

A Practical Countermeasure against Address-bit Differential Power Analysis

Kouichi Itoh, Tetsuya Izu and Masahiko Takenaka

Objective of our Work

- **A practical countermeasure against address-bit DPA**
- **Evaluation criteria of the power analysis countermeasures**

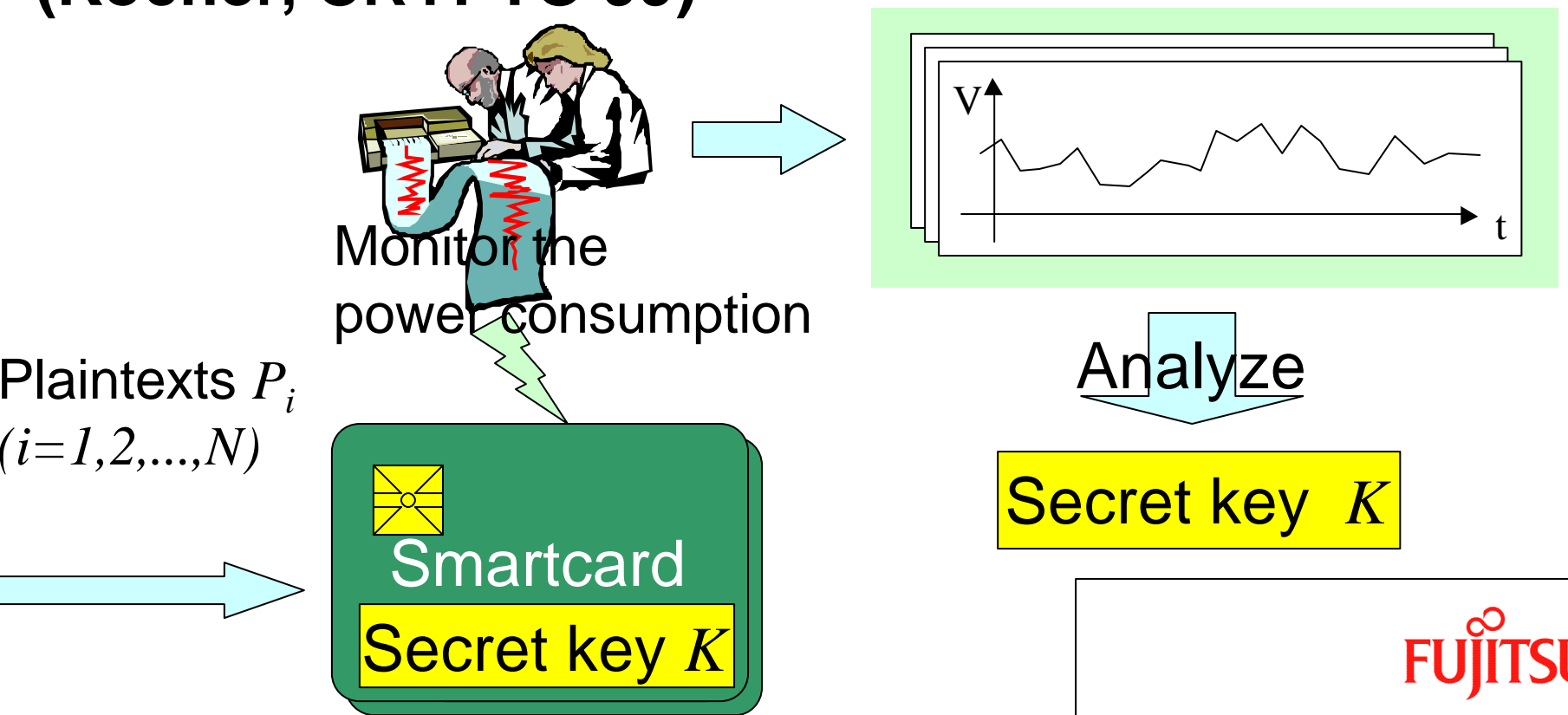
Contents

- **What is DPA?**
 - **Address-bit DPA (ADPA)**
- **Our countermeasure against ADPA**
 - **Experimental result**
- **Our evaluation criteria of countermeasures**
- **Conclusion**

