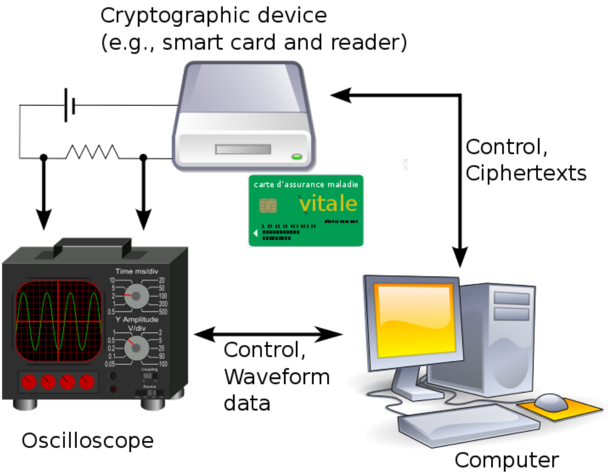
# Higher Order Masking of Look-up Tables

Jean-Sébastien Coron

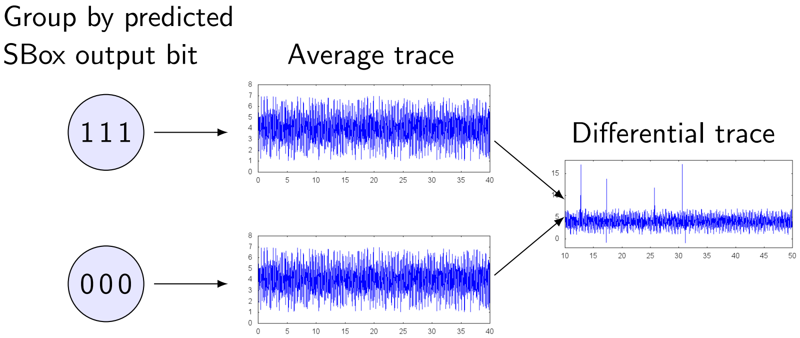
University of Luxembourg

EUROCRYPT, 2014-05-14

# Side-channel Attacks



# Differential Power Analysis [KJJ99]



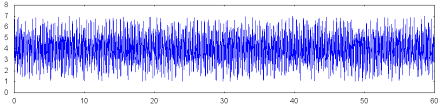
# Masking Countermeasure

- Let $x$ be some variable in a block-cipher.
- Masking countermeasure: generate a random $r$, and manipulate the masked value $x'$

$$x' = x \oplus r$$

instead of $x$.

- $r$ is random $\Rightarrow x'$ is random
$\Rightarrow$ power consumption of $x'$ is random




$\Rightarrow$ no information about $x$ is leaked

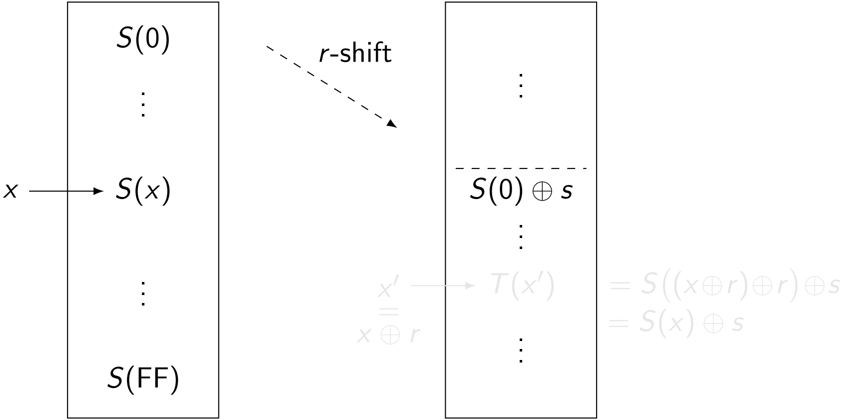
# Masking Countermeasure

- How do we compute with $x' = x \oplus r$ instead of $x$ ?
- Linear operation $f(x)$ (*e.g.* `MixColumns` in AES): easy

$$f(x') = f(x) \oplus f(r)$$

  - We compute $f(x')$ and $f(r)$ separately.
  - $f(x)$ is now masked with $f(r)$ instead of $r$.
- Non-linear operations (SBOX): randomized table [CJRR99]

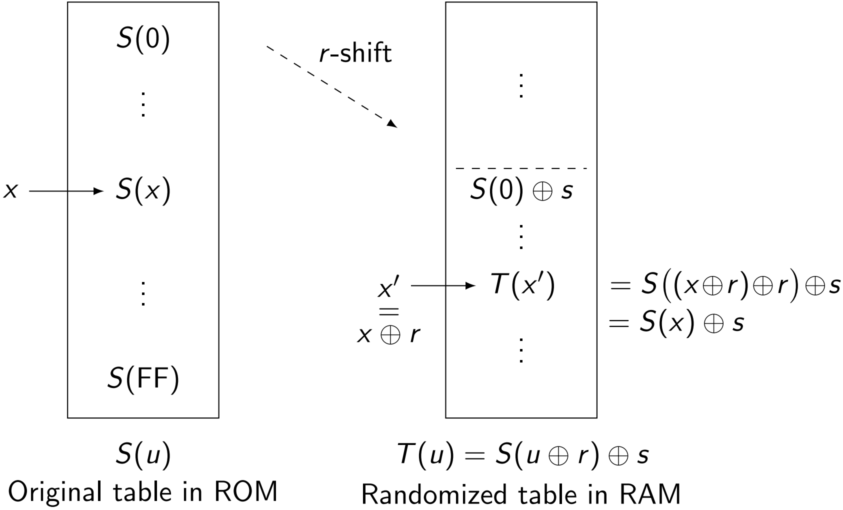
# Randomized Table Countermeasure [CJRR99]




