


Introduction to Side Channel Attack

顏嵩銘 (Sung-Ming Yen)

國立中央大學 資訊工程系所
密碼與資訊安全實驗室

Laboratory of Cryptography and Information Security 
<http://www.csie.ncu.edu.tw/~yensm/lcis.html>

Tel : (03) 4227151 Ext- 35316

Fax : (03) 4222681

E-Mail : yensm@csie.ncu.edu.tw



- An *old* research topic
“exponentiation algorithm design”
meets a very *new* issue
“hardware physical cryptanalysis”.
- Exponentiation algorithm design
becomes again a *young* research topic!



Smart Card Physical Security

- Smart card usage
 - Smart cards are considered as *tamper-proof* devices.
 - ❖ internal secret can not be unauthorized accessed??
- Components in a typical smart card
 - 8-bit CPU (or more advanced CPU) + co-processor
 - ROM (to store program)
 - EEPROM (to store secret key)
 - RAM (to store temporary data during computation)
 - Serial I/O
 - Cryptosystem library (DES, RSA, SHA, etc)



- Computation in a "real" physical device
 - Take time
 - Consume power (and make radiation)
 - Depend on the reliability of hardware



- Even a **provably secure** cryptosystem may suffer the attacks when they are **implemented!**
- Physical attack makes the design of *good* implementation algorithms for cryptosystem be *difficult*



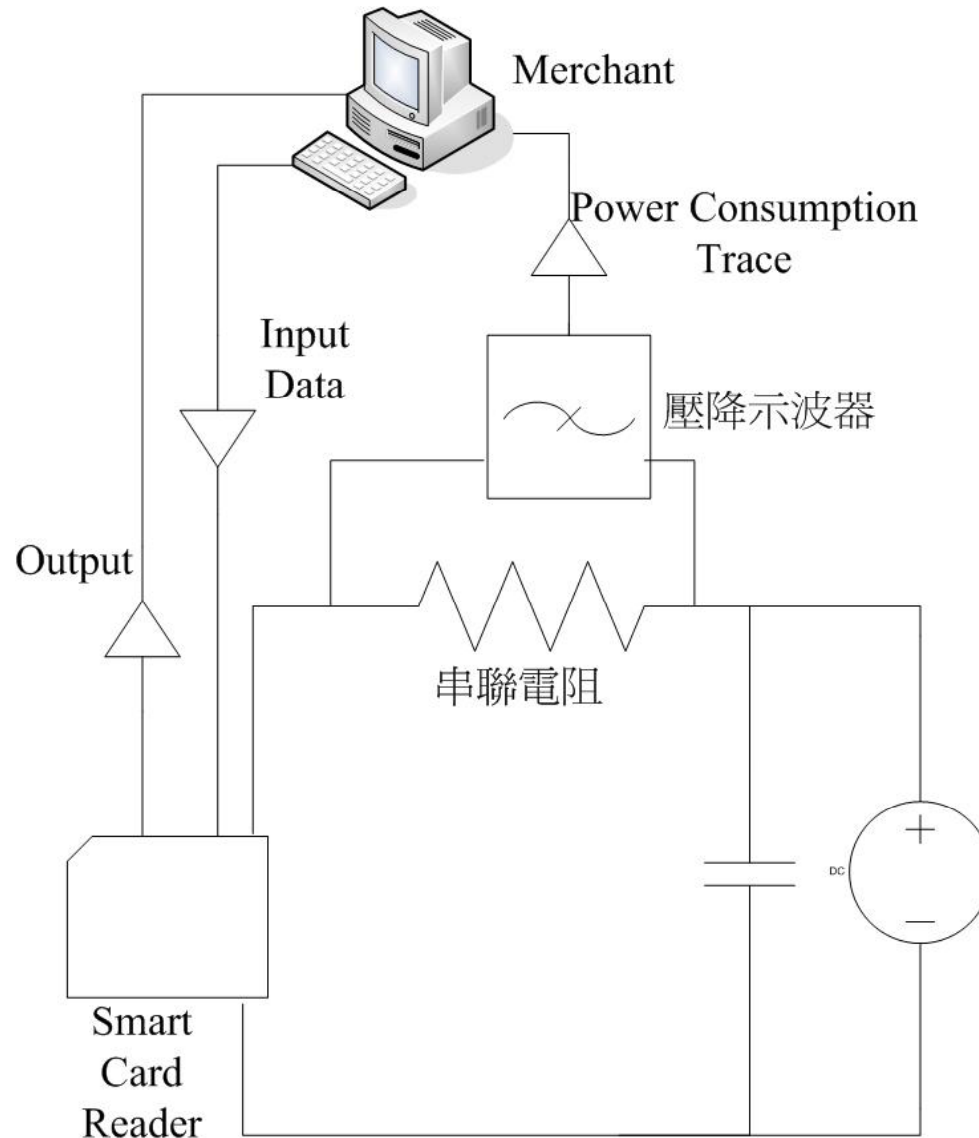
- Some reported physical cryptanalysis:
 - Side-channel attack:
 - ❖ Timing attack [[Kocher 1996](#)]
 - ❖ Power monitoring attack [[Kocher 1998 & Yen 1997 Dec. \(SPA\) in ITRI's report](#)]
 - ❖ IC card radiation attack [[Quisquator 2000](#)]
 - Fault based attack [[Boneh96 \(Bellcore\) and a series of works](#)]
 - Response based attack (a special kind of Side-channel attack) [[Yen 1998](#)]
 - Attack exploiting countermeasures [[Yen 2001 in ITRI's report; in KISA's report; in KNU's report; in a cooperation with Gemplus](#)]
 - Hybrid attack (conventional+physical attacks) [[Phan&Yen 2003; some recent results](#)]