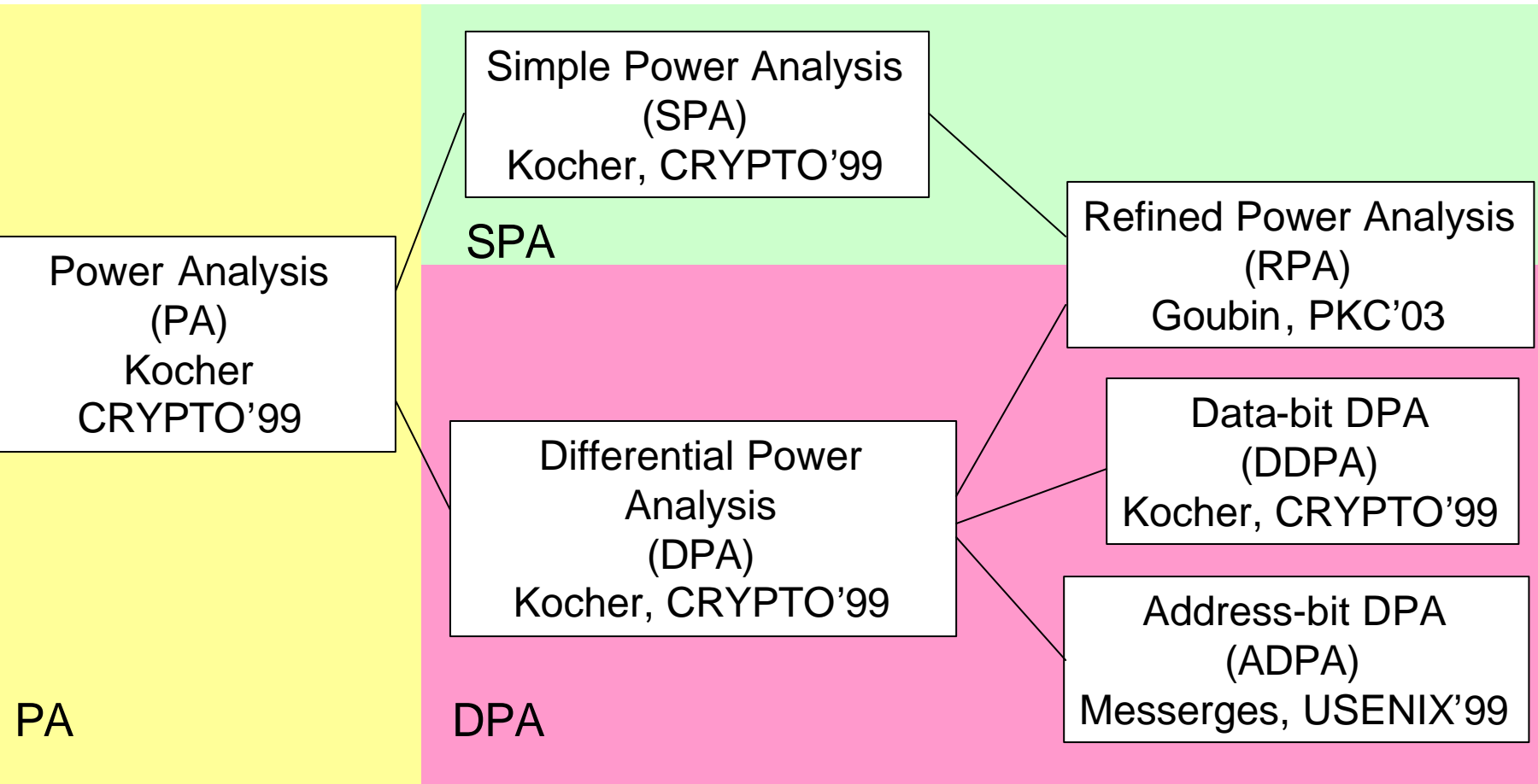
- Practical countermeasure against address-bit DPA

What is Power Analysis (PA)?