$S(u)$
Original table in ROM

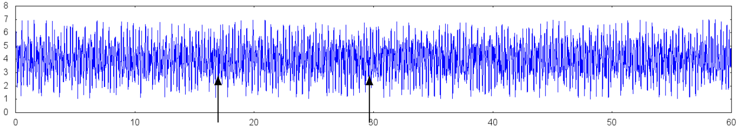
$T(u) = S(u \oplus r) \oplus s$
Randomized table in RAM

# Randomized Table Countermeasure [CJRR99]



# Second-order Attack

- Second-order attack:




$E(x')$ $E(r)$

$f(E(x'), E(r))$ correlated with $x = x' \oplus r$

- Requires more curves but can be practical

# Higher-order masking

- Solution: $n$ shares instead of 2:

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- Any subset of $n-1$ shares is uniformly and independently distributed
  - If we probe at most $n-1$ shares $x_i$, we learn nothing about $x \Rightarrow$ secure against a DPA attack of order $n-1$.
- Linear operations: still easy
  - Compute the $f(x_i)$ separately

$$f(x) = f(x_1) \oplus f(x_2) \oplus \cdots \oplus f(x_n)$$

# Higher-order computation of SBoxes

- SBox computation ?
  - We have input shares $x_1, \ldots, x_n$, with

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

  - We must output shares $y_1, \ldots, y_n$, such that

$$S(x) = y_1 \oplus y_2 \oplus \cdots \oplus y_n$$

  - without leaking information about $x$.
- **This talk**: first generalization of the previous randomized table countermeasure to $n$ shares.

# Existing Higher Order Countermeasure

- Ishai-Sahai-Wagner private circuit [ISW03]
  - Shows how to transform any boolean circuit $C$ into a circuit of size $\mathcal{O}(|C| \cdot t^2)$ perfectly secure against $t$ probes.
- Rivain-Prouff (CHES 2010) countermeasure for AES:

$$S(x) = x^{254} \in \mathbb{F}_{2^8}$$

  - Secure multiplication based on [ISW03]:

$$z = xy = \left(\bigoplus_{i=1}^{n} x_i\right) \cdot \left(\bigoplus_{i=1}^{n} y_i\right) = \bigoplus_{1 \leq i,j \leq n} x_i y_j$$

  - Provably secure against $t$-th order DPA with $n \geq 2t+1$ shares.

## Existing Higher Order Countermeasures

- Carlet *et al.* (FSE 2012) countermeasure for any Sbox.
  - Lagrange interpolation

$$S(x) = \sum_{i=0}^{2^k-1} \alpha_i \cdot x^i$$

over $\mathbb{F}_{2^k}$, for constant coefficients $\alpha_i \in \mathbb{F}_{2^k}$.

- **This talk**: alternative to Rivain-Prouff and Carlet *et al.* countermeasures
  - Generalization of the classical randomized table countermeasure.
  - No field operations, only table recomputation.

# Existing Higher Order Countermeasures

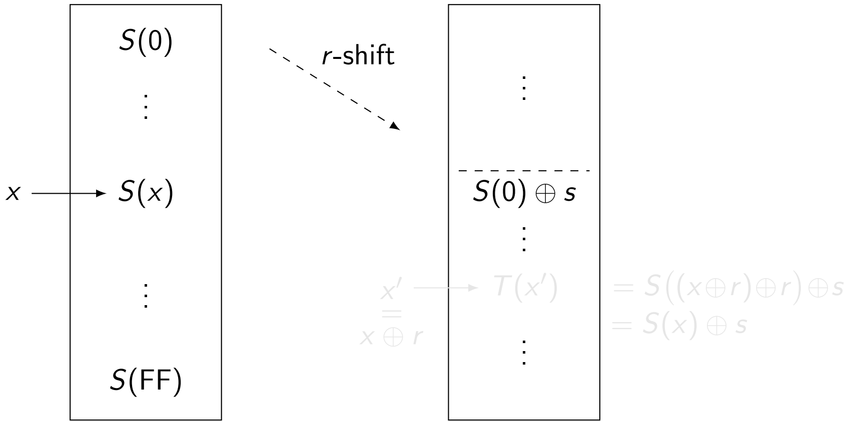
- Carlet *et al.* (FSE 2012) countermeasure for any Sbox.
  - Lagrange interpolation

$$S(x) = \sum_{i=0}^{2^k-1} \alpha_i \cdot x^i$$

    over $\mathbb{F}_{2^k}$, for constant coefficients $\alpha_i \in \mathbb{F}_{2^k}$.

- **This talk**: alternative to Rivain-Prouff and Carlet *et al.* countermeasures
  - Generalization of the classical randomized table countermeasure.
  - No field operations, only table recomputation.