Power Cryptanalysis (on RSA)

- Introduction to Simple Power Analysis (SPA)
- Introduction to Differential Power Analysis (DPA)
- SPA & DPA to RSA implementations

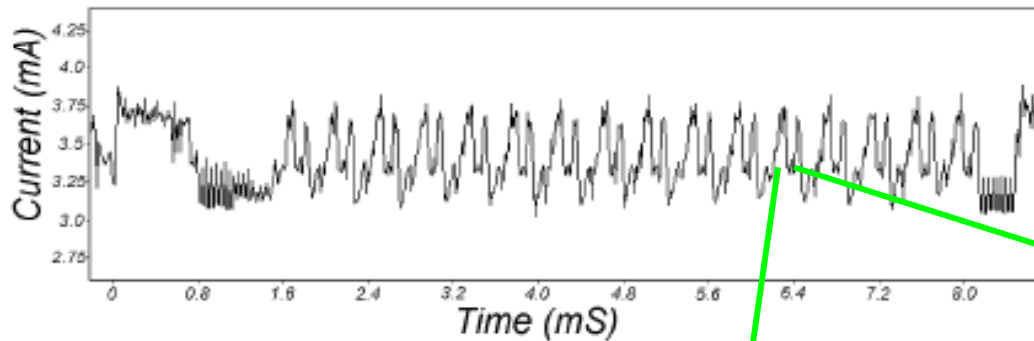
Power Attack Equipment



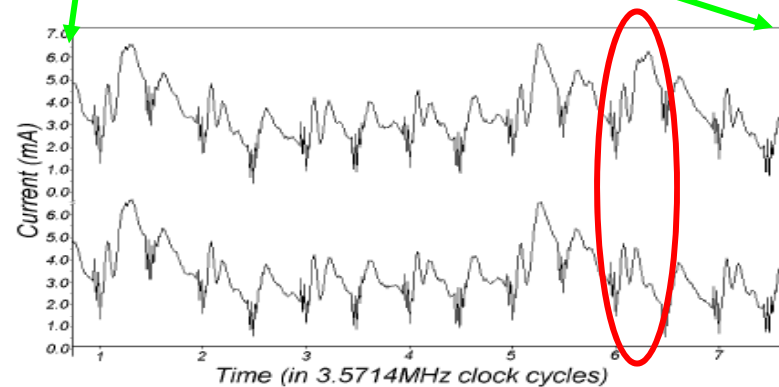
Power Attack (I)-- SPA

Exploiting difference between two *instruction* power consumptions

- **Simple power analysis (SPA)**



By observing a device's power consumption directly





- **Simple power analysis (SPA):**
 - observe on one or a few collected power traces
 - try to identify the occurrence of
 - ❖ an instruction execution or
 - ❖ a specific operand/data accesswhich are driven by a part of the *secret key*

