MI-PB-6

Druhy postranních kanálů, časovací útoky. Útoky meet-in-the-middle. Útoky na formátování a doplnění zpráv.

Postranní kanály

Postranní kanál: nežádoucí výměna informací mezi kryptografickým modulem a jeho okolím, která není součástí jeho zamýšlené funkce.

Obchází matematické principy kryptografických operací

Využívají slabiny specifické SW nebo HW implementace

Narušují předpoklady bezpečného fungování šifer (X je tajné, Y je nonce, ...)

Tajný klíč často získán po částech -- **omezení prohledávacího prostoru**

Typy postranních kanálů:

- **Časový:** doba trvání operace záleží na tajných datech
- **Chybový:** vrácený chybový kód závisí na tajných datech
- **Odběrový (proudový, výkonový):** spotřeba závisí na vnitřních hodnotách během šifrování a tajných datech
- **Elektromagnetický:** podobný odběrovému, může zahrnovat i optický, infračervený, tepelný nebo radiofrakvenční
- **"Sociální":** využití uživatelova chování

Časovací útoky

Přihlášení do systému

- kontrola uživatelského jména
  - špatné $\rightarrow$ vypsat chybu
- kontrola hesla

- špatné $\longrightarrow$ vypsat chybu

Při tomto přihlášení lze měřit čas a zjistit, jestli jméno existuje nebo ne

Porovnání polí:

```
for(i = 0; i < n; i++){
    if(a[i] != b[i])
        return false;
}
return true;
```

Měření doby trvání odhalí počet správných položek na začátku.

Správný způsob:

```
c = 0;
for(i = 0; i < n, i++){
    c |= a[i]^b[i]
}
return !c;
```

Montgomeryho obor celých čísel

Výhody: rychlé modulární násobení -- klasicky se nejdřív spočítá $a \cdot b$ a pak se od výsledku odečítají násobky m, aby se vyrušily nechtěné vyšší bity.
Montgomeryho násobení přičítá násobky m aby se vyrušily spodní bity a výsledek byl násobkem $R > m$. Nižší bity se zahodí, aby výsledek byl $< 2m$. Ten se dále zredukuje odečtením na $< m$.
Mezivýsledky jsou malé, vždy redukované $\mod m$

- **x je m-reziduum čísla y:** $x = |y \cdot R|_m$, kde R je nejmenší mocnina základu poziční soustavy větší než m
- **Montgomeryho obor celých čísel:** množina m-reziduí
- **Montgomeryho součin:** a, b m-rezidua, $c = |a \cdot b \cdot R^{-1}|_m$
- **Zpětná transformace násobením:** a je m-reziduum c, pak $c = |a \cdot 1 \cdot R^{-1}|_m$ (násobení m-rezidua s 1)
- **Montgomeryho redukce násobením:** a je celé číslo, c je m-reziduum a, pak $c = |a \cdot R^2 \cdot R^{-1}|_m = |a \cdot R|_m$

- **Montgomeryho redukce:** výpočet $T \cdot R^{-1} \mod m$ bez dělení m (dělí se R, kde často $R = 2^n$, takže rychlejší)

```
funkce redc(T):
    m' =  -m ^ (-1) (mod R)
    k = ( T (mod R) * m' ) (mod R)
    t = (T + k * m) / R
    if t >= m:
        return t-m
    else:
        return t
```

V kódu výše je $T + k \cdot m$ dělitelné R, protože $T + k \cdot m \equiv T + (((T \mod R) \cdot m') \mod R) \cdot m \equiv T + T \cdot m' \cdot m \equiv T - T \equiv 0 \pmod R$.
Pro útok důležitá **závěrečná datově závislá podmínka**

RSA časovací útok

RSA: $x = |c^d|_n$

Ve **square-and-multiply** použito **Montgomeryho násobení:** $c = |abR^{-1}| \Rightarrow c =$ `redc(a*b)`

d_{k-1} je vždy 1. Jak získat d_{k-2}?

Pokud bit $d_{k-2} = 1$, bude během square-and-multiply vypočítána hodnota $c^2 \cdot c$ *(square, pak multiply)*

Pokud bit $d_{k-2} = 0$, operace $c^2 \cdot c$ se neprovede.