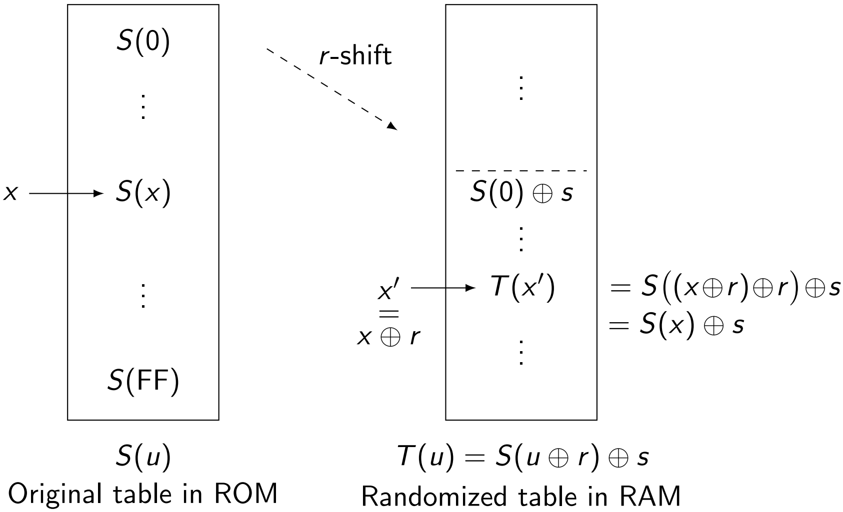
# Randomized Table Countermeasure [CJRR99]




$S(u)$
Original table in ROM

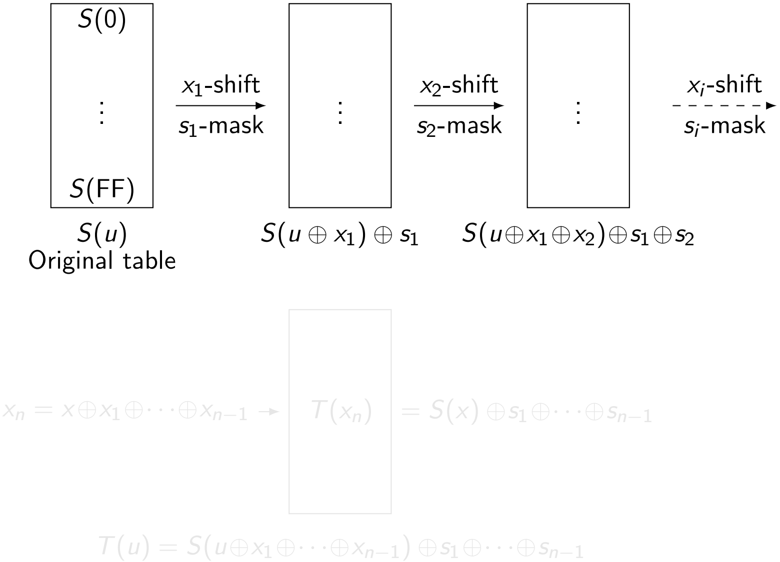
$T(u) = S(u \oplus r) \oplus s$
Randomized table in RAM

# Randomized Table Countermeasure [CJRR99]



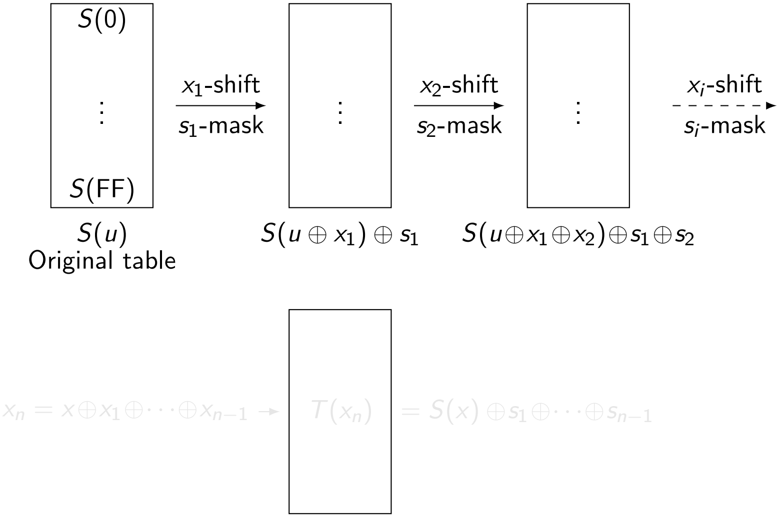
# First attempt: Schramm and Paar countermeasure [SP06]

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$



# First attempt: Schramm and Paar countermeasure [SP06]

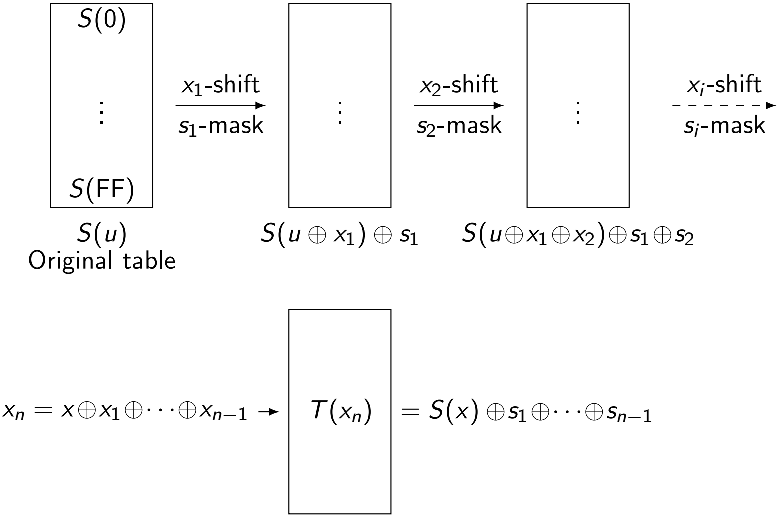
$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$




$$T(u) = S(u \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus s_1 \oplus \cdots \oplus s_{n-1}$$

# First attempt: Schramm and Paar countermeasure [SP06]

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$




$$T(u) = S(u \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus s_1 \oplus \cdots \oplus s_{n-1}$$

# Third-order Attack for any $n$

- Final randomized table:

$$
\begin{array}{|c|l}
T(0) & = S(0 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus s_1 \oplus \cdots \oplus s_{n-1} \\
 & \qquad\qquad\qquad\qquad \oplus \\
T(1) & = S(1 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus s_1 \oplus \cdots \oplus s_{n-1} \\
 & \qquad\qquad\qquad\qquad = \\
 & S(0 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus S(1 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \\
\vdots & \text{only depends on } x_1 \oplus \cdots \oplus x_{n-1}, \\
 & \text{also probe } x_n \Rightarrow \text{3rd order attack.}
\end{array}
$$

- For high-order countermeasures, do not reuse the same masks multiple times !
  - Using the same mask $r$ is OK only for first-order countermeasures.

