MI-KRY – Advanced Cryptology

Side channel attacks

Ing. Jiří Buček

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra informační bezpečnosti

bucekj@fit.cvut.cz

Lecture outline

- Side channel
- Side channel types
- Timing attacks
- Power analysis attacks
- Power models

Side channel

- An unwanted way of information exchange between a crytpographic module and its surroundings, which is not part of its normal function
- Bypasses the mathematical principle of encryption (signing, ...)
- Exploits the weaknesses of a specific physical or software implementation
- Violates the assumptions of secure function of the cipher (X is secret, Y is *nonce*, . . .)
- Often enables to get the key piecewise $\rightarrow$ limits search space

The term "side channel", also "side channel leakage" is an abstraction of information interchange commonly used in the term "side channel attack".

Side channel types

- Timing side channel
  - Time of operation depends on secret data (message, key ...)
  - Information leaks in time that can be measured
- Error side channel
  - Returned error code depends on secret data (message, key ...)
  - See padding oracle attack
- Power side channel
  - Power consumption depends on internal values during encryption, thus on secret data
  - Simple, Differential, ... Power Analysis (SPA, DPA, ...) – see MIE-HWB, MIE-BHW
- Electromagnetic side channel
  - Similar to power side channel
  - Can also include optical, IR, heat, radiofrequency
- „Social channel“
  - Exploits behavior of the user

Timing side channel 1

- Example 1: System login
  1. User name check
  2. User name wrong? $\rightarrow$ Return „User name or password error“ [1]
  3. Password check
  4. Password wrong? $\rightarrow$ Return „User name or password error“
- Password not checked for nonexistent users
  - From the timing we can uncover whether a user name exists or not, even though this information is not given in the error message
  - We can try user names, then the most frequent passwords (dictionary attack)

[1] The error message does not specify which of the two is erroneous, the attack would be too easy. But this does not suffice.

Timing side channel 2

- Example 1 (improved): System login
  1. User name check
  2. User name right? $\to u = 1$ **else** $u = 0$
  3. Password check
  4. Password right? $\to p = 1$ **else** $p = 0$
  5. **if** $(u$ **or** $p) = 0 \to$ Return „User name or password error“

Timing side channel 3

- Example 2: Password, session key, hash, etc. checking
- Compare arrays

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

- We can guess individual bytes – each additional right byte prolongs time to response

Timing side channel 4

- Correct way to compare arrays:

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

RSA timing side channel 1

- RSA timing attack
  - RSA decryption: $x = \left|c^d\right|_n$, signing is similar. d is the private exponent.
  - Usually done using square and multiply:

$k = \textbf{length}(d)$
$x = c$
for $i = k - 2$ **downto** 0
    $x = \left|x^2\right|_n$
    if $d_i = 1$ **then**
        $x = \left|x \cdot c\right|_n$
return x

- Detour – power side channel – Simple power analysis (SPA) – see MI-HWB, MI-BHW

RSA timing side channel 2

- Assume Montgomery multiplication used $c = |abR^{-1}|_n$
- Often used due to performance, $R = 2^k$, fast division by R, mod R
- Principally: $c = \mathrm{REDC}(ab)$, where REDC is the Montgomery reduction
- Multiplication and reduction are often interleaved e.g. by bits

function REDC(T)

1.	$m := \|\|T\|_R N'\|_R$	prepare $m = \|-n^{-1}T\|_R$
2.	$t := (T + mn)/R$	$T + mn$ divisible by R
3.	**if** $t \geq n$ **then return** $t - n$ **else return** t	final subtraction

