MI-PB-1

Útok SPA na implementace šifer RSA a AES, vliv šifrovacího algoritmu. Srovnání algoritmů umocňování Square and Multiply a Montgomery ladder u RSA.

Vlastnosti signálu

Spotřeba logických hradel **závisí na aktivitě** obvodu -- **intenzitě výpočtu** a **vnitřních hodnotách**.

Komponenty bodu P v průběhu spotřeby:

- P_{op}: operačně závislá (typicky využito SPA)
- P_{data}: datově závislá (typicky využito DPA)
- $P_{\text{el. noise}}$: elektronický šum
- P_{const}: konstantní komponenta

$$P_{\text{total}} = P_{\text{op}} + P_{\text{data}} + P_{\text{el. noise}} + P_{\text{const}}$$

Pouze **část** $P_{\text{op}} + P_{\text{data}}$ je **využitelná pro útok:**

$$P_{\text{op}} + P_{\text{data}} = P_{\text{exploitable}} + P_{\text{sw. noise}}$$

($P_{\text{sw. noise}}$ - *switching noise* = datově nebo operačně závislá komponenta, která ale není využitelná zvolenou metodou útoku)

$$P_{\text{total}} = P_{\text{exploitable}} + P_{\text{sw. noise}} + P_{\text{el. noise}} + P_{\text{const}}$$

SPA

Simple Power Analysis (SPA): Analýza jednoho průběhu spotřeby během kryptografické operace podél celé časové osy

Modulární umocňování

RSA dešifrování nebo podpis: $x = |c^d|_n$

d = dešifrovací klíč

c = šifrový text k dešifrování / data k podpisu

n = modul

x = dešifrovaný otevřený text / podpis

Square and Multiply:

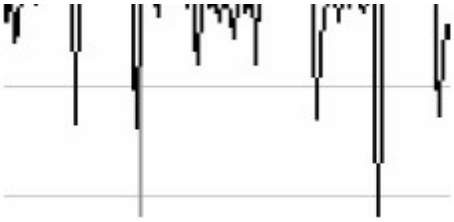
```
d = d[k-1] ... d[0] # binární zápis d, d[0] = LSb
k = length(d)
x = 1

for i = k-1 ... 0:
    x = x ^ 2 (mod n) # square
    if d[i] = 1:
        x = x * c (mod n) # multiply
return x
```

Example: $|2^{11}|_{15} = 8$

i	d	x	OP
3	$\underline{1}011$	$1^2 = 1$ $1 \cdot 2 = 2$	S M
2	$1\underline{0}11$	$2^2 = 4$	S
1	$10\underline{1}1$	$4^2 = 1$ $1 \cdot 2 = 2$	S M
0	$101\underline{1}$	$2^2 = 4$ $4 \cdot 2 = 8$	S M

Power side channel:



$\Rightarrow$ zranitelné SPA --lze **vyčíst hodnotu d -- privátního klíče**

Montgomery ladder

RSA dešifrování/podpis: $x = |c^d|_n$

```
d = d[k-1] ... d[0] # binární zápis d, d[0] = LSb
k = length(d)
R0 = 1
R1 = c

for i = k-1 ... 0:
    if d[i] = 1:
        R0 = R0 * R1 (mod n)
        R1 = R1 ^ 2 (mod n)
    else:
        R1 = R0 * R1 (mod n)
        R0 = R0 ^ 2 (mod n)
return x = R0

(invariant: R1 = c * R0)
```

Pro $d_i = 0$ i $d_i = 1$ se provedou stené operace $\Rightarrow$ **odolnost vůči SPA**
Nejde o universální řešení: stále náchylné např. na cache-collision attacks