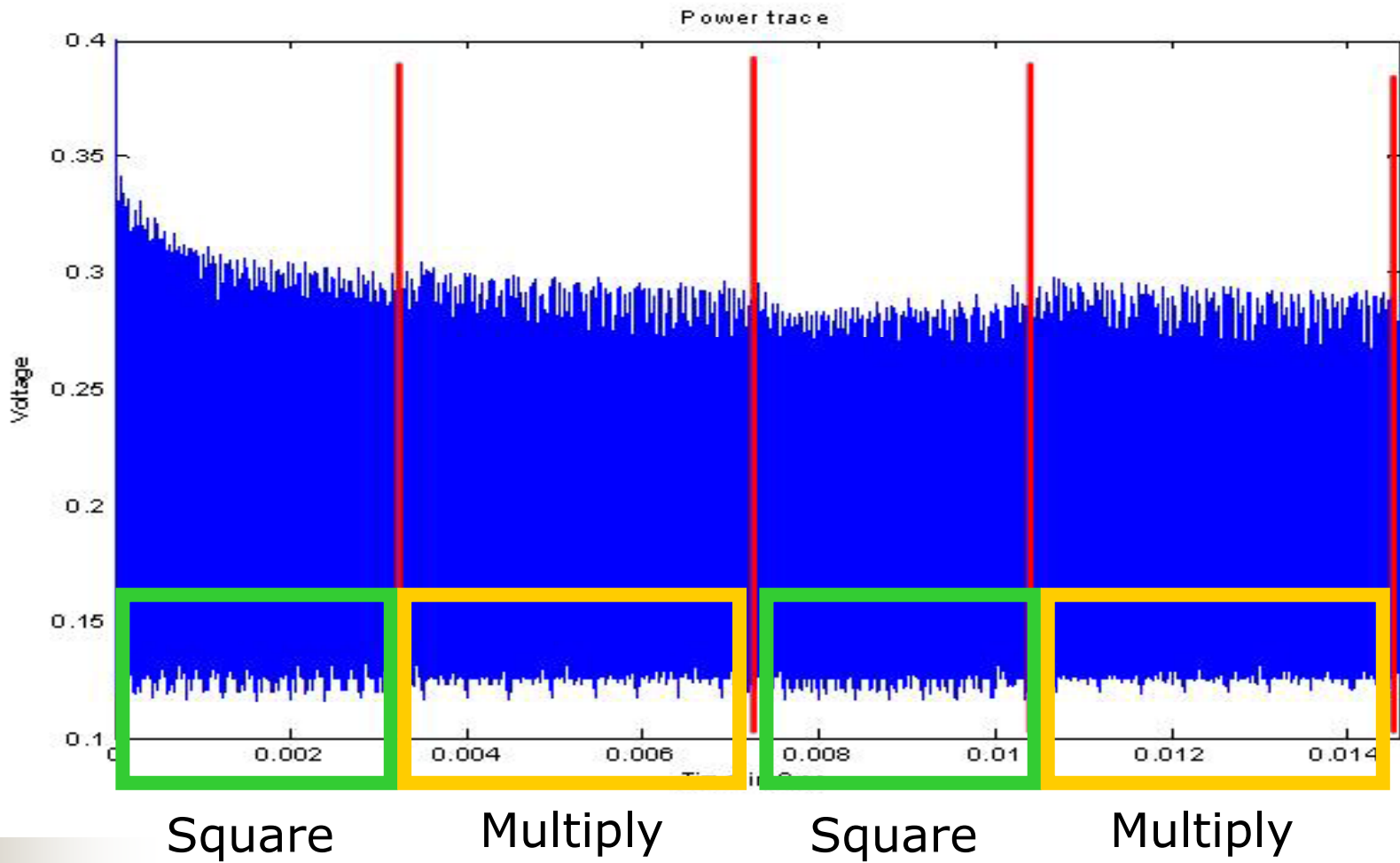


SPA by Observing Conditional Jump

- Exponentiation algorithm (g^d) for RSA:

```
01:  $R = 1$ 
02: for  $i = (k-1)$  downto 0
03:    $R = R^2$ 
04:   if ( $d_i == 1$ ) then  $R = R \times g$ 
05: return  $R$ 
```

- The “key dependent” conditional jump is vulnerable to simple power attack (SPA)
 - ❖ $R=R^2$ then $R=R \times g$ or $R=R^2$
 - ❖ $R=R \times g$ needs to access 2 operands





Countermeasures Against SPA

Hardware-level countermeasures

- To add *random power* consumption inside the chip
- To let the chip be *constant power*
 - To let each instruction be constant power

Software-level countermeasures

- To remove *key dependent* conditional jump
 - Regular process/algorithm



Countermeasure Against SPA

-- Solution (1)

- L-to-R binary exponentiation without conditional jump (*square-multiply always*)

```
01:  $R[1] = 1$ 
02: for  $i = (k-1)$  downto 0
03:    $R[1] = R[1]^2$ 
04:    $R[d_i] = R[1] \times g$ 
05: return  $R[1]$ 
```

- introduce dummy operation
 - ❖ dummy operation: $R[0] = R[1] \times g$
 - ❖ unfortunately **insecure** against *computational* safe-error (**C safe-error**)
- [Yen-ICISC 01]



Countermeasure Against SPA

-- Solution (2)

- *Montgomery ladder*--without conditional jump & without dummy operation (due to Montgomery & Yen 95 [IEE] & Joye&Yen-CHES 02)

```
01: R[0] = 1; R[1] = g
02: for i = (k-1) downto 0
03:   R[1-di] = R[0] × R[1]
04:   R[di] = R[di] × R[di]
05: return R[0]
```

- when $d_i = 0$ or 1
 $R[0] \rightarrow g^t \rightarrow g^{2t}$ or g^{2t+1}
 $R[1] \rightarrow g^{t+1} \rightarrow g^{2t+1}$ or g^{2t+2}



- Idea behind the Montgomery ladder -- consider only operation on the exponent

- Ex: $d: (1,1,1)_2 = 7$

$$(r_0, r_1) = (0, 1) \rightarrow (1, 2) \rightarrow (3, 4) \rightarrow (7, 8)$$

$$(1, 1, 1)_2 = \underbrace{(1, 1, 0)_2}_3 + \underbrace{(0, 0, 1)_2}_1 = 3 \times 2 + 1$$

$$\diamond 3 \times 2 + 1 = (3 + 3) + 1 = 3 + (3 + 1) = 3 + 4 = 7$$

- Ex: $d: (1,1,0)_2 = 6$

$$(r_0, r_1) = (0, 1) \rightarrow (1, 2) \rightarrow (3, 4) \rightarrow (6, 7)$$

$$(1, 1, 0)_2 = \underbrace{(1, 1, 0)_2}_3 + (0, 0, 0)_2 = 3 \times 2 + 0$$

$$\diamond 3 \times 2 = 3 + 3 = 6$$



- without any dummy operation
 - **secure** against *computational* safe-error attack (C safe-error) [Yen-ICISC 01]
 - still **insecure** against *memory* safe-error attack (M safe-error) [Yen-IEEE 2000]
 - ❖ enhancement is available [Joye&Yen-CHES 02]



Power Attack (II)-- DPA

- **Differential power analysis (DPA):**
 - by analyzing many executions of the same algorithm with different random inputs
 - ❖ many power traces (a few thousands or more) collected to enhance the SNR
 - design a *selection function* **S**

value of some bit(s) of a specific intermediate step
= **S**(part of key, input)