# Third-order Attack for any $n$

- Final randomized table:

$$\boxed{\begin{array}{c} T(0) \\ \\ T(1) \\ \\ \vdots \\ \\ \end{array}}\quad \begin{array}{l} = S(0\oplus x_1\oplus\cdots\oplus x_{n-1})\oplus s_1\oplus\cdots\oplus s_{n-1} \\ \qquad\qquad\qquad\oplus \\ = S(1\oplus x_1\oplus\cdots\oplus x_{n-1})\oplus s_1\oplus\cdots\oplus s_{n-1} \\ \qquad\qquad\qquad= \\ S(0\oplus x_1\oplus\cdots\oplus x_{n-1})\oplus S(1\oplus x_1\oplus\cdots\oplus x_{n-1}) \end{array}$$

only depends on $x_1\oplus\cdots\oplus x_{n-1}$,
also probe $x_n \Rightarrow$ 3rd order attack.

- For high-order countermeasures, do not reuse the same masks multiple times !
  - Using the same mask $r$ is OK only for first-order countermeasures.

# Third-order Attack for any $n$

- Final randomized table:

$$\boxed{\begin{array}{c} T(0) \\ \\ T(1) \\ \\ \vdots \\ \\ \end{array}} \begin{array}{l} = S(0 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus s_1 \oplus \cdots \oplus s_{n-1} \\ \qquad\qquad\qquad\qquad \oplus \\ = S(1 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus s_1 \oplus \cdots \oplus s_{n-1} \\ \qquad\qquad\qquad\qquad = \\ S(0 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \oplus S(1 \oplus x_1 \oplus \cdots \oplus x_{n-1}) \\ \text{only depends on } x_1 \oplus \cdots \oplus x_{n-1}, \\ \text{also probe } x_n \Rightarrow \text{3rd order attack.} \end{array}$$

- For high-order countermeasures, do not reuse the same masks multiple times !
  - Using the same mask $r$ is OK only for first-order countermeasures.

# New Countermeasure

- **This talk:** new countermeasure for SBOXes, secure against higher-order attacks:
  - Variant of Schramm and Paar countermeasure
  - but use different masks for every line of the Sbox
  - and refresh the masks between successive shifts of the table.
- Provably secure against $t$-th order DPA, in the ISW model, with $n \geq 2t+1$ shares.
  - Alternative to Rivain-Prouff and Carlet *et al.* countermeasures based on finite-fields operations.

# Initial table with $n$ shares

- Every line of the SBox is initially randomly shared among $n$ shares, independently for every line.

$$
\begin{array}{|c|c}
\hline
(s_{00,1}, \ldots, s_{00,n}) & S(00) \\
\vdots & \\
(s_{u,1}, \ldots, s_{u,n}) & S(u) \\
\vdots & \\
(s_{\mathrm{FF},1}, \ldots, s_{\mathrm{FF},n}) & S(\mathrm{FF}) \\
\hline
\end{array}
$$

Original shared table

  - Equivalent to having $n$ randomized tables instead of 1.

- The lines of the table are then progressively shifted by $x_1$, $x_2$, $\ldots, x_{n-1}$, as in Schramm and Paar, but with a RefreshMask after every shift.

# Initial table with $n$ shares

- Every line of the SBox is initially randomly shared among $n$ shares, independently for every line.

$$
\begin{array}{|c|c}
\hline
(s_{00,1}, \ldots, s_{00,n}) & S(00) \\
\vdots & \\
(s_{u,1}, \ldots, s_{u,n}) & S(u) \\
\vdots & \\
(s_{FF,1}, \ldots, s_{FF,n}) & S(FF) \\
\hline
\end{array}
$$

Original shared table

  - Equivalent to having $n$ randomized tables instead of 1.

- The lines of the table are then progressively shifted by $x_1$, $x_2$, $\ldots, x_{n-1}$, as in Schramm and Paar, but with a RefreshMask after every shift.

# Iterative input shift by $x_i$

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

| Original shared table | | | | |
|---|---|---|---|---|
| $(s_{00,1}, \ldots, s_{00,n})$ | $S(00)$ | | $\vdots$ | |
| $\vdots$ | | | | |
| $(s_{u,1}, \ldots, s_{u,n})$ | $S(u)$ | $\xrightarrow[\text{mask refresh}]{x_i\text{-shift}}$ | $(s^{(i)}_{u,1}, \ldots, s^{(i)}_{u,n})$ | $S(u \oplus x_1 \oplus \cdots \oplus x_i)$ |
| $\vdots$ | | | $\vdots$ | |
| $(s_{FF,1}, \ldots, s_{FF,n})$ | $S(FF)$ | | | |

Original shared table

| | | |
|---|---|---|
| | $\vdots$ | |
| | $(s^{(n-1)}_{u,1}, \ldots, s^{(n-1)}_{u,n})$ | $S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$ |
| $x_n = x \oplus x_1 \oplus \cdots \oplus x_{n-1} \longrightarrow$ | $(s^{(n-1)}_{x_n,1}, \ldots, s^{(n-1)}_{x_n,n})$ | $S(x)$ |

