Protected Hardware Implementation of WAGE

Yunsi Fei\textsuperscript{1} and Guang Gong\textsuperscript{2} and Cheng Gongye\textsuperscript{1} and Kalikinkar Mandal\textsuperscript{3} and Raghvendra Rohit\textsuperscript{4} and Tianhong Xu\textsuperscript{1} and Yunjie Yi\textsuperscript{2} and Nusa Zidaric\textsuperscript{2}

\textsuperscript{1}Department of Electrical and Computer Engineering, Northeastern University, USA
\textsuperscript{2}Department of Electrical and Computer Engineering, University of Waterloo, Canada
\textsuperscript{3}Faculty of Computer Science, University of New Brunswick, Canada
\textsuperscript{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: \textbf{2900 GE} (STMicro 65 nm), \textbf{3290 GE} (TSMC 65 nm)
  - LWC hw API\(^2\): \textbf{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
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

- **F₂₇**
  \[ f(x) = x^7 + x^3 + x^2 + x + 1, \quad f(\omega) = 0 \]
  Polynomial basis: \( PB = \{1, \omega, \ldots, \omega^6\} \)

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

- **WGP**
  \[ WGP_7(x^d) = x^d + (x^d + 1)^33 + (x^d + 1)^39 + (x^d + 1)^41 + (x^d + 1)^104, \quad d = 13 \]

- **SB**
  \[
  \begin{align*}
  Q(x_0, x_1, x_2, x_3, x_4, x_5, x_6) &= (x_0 \oplus (x_2 \ x_3), x_1, x_2, \overline{x_3} \oplus (x_5 \ x_6), x_4, \overline{x_5} \oplus (x_2 \ x_4), x_6) \\
  P(x_0, x_1, x_2, x_3, x_4, x_5, x_6) &= (x_6, x_3, x_0, x_4, x_2, x_5, x_1) \\
  R(x_0, x_1, x_2, x_3, x_4, x_5, x_6) &= (x_6, \overline{x_3} \oplus (x_5 \ x_6), x_0 \oplus (x_2 \ x_3), x_4, x_2, \overline{x_5} \oplus (x_2 \ x_4), x_1) \\
  \end{align*}
  \]

- **state update function**
  \[
  \begin{align*}
  S_24 &\leftarrow S_24 \oplus SB(S_27) \\
  S_30 &\leftarrow S_30 \oplus SB(S_34) \\
  S_5 &\leftarrow S_5 \oplus SB(S_8) \\
  S_11 &\leftarrow S_11 \oplus SB(S_{15}) \\
  S_{19} &\leftarrow S_{19} \oplus WGP(S_{18}) \oplus rc_0 \\
  S_j &\leftarrow S_{j+1} \quad \text{for} \quad 0 \leq j \leq 35 \\
  S_36 &\leftarrow fb
  \end{align*}
  \]
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: $[[x]] = (x^1, x^2, \ldots, x^n)$

\[
[[x]] \oplus [[y]] = (x^1 \oplus y^1, x^2 \oplus y^2, \ldots, x^n \oplus y^n)
\]
\[
[[\overline{x}]] = (\overline{x}^1, x^2, \ldots, x^n)
\]
\[
[[x]] \cap [[y]]: \text{ use } t\text{-SNI secure AND gadget}
\]
The masking scheme for WAGE

$n$ shares:

<table>
<thead>
<tr>
<th>state</th>
<th>( S = S^1 \oplus S^2 \oplus \cdots \oplus S^n ) where ( S^k = (S^k_{36}, \ldots, S^k_0), 1 \leq k \leq n )</th>
</tr>
</thead>
</table>

| stage | \([S_j] = (S^1_j, S^2_j, \ldots, S^n_j)\) or bit-wise \(([x_{j,0}], [x_{j,1}], \ldots, [x_{j,6}]), 0 \leq j \leq 36\) |

| LFSR | \( fb^k = FB(S^k), 1 \leq k \leq n \) |
| linear | \( \omega \otimes [S_0] \leftarrow ([x_6], [x_0] \oplus [x_6], [x_1] \oplus [x_6], [x_2] \oplus [x_6], [x_3], [x_4], [x_5]) \) |

| WGP | need ANF expressions for each of the 7 output bits: \( \text{WGP} \Rightarrow \text{SecWGP} \) |

| SB | \( Q(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_0 \oplus (x_2 x_3), x_1, x_2, x_3 \oplus (x_5 x_6), x_4, x_5 \oplus (x_2 x_4), x_6) \) |
| \( \downarrow \) | \( \Rightarrow \) \( (\ldots, [x_3] \oplus ([x_5] [x_6]), \ldots) \) |
| SecSB | use t-SNI secure AND gadgets |

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

$n$ shares:

| state     | $[[S_{24}]] \leftarrow [[S_{24}]] \oplus \text{SecSB}([[S_{27}]])$ | $[[S_{5}]] \leftarrow [[S_{5}]] \oplus \text{SecSB}([[S_{8}]])$ |
| update    | $[[S_{30}]] \leftarrow [[S_{30}]] \oplus \text{SecSB}([[S_{34}]])$ | $[[S_{11}]] \leftarrow [[S_{11}]] \oplus \text{SecSB}([[S_{15}]])$ |
| function  | $[[tmp]] \leftarrow \text{SecWGP}([[S_{36}]])$ | $[[S_{19}]] \leftarrow [[S_{19}]] \oplus \text{SecWGP}([[S_{18}]])$ |
|           | $fb^1 \leftarrow fb^1 \oplus tmp^1 \oplus rc_1$ | $S^1_{19} \leftarrow S^1_{19} \oplus rc_0$ |
|           | $fb^k \leftarrow fb^k \oplus tmp^k$, $2 \leq k \leq n$ |                      |

