Dummy Shuffling against Algebraic Attacks in White-box Implementations

Alex Biryukov[1], Aleksei Udovenko[2,1]

[1]DCS and SnT, University of Luxembourg

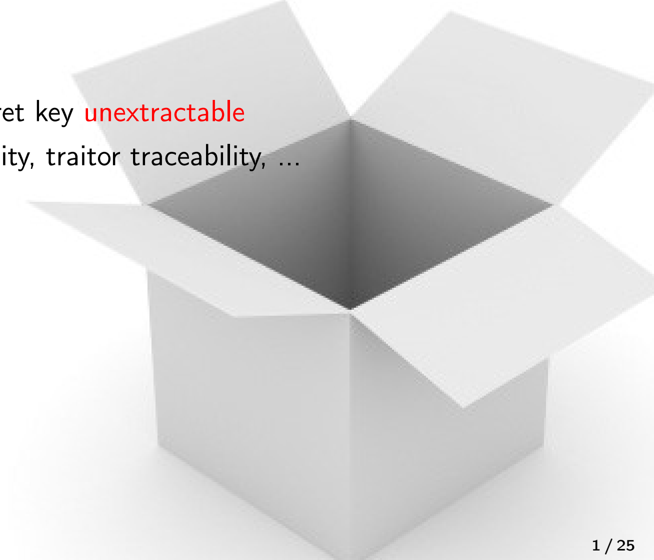
[2]CryptoExperts

EUROCRYPT 2021, October 11, 2021

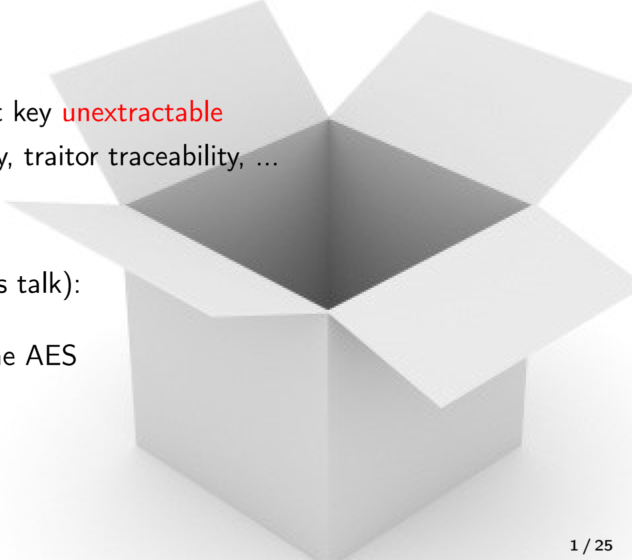
Plan

White-box implementations

- Implementation fully available, secret key unextractable
- **Extra**: one-wayness, incompressibility, traitor traceability, ...



White-box implementations



- Implementation fully available, secret key unextractable
- **Extra**: one-wayness, incompressibility, traitor traceability, ...

- The most **challenging** direction (this talk): white-box implementations of existing symmetric primitives, e.g. the AES
- “Cryptographic obfuscation”

White-box: industry vs academia

White-box: industry vs academia

- many applications
- strong need for *practical* white-box
- industry **does** WB: secret designs

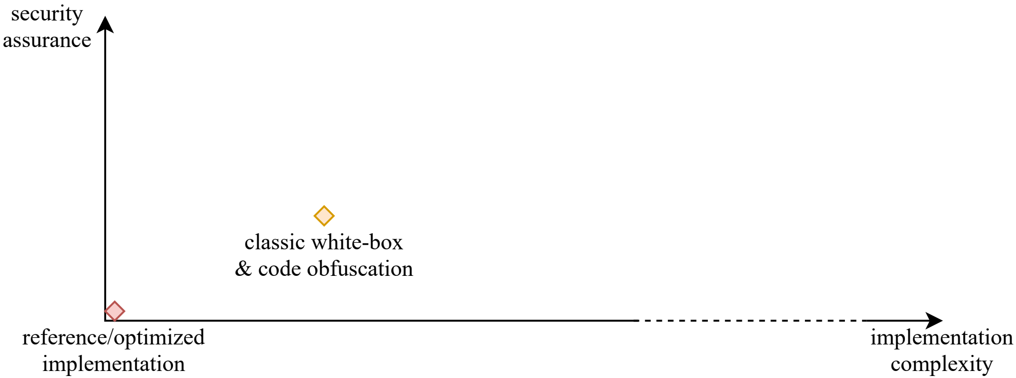
White-box: industry vs academia

- many applications
- strong need for *practical* white-box
- industry **does** WB: secret designs

- **theory**: approaches using iO, currently *impractical*
- **practical WB-AES/DES**: few attempts (2002-2017, [CEJv03], ...), all broken
- powerful **grey-box** attacks [BHMT16](CHES 2016)