Analyze a secret key stored in the cryptographic device by monitoring its power consumption (Kocher, CRYPTO'99)



Overview of Power Analysis



ADPA in ECC

Itoh-Izu-Takenaka(CHES '02)

- Breaks SPA countermeasure + DDPA countermeasure!

Add-and-always method + Randomized Projective Coordinates

```
Q[0] = RPC(P), Q[1] = ECDBL(Q[0])
for i = n-2 downto 0 {
    Q[2] = ECDBL(Q[di])
    Q[1] = ECADD(Q[0], Q[1])
    Q[0] = Q[2-di], Q[1] = Q[1+di]
}
return invRPC(Q[0])
```

$d_i=0$

Address	Data
Q[0]	*****
Q[1]	*****
Q[2]	*****

$d_i=1$

Address	Data
Q[0]	*****
Q[1]	*****
Q[2]	*****

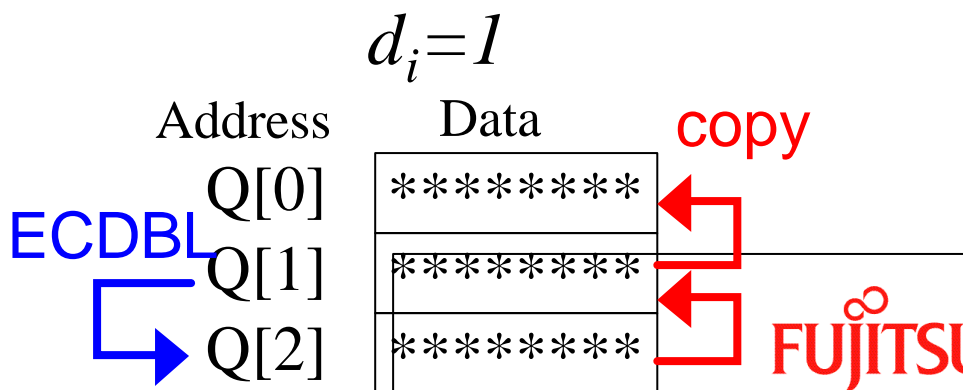
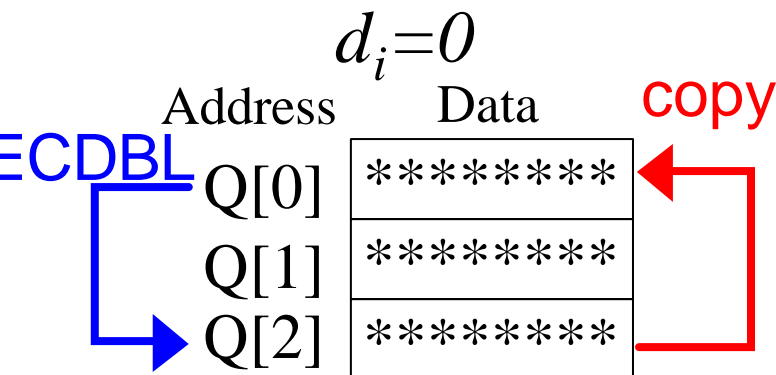
ADPA in ECC

Itoh-Izu-Takenaka(CHES '02)

- Breaks SPA countermeasure + DDPA countermeasure!