Průběh útoku na *multiply*:

- Orákulum O o zprávě c
  - $O(c) = 1 \Leftrightarrow$ `redc` je při výpočtu $(c^2) \cdot c$ se závěrečným odečtením
  - $O(c) = 0 \Leftrightarrow$ `redc` je při výpočtu $(c^2) \cdot c$ bez závěrečného odečtení
- Dvě množiny zpráv:
  - C_1 obsahuje zprávy, kde $O(c) = 1$
  - C_0 obsahuje zprávy, kde $O(c) = 0$
- Změřit časy RSA pro C_0, C_1:
  - množina F_1 obsahuje RSA časy pro zprávy z C_1, závisí na d_{k-2}
  - množina F_0 obsahuje RSA časy pro zprávy z C_0, nezávisí na d_{k-2}

- Pokud se časy F_0 a F_1 významně liší, $d_{k-2} = 1$, jinak $d_{k-2} = 0$
- Se známým d_{k-2} lze opakovat pro d_{k-3} atd.

Pokud $d_{k-2} = 1$, pak **proběhne *multiply*,** které obsahuje **datově závislé odčítání,** a časy F_1 tak budou zaručeně delší než časy F_0

Pokud $d_{k-2} = 0$, ***multiply* vůbec neproběhne** a rozdělení na C_1 a C_0 tak pozbývá smysl (bude se jevit jako **náhodné**)

Průběh útoku na *square*:
Pokud $d_{k-2} = 1$ a Square-and-Multiply zastaví těsně před podmínkou rozhodující o provedení *multiply* (vnitřní stav je c_{temp}), následující operace jsou *multiply* ($c_{temp} \cdot c$) a v další iteraci potom *square* ($(c_{temp} \cdot c)^2$).

Pokud $d_{k-2} = 0$ a Square-and-Multiply zastaví těsně před podmínkou rozhodující o provedení *multiply*, následující operace je pouze *square* (c_{temp}^2) v následující iteraci.

- Dvě orákula: O_1 pro $d_{k-2} = 1$ a O_0 pro $d_{k-2} = 0$:
  - $O_1(c) = 1 \Leftrightarrow (c^2 \cdot c)^2$ obsahuje závěrečné odečtení
  - $O_1(c) = 0 \Leftrightarrow (c^2 \cdot c)^2$ neobsahuje závěrečné odečtení
  - $O_0(c) = 1 \Leftrightarrow (c^2)^2$ obsahuje závěrečné odečtení
  - $O_0(c) = 0 \Leftrightarrow (c^2)^2$ neobsahuje závěrečné odečtení
- Množina zpráv C rozdělena do dvou podmnožin:
  - C_{11}: c, kde $O_1(c) = 1$
  - C_{10}: c, kde $O_1(c) = 0$
- Množina zpráv C rozdělena do dalších dvou podmnožin:
  - C_{01}: c, kde $O_0(c) = 1$
  - C_{00}: c, kde $O_0(c) = 0$
- Měření časů F_i pro podmnožiny C_i
- Pokud jsou časy F_{11}, F_{10} odlišné víc než F_{01} a F_{00}, potom $d_{k-2} = 1$, jinak $d_{k-2} = 0$

Pokud $d_{k-2} = 1$, dává smysl dělení do podmnožin C_{11}, C_{10}, protože se provede *multiply* a v něm se projeví časový rozdíl při závěřečném odčítání. Rozdělení do podmnožin C_{01}, C_{00} se bude jevit náhodně.

Pokud $d_{k-2} = 0$, dává smysl dělení do podmnožin C_{01}, C_{00}, protože se multiply neprovede, ale při square může odečtení taky nastat.

Tento způsob funguje i pro variantu algoritmu **square-and-multiply-always** (dummy násobení pro $d_i = 0$) -- sice se při $d_{k-2} = 0$ násobí, ale násobí se dummy proměnná a ne c, takže sledované provedení/neprovedení odečtení tím není ovlivněno

Opatření proti časovacím útokům na RSA:

- kompletně datově nezávislé operace (složité např. kvůli cache)
- **blinding (maskování):** zamaskování skutečné vnitřní hodnoty využitím aritmetických vlastností šifry
    - Multiplikativní homoorfismus RSA: $(a \cdot b)^d \equiv a^d \cdot b^d \pmod m$
    - RSA podpis: $s = |x^d|_n$
    - RSA podpis s maskou m: $s_m = |(x \cdot m^e)^d|_n = |x^d \cdot m|_n$, odmaskování: $s = |s_m \cdot m^{-1}|_n$

Meet In The Middle

Funkce $f : \{1, 2, ..., N\} \to \{1, 2, ..., N\}$, útočník zná y, kde $y = f(x)$ a potřebuje zjistit x.