Shift shared state:

$[[S_j]] \leftarrow [[S_{j+1}]]$

for $0 \leq j \leq 35$

$[[S_{36}]] \leftarrow [[fb]]$

Schematic of the masked WAGE permutation for 1-order protection ($n = 2$):
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. $S^1, \ldots, S^n$)
  - XOR gates and MUXes for non-linear components (e.g. SBmux)
  - MUXes to support the mode (e.g. Amux0, RLMux0)
Hardware implementation

recall:

\[
Q(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_0 \oplus (x_2 \ x_3), x_1, x_2, \overline{x}_3 \oplus (x_5 \ x_6), x_4, \overline{x}_5 \oplus (x_2 \ x_4), x_6)
\]

\[
\Rightarrow (\ldots, [[\overline{x}_3]] \oplus ([[x_5]] [[x_6]]), \ldots)
\]

use t-SNI secure AND gadgets

<table>
<thead>
<tr>
<th>SB</th>
<th>SecSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q(x_0, x_1, x_2, x_3, x_4, x_5, x_6) = (x_0 \oplus (x_2, x_3), x_1, x_2, \overline{x}_3 \oplus (x_5, x_6), x_4, \overline{x}_5 \oplus (x_2, x_4), x_6)$</td>
<td>$Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow R^5([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$</td>
</tr>
<tr>
<td>$Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$</td>
<td>$Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$</td>
</tr>
<tr>
<td>$Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$</td>
<td>$Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]]) \leftarrow Q ([[x_0]], [[x_1]], [[x_2]], [[x_3]], [[x_4]], [[x_5]], [[x_6]])$</td>
</tr>
</tbody>
</table>

- SecMult: t-SNI secure AND gadget
- 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

---

1Barthe et al., “Strong Non-Interference and Type-Directed Higher-Order Masking”, CCS’16
Hardware implementation

recall:

\[
WGP7(x^d) = x^d + (x^d + 1)^{33} + (x^d + 1)^{39} + (x^d + 1)^{41} + (x^d + 1)^{104}, \quad d = 13
\]

need ANF expressions for each of the 7 output bits: WGP ⇒ SecWGP

design flow:

1. ANF: obtain algebraic normal form for every component of the WGP output
2. AOIX\(^1\): 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

<table>
<thead>
<tr>
<th>Implementation Approach</th>
<th>Area [GE]</th>
<th>Number of gates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AND</td>
<td>OR</td>
</tr>
<tr>
<td>Constant array</td>
<td>258</td>
<td></td>
</tr>
<tr>
<td>ANF</td>
<td>958</td>
<td>1132</td>
</tr>
<tr>
<td>AOIX</td>
<td>825</td>
<td>146</td>
</tr>
<tr>
<td>AXI</td>
<td>759</td>
<td>172</td>
</tr>
</tbody>
</table>

\(^1\)INV is used instead of NOT to avoid confusion with number of shares \(n\)
Hardware implementation area breakdown by components using SecMult

- **unprotected**
  - 55.4% LFSR
  - 30.1% 2×WGP
  - 6.6% 4×SB
  - 7.9% other

- **n = 2**
  - 50.6% LFSR
  - 10.2% 2×WGP
  - 1.1% 4×SB
  - 38.1% other

- **n = 3**
  - 55.9% LFSR
  - 11.6% 2×WGP
  - 2.5% 4×SB
  - 30.0% other

- **n = 4**
  - 60.0% LFSR
  - 13.4% 2×WGP
  - 1.1% 4×SB
  - 25.5% other
Protected Hardware Implementation of WAGE

Hardware implementation

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

<table>
<thead>
<tr>
<th>SB</th>
<th>Algorithm</th>
<th>SecMult</th>
<th>SecMult+</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>comm. sh.</td>
<td></td>
</tr>
<tr>
<td>Unprotected</td>
<td></td>
<td></td>
<td>63</td>
</tr>
<tr>
<td>n = 2</td>
<td></td>
<td>285</td>
<td>285</td>
</tr>
<tr>
<td>n = 3</td>
<td></td>
<td>626</td>
<td>715</td>
</tr>
<tr>
<td>n = 4</td>
<td></td>
<td>1140</td>
<td>1275</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WGP</th>
<th>Algorithm</th>
<th>SecMult</th>
<th>WAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const. array</td>
<td></td>
<td>258</td>
<td>2900</td>
</tr>
<tr>
<td>Unprotected</td>
<td></td>
<td>759</td>
<td>3830</td>
</tr>
<tr>
<td>n = 2</td>
<td></td>
<td>2830</td>
<td>11177</td>
</tr>
<tr>
<td>n = 3</td>
<td></td>
<td>6030</td>
<td>21566</td>
</tr>
<tr>
<td>n = 4</td>
<td></td>
<td>10200</td>
<td>33985</td>
</tr>
</tbody>
</table>
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!