SPA na AES rozvrhování klíče

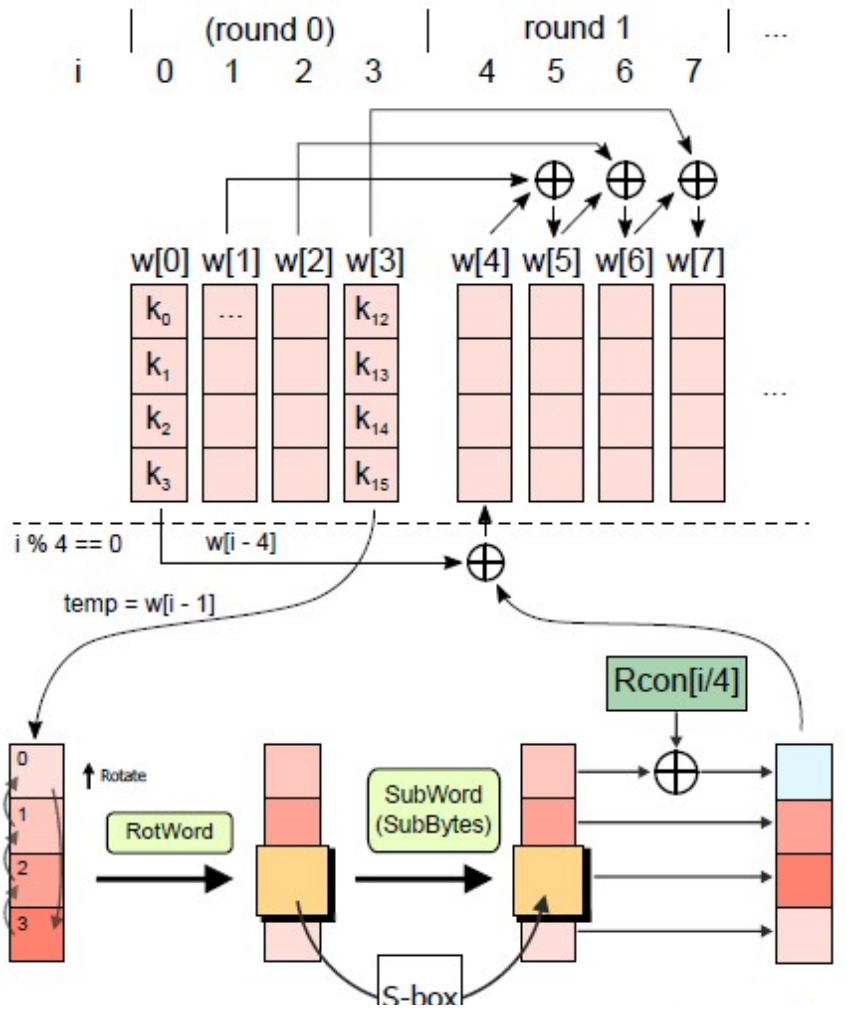
Hlavní klíč zapsán v matici $4 \times N_k$, kde $N_k = \frac{\text{velikost klíče v bitech}}{4 \cdot 8}$
AES-128: $N_k = 4$
AES-192: $N_k = 6$
AES-256: $N_k = 8$

Prvních N_k slov je použito **tak, jak jsou.** Poté se klíč **expanduje:**

- Poslední sloupec (=slovo) matice se **orotuje 1 byte nahoru k LSB** (***RotWord***)
- Orotované slovo se **substituuje** byte po bytu AES SBOXem (***SubWord***)
- K výsledku je přixorována **rundovní konstanta** (***Rcon***): $\mathrm{Rcon}[i] = \{02\}^{i-1}$, kde i je číslo rundy. Pouze první byte je změnen.
- Předchozími kroky vznikne **dočasné slovo** (sloupec) t'
- Klíč příští rundy k' se spočítá (AES-128):
  $k'_{r,0} = k_{r,0} \oplus t'_r$
  $k'_{r,i} = k'_{r,i-1} \oplus k_{r,i}$
  kde $i \in \{1, ..., N_k\}$ je sloupec klíčové matice a $r \in \{1, ..., 4\}$ je řádek klíčové matice

Expanze jinak ve zkratce:

- Vznik prvního sloupce jednoho expandovaného klíče:
  - Aplikace RotWord, SubWord, Rcon na poslední sloupec aktuálního klíče
  - XOR s prvním sloupcem aktuálního klíče
- Každý další sloupec expandovaného klíče:
  - XOR předchozího expandovaného sloupce a aktuálního neexpandovaného sloupce (např. $\text{nový 2. sloupec} = \text{nový 1. sloupec} \oplus \text{starý 2. sloupec}$)



