthesis tools

[2] source: latest report by CERG, GMU, `https://eprint.iacr.org/2020/1207`

Introduction to WAGE

- WAGE permutation has 111 iterations of a round function
- round function: an LFSR, decimated WGPs and small Sboxes SB, round constants
- defined over $\mathbb{F}_{2^7}$, with internal state 259 bits (37 stage LFSR)



Introduction to WAGE

$\mathbb{F}_{2^7}$	$f(x) = x^7 + x^3 + x^2 + x + 1, \ f(\omega) = 0$ polynomial basis: $\mathrm{PB} = \{1, \omega, \ldots, \omega^6\}$
$x \in \mathbb{F}_{2^7}$	$x = \sum_{i=0}^{6} x_i \omega^i, x_i \in \mathbb{F}_2$ vector representation: $[x]_{\mathrm{PB}} = (x_0, x_1, x_2, x_3, x_4, x_5, x_6)$

LFSR	$fb = S_{31} \oplus S_{30} \oplus S_{26} \oplus S_{24} \oplus S_{19} \oplus S_{13} \oplus S_{12} \oplus S_8 \oplus S_6 \oplus (\omega \otimes S_0)$ $\omega \otimes (x_0, x_1, x_2, x_3, x_4, x_5, x_6) \leftarrow (x_6, x_0 \oplus x_6, x_1 \oplus x_6, x_2 \oplus x_6, x_3, x_4, x_5)$

WGP	$\mathsf{WGP7}(x^d) = x^d + (x^d+1)^{33} + (x^d+1)^{39} + (x^d+1)^{41} + (x^d+1)^{104}, \quad d = 13$

SB Q	$Q(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_0 \oplus (x_2 \quad x_3), x_1, x_2, \overline{x}_3 \oplus (x_5 \quad x_6), x_4, \overline{x}_5 \oplus (x_2 \quad x_4), x_6)$
SB P	$P(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_6, x_3, x_0, x_4, x_2, x_5, x_1)$
SB R	$R(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_6, \overline{x}_3 \oplus (x_5 \quad x_6), x_0 \oplus (x_2 \quad x_3), x_4, x_2, \overline{x}_5 \oplus (x_2 \quad x_4), x_1)$
SB	$(x_0, x_1, x_2, x_3, x_4, x_5, x_6) \leftarrow R^5(x_0, x_1, x_2, x_3, x_4, x_5, x_6)$ $(x_0, x_1, x_2, x_3, x_4, x_5, x_6) \leftarrow Q \ (x_0, x_1, x_2, x_3, x_4, x_5, x_6)$ $(x_0, x_1, x_2, x_3, x_4, x_5, x_6) \leftarrow \quad (\overline{x}_0, x_1, \overline{x}_2, x_3, x_4, x_5, x_6)$

state update function	$S_{24} \leftarrow S_{24} \oplus \mathsf{SB}(S_{27})$	$S_5 \leftarrow S_5 \oplus \mathsf{SB}(S_8)$	$S_j \leftarrow S_{j+1}$ for $0 \le j \le 35$
	$S_{30} \leftarrow S_{30} \oplus \mathsf{SB}(S_{34})$	$S_{11} \leftarrow S_{11} \oplus \mathsf{SB}(S_{15})$	
	$fb \leftarrow fb \oplus \mathsf{WGP}(S_{36}) \oplus rc_1$	$S_{19} \leftarrow S_{19} \oplus \mathsf{WGP}(S_{18}) \oplus rc_0$	$S_{36} \leftarrow fb$

The masking scheme for WAGE

- masking:
  - variable x is masked with a random value r: $x' = x \oplus r$
  - notation: $[[x]] = (r, x')$

- adversarial model: attacker can probe up to t intermediate variables in the circuit
- number of shares $n = t + 1$ (t-SNI or t-strong non interference security)
- n-order masking:
  - variable x is shared among n variables: $x = x^1 \oplus x^2 \oplus \cdots \oplus x^n$
  - notation: $[[x]] = (x^1, x^2, \ldots, x^n)$

