Differential Computation Analysis
Hiding your White-Box Designs is Not Enough

Joppe W. Bos
Microsoft Research Visit, August 24, 2016
Redmond, WA, USA
NXP Semiconductors

Operations in > 35 countries, more than 130 facilities
≈ 45,000 employees

Research & Development
≈ 11,200 engineers in 23 countries
The presence of an attacker

- Where should we assume the attacker to be? What is most realistic?
  - Is the attacker only eavesdropping on the communication channel?
  - Or did one of the (trusted/authorized) end-users become the attacker?
  - Or are there any malware/viruses installed on a trusted end-user’s device?
In order to properly assess the security of (the implementation of) a cryptographic algorithm, one needs a clear definition of a security notion.

Security Notion = attacker’s goal + attacker model.

- **Attacker’s goal**: what does the attacker want to achieve?
  - This is not always key-extraction, the attacker is often satisfied with much less…

- **Attacker model**: what are the capabilities of the attacker in order for him to achieve his goal?
  - Such a model tries to capture the capabilities of an attacker as realistically as possible, i.e., modeling the hostile environment in which the implementation of a cryptographic primitive is deployed.
Initial cryptographic security model from the 1980s

- Endpoints are trusted parties
- Attacker “observes” data being transferred
When technology changed this model did not reflect reality any longer

Cryptographic algorithms implemented in hardware were originally thought to form a secure environment

In 1999 it was publicly shown that hardware implementations tend to leak key-correlated information

Kocher, Jaffe, Jun. Differential power analysis. In CRYPTO 1999
Grey box model

The research area of side-channel attacks and resistance has grown significantly: fault injections, simple power analysis, differential power analysis, correlation power analysis, template attacks, higher-order correlation attacks, mutual information analysis, linear regression analysis, horizontal analysis, vertical analysis etc. etc.
Grey box model → white box model

- When technology changed this model did not reflect reality any longer
- Increase in mobile devices without dedicated hardware support → need to rely on software solutions
- In 2002 the white-box model was introduced
  Initial focus on DRM applications.

White box model

Adversary owns the device running the software. Powerful capabilities

- has full access to the source code
- perform static analysis
- inspect and alter the memory used
- alter intermediate results
Where is this used in practice?

Original use-case for white-box crypto is *digital right management*.

For example: streaming content, protecting DVD’s etc
Where is this used in practice?
Original use-case for white-box crypto is digital right management.
For example: streaming content, protecting DVD’s etc

How Host Card Emulation Works

Recent trend
Use Host Card Emulation (HCE) to communicate using Near Field Communication (NFC)
→ Replace the secure element with software.

Protection of the cryptographic key? How?  
White-box implementation!

Source: Business Insider
Huge demand for practical + secure white-box

• 2014: VISA + Mastercard support HCE

• [Berg Insight]: 86% of the Point of Sale devices in North America and 78% in Europe will support NFC by 2017.

• [IHS research]: By 2018, 2/3 of all shipped phones will support NFC.

• → the protocols used need to use (and store!) AES / DES keys → need for secure **white-box cryptography**.
White-Box basic idea – Why?

- **Entropy attack**
  - Locate the unusual high entropy of the cryptographic key in a memory dump using sliding windows for example.

Shamir, van Someren: *Playing "Hide and Seek" with Stored Keys*. Financial Cryptography 1999
White-Box basic idea – Why?

- **Entropy attack**
  - Locate the unusual high entropy of the cryptographic key in a memory dump using sliding windows for example.

- **S-box blanking attack**
  - Locate the publicly defined S-boxes in the binary and overwrite it with all zeros such that $S(x)=0$ for any $x$.

Shamir, van Someren: *Playing "Hide and Seek" with Stored Keys*. Financial Cryptography 1999

Kerins, Kursawe: *A cautionary note on weak implementations of block ciphers*. WISSec, 2006
Security of WB solutions - Theory

White box can be seen as a form of code obfuscation

- It is known that obfuscation of any program is impossible

Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, Yang. On the (im)possibility of obfuscating programs. In CRYPTO 2001

- Unknown if a (sub)family of white-box functions can be obfuscated
- If secure WB solution exists then this is protected (by definition!) to all current and future side-channel and fault attacks!
Security of WB solutions - Theory

White box can be seen as a form of code obfuscation
• It is known that obfuscation of any program is impossible

Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, Yang. On the (im)possibility of obfuscating programs. In CRYPTO 2001
• Unknown if a (sub)family of white-box functions can be obfuscated
• If secure WB solution exists then this is protected (by definition!) to all current and future side-channel and fault attacks!