Final shared table

# Iterative input shift by $x_i$

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

| Original shared table | |
|---|---|
| $(s_{00,1}, \ldots, s_{00,n})$ | $S(00)$ |
| $\vdots$ | |
| $(s_{u,1}, \ldots, s_{u,n})$ | $S(u)$ |
| $\vdots$ | |
| $(s_{\mathrm{FF},1}, \ldots, s_{\mathrm{FF},n})$ | $S(\mathrm{FF})$ |

$\xrightarrow[\text{mask refresh}]{x_i\text{-shift}}$

| | |
|---|---|
| $\vdots$ | |
| $(s_{u,1}^{(i)}, \ldots, s_{u,n}^{(i)})$ | $S(u \oplus x_1 \oplus \cdots \oplus x_i)$ |
| $\vdots$ | |

| Final shared table | |
|---|---|
| $\vdots$ | |
| $(s_{u,1}^{(n-1)}, \ldots, s_{u,n}^{(n-1)})$ | $S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$ |
| $x_n = x \oplus x_1 \oplus \cdots \oplus x_{n-1} \rightarrow (s_{x_n,1}^{(n-1)}, \ldots, s_{x_n,n}^{(n-1)})$ | $S(x)$ |

Final shared table

# Iterative input shift by $x_i$

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

| Original shared table | | $x_i$-shift / mask refresh | | |
|---|---|---|---|---|
| $(s_{00,1}, \ldots, s_{00,n})$ | $S(00)$ | | $\vdots$ | |
| $\vdots$ | | | | |
| $(s_{u,1}, \ldots, s_{u,n})$ | $S(u)$ | $\dashrightarrow$ | $(s^{(i)}_{u,1}, \ldots, s^{(i)}_{u,n})$ | $S(u \oplus x_1 \oplus \cdots \oplus x_i)$ |
| $\vdots$ | | | $\vdots$ | |
| $(s_{\mathrm{FF},1}, \ldots, s_{\mathrm{FF},n})$ | $S(\mathrm{FF})$ | | | |

Original shared table

| | | |
|---|---|---|
| | $\vdots$ | |
| | $(s^{(n-1)}_{u,1}, \ldots, s^{(n-1)}_{u,n})$ | $S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$ |
| $x_n = x \oplus x_1 \oplus \cdots \oplus x_{n-1} \longrightarrow$ | $(s^{(n-1)}_{x_n,1}, \ldots, s^{(n-1)}_{x_n,n})$ | $S(x)$ |

Final shared table

# Final randomized table

- In the final shared table, the inputs are shifted by $x_1 \oplus \cdots \oplus x_{n-1}$ :

$$\vdots$$

$$(s_{u,1}^{(n-1)}, \ldots, s_{u,n}^{(n-1)}) \quad S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$$

$$x_n = x \oplus x_1 \oplus \cdots \oplus x_{n-1} \rightarrow (s_{x_n,1}^{(n-1)}, \ldots, s_{x_n,n}^{(n-1)}) \quad S(x)$$

Final shared table

- The $n$ output shares $T(x_n) = (s_{x_n,1}^{(n-1)}, \ldots, s_{x_n,n}^{(n-1)})$ correspond to the output $S(x)$

# Final randomized table

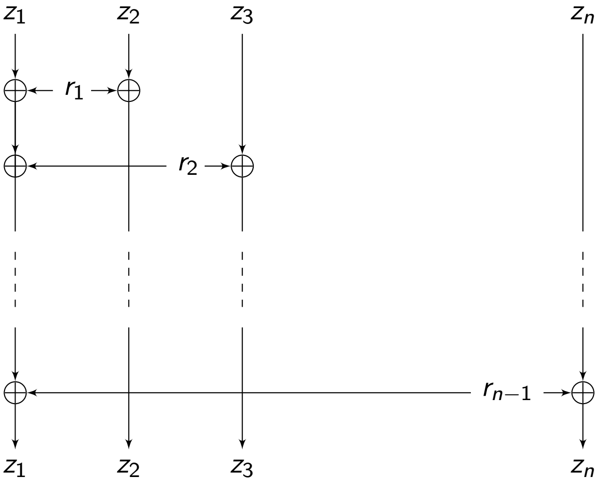
- In the final shared table, the inputs are shifted by $x_1 \oplus \cdots \oplus x_{n-1}$ :

$$
\begin{array}{rc|c|l}
\cline{3-3}
 & & \vdots & \\
 & & (s^{(n-1)}_{u,1}, \ldots, s^{(n-1)}_{u,n}) & S(u \oplus x_1 \oplus \cdots \oplus x_{n-1}) \\
x_n = x \oplus x_1 \oplus \cdots \oplus x_{n-1} & \longrightarrow & (s^{(n-1)}_{x_n,1}, \ldots, s^{(n-1)}_{x_n,n}) & S(x) \\
 & & & \\
\cline{3-3}
\end{array}
$$

Final shared table

- The $n$ output shares $T(x_n) = (s^{(n-1)}_{x_n,1}, \ldots, s^{(n-1)}_{x_n,n})$ correspond to the output $S(x)$

# Mask Refreshing




- Required property: any subset of $n-1$ output shares $z_i$ is uniformly and independently distributed.

# Why are the mask refreshing necessary ?

- Without mask refreshing:

$s_{0,1}\ ,\ \cdots\ ,\ s_{0,n}$ $\quad$ $S(00)$

$\vdots$

Original shared table

If collision, then presumably $x_1 \oplus \cdots \oplus x_{n-1} = 0$

also probe $x_n \Rightarrow$ 3rd order attack

$s_{0,1}^{(n-1)}, \ldots, s_{0,n}^{(n-1)}$ $\quad$ $S(00 \oplus x_1 \oplus \cdots \oplus x_{n-1})$

$\vdots$

Final shared table

- The mask refreshing prevents from correlating shares between different shifts of the tables.