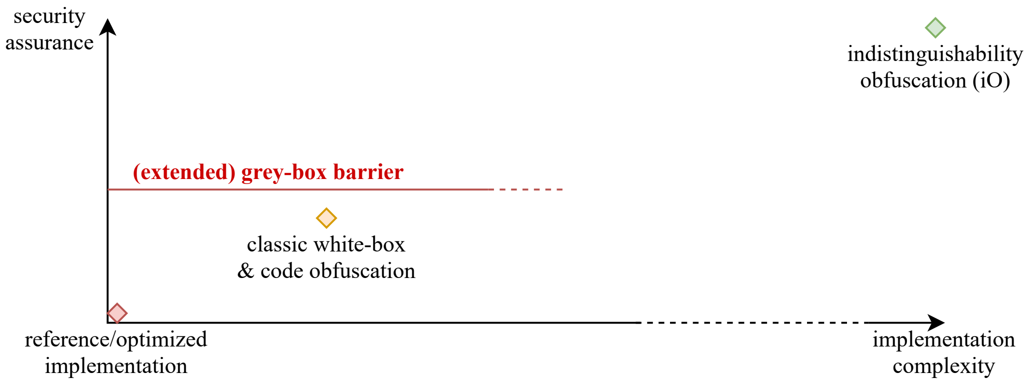
Shades of obfuscation

Shades of obfuscation

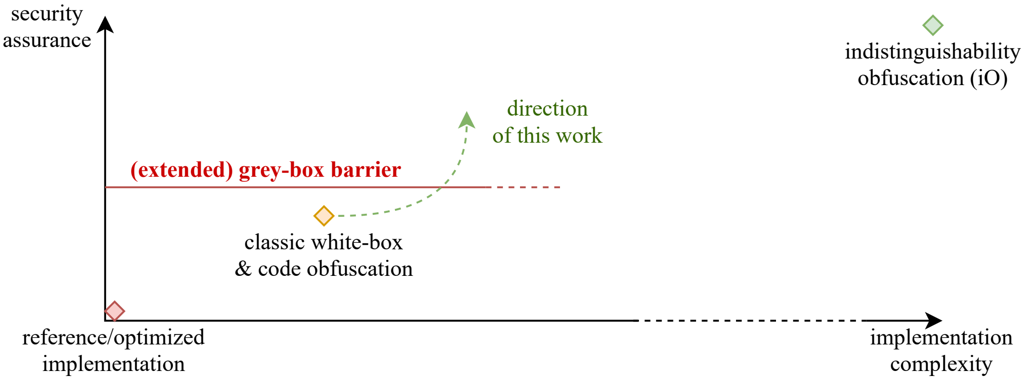


Shades of obfuscation

Shades of obfuscation

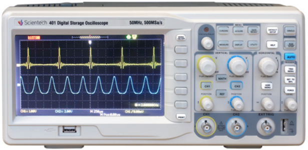


Shades of obfuscation



Plan

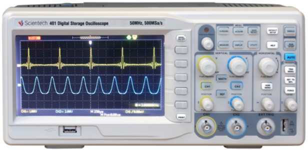
White-box: Differential Computation Analysis (DCA)



- **DCA = Differential Power Analysis (DPA)** applied to white-box implementations [BHMT16][1]
- Most of the existing implementations broken automatically

[1] (CHES 2016) Bos, Hubain, Michiels, Teuwen. Differential computation analysis: Hiding your white-box designs is not enough.

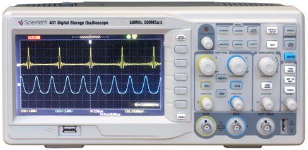
White-box: Differential Computation Analysis (DCA)



- **DCA = Differential Power Analysis (DPA)** applied to white-box implementations [BHMT16][1]
- Most of the existing implementations broken automatically
- Side-channel protections: **masking schemes**, **shuffling**

[1](CHES 2016) Bos, Hubain, Michiels, Teuwen. Differential computation analysis: Hiding your white-box designs is not enough.

White-box: Differential Computation Analysis (DCA)



- **DCA = Differential Power Analysis (DPA)** applied to white-box implementations [BHMT16][1]
- Most of the existing implementations broken automatically
- Side-channel protections: **masking schemes**, **shuffling**

Can we apply these countermeasures to white-box implementations?

[1](CHES 2016) Bos, Hubain, Michiels, Teuwen. Differential computation analysis: Hiding your white-box designs is not enough.

Weakness of linear masking

- assume a sensitive function s being protected
- **linear masking**: $\exists v_1, \ldots, v_t$ *shares*,

$$v_1 \oplus \ldots \oplus v_t = s$$

Weakness of linear masking

- assume a sensitive function s being protected
- **linear masking**: $\exists v_1, \ldots, v_t$ *shares*,

$$v_1 \oplus \ldots \oplus v_t = s$$

Problem: there exists a linear combination of computed functions that equals the sensitive function!

Linear algebraic attack - illustration

$$\begin{pmatrix} 010000111000 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}$$

Linear algebraic attack - illustration

$$\begin{pmatrix} 010000111000 \\ 000100011101 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}$$

Linear algebraic attack - illustration

$$\begin{pmatrix} 010000111000 \\ 000100011101 \\ 000101011110 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}$$

Linear algebraic attack - illustration

$$s = \mathrm{Sbox}_i(pt_i \oplus k_i)$$

$$\begin{pmatrix} 010000111000 \\ 000100011101 \\ 000101011110 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} \qquad \begin{pmatrix} 1 \\ 0 \\ 1 \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}$$

Linear algebraic attack - illustration

$$
\begin{array}{cc}
 & \leftarrow \text{trace length} \rightarrow \\
\begin{array}{c}\uparrow \\ \#\ \text{traces} \\ \downarrow\end{array} &
\begin{pmatrix}
010000111000 \\
000100011101 \\
000101011110 \\
101010111011 \\
001011100011 \\
011011011100 \\
000101100111 \\
001010001010 \\
110110101101 \\
111101100110 \\
010111111010 \\
111001110110 \\
101000000101 \\
010011100000 \\
011011000100 \\
100101010010
\end{pmatrix}
\end{array}
\qquad
\begin{array}{c}
{\color{red}s} = \mathrm{Sbox}_i(pt_i \oplus k_i) \\
\begin{pmatrix}
1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1
\end{pmatrix}
\end{array}
$$

Linear algebraic attack - illustration

$$
\begin{array}{cc}
 & \leftarrow \text{trace length} \rightarrow \\
\uparrow \\ \# \text{ traces} \\ \downarrow &
\begin{pmatrix}
010000111000 \\
000100011101 \\
000101011110 \\
101010111011 \\
001011100011 \\
011011011100 \\
000101100111 \\
001010001010 \\
110110101101 \\
111101100110 \\
010111111010 \\
111001110110 \\
101000000101 \\
010011100000 \\
011011000100 \\
100101010010
\end{pmatrix}
\end{array}
\times
\begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}
=
\begin{array}{c}
{\color{red} s} = \mathrm{Sbox}_i(pt_i \oplus k_i) \\
\begin{pmatrix}
1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1
\end{pmatrix}
\end{array}
$$