Practice
• Only results known for symmetric crypto (all academic designs broken)
• Convert algorithms to sequence of LUTs
• Embed the secret key in the LUTs
• Obfuscate the LUTs by using encodings
No! “Ideal” WB AES implementation
One big lookup table $\Rightarrow 2^{92}$ TB storage required

Practical WB AES?
Network of smaller tables: $\approx 700$ kB
Encoding on intermediate values using ideas by Chow


Generic idea.
Transform a cipher into a network of randomized key-instantiated look-up tables
AES with look-up tables: example, Chow

- The key addition and S-box operations are merged into a single operation
  (8 bit → 8 bit table → 256 byte)  
  \[ b_{i,j} = Sbox(a_{i,j} \oplus k_{i,j}) = T_{i,j}(a_{i,j}) \]

- To simplify: we omit ShiftRow operation
  - Corresponds to renumbering of indices

- The MixColumn operation can be split into four byte-to-32-bit (8 bit → 32 bit table → 1024 byte) operations:
  \[ c_{j} = M_{0}T_{0,j}(a_{0,j}) \oplus M_{1}T_{1,j}(a_{1,j}) \oplus M_{2}T_{2,j}(a_{2,j}) \oplus M_{3}T_{3,j}(a_{3,j}) \]

- We can now implement a round by only using the following 2 types of lookup tables:

\[ \begin{array}{c}
8 & T_{i,j} & 8 & 32 & \oplus & 4 \\
\end{array} \]
AES (Chow) with look-up tables + obfuscation

• Since S-boxes and matrix $M$ are known, the key can easily be extracted from the lookup tables.

• **Solution**: obfuscating lookup tables by encoding their input and output.
AES (Chow) with look-up tables + obfuscation

- Since S-boxes and matrix $M$ are known, the key can easily be extracted from the lookup tables.

- **Solution**: obfuscating lookup tables by encoding their input and output.

- First, we apply **linear** encodings:
  - $A_i$: random 8-bit linear mapping
  - $MB$: random 32-bit linear mapping

$$A_i^{-1} \cdot a_{i,j} \rightarrow \begin{array}{c}
8 \rightarrow \begin{array}{c}
A_i \rightarrow \begin{array}{c}
8 \rightarrow \begin{array}{c}
T_{i,j} \rightarrow \begin{array}{c}
8 \rightarrow \begin{array}{c}
MB \cdot M_i \rightarrow \begin{array}{c}
32 \rightarrow \begin{array}{c}
8 \rightarrow \begin{array}{c}
\oplus \rightarrow \begin{array}{c}
4 \rightarrow \begin{array}{c}
MB \cdot c_j
\end{array}\end{array}\end{array}\end{array}\end{array}\end{array}\end{array}\end{array}\end{array}\end{array}\end{array}\end{array}$$
AES (Chow) with look-up tables + obfuscation

- Since S-boxes and matrix $M$ are known, the key can easily be extracted from the lookup tables.

- **Solution**: obfuscating lookup tables by encoding their input and output.
- First, we apply *linear* encodings:
  - $A_i$: random 8-bit linear mapping
  - $MB$: random 32-bit linear mapping

- Matrix $MB$ is removed from the computed output columns. Implemented in the same way as the MixColumn operations

\[
MB^{-1}(x) = MB_0^{-1}(x_0) \oplus MB_1^{-1}(x_1) \oplus MB_2^{-1}(x_2) \oplus MB_3^{-1}(x_3)
\]

- Merge the $MB_i$-tables by the linear encodings used in the next round.
Obfuscation, obfuscation, obfuscation

- In addition to the linear encodings, also add non-linear encodings $f$.

\[(f_{0,i}, f_{1,i})A_i^{-1} \cdot a_{i,j} \rightarrow A_i \rightarrow T_{i,j} \rightarrow MB \cdot M_i \rightarrow \bigoplus \rightarrow (f_{0,i}, f_{1,i})A_i^{-1} \cdot c_{i,j}\]

Size of implementation: $\approx 700$ kB

In practice the white box is the most essential but a **small part** of the entire software implementation

- Strong code obfuscation
- Binary is “glued” to the environment
  - Prevent code-lifting
- Support for traitor tracing
- Mechanism for frequent updating

More details see the invited talk at EC 2016 *Engineering Code Obfuscation* by Christian Collberg
Previous effort

Previous WB attacks were WB specific which means knowing:

- the *encodings*
- which *cipher operations* are implemented by
- which (network of) *lookup tables*

**Attack**

1. time-consuming reverse-engineering of the code
2. identify which WB scheme is used + target the correct LUTs
3. apply an algebraic attack
Effort and expertise required

**Previous effort**
Previous WB attacks were **WB specific** which means knowing
- the *encodings*
- which *cipher operations* are implemented by
- which (network of) *lookup tables*

**Attack**
1. time-consuming *reverse-engineering* of the code
2. identify which WB scheme is used + target the correct LUTs
3. apply an algebraic attack

**Our approach**
Assess the security of a WB implementation
- Automatically and very simply (see CHES challenge)
- Without knowledge of any implementation choices
  → only the algorithm itself
- Ignores all (attempts) at code-obfuscation
SOFTWARE TRACES
Tracing binaries

- Academic attacks are on open design
- In practice: what you get is a binary blob

Idea: create software traces using *dynamic binary instrumentation* tools
(→ visual representation → use traces to find correlation)

- Record all instructions and memory accesses.