- ❖ 1-bit **S** partitions power traces into 2 groups
- ❖ 2-bit **S** partitions power traces into $2^2=4$ groups
- a *differential step* is performed
- to verify a guess on the secret key by using selection function **S** and the differential step



Sample DPA on RSA

- Given $(d_{k-1}=1, d_{k-2}, \dots, d_{t+1})$ and to derive d_t
 - suppose we already have $d_{k-1}=1$
 - what's the **next** computation after the first
 $R = R \times R$
 - to guess d_{k-2} and to monitor on MSB of R

Algorithm for m^d

```

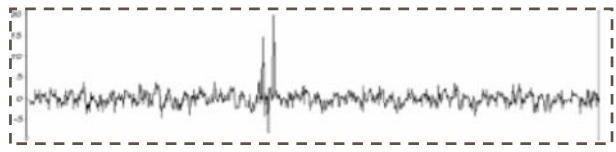
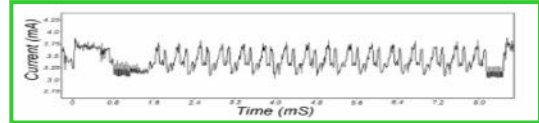
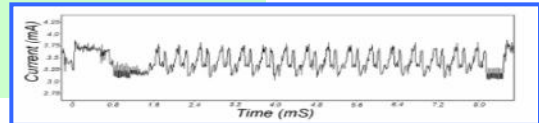
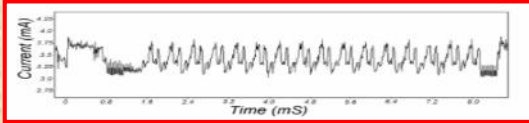
01:  $R = m$ 
02: for  $i = (k-2)$  downto 0
03:    $R = R \times R$ 
04:   if  $(d_i == 1)$  then  $R = R \times m$ 
05: return  $R$ 

```

Use MSB of R as "selection function" S

m_1
 m_2
 m_3

0 1 1



Find average power trace P_L

Find average power trace P_H



Countermeasures Against DPA

Hardware-based countermeasures

- To add *random power* consumption inside the chip.
BUT it is not so useful!
- To let the chip be *constant power*
 - to let each instruction+operand be constant power

Software-based countermeasures

- To *randomize/mask operands* within the cryptographic operations
 - a post-mask process is also required
- To *randomize/mask secret key*
- To *randomize* instruction execution order
 - hardware level *non-deterministic processor*
 - software level *non-deterministic software*



Examples of software countermeasures

- To randomize *operand* (e.g. input message) in order to *disable* cryptanalysis -- to blind operand (de-blinding at the end)
 - *Blinding technique for RSA system*
$$m \rightarrow m * r^e \quad (r \text{ is a random integer})$$
$$S' = (m * r^e)^d ; \quad S' * r^{-1} \rightarrow S$$
- To randomize *secret key* -- to blind key
 - for example, in RSA, let $d' = d + k * \varphi(n)$, then $m^d \equiv m^{d'} \pmod{n}$
 - to randomly *recode* secret key (better performance than above one)



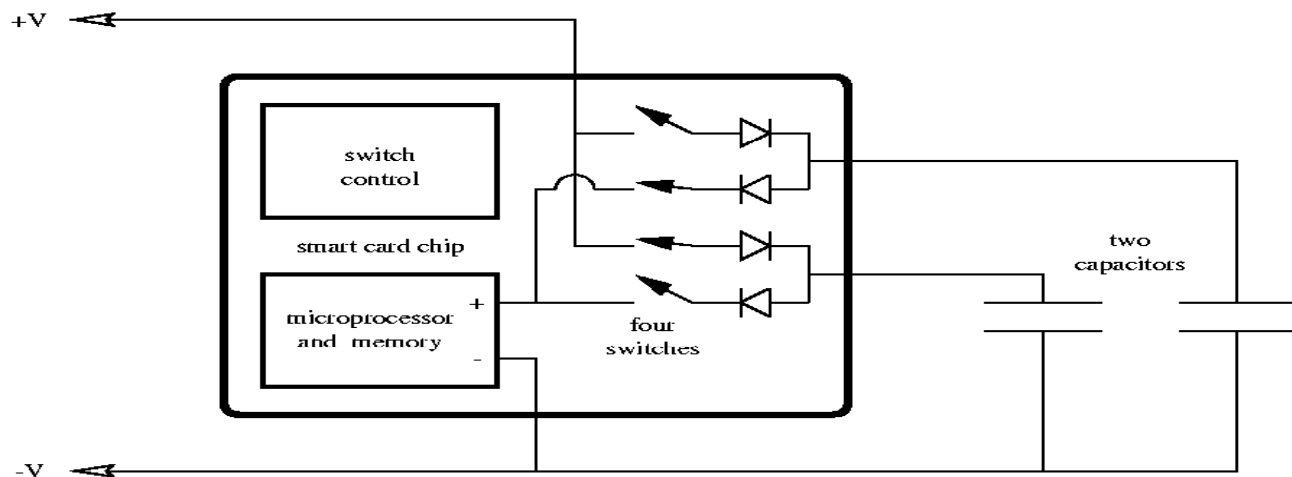
Overall Secure Implementations against Power Attacks

- To achieve uniform power consumption by **re-designing circuits** (e.g. differential logic)
- **Software**-based **randomization** techniques
- **Hardware**-based random **noise** injection
- To replace external power supply by an internal battery
 - alternative solution: **rechargeable battery**