Linear algebraic attack - illustration

$$s = \mathrm{Sbox}_i(pt_i \oplus k_i) = v_1 \oplus v_2 \oplus v_3$$

$$\begin{array}{c} \qquad v_1 \qquad v_2 \;\; v_3 \\ \begin{pmatrix} 010000111000 \\ 000100011101 \\ 000101011110 \\ 101010111011 \\ 001011100011 \\ 011011011100 \\ 000101100111 \\ 001010001010 \\ 110110101101 \\ 111101100110 \\ 010111111010 \\ 111001110110 \\ 101000000101 \\ 010011100000 \\ 011011000100 \\ 100101010010 \end{pmatrix} \end{array} \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Generalization I: higher-degree attack vs nonlinear masking (1/2)

$$\color{red}{s}\color{black} = \mathrm{Sbox}_i(pt_i \oplus k_i)$$

$$\begin{pmatrix}
010000111000 \mid & \qquad\qquad \\
000100011101 \mid & \\
000101011110 \mid & \\
101010111011 \mid & \\
001011100011 \mid & \\
011011011100 \mid & \\
000101100111 \mid & \\
001010001010 \mid & \\
110110101101 \mid & \\
111101100110 \mid & \\
010111111010 \mid & \\
111001110110 \mid & \\
101000000101 \mid & \\
010011100000 \mid & \\
011011000100 \mid & \\
100101010010 \mid &
\end{pmatrix}
\qquad\qquad
\begin{pmatrix}
0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1
\end{pmatrix}$$

Generalization I: higher-degree attack vs nonlinear masking (1/2)

$\otimes$

$$\begin{pmatrix}
010000111000 & | & 0 \\
000100011101 & | & 0 \\
000101011110 & | & 0 \\
101010111011 & | & 0 \\
001011100011 & | & 0 \\
011011011100 & | & 0 \\
000101100111 & | & 0 \\
001010001010 & | & 0 \\
110110101101 & | & 1 \\
111101100110 & | & 1 \\
010111111010 & | & 0 \\
111001110110 & | & 1 \\
101000000101 & | & 0 \\
010011100000 & | & 0 \\
011011000100 & | & 0 \\
100101010010 & | & 0
\end{pmatrix}
\qquad
\color{red}{s} = \mathrm{Sbox}_i(pt_i \oplus k_i)
\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Generalization I: higher-degree attack vs nonlinear masking (1/2)

$$\otimes$$

$$\textcolor{red}{s} = \mathrm{Sbox}_i(pt_i \oplus k_i)$$

$$\begin{pmatrix}
010000111000 \mid 00 \\
000100011101 \mid 00 \\
000101011110 \mid 00 \\
101010111011 \mid 01 \\
001011100011 \mid 00 \\
011011011100 \mid 00 \\
000101100111 \mid 00 \\
001010001010 \mid 00 \\
110110101101 \mid 10 \\
111101100110 \mid 11 \\
010111111010 \mid 00 \\
111001110110 \mid 11 \\
101000000101 \mid 01 \\
010011100000 \mid 00 \\
011011000100 \mid 00 \\
100101010010 \mid 00
\end{pmatrix}
\qquad
\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Generalization I: higher-degree attack vs nonlinear masking (1/2)

$$\begin{pmatrix} 010000111000 & 000\ldots0 \\ 000100011101 & 000\ldots0 \\ 000101011110 & 000\ldots0 \\ 101010111011 & 010\ldots1 \\ 001011100011 & 000\ldots1 \\ 011011011100 & 000\ldots0 \\ 000101100111 & 000\ldots1 \\ 001010001010 & 000\ldots0 \\ 110110101101 & 101\ldots0 \\ 111101100110 & 111\ldots0 \\ 010111111010 & 000\ldots0 \\ 111001110110 & 110\ldots0 \\ 101000000101 & 010\ldots0 \\ 010011100000 & 000\ldots0 \\ 011011000100 & 000\ldots0 \\ 100101010010 & 001\ldots0 \end{pmatrix} \qquad \begin{array}{c} \color{red}{s} = \mathrm{Sbox}_i(pt_i \oplus k_i) \\ \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{array}$$

Generalization I: higher-degree attack vs nonlinear masking (1/2)

$$s = \mathrm{Sbox}_i(pt_i \oplus k_i) = v_1 v_2 \oplus v_3$$

$$\begin{array}{c} \otimes \qquad v_3 \quad v_1 v_2 \\ \begin{pmatrix} 010000111000 & | & 000\ldots0 \\ 000100011101 & | & 000\ldots0 \\ 000101011110 & | & 000\ldots0 \\ 101010111011 & | & 010\ldots1 \\ 001011100011 & | & 000\ldots1 \\ 011011011100 & | & 000\ldots0 \\ 000101100111 & | & 000\ldots1 \\ 001010001010 & | & 000\ldots0 \\ 110110101101 & | & 101\ldots0 \\ 111101100110 & | & 111\ldots0 \\ 010111111010 & | & 000\ldots0 \\ 111001110110 & | & 110\ldots0 \\ 101000000101 & | & 010\ldots0 \\ 010011100000 & | & 000\ldots0 \\ 011011000100 & | & 000\ldots0 \\ 100101010010 & | & 001\ldots0 \end{pmatrix} \end{array} \times \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \underline{0} \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Generalization I: higher-degree attack vs nonlinear masking (2/2)

Takeaway:

- higher-degree schemes can be attacked, but at a higher cost

Generalization II: noisy linear attack (1/2)

$$\begin{pmatrix} 010111111010 \\ 001010001010 \\ 011011000100 \\ 110110101101 \\ 101010111011 \\ 100101010010 \\ 001011100011 \\ 010011100000 \\ 000100011101 \\ 101000000101 \\ 010000111000 \\ 011011011100 \\ 000101100111 \\ 000101011110 \\ 111101100110 \\ 111001110110 \end{pmatrix} \times \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}}^{s = \mathrm{Sbox}_i(pt_i \oplus k_i)} \oplus \overbrace{\begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix}}^{e \text{ (low weight)}}$$

