# Side-Channel Attacks Countermeasure Based on Decomposed S-Boxes for Kuznyechik

Tamara Lavrenteva, Sergey Matveev

CTCrypt'2020

# Masking

The idea of masking

- $x$ - the intermediate value
- $m$ - the random mask
- $x^m = x * m$ - the masked value
- $*$ - operations used in a cryptographic algorithm

The masking implementation

* - linear function, simple implementation

- $f(x^m) = f(x) * f(m)$

* - non-linear function (S-box),

- $f(x^m) \neq f(x) * f(m)$
- $f(x^m) =?$

# Masking of the S-box

- Re–computation table (Method 1)
- Compression of the lookup Table (Method 2)
- Global lookup table (Method 3)
- "on-the-fly"lookup table computation (Method 4)

# Method 1: Re–computation table

ALGORITHM 1.
Input: S($x$) - S-box, $a, b \in V^n$ - random masks
Output. $(S_{a,b}(x), b)$

1. for $x$=0 to $2^{n-1}$ do
2. $S_{a,b}(x) := S(x \oplus a) \oplus b$
3. end

- $S(x) : \{0,1\}^n \to \{0,1\}^n$ - the lookup table of S-Box
- $S_{a,b}(x) = S(x \oplus a) \oplus b$ - random masks

The new lookup table is calculated every time after $a$ and $b$ masks changing.

# Method 2: Compression of the lookup Table

Praveen Kumar Vadnala, "Time-Memory Trade-Offs for Side-Channel Resistant Implementations of Block Ciphers CT-RCA 2017
ALGORITHM 2.
*ALGORITHM 2.1* The lookup table creation.
Input: $S(x)$ - S-box, where $S(x) = S_0(x) \parallel S_1(x)$ for all $x \in \{0,1\}^n$ , $a$,$b \in V^{\frac{n}{2}}$, $c \in V^n$ - masks.
Output: $S_{a,b}(x) = S_0(x \oplus a) \oplus S_1(x \oplus b) \oplus c$

1. for $x$=0 to $2^{n-1}$ do
2. $S_{a,b}(x) := S_0(x \oplus a) \oplus S_1(x \oplus b) \oplus c$
3. end

*ALGORITHM 2.2.* The lookup table use
Input: $S(x), S_{a,b}(x)$ - S-box (original and from ALGORITHM 2.1), $a, b \in V^n$, $c \in V^{\frac{n}{2}}$ - masks, masked input $x_{a,b} = x \oplus a \oplus b$, the new mask $d \in V^{\frac{n}{2}}$
Output: $(a \parallel b, c \parallel (c \oplus d))$

1. $a := S_{a,b}(x_1 \oplus b) \oplus S_1(x_{a,b})$
2. $b := S_{a,b}(x_1 \oplus a) \oplus S_0(x_{a,b}) \oplus d$
3. $c \oplus d$

Output (a|b,C|(C⊕d))

# Method 3: Global lookup table method

One general look-up table is generated for all possible masks.
That is, the lookup table $S_{a,b}(x))$ are formed for all $x, a, b$ according to ALGORITHM 1.

# Method 4: "On-the-fly"a lookup table computation

A lookup table is computed on-the-fly by using mathematical representation of the substitution.
Each time the masked value $S(x) \oplus b$ may be computed from the tuple $(x \oplus a, a, b)$ and S-boxes representation algorithm.

# The complexity of evaluation and memory requirements

***Table 1.***

| Method | RAM | Table creation complexity | Re-masking complexity | Call complexity |
|---|---|---|---|---|
| Method 1 | $2^n$ | 0 | $2 \cdot 2^n \cdot A_n + 2^n \cdot R_n + 2n \cdot W_n$ | $R_n$ |
| Method 2 | $2^{n-1}$ | 0 | $4 \cdot 2^n \cdot A_n + 2 \cdot 2^n \cdot R_n + 2^n \cdot W_n$ | $4 \cdot R_n + 6 \cdot A_n$ |
| Method 3 | $2^{3n}$ | $2 \cdot 2^n(2 \cdot 2^n \cdot A_n + 2^n \cdot R_n + 2^n \cdot W_n)$ | 0 | $R_n$ |
| Method 4 | 0 | 0 | 0 | $2^n \cdot R_n + 2 \cdot 2^n \cdot A_n + 2n \cdot W_n$ |