Example of internal battery: [Shamir CHES2000] Detached Power Supply as countermeasure

- A patent owned by Shamir
- Basic idea: (non-mathematical solution)
 - capacitors as **power isolation** element
 - 1st capacitor disconnected from **external power** & **supply power** to the **chip**
 - meanwhile, 2nd capacitor disconnected from the chip & **recharged** by the external power





Fault Cryptanalysis (on RSA)

- Introduction to Fault Attack
- Fault Attack on RSA with CRT

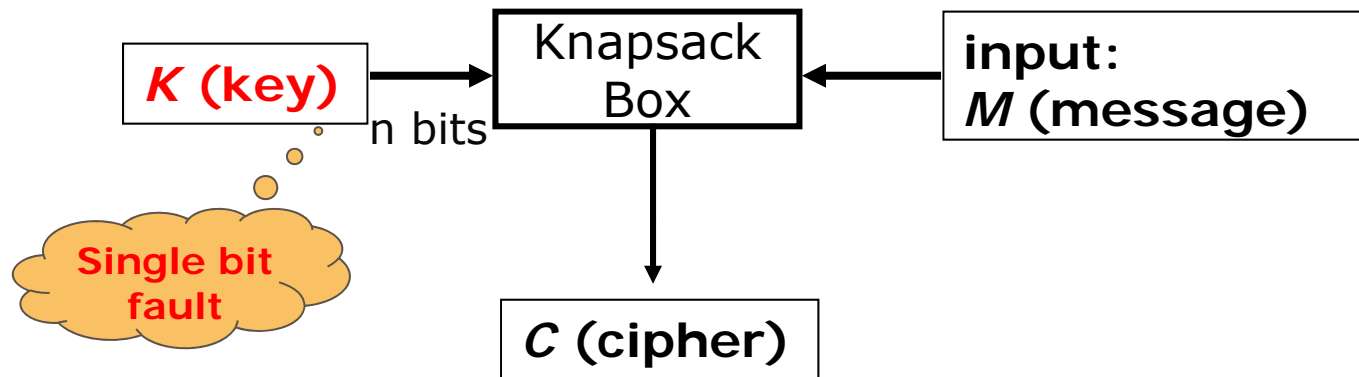


Basic Idea of Hardware Fault Attack (FA)

Exploiting the relationship between a correct and an erroneous results.

$K = \{K_1, K_2, \dots, K_n\}$
 K_i are all binary bits

$M = \{M_1, M_2, \dots, M_n\}$
 M_i are all integers



$$\Sigma K_i * M_i$$

- Given M , C' (with **single** bit fault in K), and C , the attacker can obtain one bit of K_i .



Countermeasures against fault attack

- **Not** to send out incorrect result (by checking)
 - cryptography **dependent** checking
 - cryptography **independent** detection

- But, sometimes, “response” is enough to attack the implementation
 - ==> **Safe-error** attack [**Yen 1999 & 2001**]
 - see the following 2 kinds of attacks
 - M-safe error & C-safe error attacks



Memory Safe-error attack [Yen 1999]

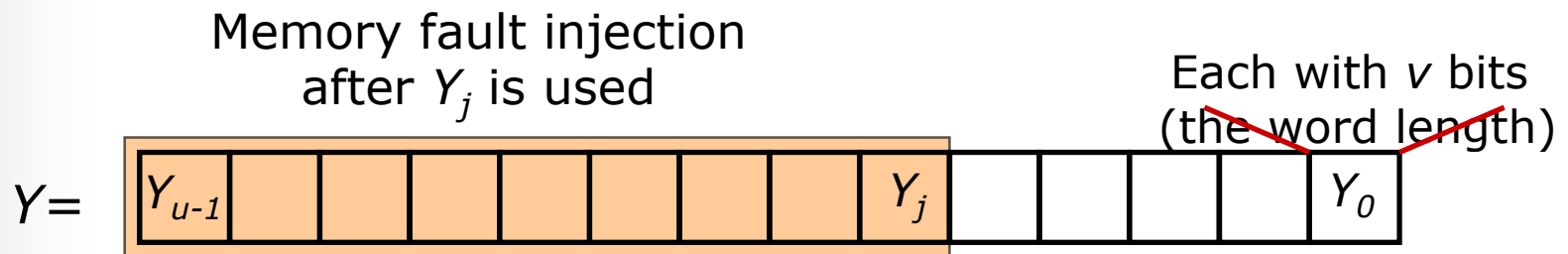
- L-to-R binary exponentiation $g^d \bmod n$

```
01:  $R = 1$ 
02: for  $i = (k-1)$  downto 0
03:    $R = \text{Mul}(R, R)$ 
04:   if ( $d_i == 1$ ) then  $R = \text{Mul}(g, R)$ 
05: return  $R$ 
```

Mul(input: X, Y ; output: T) % pass by address

```
01:  $T = 0$ 
02: for  $j = (u-1)$  downto 0
03:    $T = (T * 2^v + X * Y_j) \bmod n$ 
```

- insecure against **M** safe-error





Computational Safe-error attack

[Yen 2001]

- L-to-R binary exponentiation without conditional jump (*square-multiply always*)

```
01: R[1] = 1
02: for i = (n-1) downto 0
03:   R[1] = R[1]2
04:   R[di] = R[1] × g
05: return R[1]
```