# Why are the mask refreshing necessary ?

- Without mask refreshing:

$s_{0,1}$ , $\cdots$ , $s_{0,n}$ $\qquad S(00)$

$\vdots$

Original shared table

If collision, then presumably $x_1 \oplus \cdots \oplus x_{n-1} = 0$

also probe $x_n \Rightarrow$ 3rd order attack

$s_{0,1}^{(n-1)}, \ldots, s_{0,n}^{(n-1)}$ $\qquad S(00 \oplus x_1 \oplus \cdots \oplus x_{n-1})$

$\vdots$

Final shared table

- The mask refreshing prevents from correlating shares between different shifts of the tables.

# Why are the mask refreshing necessary ?

- Without mask refreshing:

$s_{0,1}$ , $\cdots$ , $s_{0,n}$ $\qquad S(00)$

$\vdots$

Original shared table

If collision, then presumably $x_1 \oplus \cdots \oplus x_{n-1} = 0$

also probe $x_n \Rightarrow$ 3rd order attack

$s_{0,1}^{(n-1)}, \ldots, s_{0,n}^{(n-1)} \qquad S(00 \oplus x_1 \oplus \cdots \oplus x_{n-1})$

$\vdots$

Final shared table

- The mask refreshing prevents from correlating shares between different shifts of the tables.

# Full Algorithm

**Algorithm 1** Masked computation of $y = S(x)$

**Input:** $x_1, \ldots, x_n$ such that $x = x_1 \oplus \cdots \oplus x_n$
**Output:** $y_1, \ldots, y_n$ such that $y = S(x) = y_1 \oplus \cdots \oplus y_n$

**for all** $u \in \{0,1\}^k$ **do**
  $T(u) \leftarrow \big(S(u), 0, \ldots, 0\big) \in (\{0,1\}^{k'})^n$   $\triangleright \oplus\big(T(u)\big) = S(u)$
**end for**
**for** $i = 1$ to $n-1$ **do**
  **for all** $u \in \{0,1\}^k$ **do**
    **for** $j = 1$ **to** $n$ **do** $T'(u)[j] \leftarrow T(u \oplus x_i)[j]$   $\triangleright T'(u) \leftarrow T(u \oplus x_i)$
  **end for**
  **for all** $u \in \{0,1\}^k$ **do**
    $T(u) \leftarrow \text{RefreshMasks}\big(T'(u)\big)$   $\triangleright \oplus\big(T(u)\big) = S(u \oplus x_1 \oplus \cdots \oplus x_i)$
  **end for**
**end for**   $\triangleright \oplus\big(T(u)\big) = S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$ for all $u \in \{0,1\}^k$.
$(y_1, \ldots, y_n) \leftarrow \text{RefreshMasks}\big(T(x_n)\big)$   $\triangleright \oplus\big(T(x_n)\big) = S(x)$
**return** $y_1, \ldots, y_n$

# Mask refreshing

**Algorithm 2** RefreshMasks

**Input:** $z_1, \ldots, z_n$ such that $z = z_1 \oplus \cdots \oplus z_n$
**Output:** $z_1, \ldots, z_n$ such that $z = z_1 \oplus \cdots \oplus z_n$

```
for i = 2 to n do
    tmp ← {0,1}^{k'}
    z_1 ← z_1 ⊕ tmp
    z_i ← z_i ⊕ tmp
end for
return z_1, ..., z_n
```

$$
\begin{array}{l}
\textbf{for } i = 2 \text{ to } n \textbf{ do} \\
\quad tmp \leftarrow \{0,1\}^{k'} \\
\quad z_1 \leftarrow z_1 \oplus tmp \\
\quad z_i \leftarrow z_i \oplus tmp \\
\textbf{end for} \\
\textbf{return } z_1, \ldots, z_n
\end{array}
$$

# Asymptotic Complexity

- Asymptotic complexity for $k$-bit SBox and $n$ shares:

| Countermeasure | Time comp. | Memory comp. |
|---|---|---|
| Carlet *et al.* | $\mathcal{O}(2^{k/2} \cdot n^2)$ | $\mathcal{O}(2^{k/2} \cdot n)$ |
| New countermeasure. | $\mathcal{O}(2^{k} \cdot n^2)$ | $\mathcal{O}(2^{k} \cdot n)$ |
| New count. (large register) | $\mathcal{O}(2^{k/2} \cdot n^2)$ | $\mathcal{O}(2^{k} \cdot n)$ |

- Large register variant: pack multiple Sbox outputs in a single register
  - For DES, pack 8 output 4-bit nibbles into a 32-bit register
  - Running time divided by 8

# Asymptotic Complexity

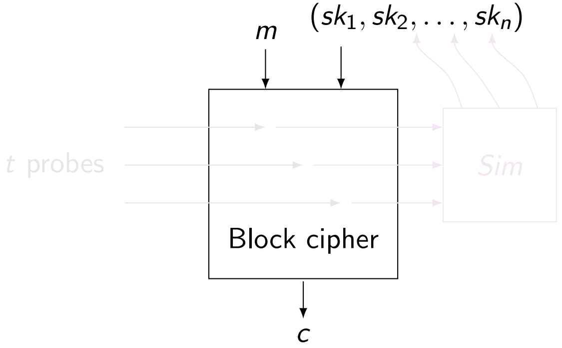
- Asymptotic complexity for $k$-bit SBox and $n$ shares:

| Countermeasure | Time comp. | Memory comp. |
|---|---|---|
| Carlet *et al.* | $\mathcal{O}(2^{k/2} \cdot n^2)$ | $\mathcal{O}(2^{k/2} \cdot n)$ |
| New countermeasure. | $\mathcal{O}(2^{k} \cdot n^2)$ | $\mathcal{O}(2^{k} \cdot n)$ |
| New count. (large register) | $\mathcal{O}(2^{k/2} \cdot n^2)$ | $\mathcal{O}(2^{k} \cdot n)$ |