Examples of the tools we extended / modified
- Intel PIN (x86, x86-64, Linux, Windows, Wine/Linux)
- Valgrind (idem+ARM, Android)
Trace visualization

Based on Ptra, an unreleased Quarkslab tool presented at SSTIC 2014
Visual crypto identification: code
Visual crypto identification: code?
Visual crypto identification: code? data!
Visual crypto identification: code? data?
Visual crypto identification: stack!
Differential Power Analysis and friends

Very powerful grey box attack!

Requirements
- known input or known output
- ability to trace power consumption (or EM radiations, or …)


For example in AES: $SubBytes(p \oplus \kappa)$

Key Expansion → AddRoundKey → MixColumns → ShiftRows → SubBytes → R?
Differential Computation Analysis

Port the white-box to a smartcard and measure power consumption
Differential Computation Analysis

Port the white-box to a smartcard and measure power consumption
Make pseudo power traces from our software execution traces
→ this are lists of memory accesses / data + stack writes / …

E.g. build a trace of all 8-bit data reads:

→ 256 possible discrete values
Differential Computation Analysis

256 possible discrete values but bit values dominated by the MSB
→ Build Hamming weight traces?

→ 8 possible discrete values

That works but we can do better…

recall: Hamming weight was a hardware model for combined bit leaks
Differential Computation Analysis

Each bit of those bytes is equally important address bits represent a different way to partition the look-up tables

→ Serialize bytes in a succession of bits

→ 2 possible discrete values: 0's and 1's
DCA: DPA on software traces

HW analogue: this is like probing each bus-line individually \textit{without any error}
## Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let’s check!

<table>
<thead>
<tr>
<th>WB implementation</th>
<th>Algorithm</th>
<th>#traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wyseur challenge, 2007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hack.lu challenge, 2009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSTIC challenge, 2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Klinec implementation, 2013</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chow+: Chow-based plus personal improvements by Brecht Wyseur


E. Link and W. D. Neumann. Clarifying obfuscation: Improving the security of white-box DES. In International Symposium on Information Technology: Coding and Computing (ITCC 2005)
WB implementations should not leak any side-channel information (by definition of the WB attack model): let’s check!

<table>
<thead>
<tr>
<th>WB implementation</th>
<th>Algorithm</th>
<th>#traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wyseur challenge, 2007</td>
<td>DES (Chow+)</td>
<td>65</td>
</tr>
<tr>
<td>Hack.lu challenge, 2009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSTIC challenge, 2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Klinec implementation, 2013</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hack.lu challenge

Zoom on the stack

✓ AES-128
✓ Very easy to break
   (designed for a one-day challenge)
Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let’s check!

<table>
<thead>
<tr>
<th>WB implementation</th>
<th>Algorithm</th>
<th>#traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wyseur challenge, 2007</td>
<td>DES (Chow+)</td>
<td>65</td>
</tr>
<tr>
<td>Hack.lu challenge, 2009</td>
<td>AES (Chow)</td>
<td>16 (no encodings)</td>
</tr>
<tr>
<td>SSTIC challenge, 2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Klinec implementation, 2013</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let’s check!

<table>
<thead>
<tr>
<th>WB implementation</th>
<th>Algorithm</th>
<th>#traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wyseur challenge, 2007</td>
<td>DES (Chow+)</td>
<td>65</td>
</tr>
<tr>
<td>Hack.lu challenge, 2009</td>
<td>AES (Chow)</td>
<td>16 (no encodings)</td>
</tr>
<tr>
<td>SSTIC challenge, 2012</td>
<td>DES</td>
<td>16 (no encodings)</td>
</tr>
<tr>
<td>Klinec implementation, 2013</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• AES using Karroumi’s approach (using dual ciphers)
• More difficult, not all correct key bytes are #1
Klinec

- Balanced encodings?
  - It may become the least candidate, this is still standing out!