If $d_i = "0"$
then $R[0]$ is
ignored
(safe-error)

Any
computational
fault in ALU

- introduce dummy operation to avoid SPA
 - ❖ dummy operation: $R[0] = R[1] \times g$
 - ❖ **insecure** against C safe-error [Yen-ICISC 01]



Preliminary Background of CRT-based Cryptanalysis

- RSA speedup with CRT
- The CRT-based cryptanalysis



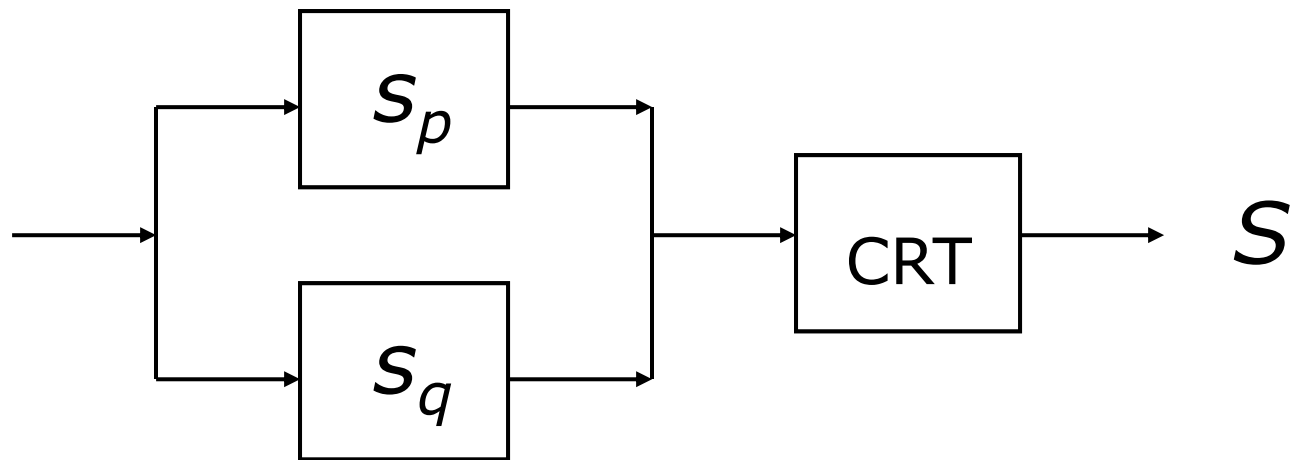
RSA Speedup with CRT

RSA speedup based on CRT:

- Given $p, q, (n=p*q), d,$ and $m,$
 $S=m^d \bmod n$ can be sped up by

$$s_p = (m \bmod p)^{d \bmod (p-1)} \bmod p$$

$$s_q = (m \bmod q)^{d \bmod (q-1)} \bmod q$$





CRT recombination algorithms:

- **Gauss's** CRT recombination
 - a standard representation but it takes more memory space & time

$$\begin{aligned} S &= \text{CRT}(s_p, s_q) \\ &= [(s_p \times q \times (q^{-1} \bmod p)) + s_q \times p \times (p^{-1} \bmod q)] \bmod n \\ &= [s_p \times X_p + s_q \times X_q] \bmod n \end{aligned}$$

- **Garner's** CRT recombination
 - **widely used** because it takes *fewer memory space & time*

$$\begin{aligned} S &= \text{CRT}(s_p, s_q) \\ &= \{s_q + [(s_p - s_q) \times (q^{-1} \bmod p)] \times q\} \bmod n \\ &= s_q + [(s_p - s_q) \times (q^{-1} \bmod p) \bmod p] \times q \end{aligned}$$