  - $[[x]] \oplus [[y]] = (x^1 \oplus y^1, x^2 \oplus y^2, \ldots, x^n \oplus y^n)$
  - $[[\bar{x}]] = (\bar{x}^1, x^2, \ldots, x^n)$
  - $[[x]]$ $[[y]]$: use t-SNI secure AND gadget

The masking scheme for WAGE

n shares:

state	$\mathbf{S} = \mathbf{S}^1 \oplus \mathbf{S}^2 \oplus \cdots \oplus \mathbf{S}^n$ where $\mathbf{S}^k = (S^k_{36}, \ldots, S^k_0)$, $1 \le k \le n$
stage	$[[S_j]] = (S^1_j, S^2_j, \ldots, S^n_j)$ or bit-wise $([[x_{j,0}]], [[x_{j,1}]], \ldots, [[x_{j,6}]])$, $0 \le j \le 36$

LFSR (linear)	$fb^k = \mathsf{FB}(\mathbf{S}^k)$, $1 \le k \le n$ $\omega \otimes [[S_0]] \leftarrow ([[x_6]], [[x_0]] \oplus [[x_6]], [[x_1]] \oplus [[x_6]], [[x_2]] \oplus [[x_6]], [[x_3]], [[x_4]], [[x_5]])$

WGP	need ANF expressions for each of the 7 output bits: WGP $\Rightarrow$ SecWGP

SB	$Q(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_0 \oplus (x_2 \quad x_3), x_1, x_2, \overline{x}_3 \oplus (x_5 \quad x_6), x_4, \overline{x}_5 \oplus (x_2 \quad x_4), x_6)$
$\Downarrow$	$\Rightarrow \quad (\ldots, [[\overline{x_3}]] \oplus ([[x_5]] \quad [[x_6]]), \ldots)$
SecSB	use t-SNI secure AND gadgets

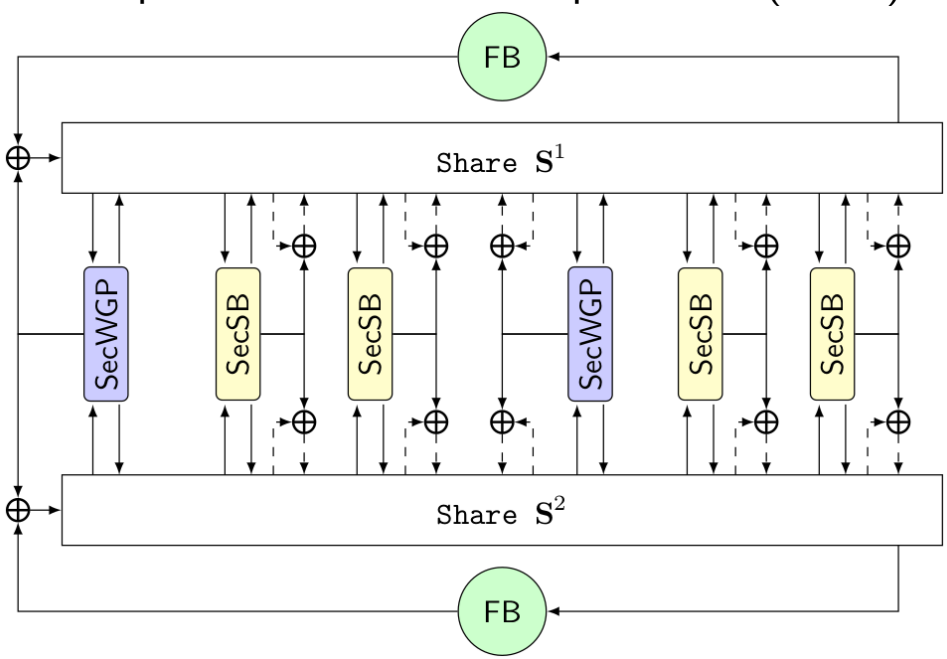
state update function	$[[S_{24}]] \leftarrow [[S_{24}]] \oplus \mathsf{SecSB}([[S_{27}]])$	$[[S_5]] \leftarrow [[S_5]] \oplus \mathsf{SecSB}([[S_8]])$	
	$[[S_{30}]] \leftarrow [[S_{30}]] \oplus \mathsf{SecSB}([[S_{34}]])$	$[[S_{11}]] \leftarrow [[S_{11}]] \oplus \mathsf{SecSB}([[S_{15}]])$	shift shared state:
	$[[tmp]] \leftarrow \mathsf{SecWGP}([[S_{36}]])$	$[[S_{19}]] \leftarrow [[S_{19}]] \oplus \mathsf{SecWGP}([[S_{18}]])$	$[[S_j]] \leftarrow [[S_{j+1}]]$
	$fb^1 \leftarrow fb^1 \oplus tmp^1 \oplus rc_1$	$S^1_{19} \leftarrow S^1_{19} \oplus rc_0$	for $0 \le j \le 35$
	$fb^k \leftarrow fb^k \oplus tmp^k, 2 \le k \le n$		$[[S_{36}]] \leftarrow [[fb]]$