- Large register variant: pack multiple Sbox outputs in a single register
  - For DES, pack 8 output 4-bit nibbles into a 32-bit register
  - Running time divided by 8

# ISW security model

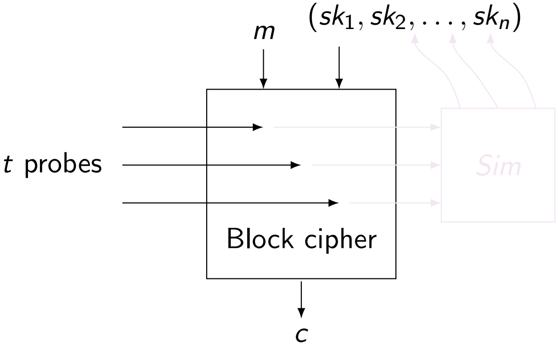
- Simulation framework of [ISW03]:




- Show that any $t$ probes can be perfectly simulated from at most $n-1$ of the $sk_i$'s.
- Those $n-1$ shares $sk_i$ are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the $t$ probes, since he could perfectly simulate those $t$ probes by himself.

# ISW security model

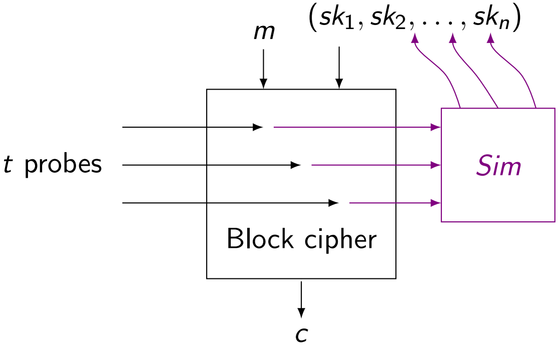
- Simulation framework of [ISW03]:




- Show that any $t$ probes can be perfectly simulated from at most $n-1$ of the $sk_i$'s.
- Those $n-1$ shares $sk_i$ are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the $t$ probes, since he could perfectly simulate those $t$ probes by himself.

# ISW security model

- Simulation framework of [ISW03]:




- Show that any $t$ probes can be perfectly simulated from at most $n-1$ of the $sk_i$'s.
- Those $n-1$ shares $sk_i$ are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the $t$ probes, since he could perfectly simulate those $t$ probes by himself.

# Security of high-order table recomputation

Theorem

*The table recomputation countermeasure is secure against $t$ probes in the ISW model, for $n \geq 2t + 1$.*

# Proof sketch

Initial table $(s_{u,1}, \ldots, s_{u,n})$ $S(u)$

Probes

No probes

$x_i$-shift

RefreshMask

Perfect simulation of any $n-1$ shares without knowing $x_i$

$(s^{(i)}_{u,1}, \ldots, s^{(i)}_{u,n})$ $S(u \oplus x_1 \oplus \cdots \oplus x_i)$

Probes

Final table $(s^{(n-1)}_{u,1}, \ldots, s^{(n-1)}_{u,n})$ $S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$

Table look-up $T(x_n) = (y_1, \ldots, y_n)$ $S(x)$

# Proof sketch

Initial table $(s_{u,1}, \ldots, s_{u,n})$ $S(u)$

Probes

No probes

$x_i$-shift

RefreshMask

Perfect simulation of any $n-1$ shares without knowing $x_i$

$(s^{(i)}_{u,1}, \ldots, s^{(i)}_{u,n})$ $S(u \oplus x_1 \oplus \cdots \oplus x_i)$

Probes

Final table $(s^{(n-1)}_{u,1}, \ldots, s^{(n-1)}_{u,n})$ $S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$

Table look-up $T(x_n) = (y_1, \ldots, y_n)$ $S(x)$

# Proof sketch

Initial table $(s_{u,1}, \ldots, s_{u,n})$ $S(u)$

Probes

No probes $x_i$-shift

RefreshMask

Perfect simulation of any $n-1$ shares without knowing $x_i$ $(s^{(i)}_{u,1}, \ldots, s^{(i)}_{u,n})$ $S(u \oplus x_1 \oplus \cdots \oplus x_i)$

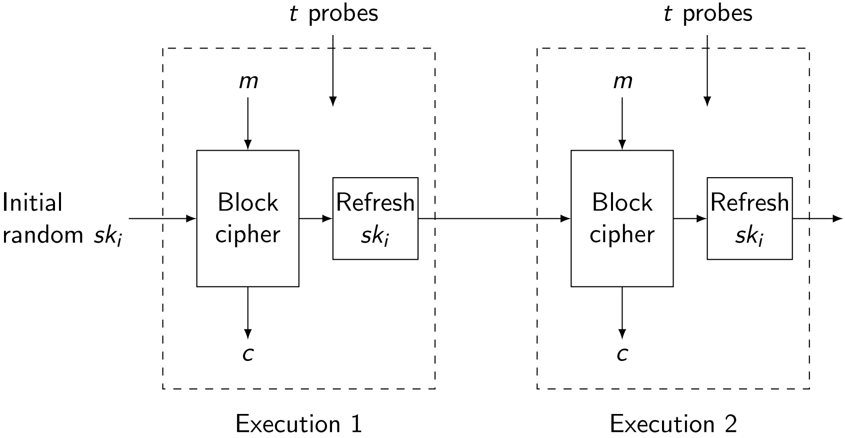
Probes

Final table $(s^{(n-1)}_{u,1}, \ldots, s^{(n-1)}_{u,n})$ $S(u \oplus x_1 \oplus \cdots \oplus x_{n-1})$