<table>
<thead>
<tr>
<th>target hit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>256</td>
<td>255</td>
<td>256</td>
<td>255</td>
<td>256</td>
<td>256</td>
<td>253</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>239</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>226</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>222</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>255</td>
<td>251</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>254</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>253</td>
<td>254</td>
<td>254</td>
<td>254</td>
<td>255</td>
</tr>
<tr>
<td>4</td>
<td>256</td>
<td>256</td>
<td>74</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>255</td>
<td>256</td>
<td>254</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>50</td>
<td>256</td>
<td>253</td>
<td>1</td>
<td>251</td>
<td>256</td>
<td>253</td>
<td>1</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>6</td>
<td>254</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>254</td>
<td>256</td>
<td>248</td>
<td>256</td>
<td>252</td>
<td>256</td>
<td>1</td>
<td>14</td>
<td>255</td>
<td>256</td>
<td>250</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>252</td>
<td>256</td>
<td>253</td>
<td>256</td>
<td>256</td>
<td>255</td>
<td>256</td>
<td>1</td>
<td>251</td>
<td>1</td>
<td>1</td>
<td>254</td>
</tr>
</tbody>
</table>

Correct key, ranked #256
### Table 1. DCA ranking for a Karroumi white-box implementation when targeting the output of the *SubBytes* step in the first round based on the least significant address byte on memory reads.

<table>
<thead>
<tr>
<th>target bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>256</td>
<td>1</td>
<td>56</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>253</td>
<td>256</td>
<td>256</td>
<td>239</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>255</td>
</tr>
<tr>
<td>1</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>255</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>226</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>22</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>50</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td>6</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>254</td>
<td>256</td>
<td>254</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>14</td>
<td>255</td>
<td>256</td>
<td>250</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>252</td>
<td>256</td>
<td>253</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>251</td>
<td>1</td>
<td>254</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. DCA ranking for a Karroumi white-box implementation when targeting the output of the multiplicative inversion inside the *SubBytes* step in the first round based on the least significant address byte on memory reads.

<table>
<thead>
<tr>
<th>target bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>253</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>249</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>226</td>
<td>1</td>
<td>254</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>254</td>
<td>256</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>254</td>
<td>1</td>
<td>69</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>252</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>254</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>238</td>
<td>256</td>
<td>253</td>
<td>256</td>
</tr>
<tr>
<td>5</td>
<td>254</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>241</td>
<td>256</td>
<td>255</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>231</td>
<td>256</td>
<td>208</td>
<td>254</td>
</tr>
<tr>
<td>6</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>1</td>
<td>241</td>
</tr>
<tr>
<td>7</td>
<td>63</td>
<td>256</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>255</td>
<td>231</td>
<td>256</td>
<td>255</td>
<td>1</td>
<td>255</td>
<td>256</td>
<td>255</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

48.
Results

WB implementations should not leak any side-channel information (by definition of the WB attack model): let’s check!

<table>
<thead>
<tr>
<th>WB implementation</th>
<th>Algorithm</th>
<th>#traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wyseur challenge, 2007</td>
<td>DES (Chow+)</td>
<td>65</td>
</tr>
<tr>
<td>Hack.lu challenge, 2009</td>
<td>AES (Chow)</td>
<td>16 (no encodings)</td>
</tr>
<tr>
<td>SSTIC challenge, 2012</td>
<td>DES</td>
<td>16 (no encodings)</td>
</tr>
<tr>
<td>Klinec implementation, 2013</td>
<td>AES (Karroumi, dual ciphers)</td>
<td>2000 → 500</td>
</tr>
</tbody>
</table>
A lot of potential for follow-up work!

Use the extended research results from the grey box world

**Countermeasures**
- Use random masks / delays → white-box model allows to disable entropy source
- Use static random data within the white-box itself?
- Use ideas from threshold implementation? [TI]
- Better DBI framework detection mechanisms
- DCA might fail when using large encodings → larger LUTs → algebraic attacks still work
  

**Other attacks**
Riscure has proven software fault attacks (DFA) work too [RISCURE].

Once there are countermeasures against DCA and DFA, can we use any of the other known advanced SCA in this setting?

Any help to complete our collection of open whitebox challenges and attacks or to improve our tools is highly appreciated!

https://github.com/SideChannelMarvels
Conclusions

• Software-only solutions are becoming more popular
  • white-box crypto

• Traditional (DRM) and new use-cases HCE (payment, transit, …)

• Level of security / maturity of many (all?) WB schemes is questionable
  • Open problem to construct asymmetric WB crypto
  • Industry keeps design secret

• DCA is an automated attack which can be carried out without any expertise
  • Counterpart of the DPA from the crypto HW community

• This hopefully sparks more interest in both cryptographic and cryptanalytic white-box research!
References