$N' = |-n^{-1}|_R$

- Line 3 is important – final subtraction, data dependent

RSA timing side channel 3

- $d_{k-1} = 1$ always. How to get d_{k-2}?
- Introduce an oracle O about the message c:
  - $O(c) = 1$, when $(c^2) \cdot c$ is with final subtraction
  - $O(c) = 0$, when $(c^2) \cdot c$ is without final subtraction
- Create 2 sets of messages C_1 and C_2
  - C_1 contains messages where $O(c) = 1$
  - C_2 contains messages where $O(c) = 0$
- Measure times of RSA for these message sets:
  - F_1 contains RSA times for messages from C_1, depend on d_{k-2}
  - F_2 contains RSA times for messages from C_2, independent of d_{k-2}
- If the times from F_1 a F_2 differ significantly, then $d_{k-2} = 1$ (at a suitable statistical test's significance level)
- Else $d_{k-2} = 0$. Knowing d_{k-2}, we can repeat the process for d_{k-3} etc. and get the whole key.
- Problem – what is a significant difference between F_1 and F_2, so that $d_{k-2} = 1$? (we have nothing for comparison)

RSA timing side channel – attacking squaring (1)

- Variant 2: Attack on squaring.
- 2 oracula about the message c: O_1 for $d_{k-2} = 1$, O_2 for $d_{k-2} = 0$
  - $O_1(c) = 1$, when $(c \cdot c^2)^{\mathbf{2}}$ is with final subtraction
  - $O_1(c) = 0$, when $(c \cdot c^2)^{\mathbf{2}}$ is without final subtraction
  - $O_2(c) = 1$, when $(c^2)^{\mathbf{2}}$ is with final subtraction
  - $O_2(c) = 0$, when $(c^2)^{\mathbf{2}}$ is without final subtraction
- The message set C is divided into C_1, C_2, and again to C_3, C_4
  - C_1 contains messages where $O_1(c) = 1$
  - C_2 contains messages where $O_1(c) = 0$
  - C_3 contains messages where $O_2(c) = 1$
  - C_4 contains messages where $O_2(c) = 0$
- Measure times F_i for messages form C_i. Division into F_1 and F_2 assumes $d_{k-2} = 1$, while division into F_2 and F_3 assumes $d_{k-2} = 0$.
- If times from F_1, F_2 differ more, than those from F_3, F_4, then $d_{k-2} = 1$, else $d_{k-2} = 0$. Knowing d_{k-2}, we can repeat for d_{k-3}, and get the whole key.

RSA timing side channel – attacking squaring (2)

Assume we have guessed a few first bits correctly. Ex: $d = \mathbf{110}1101001$, $k = 10$. The first few iterations were $\rightarrow$ (see table)

i	k_i	x
–	**1**	c
8		c^2
8	**1**	$c^2 \cdot c = c^3$
7	**0**	$(c^3)^2 = c^6$
6		$(c^6)^2 = c^{12} = c_{temp}$
6	1?	$?c^{12} \cdot c = c^{13}?$
5		$?^2$

Computing until the unknown multiplication, we have an intermediate value $c_{temp} = c^b = c^{12} = c^{1100_2}$. If the bit $d_i = 1$, the operations will be (using Montgomery)

1. multiply c_{temp} by c (part of iteration i)
2. square the result (part of iteration $i+1$)

Execute multiplication, then determine if the **squaring** needs final subtraction $\rightarrow$ oracle $\mathbf{O_1}$ divides all messages into C_1 (final subtraction), and C_2 (no final subtraction).

RSA timing side channel – attacking squaring (3)

If the bit $d_i = 0$, no multiplication will occur, the operation is

- square the result: c_{temp}^2 (part of iteration $i + 1$)

Determine if the squaring needs final subtraction $\rightarrow$ oracle $\mathbf{O_2}$ divides all messages into C_3 (final subtraction), and C_4 (no final subtraction).