Table look-up $T(x_n) = (y_1, \ldots, y_n)$ $S(x)$

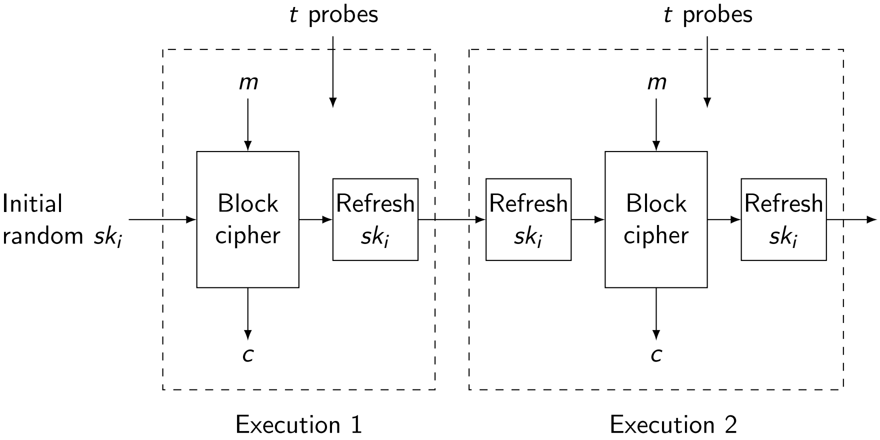
# Protecting a full block-cipher

- Adaptive model of [ISW03]:
  - The adversary can move its $t$ probes between successive executions of the block-cipher.
  - $n \geq 4t + 1$ are sufficient to guarantee security in the adaptive model



# Protecting a full block-cipher

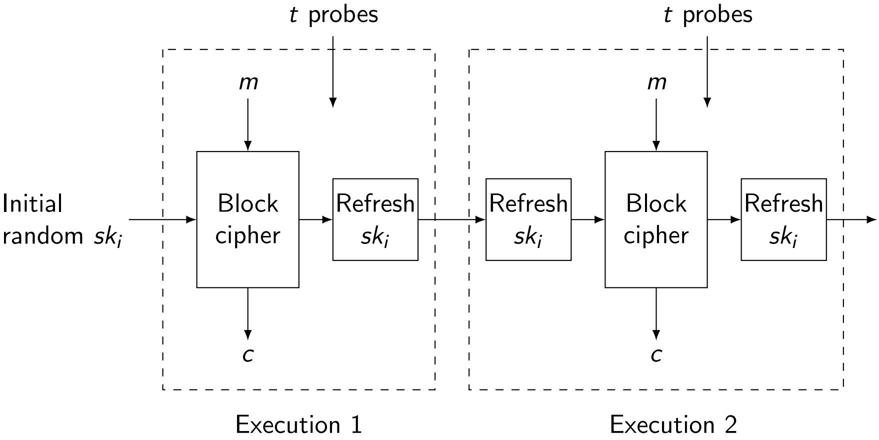
- Improvement: $n \geq 2t + 1$ are sufficient to guarantee security in the adaptive model




- Optimal: $\mathcal{A}$ can probe $t$ shares $sk_i$ at the end of one execution and $t$ shares $sk_i$ at the beginning of the next.

# Protecting a full block-cipher

- Improvement: $n \geq 2t+1$ are sufficient to guarantee security in the adaptive model




- Optimal: $\mathcal{A}$ can probe $t$ shares $sk_i$ at the end of one execution and $t$ shares $sk_i$ at the beginning of the next.

# Performances for AES

| | $t$ | $n$ | Time (ms) | Penalty |
|---|---|---|---|---|
| AES, unmasked | | | 0.0018 | 1 |
| AES, Rivain-Prouff | 1 | 3 | 0.092 | 50 |
| AES, table recomputation | 1 | 3 | 0.80 | 439 |
| AES, Rivain-Prouff | 2 | 5 | 0.18 | 96 |
| AES, table recomputation | 2 | 5 | 2.2 | 1205 |
| AES, Rivain-Prouff | 3 | 7 | 0.31 | 171 |
| AES, table recomputation | 3 | 7 | 4.4 | 2411 |
| AES, Rivain-Prouff | 4 | 9 | 0.51 | 276 |
| AES, table recomputation | 4 | 9 | 7.3 | 4003 |

- Table recomputation an order of magnitude slower than RP
  - RP can take advantage of the special structure of the AES SBox (only 4 mults in $\mathbb{F}_{2^8}$).

# Performances for DES

| | $t$ | $n$ | Time (ms) | Penalty |
|---|---|---|---|---|
| DES, unmasked | | | 0.010 | 1 |
| DES, Carlet *et al.* | 1 | 3 | 0.47 | 47 |
| DES, table recomputation | 1 | 3 | 0.31 | 31 |
| DES, Carlet *et al.* | 2 | 5 | 0.78 | 79 |
| DES, table recomputation | 2 | 5 | 0.59 | 59 |
| DES, Carlet *et al.* | 3 | 7 | 1.3 | 129 |
| DES, table recomputation | 3 | 7 | 0.90 | 91 |
| DES, Carlet *et al.* | 4 | 9 | 1.9 | 189 |
| DES, table recomputation | 4 | 9 | 1.4 | 142 |

- For DES the performances are similar
- `http://github.com/coron/htable/`

Questions ?