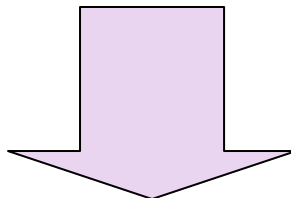
Add-and-always method + Randomized Projective Coordinates

```
Q[0] = RPC(P), Q[1] = ECDBL(Q[0])
for i = n-2 downto 0 {
    Q[2] = ECDBL(Q[d_i])
    Q[1] = ECADD(Q[0], Q[1])
    Q[0] = Q[2-d_i], Q[1] = Q[1+d_i]
}
return invRPC(Q[0])
```



Previous Countermeasures against ADPA in ECC

- **Exponent splitting(ES)** : $d=d_1+ d_2, Q=d_1P +d_2P, (d_1, d_2:random)$
→ 2 times slower than without countermeasures
- **Randomized Exponent(REXP)**: $d \xrightarrow{P} d' = d + r \cdot f, Q = d'P$
($r:random, f:order$)
→ 1.125 times slower than without countermeasures
(in 160-bit ECC)



All of them involve overheads!

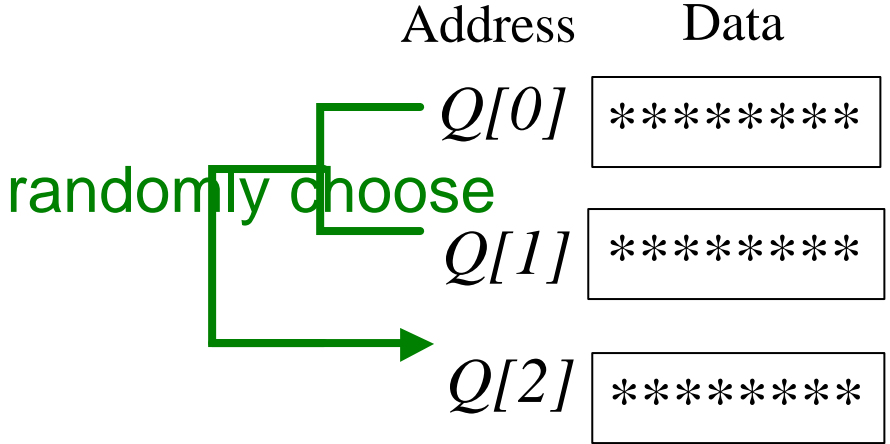
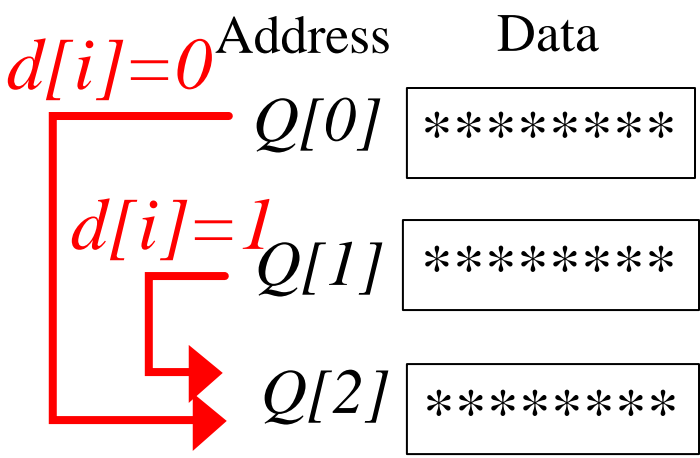
Outline of our Countermeasures against ADPA in ECC

- **Randomized Addressing method (RA)**
 - Approach of RA is similar to Random Register Renaming (RRR, May, CHES '01), a hardware countermeasure by randomly mapping between virtual and physical registers.
- **Advantages of RA to RRR:**
 - No special hardware is required
 - Easily implemented with simple software code and same as RRR, RA involves no overheads!

Basic Idea of RA(our proposal)

- Directly blind the address value of registers with the random number.