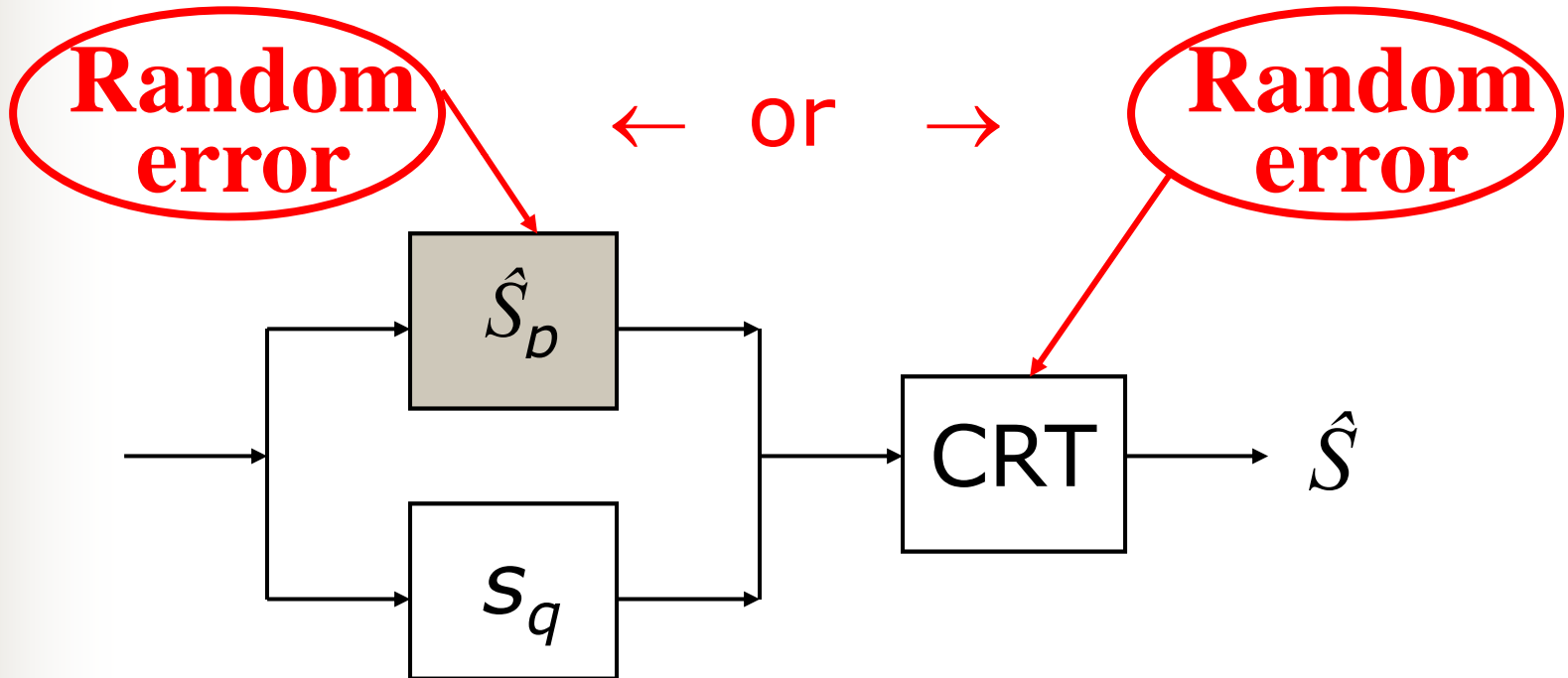
Fault Attack on RSA with CRT

Fault attack on the computation of s_p or s_q

- Given a faulty result of $\hat{S} = \text{CRT}(\hat{S}_p, s_q)$

$$q = \text{gcd}((\hat{S} - S) \bmod n, n)$$

$$q = \text{gcd}((\hat{S}^e - m) \bmod n, n)$$





Importance of CRT-based Attack

False alarm attack on RSA+CRT (a new proposal)

- The false alarm may be initiated by a malicious outside attacker
- False alarm attack
 - Consider that you are the administrator of a **CA** and received an **anonymous** e-mail claiming that a **wrong** signature/certificate has been produced. What will you do?)
 - **Denial of service attack**
- So, any potential CRT-based attack should be carefully considered.



Some Countermeasure and Possible Attacks

- Shamir's countermeasure
- Enhanced Shamir's countermeasure
- (Fault-tolerant computation platform)



Shamir's Countermeasure

- Shamir's countermeasure (a **patent**)
(extend modulus then reduce modulus)

$$s_{pr} = m_{pr}^{d_{pr}} \bmod p^*r \quad \blacksquare \quad \varphi(p^*r)$$
$$s_{qr} = m_{qr}^{d_{qr}} \bmod q^*r \quad = (p-1)^*(r-1)$$

where $m_{pr} = m \bmod p^*r$ & $d_{pr} = d \bmod \varphi(p^*r)$ & r is a random **prime**

- output S only if $(s_{pr} \bmod r) = (s_{qr} \bmod r)$
 $S = \text{CRT}(s_{pr}, s_{qr})$
 $= \text{CRT}(s_{pr} \bmod p, s_{qr} \bmod q)$



■ CRT-based attack on Shamir's method [Yen-ICISC 02]

- The approach
 - ❖ To produce **incorrect** \hat{S}_p but correct s_q
 - ❖ However, both s_{pr} and s_{qr} are **correct**, so Shamir's method **cannot** detect the attack.
- How to produce \hat{S}_p without being detected?
 - ❖ **computational** fault:
when modulo s_{pr} by p
 - ❖ **memory access** fault (on operand):
when accessing s_{pr} or p
 - ❖ **memory storing** fault (storing result):
when storing the result of $(s_{pr} \bmod p)$



Enhanced Shamir's Method

[Yen ICISC 02]

- Note: only operations for p are described

1) Compute $d_{pr} = d \bmod \varphi(pr)$ and $p_r = p^*r$
and **checks** whether

$$(d - d_{pr}) \bmod (p - 1) \stackrel{?}{=} 0 \quad \& \quad p_r \bmod p \stackrel{?}{=} 0$$

2) Compute $s_{pr} = m^{d_{pr}} \bmod p_r$

$$\text{and } s_p = s_{pr} \bmod p$$

3) **Checks** whether

$$s_{pr} \stackrel{?}{=} s_{qr} \pmod{r}$$

If correct, then compute $S = \text{CRT}(s_p, s_q)$.