Generalization II: noisy linear attack (1/2)

$s = \mathrm{Sbox}_i(pt_i \oplus k_i) \qquad e$ (low weight)

$$\begin{pmatrix} 111101100110 \\ 111001110110 \\ 110110101101 \\ 010000111000 \\ 000101100111 \\ 010111111010 \\ 011011011100 \\ 100101010010 \\ 101000000101 \\ 001011100011 \\ 010011100000 \\ 101010111011 \\ 000100011101 \\ 011011000100 \\ 001010001010 \\ 000101011110 \end{pmatrix} \times \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Generalization II: noisy linear attack (1/2)

$$\begin{pmatrix} 011011011100 \\ 101010111011 \\ 001011100011 \\ 001010001010 \\ 000101011110 \\ 010011100000 \\ 000101100111 \\ 100101010010 \\ 101000000101 \\ 011011000100 \\ 111101100110 \\ 010111111010 \\ 110110101101 \\ 010000111000 \\ 111001110110 \\ 000100011101 \end{pmatrix} \times \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}}_{s = \mathrm{Sbox}_i(pt_i \oplus k_i)} \oplus \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{e \text{ (low weight)}}$$

Generalization II: noisy linear attack (1/2)

$s = \mathrm{Sbox}_i(pt_i \oplus k_i) \quad e$ (low weight)

$$\begin{pmatrix} 010000111000 \\ 000100011101 \\ 000101011110 \\ 101010111011 \\ 001011100011 \\ 011011011100 \\ 000101100111 \\ 001010001010 \\ 110110101101 \\ 111101100110 \\ 010111111010 \\ 111001110110 \\ 101000000101 \\ 010011100000 \\ 011011000100 \\ 100101010010 \end{pmatrix} \times \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Generalization II: noisy linear attack (1/2)

$$\begin{pmatrix} 010000111000 \\ 000100011101 \\ 000101011110 \\ 101010111011 \\ 001011100011 \\ 011011011100 \\ 000101100111 \\ 001010001010 \\ 110110101101 \\ 111101100110 \\ 010111111010 \\ 111001110110 \\ 101000000101 \\ 010011100000 \\ 011011000100 \\ 100101010010 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}}_{s = \mathrm{Sbox}_i(pt_i \oplus k_i)} \oplus \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{e \text{ (low weight)}}$$

Generalization II: noisy linear attack (2/2)

- General attack framework: *Learning Parity with Noise* (LPN)

Takeaway:

- good low-degree approximations are sufficient for an attack
- need to ensure sufficient weight of the error term e

Generalization III: input restriction (1/3)

- assume $f = s \cdot r$ is computed/shared in the circuit, r is pseudorandom
- linear algebra attack would fail

Generalization III: input restriction (1/3)

- assume $f = s \cdot r$ is computed/shared in the circuit, r is pseudorandom
- linear algebra attack would fail
- observation: $s = 0 \quad \Rightarrow \quad f = 0$

Generalization III: input restriction (2/3)

$$s = \mathrm{Sbox}_i(pt_i \oplus k_i)$$

$$\begin{pmatrix} 010000111000 \\ 000100011101 \\ 000101011110 \\ 101010111011 \\ 001011100011 \\ 011011011100 \\ 000101100111 \\ 001010001010 \\ 110110101101 \\ 111101100110 \\ 010111111010 \\ 111001110110 \\ 101000000101 \\ 010011100000 \\ 011011000100 \\ 100101010010 \end{pmatrix} \times \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Generalization III: input restriction (2/3)

$$\textcolor{red}{s} = \text{Sbox}_i(pt_i \oplus k_i)$$

$$\begin{pmatrix} 000\textcolor{blue}{1}000\textcolor{blue}{1}11\textcolor{blue}{0}1 \\ 101\textcolor{blue}{0}101\textcolor{blue}{1}10\textcolor{blue}{1}1 \\ 000\textcolor{blue}{1}011\textcolor{blue}{0}01\textcolor{blue}{1}1 \\ 111\textcolor{blue}{1}011\textcolor{blue}{0}01\textcolor{blue}{1}0 \\ 111\textcolor{blue}{0}011\textcolor{blue}{1}01\textcolor{blue}{1}0 \\ 101\textcolor{blue}{0}000\textcolor{blue}{0}01\textcolor{blue}{0}1 \\ 010\textcolor{blue}{0}111\textcolor{blue}{0}00\textcolor{blue}{0}0 \\ 011\textcolor{blue}{0}110\textcolor{blue}{0}01\textcolor{blue}{0}0 \end{pmatrix} \times \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} \textcolor{red}{0} \\ \textcolor{red}{0} \\ \textcolor{red}{0} \\ \textcolor{red}{0} \\ \textcolor{red}{0} \\ \textcolor{red}{0} \\ \textcolor{red}{0} \\ \textcolor{red}{0} \end{pmatrix}$$

Generalization III: input restriction (2/3)

$$s \cdot r \;= s = \mathrm{Sbox}_i(pt_i \oplus k_i) = (v_1 \oplus v_2 \oplus v_3) \cdot r$$

$$\begin{pmatrix} 000100011101 \\ 101010111011 \\ \\ 000101100111 \\ \\ 111101100110 \\ \\ 111001110110 \\ 101000000101 \\ 010011100000 \\ 011011000100 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \\ 0 \\ \\ 0 \\ \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Generalization III: input restriction (3/3)