$R_n$ - n-bit operation for reading from RAM

$A_n$ - $\oplus$ operation for binary addition of n-bit vectors

$W_n$ - n-bit operation for writing to RAM

# Kuznechik (GOST R 34.13-2015) short description

$\mathrm{E}(a)= \mathrm{X}[K_{10}]\mathrm{LSX}[K_9]...\mathrm{LSX}[K_1](a)$,
where
$K_i \in F_{2^8}^n$, $i$=1,$\cdots$,10, $n$=16 - the sequence of round keys
$\mathrm{X}[K](a)=K \oplus a$
$\mathrm{S}(y)=\mathrm{S}(a_{15}, \cdots, a_0)=\pi(a_{15}) \parallel \cdots \parallel \pi(a_0)$,
$\pi$ - 8-bit substitution,
$\mathrm{L}(a)=\mathrm{R}^{16}(a)$,
$\mathrm{R}(a)=\mathrm{R}(a_{15}, \cdots, a_0)=l(a_{15}, \cdots, a_0) \parallel a_{15} \cdots \parallel y_1$,
$l$ - a linear transformation in the field $GF(2^8)$
$K_1, K_2$ - a master key,
$K_{2i+1} \parallel K_{2i+2}$,$i$=1,2,3,4.
$C_i = L(i)$, $i$=1,2,...,32,
$\mathrm{F}[k](a_1, a_0)=(\mathrm{LSX}[k](a_1) \oplus a_0, a_1)$

# S-box decomposition

Biryukov A., Perrin L., Udovenko A. Reverse-engineering the S-box of streebog, kuznyechik and STRIBOBr1, http://eprin.iacr.org/2016/071.pdf
We can consider $\pi$ as a composition $\pi = \omega\pi_0\alpha$, where $\pi_0$ is a substitution defined by steps 2,3 ALGORITHM 3, $\omega$ and $\alpha$ is a linear maps.
ALGORITHM 3

1. $(l \parallel r) \leftarrow \alpha \cdot a$
2. $ll \leftarrow \begin{cases} \nu_0(l), & \textit{if } r = 0 \\ \nu_1(l \circ \chi(r)), & \textit{otherwise} \end{cases}$
3. $r \leftarrow \sigma(rr \circ \phi(ll))$
4. $b \leftarrow \omega \cdot (ll \parallel rr)$

Where

$l, r, ll, rr \in F_2^4$

$\circ$ - multiplication in finite field $GF(2^4)$

$\alpha$, $\omega$ - 8-bit linear permutations

$\chi$, $\nu_0$, $\nu_1$, $\phi$, $\sigma$ - non-linear 4-bit functions

# S-box decomposition based masking method

Generation masking look-up tables for 4-bits non-linear function:

$$\chi^{a,b}(x) = \chi(x \oplus a) \oplus b$$
$$\nu_0^{a,b}(x) = \nu_0(x \oplus a) \oplus b$$
$$\nu_1^{a,b}(x) = \nu_1(x \oplus a) \oplus b$$
$$\phi^{a,b}(x) = \phi(x \oplus a) \oplus b$$
$$\sigma^{a,b}(x) = \sigma(x \oplus a) \oplus b.$$

$a, b \in V^4$

# S-box decomposition based masking method

ALGORITHM 4.
Input: masked values: $(I^{m_1} \parallel r^{m_0})$, random masks: $(m_0,\ m_1,\ m_2,\ m_3,\ m_4,\ m_5)$, lookup tables: $\chi^{m_0,m_2}(x)$, $\nu_0^{m_1,m_3}(x)$, $\nu_1^{m1\circ m_2,m_3}(x)$, $\phi^{m_3,m_4}(x)$, $\sigma^{m_0\circ m_4,m_5}(x)$
Output: $(II^{m_3} \parallel rr^{m_5})$

