WHITE-BOX CRYPTOGRAPHY IN THE GRAY BOX
− A HARDWARE IMPLEMENTATION AND ITS SIDE CHANNELS −

PASCAL SASDRICH, AMIR MORADI, TIM GÜNEYSU

23RD INT. CONFERENCE ON FAST SOFTWARE ENCRYPTION, BOCHUM, GERMANY
MARCH 21, 2016
THE STORY OF THIS WORK

HOW DID THIS WORK START?

“The challenge that white-box cryptography aims to address is to implement a cryptographic algorithm in software in such a way that cryptographic assets remain secure even when subject to white-box attacks.”

(Sources: www.whiteboxcrypto.com)

SOME QUESTIONS AROSE:

1. If an implementation is secure against white-box attacks, will it be secure against grey-box (i.e. side-channel) attacks as well?

2. Can we use white-box cryptography or adopt its ideas to build side-channel secure implementations?

3. Why do we only address software implementations? Can we implement white-box cryptography in hardware, too?

THIS IS THE STORY OF A

WHITE-BOX HARDWARE IMPLEMENTATION AND ITS SIDE CHANNELS.
Modern cryptography differentiates between three models to estimate the capabilities of an adversary:

**BLACK-BOX ADVERSARY MODEL:**
- trusted environment
- secure communication endpoints
- adversary can only observe input/output behavior (black-box)

**GREY-BOX ADVERSARY MODEL:**
- adversary has limited access to implementation internals
- usually targets implementations rather than algorithms

**WHITE-BOX ADVERSARY MODEL:**
- capabilities are virtually unlimited
- full control over implementation and execution environment
- white-box secure implementation behaves as virtual black-box
GENERAL IDEA OF WHITE-BOX CRYPTOGRAPHY

An ideal white-box implementation would be a single look-up table (for a fixed secret key).
– Obviously this is impractical for modern ciphers with block and key sizes of 128 bits and more.

So, practically feasible approaches for round-based symmetric block ciphers look like:

\[
\left( f^{(r+1)} \right)^{-1} \circ E_1 \circ f^1 \circ \ldots \circ \left( f^{(3)} \right)^{-1} \circ E_2 \circ f^2 \circ \left( f^{(2)} \right)^{-1} \circ E_1 \circ f^1
\]

\[
= \left( f^{(r+1)} \right)^{-1} \circ E_1 \circ \ldots \circ E_2 \circ E_1 \circ f^1 = \left( f^{(r+1)} \right)^{-1} \circ E_K \circ f^1,
\]

This principle was initially proposed by Chow et al. for DES [1] and AES [2] in 2002.

WHITE-BOX IMPLEMENTATIONS CAN BE SEEN AS NETWORK OF RANDOMIZED LOOK-UP TABLES.

---

HARDWARE WHITE-BOX IMPLEMENTATION OF AES

DESIGN AND CONSTRUCTION IN FOUR STEPS:

1. **PARTIAL EVALUATION**
   - S-box and key addition are merged (T-Box)

2. **MATRIX PARTITIONING**
   - MixColumns is added to T-Box (TMC-Box)

3. **MIXING BIJECTIONS**
   - Linear encodings (8-bit and 32-bit) are added

4. **NIBBLE ENCODINGS**
   - 4-bit non-linear nibble encodings are applied to all tables

**HARDWARE (FPGA) IMPLEMENTATION:**
- \( \mathcal{L} \)-II and \( \mathcal{L} \)-III are mapped into BRAM
- \( \mathcal{L} \)-IV is mapped into LUTs
RESULTS FOR FPGA BASED IMPLEMENTATION

<table>
<thead>
<tr>
<th>Look-Up Tables</th>
<th>Resources</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>Type</td>
<td>Size</td>
</tr>
<tr>
<td>16</td>
<td>$L$-Ia</td>
<td>(8 x 32-bit)</td>
</tr>
<tr>
<td>16</td>
<td>$L$-Ib</td>
<td>(8 x 8-bit)</td>
</tr>
<tr>
<td>144</td>
<td>$L$-II</td>
<td>(8 x 32-bit)</td>
</tr>
<tr>
<td>144</td>
<td>$L$-III</td>
<td>(8 x 32-bit)</td>
</tr>
<tr>
<td>1728</td>
<td>$L$-IV</td>
<td>(8 x 4-bit)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utilization (for XC7K160T)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SIDEC-CHANNEL ANALYSIS

OUR SETUP:
- SAKURA-X Board (Kintex-7)
- 500 MS/s, FPGA@3MHz

EVALUATION:
- 10,000,000 power traces
- classical (single bit) DPA

RESULTS:
- target value: 5th S-Box output
- key hypotheses: 8-bit (256)
- one bit allowed to recover key (bit 2)

WHY IS A CLASSICAL DPA POSSIBLE?
MATHEMATICAL ANALYSIS

TO UNDERSTAND THE PROBLEM, WE APPLIED A WELL KNOWN TOOL FOR BOOLEAN FUNCTIONS:

Definition 1. Let \( x = \langle x_1, ..., x_n \rangle \), \( \omega = \langle \omega_1, ..., \omega_n \rangle \) be elements of \( \{0, 1\}^n \) and \( x \cdot \omega = x_1\omega_1 \oplus ... \oplus x_n\omega_n \). Let \( f(x) \) be a Boolean function of \( n \) variables. Then the Walsh transform of the function \( f(x) \) is a real valued function over \( \{0, 1\}^n \) that can be defined as \( W_f(\omega) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus x \cdot \omega} \).

MATHEMATICAL EVALUATION OF \( L \)-Ia TABLE:
- assume external encodings are known or non-existing
- consider table as 32 different Boolean functions \( f_i \)
- calculate Walsh transform for all \( f_i \) and all key candidates (for different \( \omega \))

RESULTS:
- Walsh transform for \( \omega \) with \( HW(\omega) = 1 \) confirm results of side-channel analysis
- directly related to single bit DPA
HOW TO PREVENT SUCH ATTACKS?

WE HAVE TO INTRODUCE A SECOND CONCEPT:

Definition 2. *If the Walsh transform* $W_f$ *of a Boolean function* $f(x_1, ..., x_n)$ *satisfies* $W_f(\omega) = 0$, *for* $0 \leq \text{HW}(\omega) \leq m$, *it is called a balanced m-th order correlation immune (CI) function or an m-resilient function, where* $\text{HW}$ *stands for Hamming weight.*

CAN WE AVOID ATTACKS BY USING 1ST-ORDER CORRELATION IMMUNE FUNCTIONS?

- all $f_i$ will be *m*-th order correlation immune ($m \geq 1$) for the correct key guess
- not necessary the case for a wrong key guess
- then, simply compute:

\[
\Delta_k = x \times 10^4
\]

![Graph](image)
CONCLUSION AND FUTURE WORK

THE END OF THE STORY:

1. We presented the first AES white-box implementation realized in hardware.
2. Provided results of a practical grey-box (side-channel) analysis and revealed side channels.
3. Investigated underlying mathematical reasons for discovered vulnerabilities.

WHAT HAS TO BE DONE IN FUTURE WORK?

1. Further investigations for linear/non-linear encodings. Specify requirements to prevent analysis through imbalances in Walsh transformations.
2. Enhance white-box security by countermeasures to prevent grey-box attacks, e.g. using dynamically updated encodings.
Thank you for your attention!
Any Questions?