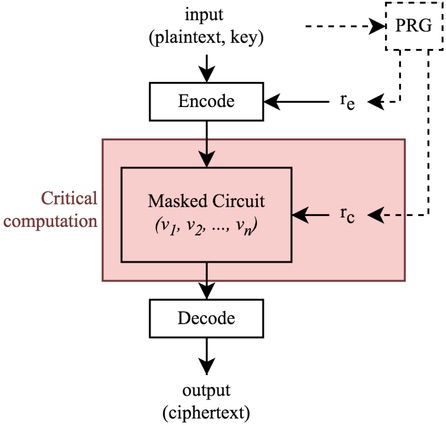
Takeaway:

- protection should not depend critically on “predictable” values such as s

Plan

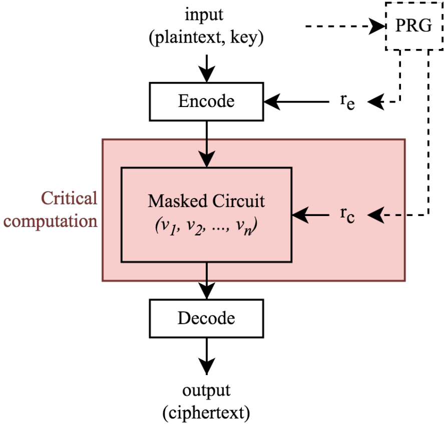
Algebraic security (BU-model [BU18], ASIACRYPT 2018)

1. random bits allowed
   - as in classic masking
   - model unpredictability
   - in WB impl. as **pseudorandom**



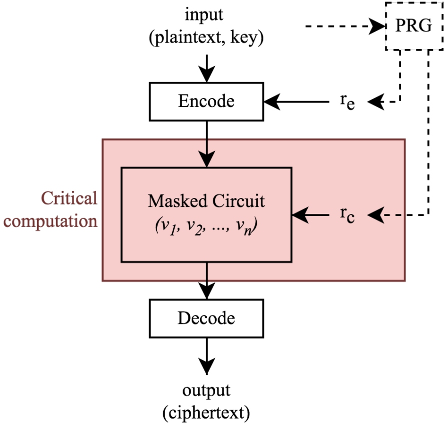
Algebraic security (BU-model [BU18], ASIACRYPT 2018)

1. random bits allowed
   - as in classic masking
   - model unpredictability
   - in WB impl. as **pseudorandom**
2. **security requirement:**
   any non-constant $f \in \mathrm{span}\{v_i\}_i$ must have sufficient error/noise in any fixed input



Algebraic security (BU-model [BU18], ASIACRYPT 2018)

1. random bits allowed
   - as in classic masking
   - model unpredictability
   - in WB impl. as **pseudorandom**
2. **security requirement:**
   any non-constant $f \in \mathrm{span}\{v_i\}_i$
   must have sufficient error/noise in any fixed input
3. protects against the generalizations



Nonlinear Masking

- **Nonlinear masking**: $\exists v_1, \ldots, v_t$ *shares*

$$f(v_1, \ldots, v_t) = s$$

Nonlinear Masking

- **Nonlinear masking**: $\exists v_1, \ldots, v_t$ *shares*

$$f(v_1, \ldots, v_t) = s$$

- [BU18][2]: *provably secure* minimalist quadratic masking ($\deg f = 2$):

$$f(v_1, v_2, v_3) = v_1 v_2 \oplus v_3$$

[2] (ASIACRYPT 2018) Biryukov, Udovenko. Attacks and countermeasures for white-box designs.

Nonlinear Masking

- **Nonlinear masking**: $\exists v_1, \ldots, v_t$ *shares*

$$f(v_1, \ldots, v_t) = s$$

- [BU18][2]: *provably secure* minimalist quadratic masking ($\deg f = 2$):

$$f(v_1, v_2, v_3) = v_1 v_2 \oplus v_3$$

- [SEL21][3]: generalization and combination with linear masking, concrete & proof for $\deg f \leq 3$:

$$f(v_1, \ldots, v_t) = (v_1 v_2 \ldots v_d) \oplus v_{d+1} \oplus \ldots \oplus v_t$$

[2] (ASIACRYPT 2018) Biryukov, Udovenko. Attacks and countermeasures for white-box designs.
[3] (TCHES 2021) Seker, Eisenbarth, Liskiewicz. A white-box masking scheme resisting computational and algebraic attacks.

Nonlinear Masking

- **Nonlinear masking**: $\exists v_1, \ldots, v_t$ *shares*

$$f(v_1, \ldots, v_t) = s$$

- [BU18][2]: *provably secure* minimalist quadratic masking ($\deg f = 2$):

$$f(v_1, v_2, v_3) = v_1 v_2 \oplus v_3$$

- [SEL21][3]: generalization and combination with linear masking, concrete & proof for $\deg f \le 3$:

$$f(v_1, \ldots, v_t) = (v_1 v_2 \ldots v_d) \oplus v_{d+1} \oplus \ldots \oplus v_t$$

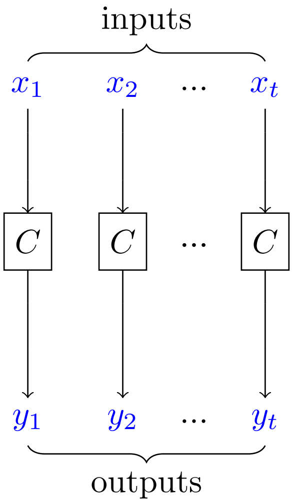
- Proofs are involved, larger security degrees are hard

[2] (ASIACRYPT 2018) Biryukov, Udovenko. Attacks and countermeasures for white-box designs.