1. $x \leftarrow \nu_0^{m_1,m_3}(I^{m_1})$
2. $y \leftarrow I^{m_1} \circ \chi^{m_0,m_2}(r^{m_0})$
3. $y \leftarrow y \oplus m_1 \circ \chi^{m_0,m_2}(r^{m_0})$
4. $y \leftarrow y \oplus m_2 \circ I^{m_1}$
5. $y \leftarrow \nu_1^{m_1\circ m_2,m_3}(y)$
6. $z \leftarrow MSB_4(16 - ((r^{m_0} + (\neg m_0 + 1)) \vee 0\text{x}10))$
7. $II^{m_3} \leftarrow x \oplus x \cdot z \oplus y \cdot z$
8. $y \leftarrow \phi^{m_3,m_4}(II^{m_3}) \circ r^{m_0}$
9. $y \leftarrow y \oplus m_0 \circ \phi^{m_3,m_4}(II^{m_3})$
10. $y \leftarrow y \oplus m_4 \circ r^{m_0}$
11. $rr^{m_5} \leftarrow \sigma^{m_0\circ m_4,m_5}$

# The complexity of S-box masking methods

***Table 2***

| Method | RAM | Table creation | Re-masking | The call lookup table |
|---|---|---|---|---|
| 1 | $5\frac{n}{2}2^{\frac{n}{2}}$ | 0 | $5(2 \cdot 2^{\frac{n}{2}}A_{\frac{n}{2}} + 2^{\frac{n}{2}}R_{\frac{n}{2}} + 2^{\frac{n}{2}}W_{\frac{n}{2}})$ | $R_{\frac{n}{2}}$ |
| 2 | $5\frac{n}{2}2^{\frac{n}{2}-1}$ | 0 | $5(4 \cdot 2^{\frac{n}{2}}A_{\frac{n}{2}} + 2 \cdot 2^{\frac{n}{2}}R_{\frac{n}{2}} + 2^{\frac{n}{2}}W_{\frac{n}{2}})$ | $5(4R_{\frac{n}{2}} + 6A_{\frac{n}{2}})$ |
| 3 | $5\frac{n}{2}2^{3\frac{n}{2}}$ | $5(2 \cdot 2^{\frac{n}{2}}(2 \cdot 2^{\frac{n}{2}}A_{\frac{n}{2}} + 2^{\frac{n}{2}}R_{\frac{n}{2}} + 2^{\frac{n}{2}}W_{\frac{n}{2}})$ | 0 | $5 \cdot R_{n/2}$ |
| 4 | 0 | 0 | 0 | $5 \cdot (2^{n/2} \cdot R_{n/2} + 2 \cdot 2^{n/2} \cdot A_{n/2} + 2^{n/2} \cdot W_{n/2})$ |

$R_n$ - n-bit operation for reading from RAM

$A_n$ - $\oplus$ operation for binary addition of n-bit vectors

$W_n$ - n-bit operation for writing to RAM

# The complexity of masking methods (in comparison)

***Table 3.***

| Method | RAM | | Table creation | | Re-masking | | The call lookup table | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 1 | $5 \cdot 2^6$ | $2^{11}$ | 0 | 0 | $5 \cdot 2^6 A_4$ | $2^{10} A_8$ | $R_4$ | $R_8$ |
| 2 | $5 \cdot 2^5$ | $2^{10}$ | 0 | 0 | $5 \cdot 2^6 A_4$ | $2^{10} A_8$ | $50 A_4$ | $10 A_8$ |
| 3 | $5 \cdot 2^{14}$ | $2^{27}$ | $5 \cdot 2^{14} A_4$ | $2^{27} A_8$ | 0 | 0 | $5 R_4$ | $R_8$ |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | $5 \cdot 2^6 A_4$ | $2^{10} A_8$ |

# Security analysis

- On the each step of the algorithm the intermediate result computed from the some sensitive variable and masks
- Attacker can to observe the "zero value"for $z$ variable at step 6 of the algorithm 4. The average complexity of recovery 128-bit round key $K_1$ is $S = 2^{127.994}$

Thanks for attentions!
Questions?