The masking scheme for WAGE

n shares:

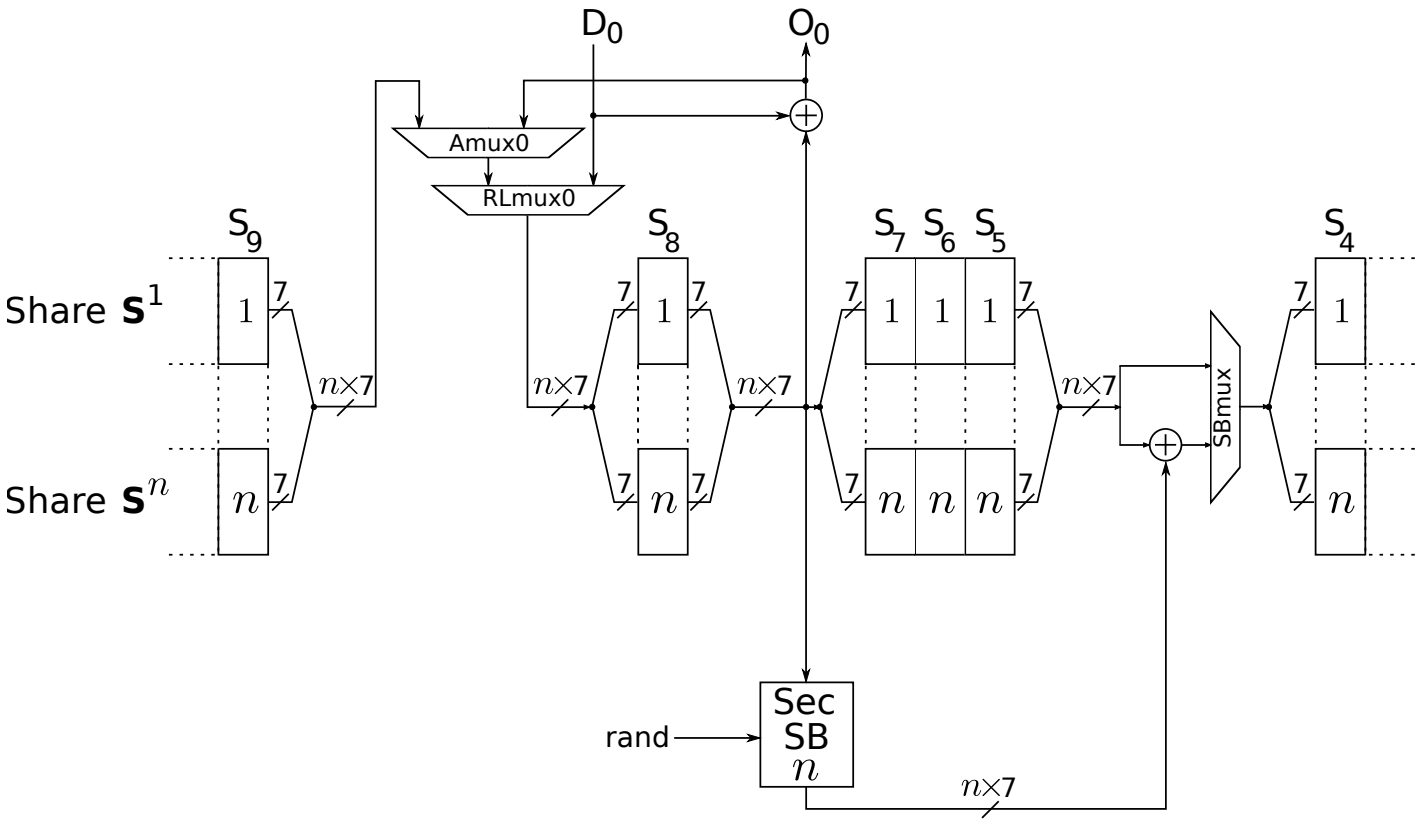
state update function	$[[S_{24}]] \leftarrow [[S_{24}]] \oplus \mathsf{SecSB}([[S_{27}]])$	$[[S_5]] \leftarrow [[S_5]] \oplus \mathsf{SecSB}([[S_8]])$	
	$[[S_{30}]] \leftarrow [[S_{30}]] \oplus \mathsf{SecSB}([[S_{34}]])$	$[[S_{11}]] \leftarrow [[S_{11}]] \oplus \mathsf{SecSB}([[S_{15}]])$	shift shared state:
	$[[tmp]] \leftarrow \mathsf{SecWGP}([[S_{36}]])$	$[[S_{19}]] \leftarrow [[S_{19}]] \oplus \mathsf{SecWGP}([[S_{18}]])$	$[[S_j]] \leftarrow [[S_{j+1}]]$
	$fb^1 \leftarrow fb^1 \oplus tmp^1 \oplus rc_1$	$S^1_{19} \leftarrow S^1_{19} \oplus rc_0$	for $0 \leq j \leq 35$
	$fb^k \leftarrow fb^k \oplus tmp^k, 2 \leq k \leq n$		$[[S_{36}]] \leftarrow [[fb]]$