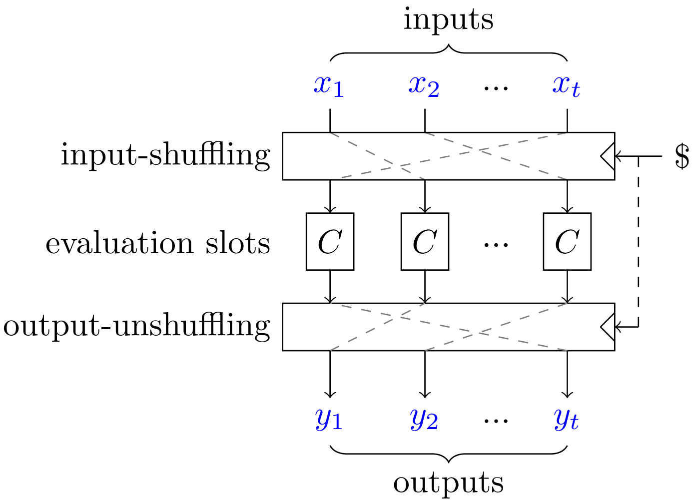
[3] (TCHES 2021) Seker, Eisenbarth, Liskiewicz. A white-box masking scheme resisting computational and algebraic attacks.

Plan

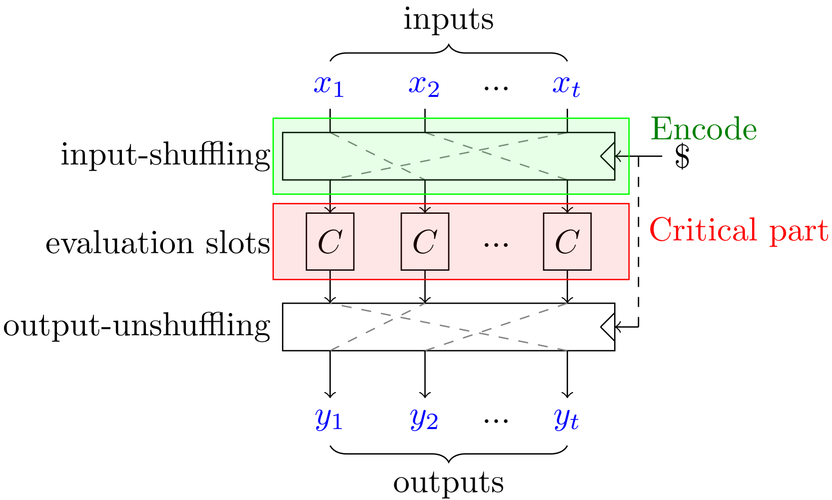
Basic shuffling



Basic shuffling

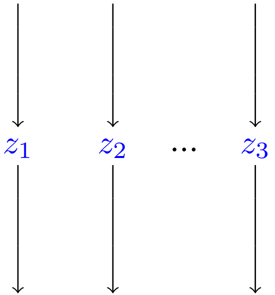


Basic shuffling



Plan

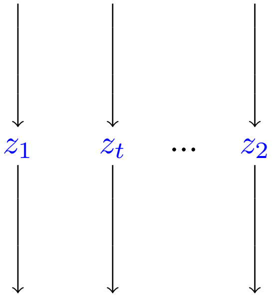
Algebraic insecurity of dummyless shuffling (1/3)



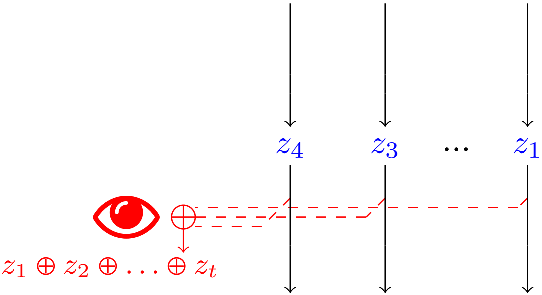
Algebraic insecurity of dummyless shuffling (1/3)

$$z_6 \quad z_4 \quad \cdots \quad z_5$$

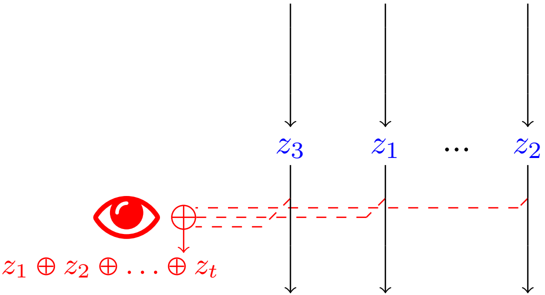
Algebraic insecurity of dummyless shuffling (1/3)



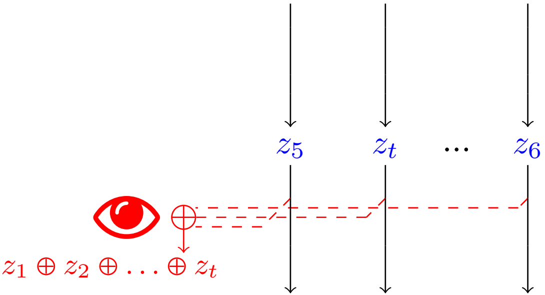
Algebraic insecurity of dummyless shuffling (1/3)



Algebraic insecurity of dummyless shuffling (1/3)



Algebraic insecurity of dummyless shuffling (1/3)



Algebraic insecurity of dummyless shuffling (3/3)

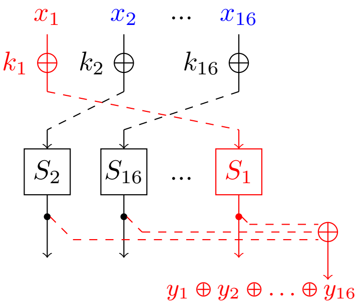
- Generally, all symmetric functions are leaked!
- No matter how shuffling is implemented...

Algebraic insecurity of dummyless shuffling (3/3)

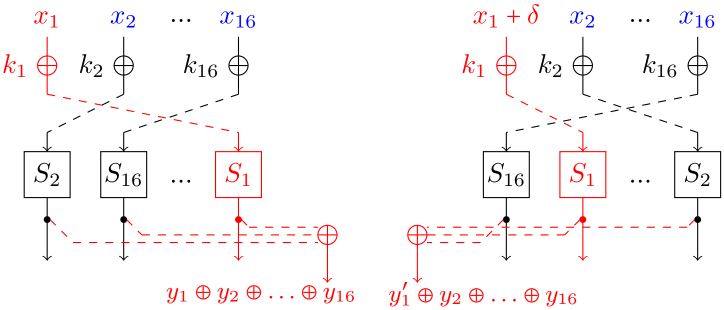
- Generally, all symmetric functions are leaked!
- No matter how shuffling is implemented...