One of these separations makes sense, depending on the actual value of $d_i \rightarrow$ compare the separations

- If the difference between C_1 and C_2 is more important than between C_3 and C_4, then decide $d_i = 1$
- Otherwise decide $d_i = 0$

RSA timing side channel 5

- Variant 2 also works for "square and multiply *always*":

$k = \textbf{length}(d)$
$x = c$
for $i = k - 2$ **downto** 0
    $x = \left|x^2\right|_n$
    if $d_i = 1$ **then**
        $x = |x \cdot c|_n$
    else
        $dummy = |x \cdot c|_n$ (discard the result)
return x

- Countermeasures
  - Completely data-independent execution time (not trivial to implement due to e.g. caches)
  - Blinding (a type of masking, see next)

Countermeasures – masking

- In digital signatures, usually called *Blinding*
  - Exploit the arithmetic properties of the cipher to obscure the real internal value, so that the attacker is unable to guess, what is being computed.
  - E.g. RSA multiplicative homomorphism:

$$(a \cdot b)^d \equiv a^d \cdot b^d \pmod{n}$$

  - Choose a mask m, compute $|m^e|_n$ (e is public)
  - RSA signature normally: $s = \left|x^d\right|_n$
  - RSA signature of x with a mask m:

$$s_m = \left|(x \cdot m^e)^d\right|_n = \left|x^d \cdot m\right|_n$$

  - RSA signature is then unmasked m:

$$s = \left|s_m \cdot m^{-1}\right|_n$$

  - In advance, we can prepare a random m , $|m^e|_n$, $\left|m^{-1}\right|_n$.

Differential Power Analysis (recap from MIE-BHW)

- Choose an **intermediate value** that depends on data and key $v = f(d, k)$
- Measure **power traces** $t_{i,j}$ while encrypting data d_i
  - for each data block i and time j
- Build a matrix of **hypothetical intermediate values** inside the cipher
  - for every trace i and key k: $v_{i,k} = f(d_i, k)$
- Using a power model, compute the matrix of **hypothetical power consumption**
  - for every trace i and key k: $h_{i,k} = \mathrm{hwmodel}(v_{i,k})$
- **Statistically evaluate** which key hypothesis k best matches the measured power at each individual time j (across all traces i).
  - There are multiple methods, we will focus on the correlation coefficient $r_{\{h_{\dots,k}\},\{t_{\dots,j}\}}$.

Correlation power analysis

- A type of Differntial Power Analysis (DPA),
- sometimes denoted CPA
- Uses the (Pearson's) correlation coefficient as the statistical method
  - Determines the measure of linear relationship between two random variables
- $\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sqrt{\mathrm{var}\,X\,\mathrm{var}\,Y}} = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{\mathrm{E}(X-\mu_X)^2\,\mathrm{E}(Y-\mu_Y)^2}} = \frac{\mathrm{E}(XY)-\mathrm{E}(X)\,\mathrm{E}(Y)}{\sqrt{\mathrm{E}(X^2)-\mathrm{E}^2(X)}\sqrt{\mathrm{E}(Y^2)-\mathrm{E}^2(Y)}}$
- Because we have a limited sample, use the point estimate
- $r_{X,Y} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}}$

Power Models

- Models how power consumption depends on intermediate value
- Single bit model – eg. $\mathrm{hwmodel}(v) = \mathrm{LSB}(v)$
  - In real circuits, not every bit affects consumption the same way
- Hamming weight model – consumption depends on value's Hamming weight
  - $\mathrm{hwmodel}(v) = \mathrm{HW}(v)$ Applies mainly to processors, buses with pullups etc.
  - Not suitable for an ideal CMOS circuit, where the consumption depends on the *change* of value
- Hamming distance model – consumption depends on Hamming distance of two values
  - $\mathrm{hwmodel}(v) = \mathrm{HD}(v, v') = \mathrm{HW}(v \oplus v')$, where v' is the previous value in the circuit (in a register, on a bus, in a logic network)
  - Works also for ASICs, FPGAs
  - If v' is constant, or depends on v, or has a significantly non-uniform distribution, Hamming weight model works usually, too.
- Zero value model - consumption is different for zero / nonzero value – $\mathrm{hwmodel}(v) = (v == 0)$

Bibliography

J.F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, "A practical implementation of the timing attack", at www.cs.jhu.edu/ fabian/courses/CS600.624/Timing-full.pdf

Mangard, S., Oswald, E., Popp., T.: Power Analysis Attacks – Revealing the secrets of smart cards, Springer, 2007, ISBN 0-387-30857-1