Vulnerable : $Q[2]=ECDBL(Q[d[i]])$ Ours : $Q[2]=ECDBL(Q[d[i] \oplus 1\text{-bit random}])$



Algorithm of our Countermeasure

- No overheads are involved
- Easily implemented with simple program code

SPA- and DPA-countermeasure

INPUT: $d; P$
OUTPUT: dP

```
1:  $P' = \text{RPC}(P)$ ,  $Q[0] = P'$ 
2:  $Q[1] = \text{ECDBL}(P')$ 
3: for  $i = m-2$  downto 0 {
4:  $Q[2] = \text{ECDBL}(Q[d[i]])$ 
5:  $Q[1] = \text{ECADD}(Q[0], Q[1])$ 
6:  $Q[0] = Q[2-d[i]]$ 
7:  $Q[1] = Q[1+d[i]]$ 
8: }
9: return  $\text{invRPC}(Q[0])$ 
```

SPA- and DPA-countermeasure + RA

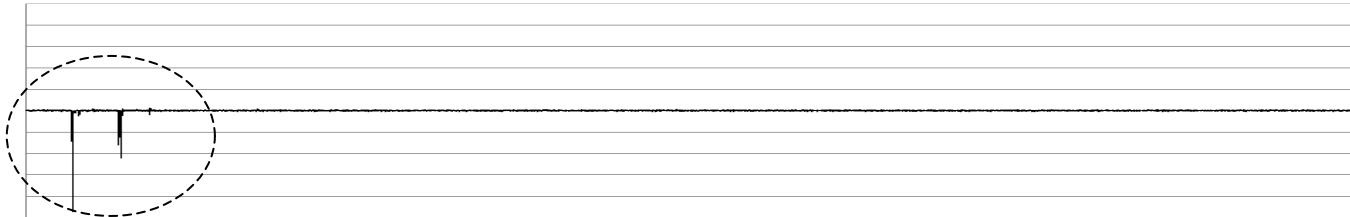
INPUT: $d; P$
OUTPUT: dP

```
1:  $P' = \text{RPC}(P)$ ,  $Q[r[m-1]] = P'$ 
2:  $Q[1-r[m-1]] = \text{ECDBL}(Q[r[m-1]])$ 
3: for  $i = m-2$  downto 0 {
4:  $Q[2] = \text{ECDBL}(Q[d[i] \hat{A}r[i+1]])$ 
5:  $Q[1] = \text{ECADD}(Q[0], Q[1])$ 
6:  $Q[0] = Q[2-(d[i] \hat{A}r[i])]$ 
7:  $Q[1] = Q[1+(d[i] \hat{A}r[i])]$ 
8: }
9: return  $\text{invRPC}(Q[r[0]])$ 
```

Experimental result for ADPA Attack

■ Without RA ($d_a \neq d_b$)

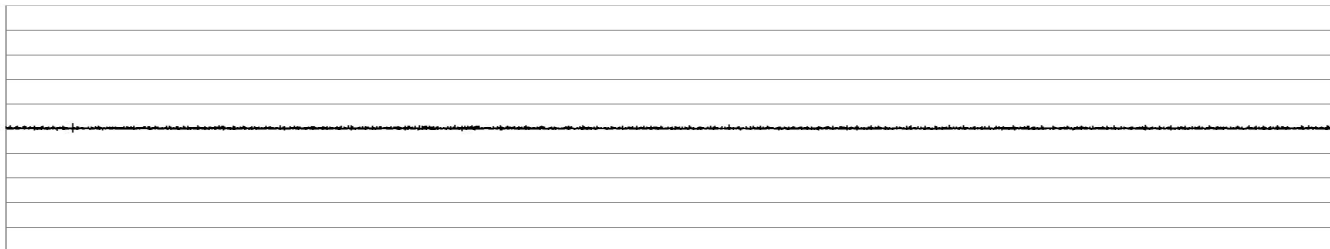
(loading $Q[d_a]$ 10000 times) – (loading $Q[d_b]$ 10000 times)



Some spikes are observed

■ With RA ($d_a = d_b$)

(loading $Q[d_a \hat{A}r_a]$ 10000 times) – (loading $Q[d_b \hat{A}r_b]$ 10000 times)



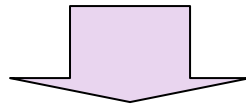
No spikes are observed

→ Experimental result showed RA is secure against ADPA attack.