- But is it exploitable?

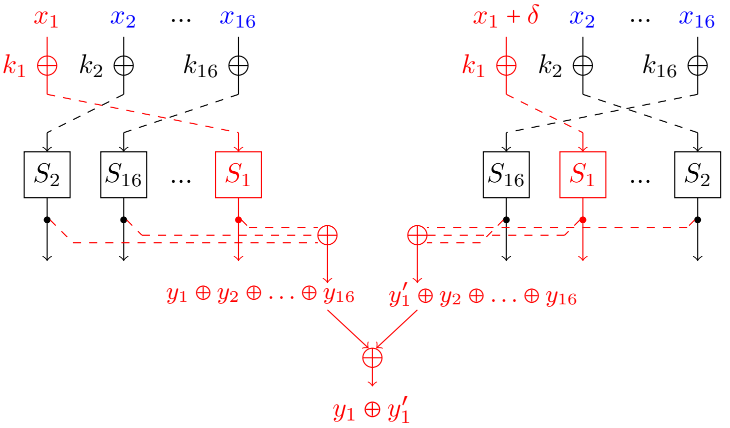
Differential Algebraic Attack

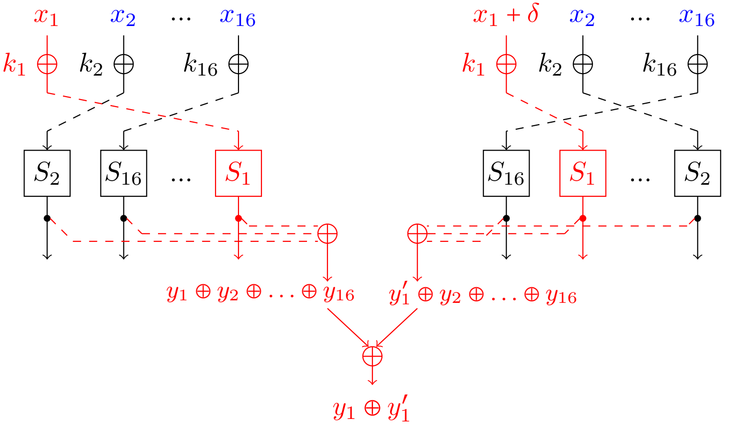


Differential Algebraic Attack



Differential Algebraic Attack

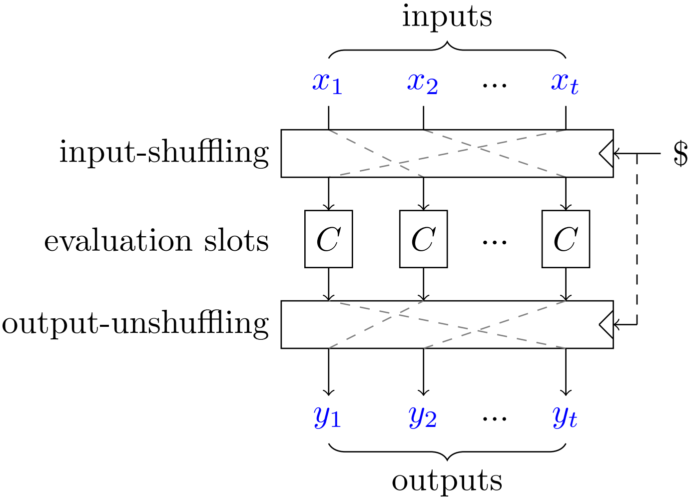




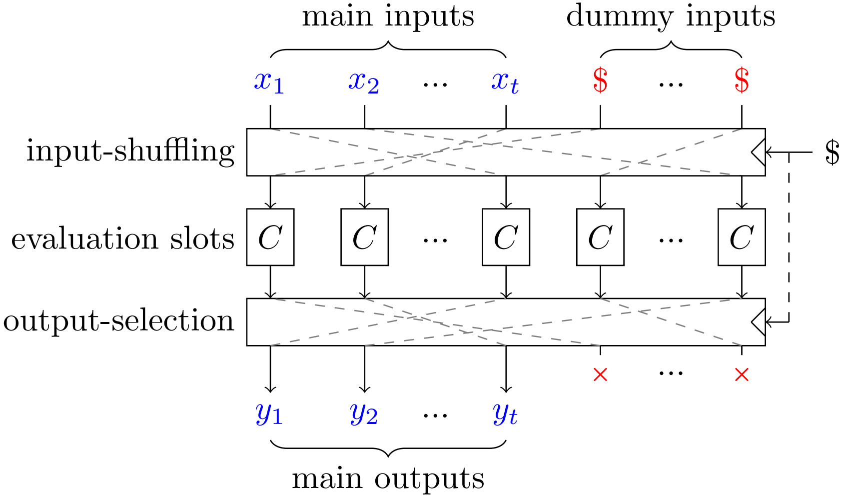
$$= \mathbf{S_1}(\mathbf{x_1} \oplus \mathbf{k_1}) \oplus \mathbf{S_1}(\mathbf{x_1} \oplus \delta \oplus \mathbf{k_1})$$