4) **Checks** "before" sending out S :

$$S \stackrel{?}{=} s_{pr} \pmod{p} \dots\dots (\mathbf{A})$$

to prevent/detect:
(a) $\varphi(p^*r) \rightarrow \varphi(\mathbf{p}'^*r)$
(b) $p^*r \rightarrow \mathbf{p}''^*r$



- Analysis of Enhanced Shamir's method (A)
 - Checking in **Step (4)** can:
 - ❖ avoid attack during CRT recombination [Yen ICISC 01], i.e., $q^{-1} \bmod p$ storage attack
$$S =? s_p \pmod{p} \dots\dots (A1)$$
 - ❖ avoid the attack during finding s_p [Yen ICISC 02], i.e., **modulo p attack**
$$s_p =? s_{pr} \pmod{p} \dots\dots (A2)$$
 - But, Step (4) **cannot** detect an attack which corrupts the values of s_{pr} **prior to** s_p is computed.
 - ❖ This attack **can** however be detected by checking in **Step (3)**

$$s_{pr} =? s_{qr} \pmod{r}$$



- Analysis of Enhanced Shamir's method (B)
 - *Permanent* fault on the storage of d
 $d' \leftarrow d$
 - ❖ **Both** s_p and s_q will be **incorrect** (as \hat{S}_p and \hat{S}_q) and the CRT-based attack is not applicable on $\hat{S} = \text{CRT}(\hat{S}_p, \hat{S}_q)$.



- Analysis of Enhanced Shamir's method (C)
 - *Permanent* fault on p or q

Ex: $p^{\wedge} \leftarrow p$

❖ **Step (4)**: Checks before output S

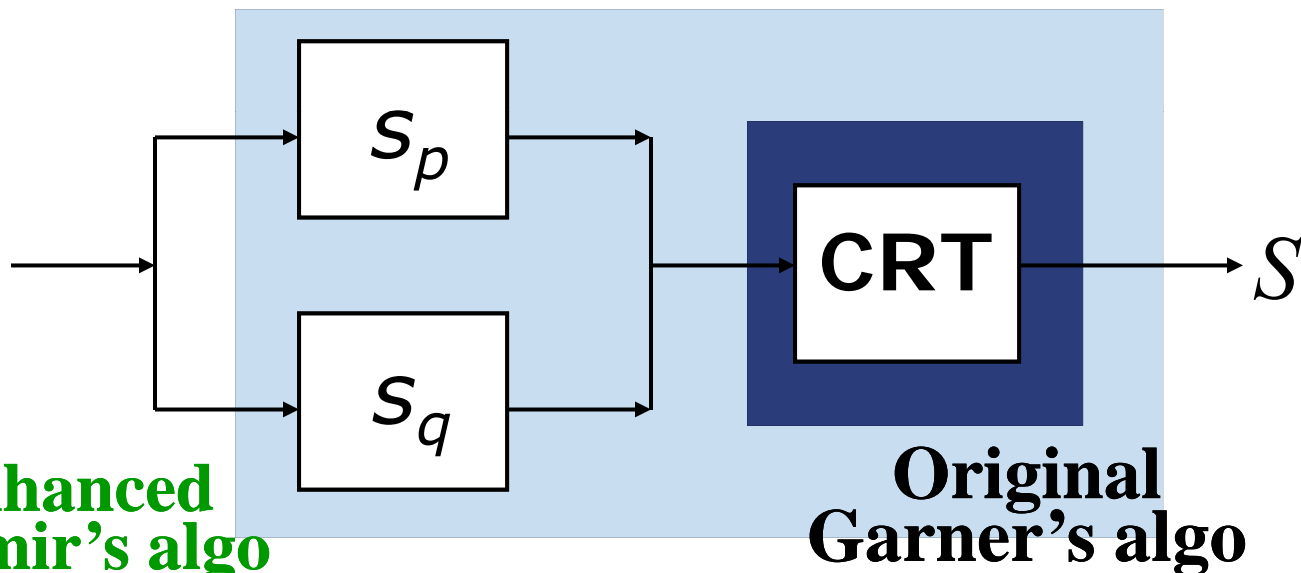
$$S \stackrel{?}{=} s_{pr} \pmod{p}$$

The **Step (4)** (in fact, the **A1 checking**) is itself “ p or q permanent fault” attack **immune**



Overall Suggestion

- This design can be immune against:
 - basic CRT-based factorization attack
 - modulo p attack
 - $(q^{-1} \bmod p)$ storage attack
 - p or q storage attack
 - d storage attack



**Enhanced
Shamir's algo
[Yen ICISC 2002]**

**Original
Garner's algo**



That is all ?

- There are much more to examine:
 - more fault models to check – computational & storage
 - fault on basic parameters & pre-computation results
 - CRT recombination formula to check
 - is it good to use so many checking procedures?



Some Remarks on RSA with CRT

- **Popularity of RSA with CRT:**
 - RSA+CRT is widely adopted for efficiency for either small devices and larger servers.
- **Pitfalls of RSA+CRT (fault attacks):**
 - Many pitfalls are found recently. Other pitfalls may still be found in the near future!
 - A single erroneous result is enough!
 - The false alarm attack may lead to the “denial of service” attack. (non-technical issue but it is very important!)
 - More “checking” procedures being used will lead to a **less reliable** countermeasure.
- **Conclusion:**
 - More research is still necessary.



Remarks on Further Research of Physical Cryptanalysis

- New attacks research
 - Detailed examination on the relationship between any format of *output*
 - ❖ power (energy)
 - ❖ timing
 - ❖ data
 - ❖ reliability and response, etcand any *internal information* of the cryptographic device.
- *Good* countermeasure design



Good countermeasure design

- What is a good countermeasure?
 - an **art** to counteract all/most existing attacks and with good performance
 - ❖ **too much overhead** on time and power (or silicon space) is **NOT acceptable**
 - ❖ **quick switch** from one countermeasure to another one without altering the **hardware**