Summary of RA

- RA has following merits:
 - No overheads are involved
 - Special hardware is never required
 - Easily implemented with simple program codes
- And It also can be applied to:
 - Window method(s)
 - RSA

RA is best solution to prevent ADPA, but for preventing other PA attacks, it should be combined with other countermeasures.

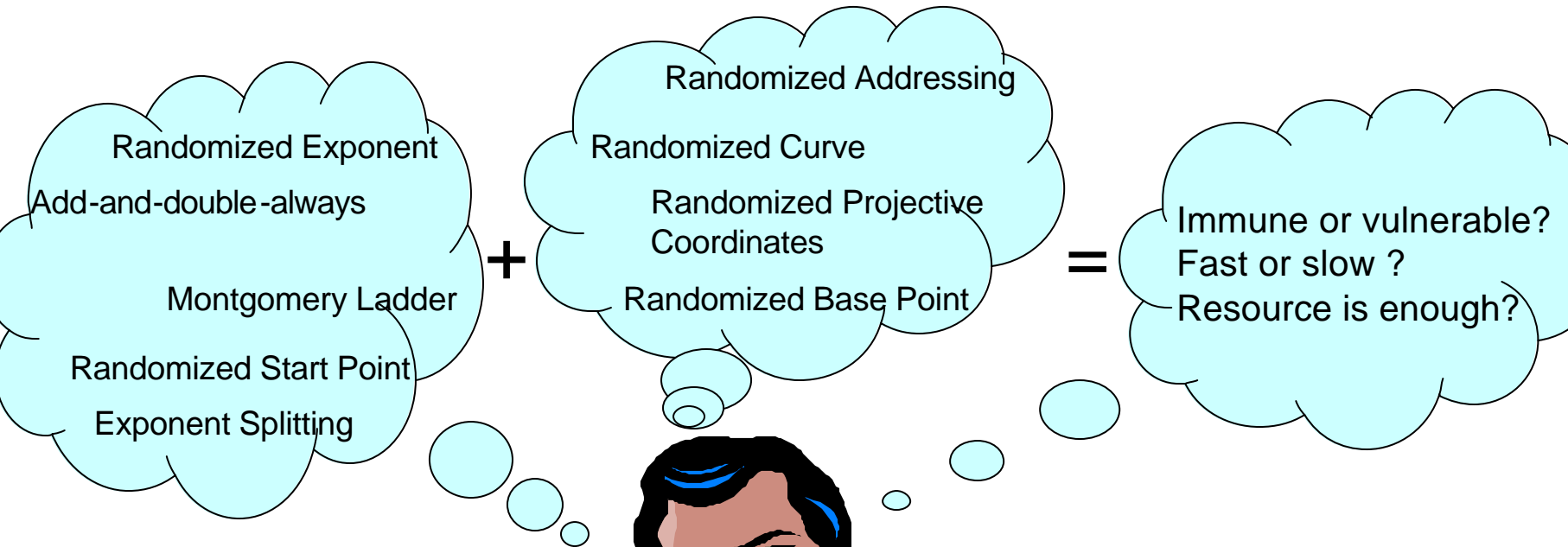


We study for the combination of the countermeasures

- Evaluation Criteria for Countermeasures

Background

■ Question : What is the best choice of the countermeasures?



Security evaluation of Countermeasures

- **Security is attained by the combination of the countermeasures, e.g.:**

Add-and-double-always

SPA:immune, DDPA: vulnerable, ADPA:vulnerable

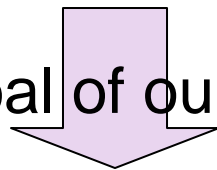
Randomized Projective Coordinates (RPC)