Útok hrubou silou: Zkoušení x dokud není nalezeno takové, že $y = f(x)$
Složitost: $O(N) = O(2^n)$

Dvojité šifrování: $c = E''_{K_2}(E'_{K_1}(p))$
Hrubá síla: Získat $K_1 \in \{0,1\}^n, K_2 \in \{0,1\}^m$, složitost $O(2^{n+m})$

Nalezení kolize uprostřed výpočtu:
$c = E''_{K_2}(E'_{K_1}(p))$
$D''_{K_2}(c) = E'_{K_1}(p)$

Průběh Meet-in-the-middle:

- Předpočítat všechny možné $E'_{K_1}(p)$ pro plaintext a uložit páry $(K_1, E'_{K_1}(p))$ do tabulky T
- Najít K_2 takové, že $D''_{K_2}(c)$ je v tabulce T: $\exists K_1 : (K_1, D''_{K_2}(c)) \in T$
- Nalezená dvojice (K_1, K_2) je hledaný klíč -- ověřit na ostatních šifrových textech a pokud nesedí, hledat dál

Složitost MITM:
Předpočítání: $O(2^n)$ šifrování a $O(2^n)$ paměti
Hledání: $O(2^m)$ šifrování a hledání v tabulce
Výsledná složitost: $O(2^n + 2^m)$

Útok na 3DES: (56 bitů každý klíč)

$c = E_{K_3}(D_{K_2}(E_{K_1}(p)))$
$D_{K_3}(c) = D_{K_2}(E_{K_1}(p))$
$E_{K_2}(D_{K_3}(c)) = E_{K_1}(p)$

- Předpočítat všechny možné $E_{K_1}(p)$ a uložit do tabulky T
- Hledat (K_2, K_3) t.ž. $\exists K_1 : (K_1, E_{K_2}(D_{K_3}(c))) \in T$
- (K_1, K_2, K_3) je pravděpodobně správný klíč

Složitost 3DES MITM:
Předpočítání: $O(2^{56})$ šifrování a $O(2^{56})$ paměti
Hledání: $O(2^{112})$ šifrování a hledání v tabulce
Výsledná složitost: $O(2^{56} + 2^{112}) = O(2^{112})$

Útok na RSA bez paddingu:
Náhodná 64b zpráva M šifrovná RSA s veřejným klíčem (n, e): $C = |M^e|_n, M < 2^m$
Cíl: zjistit m-bitovou zprávu M při znalosti C, n, e
Velká šance, že M je složené číslo:
$M = M_1 M_2, M_1 < 2^{m_1}, M_2 < 2^{m_2}$ (např. pokud $m = 64$, je 18 % šance, že je M je složeno ze dvou 32-bitových čísel a 29 % šance, že z nejvýš 33-bitových)
Multiplikativní homomorfismus RSA:
$C = |M^e|_n = |(M_1 M_2)^e|_n = |M_1^e M_2^e|_n$ a proto:
$|C M_2^{-e}|_n = |M_1^e|_n$
Možno hledat kolizi -- není zaručeno, že útok vždy uspěje, ale je velká šance, že ano

- Předpočítat všechny možné $|M_1^e|_n$, uložit do tabulky páry $(M_1, |M_1^e|_n)$
- Najít M_2 t.ž. $(M_1, |C M_2^{-e}| : n) \in T$

Složitost MITM na RSA:
Není nutné ukládat celou $|M_1^e|_n$, stačí uložit dost bitů, aby bylo možné rozlišit různé zprávy -- mělo by stačit $2\max(m_1, m_2)$ nejnižších bitů.
$\Rightarrow$ potřeba $2^{m_1+1}\max(m_1, m_2)$ paměti a $2^{m_1} + 2^{m_2}$ modulárních umocnění

Padding

Kryptografická techinika umožňující dosáhnout určitých vlastností prostřednictvím vhodných úprav otevřeného textu.

Použití:

- maskování vlastností (délka, struktura)
- doplnění na potřebnou délku (blok)
- ochrana proti pozměnění

Požadavky:

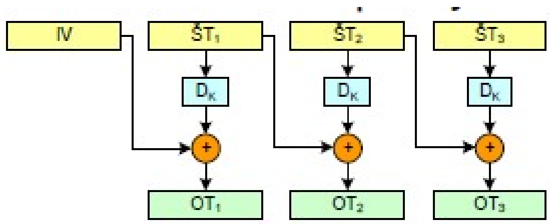
- musí jít opět odstranit
- měl by být deterministický
- nesmí dát útočníkovi informaci o OT

Druhy:

- **Bitový:** 1000...00 připojeno za konec OT
  - Odstranění: postup od konce OT, dokud se nenarazí na 1
- **Bytový:** různá schémata:
  - **ANSI X.923:** byty `00 00 ... 00 x`, kde `x` je počet `00`
  - **ISO 10126:** Obdoba ANSI, ale místo `00` jsou náhodná čísla (není deterministický)
  - **PKCS7:** Byty `x x ... x`, kde `x` je délka paddingu
  - **ISO/IEC 7816-4:** Obdoba bitového, ale na úrovni bytů

Padding Oracle

Server zpracovávající zprávy šifrované blokovou šifrou v režimu CBC



Napřed testuje **padding,** potom ověřuje **integritu zprávy**, umožňuje odlišit chybu paddingu od chyby integrity.

Útočník zná $IV, ŠT_1, ŠT_2, ŠT_3$ a použité schéma paddingu (zde PKCS). **Chce zjistit OT_2.**

- **Poslední byte OT_2** má skutečnou (neznámou) hodnotu a_1
- **Útočník si tipne,** že hodnota posledního bytu OT_2 je b_1
- Útočník k poslednímu bytu $ŠT_1$ **naxoruje** $b_1 \oplus 0x01$
  - Poslední byte OT_2 bude po dešifrování $a_1 \oplus b_1 \oplus 0x01$
- **Útočník pošle $IV, ŠT_1, ŠT_2$** serveru a sleduje odpověď:

    - chyba integrity $\Rightarrow a_1 = b_1$ (útočník se trefil)
      Poslední byte OT_2 je po dešifrování $a_1 \oplus a_1 \oplus 0x01 = 0x01$ (platný padding)
    - chyba paddingu $\Rightarrow$ útočník b_1 neuhodl, volí další hodnotu
- **Útočník opakuje** pro předposlední byte OT_2 (skutečná hodnota je a_2):
    - Tipne, že má hodnotu b_2 a na $\check{S}T_1$ naxoruje 2 byty:
        - $\text{předposlední byte} = \text{předposlední byte} \oplus b_2 \oplus 0x02$
        - $\text{poslední byte} = \text{poslední byte} \oplus a_1 \oplus 0x02$
- **Útočník opakuje** pro všechny bloky

Obrana proti padding oracle:

Upravit server tak, aby útočník nemohl rozlišit chybu paddingu od chyby integrity.

Použít šifru, která padding nepotřebuje (proudovou)

Použít šifrovací mód, které nepotřebuje padding (CTR)

Padding u asymetrických šifer

Malleabilita: útočník může pozměnit OT předvídatelným způsobem, aniž by prolomil šifru

ElGamal:

- x soukromý klíč
- $(m, g, c = g^x)$ veřejný klíč
- p otevřený text, y nonce zvolená odesílatelem
- $\check{S}T = (d, e)$, kde $d = |g^y|_m, e = |c^y \cdot p|_m$
- Děšifrování:
  $c^y = |(g^x)^y|_m = |(g^y)^x|_m = |d^x|_m$
  $p' = |(c^y)^{-1} \cdot e|_m = |(c^y)^{-1} \cdot c^y \cdot p|_m = p$

Malleabilita ElGamal:
$\check{S}T' = (d, s \cdot e)$
$\Rightarrow p' = s \cdot p$

Malleabilita RSA:
$\check{S}T' = s^e \cdot \check{S}T$
$\Rightarrow p' = |\check{S}T'^d|_n = |(s^e \cdot p^e)^d|_n = s \cdot p$
Pomůže **padding:** útočník změní i jeho hodnotu a oběť ji dokáže detekovat (u deterministického paddingu)

Prolomení ŠT u RSA:

Pokud modul $2^{2047} < n < 2^{2048}$, slabý veřený klíč (např. 3), a dostatečně malá zpráva $p \approx 2^{512}$, zpráva po umocnění neptřeteče modul: $\check{S}T = |p^e|_n \approx |2^{1536}|_n = 2^{1536}$

Útočník může zachytit $\check{S}T$ a spočítat $\sqrt[3]{\check{S}T}$ v **reálné aritmetice** (= rychle) a dostane p

Pomůže **padding:** malé p zvětší tak, že po umocnění (i na malý exponent) přeteče n

Determinističnost RSA:

Šifrování je $\check{S}T = |p^e|_n$, kde n je konstantní a e taky

$\Rightarrow p_1 = p_2 \Leftrightarrow \check{S}T_1 = \check{S}T_2$

Padding pomůže, ale pouze pokud jeho část je náhodná