Schematic of the masked WAGE permutation for 1-order protection ($n = 2$):



Hardware implementation

- protected hardware implementation is built on top of original WAGE hardware
- the datapath was modified with additional:
  - state registers and LFSR feedback XOR gates (e.g. $\mathbf{S}^1, \ldots, \mathbf{S}^n$)
  - XOR gates and MUXes for non-linear components (e.g. `SBmux`)
  - MUXes to support the mode (e.g. $\texttt{Amux}_0$, $\texttt{RLmux}_0$)



Hardware implementation

recall:

SB $\Downarrow$ SecSB	$Q(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_0 \oplus (x_2 \quad x_3), x_1, x_2, \overline{x}_3 \oplus (x_5 \quad x_6), x_4, \overline{x}_5 \oplus (x_2 \quad x_4), x_6)$ $\Rightarrow \quad (\ldots, [[\overline{x_3}]] \oplus ([[x_5]] \quad [[x_6]]), \ldots)$ use t-SNI secure AND gadgets
SecSB	$([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow R^5([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$ $([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow Q \ ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$ $([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow \quad ([[\overline{x_0}]], [[x_1]], [[\overline{x_2}]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$

- SecMult: t-SNI secure AND gadget [1]
- number of random bits needed: $\frac{n(n-1)}{2}$
- assume random bits always available
- unrolled: $[[z]]$ computed in a single clock cycle
- replace each in SB with SecMult gadget
- SB also unrolled

```
1: for i = 1 to n do
2:     z^i ← x^i y^i
3: end for
4: for i = 1 to n do
5:     for j = i + 1 to n do
6:         r ←$ F_2,  z^i ← z^i ⊕ r
7:         t ← x^i y^j,  r ← r ⊕ t
8:         t ← x^j y^i,  r ← r ⊕ t
9:         z^j ← z^j ⊕ r
10:    end for
11: end for
```

[1] Barthe *et al.* , "Strong Non-Interference and Type-Directed Higher-Order Masking", CCS'16

Hardware implementation

recall:

WGP	WGP7$(x^d) = x^d + (x^d+1)^{33} + (x^d+1)^{39} + (x^d+1)^{41} + (x^d+1)^{104}, \quad d = 13$
	need ANF expressions for each of the 7 output bits: WGP $\Rightarrow$ SecWGP

design flow:

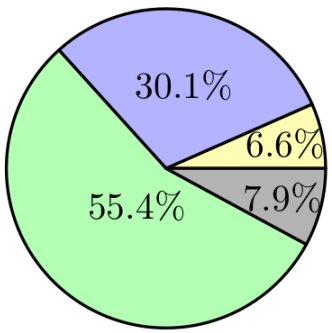
1. ANF: obtain algebraic normal form for every component of the WGP output
2. AOXI[1]: synthesis tools restricted to use only AND, OR, XOR, and NOT gates
3. AXI: replace OR gates by AND and NOT gates, optimize
4. SecWGP (protected AXI): replace AND gates by SecMult gadgets

Implementation Approach	Area [GE]	Number of gates AND	OR	XOR	INV
Constant array	258	–			
ANF	958	1132	–	439	–
AOIX	825	146	30	310	39
AXI	759	172	–	313	66

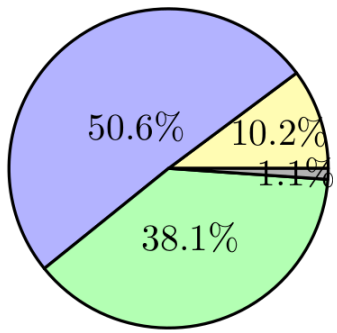
[1] INV is used instead of NOT to avoid confusion with number of shares n

Hardware implementation

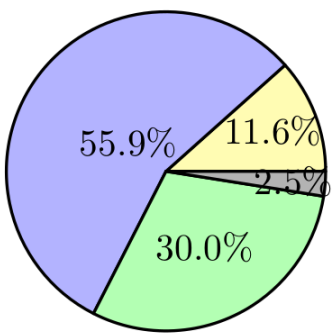
- Hardware implementation area breakdown by components using SecMult



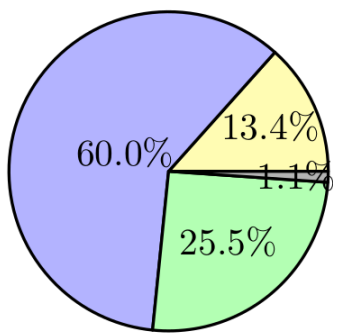
unprotected



$n = 2$



$n = 3$



$n = 4$

Hardware implementation

- Implementation area for ST Micro 65nm (post-PAR) [GE]
- Common share SecMult for SB: $(x_2 \quad x_3)$ and $(x_2 \quad x_4)$
  common share SecMult implementation of WAGE omitted for brevity

SB		
Algorithm	SecMult	SecMult+ comm. sh.
Unprotected	63	
$n = 2$	285	285
$n = 3$	626	715
$n = 4$	1140	1275

	WGP	WAGE
Algorithm	SecMult	SecMult
Const. array	258	2900
Unprotected	759	3830
$n = 2$	2830	11177
$n = 3$	6030	21566
$n = 4$	10200	33985

Conclusion and future work

- we presented the first high-order masking scheme of WAGE
- implementation results for ST Micro 65 nm and TSMC 65nm
- comparison with other candidates is difficult

  different countermeasures, ASIC libraries, optimization levels for synthesis tools

- future work:
  - iterative implementation of SecSB and SecWGP
    to reduce number of random bits needed per single clock cycle
  - explore tradeoffs among the throughput, hardware area, and
    the amount of randomness available per single clock cycle
  - exploring other countermeasures
    threshold, unified masked multiplication, domain oriented masking

- more details available at `https://eprint.iacr.org/2020/1202`

  this work will also be presented in SAC2020 tomorrow

- Acknowledgements: Hardware implementations in this work are based on the original WAGE hardware. Authors would like to thank Dr. Mark Aagaard for the help with synthesis tools.

Thank You!