SPA:vulnerable, DDPA:immune, ADPA:vulnerable

Add-and-double-always + RPC

SPA:immune, DDPA: immune, ADPA:vulnerable

Goal of our criteria



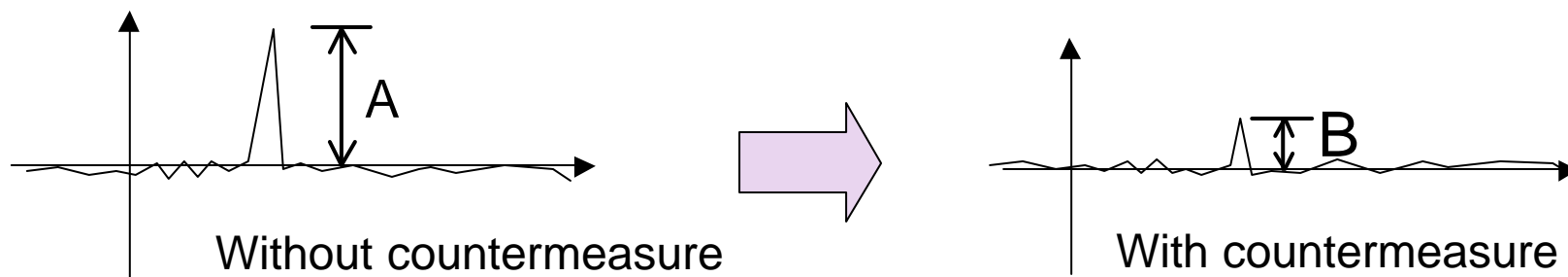
- Choose the **best combination of countermeasures** to attain the **security** within the system requirement, that is, **performance** and **memory size**

Overview of Our Criteria

- **Evaluates a combination of the countermeasures for following points :**
 - **Security**
 - **Performance**
 - **Memory size**
- **Assumption :**
 - **Use 160-bit ECC parameters on prime field**
 - **PA are SPA, DDPA and ADPA**
(In the current result, RPA is not included)
 - **Evaluation is limited to software countermeasures**
→ We do not deal RRR

Security Evaluation in Our Criteria

Security Evaluation with attenuation ratio (AR) (Itoh-Yajima-Takenaka-Torii CHES'02)



A : size of the spikes without countermeasure

B : size of the spikes with countermeasure

AR is evaluated by B/A ($0 \leq AR \leq 1$).

→ As AR is lower, security is higher.

Note : AR is not RA!

FUJITSU

THE POSSIBILITIES ARE INFINITE

Evaluation Parameters in our Criteria

■ Security (AR_s , AR_d , AR_a)

It is evaluated by the AR in SPA (AR_s), DDPA (AR_d) and ADPA(AR_a)

■ Performance (D , A)

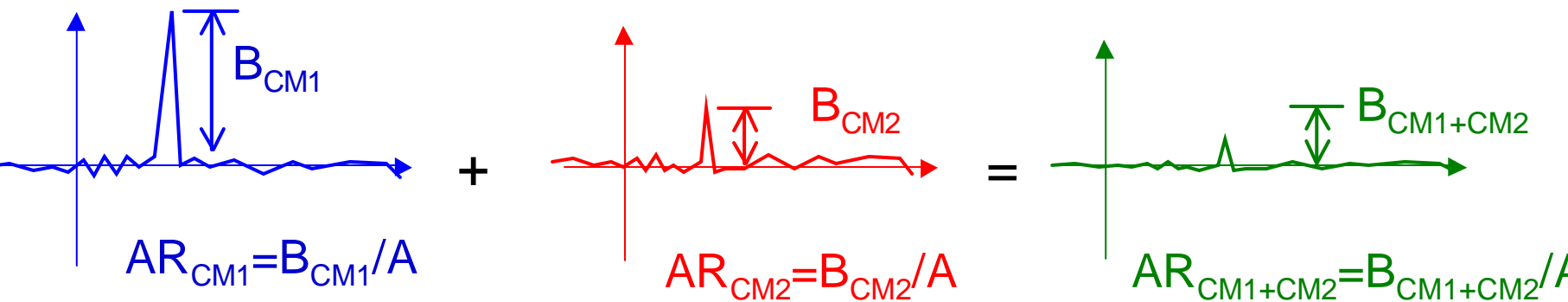
It is evaluated by the number of EC doublings (D) and EC additions (A)

■ Memory size (R_p , R_s)