Plan

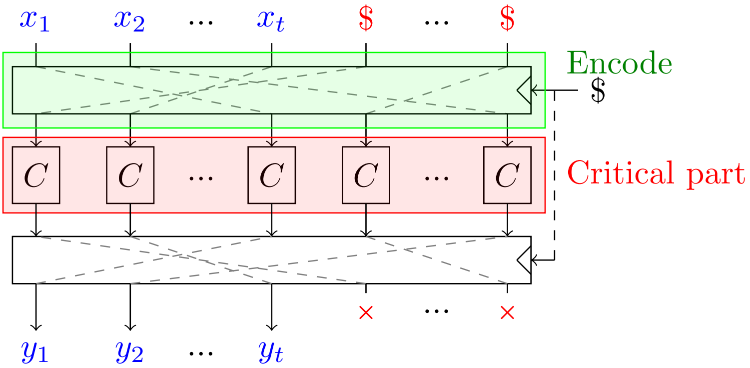
Dummy shuffling



Dummy shuffling



Dummy shuffling



Plan

Linear security of dummy shuffling

Definition

Let C be a slot implementation. Denote by $e_1(C)$ the minimum error[4] of a non-constant function from the linear span of C:

$$e_1(C) := \min \{\mathrm{err}(f) \mid f \in (\mathrm{span}\ C) \setminus \{0,1\}\}$$

Theorem

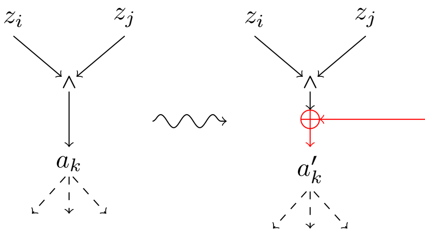
The dummy shuffling scheme with slots C is algebraically secure with the minimum error τ lower bounded as:

$$\tau \geq \frac{\#\textit{dummy slots}}{\#\textit{slots}} \cdot e_1(C)$$

[4] The minimum distance to a constant function

Plan

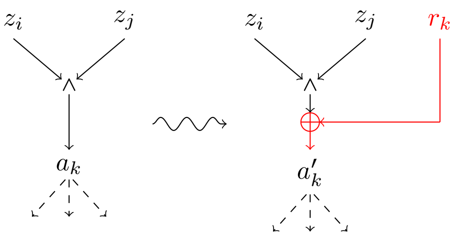
Provably security via refreshing



$$r_k \begin{cases} \leftarrow 0, & \text{if main slot,} \\ \xleftarrow{\$} \mathbb{F}_2, & \text{if dummy slot,} \end{cases}$$

(*encoding phase setup*)

Provably security via refreshing



$$r_k \begin{cases} \leftarrow 0, & \text{if main slot,} \\ \xleftarrow{\$} \mathbb{F}_2, & \text{if dummy slot,} \end{cases}$$

(*encoding phase setup*)

Theorem

*The dummy shuffling scheme with **refreshed** implementations of slots is degree-1 algebraically secure with the minimum error τ lower bounded as:*

$$\tau \geq \frac{\#\text{dummy slots}}{\#\text{slots}} \cdot \frac{1}{4} \qquad (e_1(\tilde{C}) \geq \frac{1}{4})$$

Plan

Full degree security

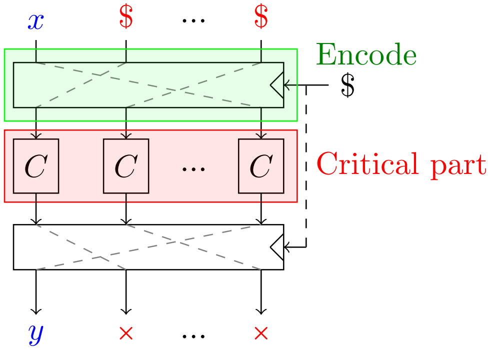
Theorem (Main)

*The dummy shuffling scheme with **refreshed** implementations using 1 main slot is*

$$\textit{degree-}d \textit{ algebraically secure}$$
$$(1 \leq d \leq \#\textit{dummy slots})$$

with the minimum error τ lower bounded as:

$$\tau \geq \frac{\#\mathit{slots} - d}{\#\mathit{slots}} \cdot \frac{1}{2^{2d}}$$



Plan

Implementation cost estimation

Protection degree	XOR	AND	Error τ	Ref.
1	$33 + 6\$$	$43 + 6\$$	$1/16$	[BU18, Alg. 3]
1	7	$16 + 2\$$	$1/16$	[SEL21]
1 $(t = 1)$	2	$8 + 1\$$	$1/8$	This work
2	16	$46 + 3\$$	$1/4096$	[SEL21]
2 $(t = 2)$	3	$14 + 2\$$	$1/48$	This work
d $(t \geq d)$	$t + 1$	$(6t + 2) + t\$$	$\frac{t+1-d}{t+1} \cdot \frac{1}{2^{2d}}$	This work

Estimation of gate complexity for protections against algebraic attacks per original AND/XOR gate. \$ stands for one random bit generation. t is the number of dummy slots.

The End

More in the paper:

1. Matching degree-$(d+1)$ attack on d dummy slots
2. A proof-of-concept implementation of *public* dummy shuffling, relying on a single slot implementation (used in the WhibOx 2019 contest surviving challenge #100)

Open problems:

1. Concrete evaluation of security against LPN-based attacks
2. Extending the algebraic security model to cover the full implementation
3. Are there more generic value-based attacks to consider, or does algebraic&correlation security cover it? (excluding fault attacks and data-dependency analysis).

ia.cr/2021/290 github.com/CryptoExperts/EC21-dummy-shuffling

[BHMT16] Joppe W. Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen.
Differential computation analysis: Hiding your white-box designs is not enough.
In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 215–236. Springer, Heidelberg, 2016.

[BU18] Alex Biryukov and Aleksei Udovenko.
Attacks and countermeasures for white-box designs.
In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 373–402. Springer, Heidelberg, 2018.

[CEJv03] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot.
White-box cryptography and an AES implementation.
In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 250–270. Springer, Heidelberg, 2003.

[SEL21] Okan Seker, Thomas Eisenbarth, and Maciej Liskiewicz.
A white-box masking scheme resisting computational and algebraic attacks.
IACR TCHES, 2021(2):61–105, 2021.