SPA útok: využití Hammingovy váhy (HW) k odhalení tajného klíče

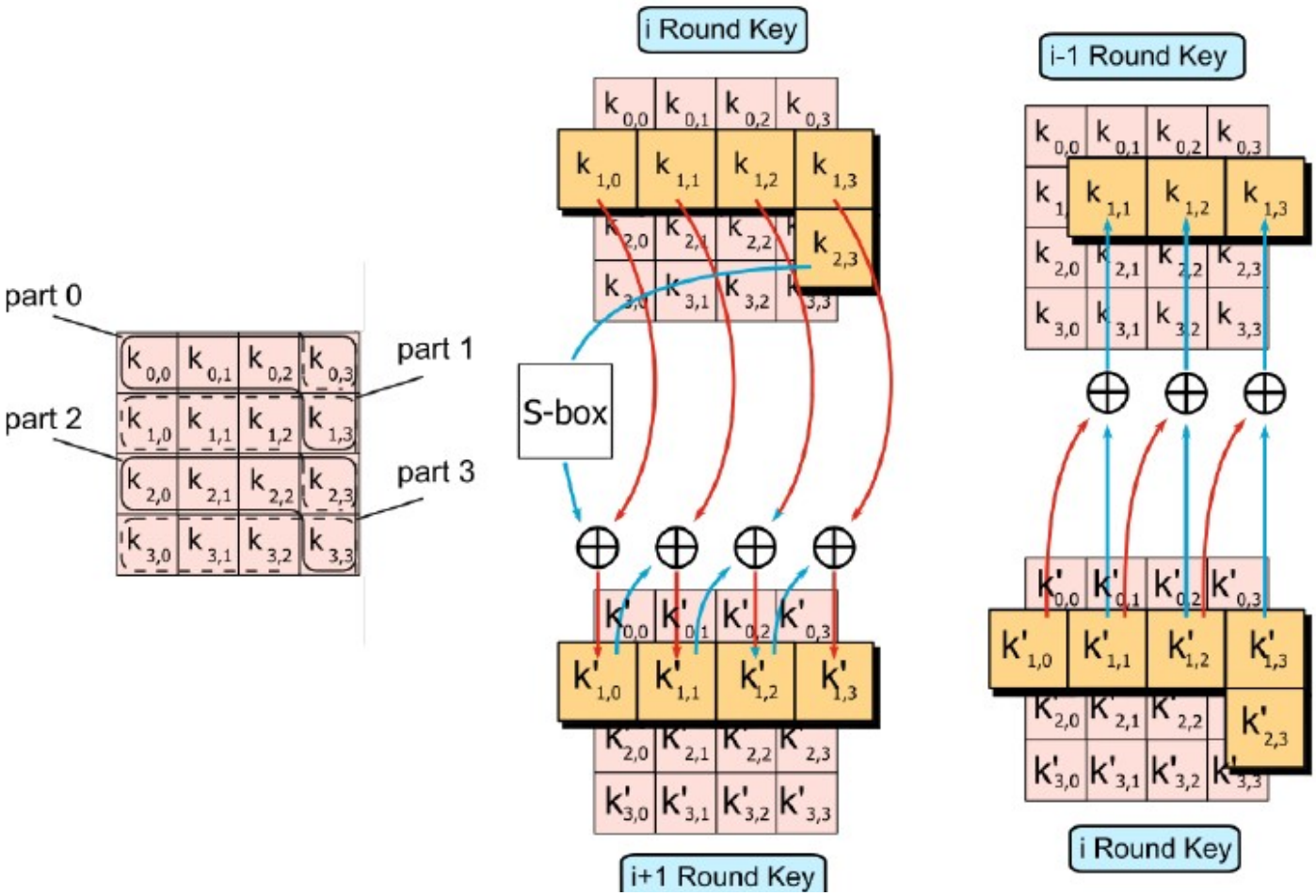
HW hlavního klíče: příliš málo informací

$\Rightarrow$ zjistit HW **bitů expandovaného klíče,** omezit prohledávací prostor

Ideálně změřit všech $11 \cdot 16 = 176$, lze použít i méně

Útok:

- Expandovaný klíč se dá rozdělit na čtyři pětibytové části (dělení na obrázku níž)
- Z jedné pětibytové části lze spočítat následující 4 byty (pro $i \in \{1, ...N_k\}, r \in \{1, 2, 3, 4\}$):
  $k'_{r,0} = k_{r,0} \oplus \mathrm{SubBytes}(k_{r+1,3})$
  $k'_{r,i} = k'_{r,i-1} \oplus k_{r,i}$
- Z pětibytové části lze spočítat i část předchozího klíče:
  $k'_{r,i} = k'_{r,i-1} \oplus k_{r,i}$
- Tento postup lze opakovat (= deexpandovat odhad klíče), ale s každou iterací je známo méně a méně bytů dalšího klíče



Příklad průběhu útoku:

Naměřeno 22 HW nějakých bytů expandovaného klíče

- Zvolit hodnoty 5 bytů tak, aby odpovídaly naměřeným HW
- Spočítat všechny bajty, které lze z těchto 5 bytů odvodit postupem uvedeným výše
- Zkontrolovat HW odvozených bytů:
  - Pokud všechny sedí s naměřenými hodnotami, zapamatovat si těchto 5 bytů jako kandidáty na klíč, jinak zahodit
- Opakovat pro všechny možné hodnoty
  - Worst-case: $70^5 = 1,6 \cdot 10^9$ opakování
- Opakovat pro další pětici

Výše uvedený postup útoku sníží počet kandidátů na klíč na minimum. Pokud se změří HW všech $4 \cdot 11 = 88$ rundovních klíčů, zbyde pouze několik málo možných hlavních klíčů.