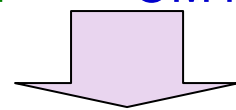
It is evaluated by the number of registers for EC points (R_p) and scalar value (R_s)

Basic Idea for evaluating the combination (1)

■ How to evaluate the AR of **CM1+CM2=???**



$$B_{CM1+CM2} = B_{CM1} \times AR_{CM2}$$



$$AR_{CM1+CM2} = B_{CM1+CM2}/A = AR_{CM1} \times AR_{CM2}$$

Basic Idea for evaluating the combination (2)

■ Performance : A (or D)

$$A_{CM1+CM2} = A_{CM1} \times A_{CM2} \quad (+Ae_{CM2} \text{ in some cases})$$

■ Memory size : R

$$R_{CM1+CM2} = R_{CM1} + R_{CM2}$$

Our Evaluation for Combination of Countermeasures

Parameters

- Security: AR_s (vs. SPA), AR_d (vs. DDPA), AR_a (vs. ADPA)
- Performance : D (ECDBLs), A(ECADDs)
- memory size : R_p (Number of EC Points), R_s (Number of Scalars)

e.g. “**Montgomery Ladder**” + “**Randomized Curve**” = ??????

$$\begin{aligned}
 (AR_s, AR_d, AR_a) &= ((0, 1, 1), (\times 1, \times 2^{-160}, \times 1)) \\
 &= (0 \times 1, 1 \times 2^{-160}, 1 \times 1) = (0, 2^{-160}, 1)
 \end{aligned}
 \left. \vphantom{\begin{aligned} (AR_s, AR_d, AR_a) &= ((0, 1, 1), (\times 1, \times 2^{-160}, \times 1)) \\ &= (0 \times 1, 1 \times 2^{-160}, 1 \times 1) = (0, 2^{-160}, 1) \end{aligned}} \right\} \begin{array}{l} \text{ARs in} \\ \text{(SPA, DDPA, ADPA)} \end{array}$$

$$\begin{aligned}
 (D, A) &= ((160, 160), (\times 1, \times 1)) \\
 &= (160 \times 1, 160 \times 1) = (160, 160)
 \end{aligned}
 \left. \vphantom{\begin{aligned} (D, A) &= ((160, 160), (\times 1, \times 1)) \\ &= (160 \times 1, 160 \times 1) = (160, 160) \end{aligned}} \right\} \begin{array}{l} \text{Number of} \\ \text{(ECDBL, ECADD)} \end{array}$$

$$(R_p, R_s) = ((3, 0), (+0, +3)) = (3+0, 0+3) = (3, 3) \left. \vphantom{(R_p, R_s)} \right\} \begin{array}{l} \text{Register number for} \\ \text{(Point, Scalar)} \end{array}$$

→ **Combination is easily evaluated!**



Basic Data Table of our Evaluation Criteria (Binary-method)

Method	Security (SPA,DDPA,ADPA)			Performance (ECDBL,ECADD)		Registers (Point, Scalar)	
	AR_s	AR_d	AR_a	D	A	R_p	R_s
<i>Binary method (from MSB)</i>	1	1	1	160	80	1	0
<i>Binary method (from LSB)</i>	1	1	1	160	80	2	0
Add-and-double-always	$\times 0$	$\times 1$	$\times 1$	$\times 1$	$\times 2$	+1	+0
<i>Montgomery ladder</i>	0	1	1	160	160	3	0
Randomized projective coordinate (RPC)	$\times 1$	$\times 2^{-160}$	$\times 1$	$\times 1$	$\times 1$	+1	+1
Randomized curve (RC)	$\times 1$	$\times 2^{-160}$	$\times 1$	$\times 1$	$\times 1$	+0	+3
Randomized base point	$\times 1$	$\times 2^{-160}$	$\times 1$	$\times 2$	$\times 2$	+2	+0
Randomized exponent ($ r = 20$)	$\times 1$	$\times 2^{-20}$	$\times 2^{-20}$	$\times 1.13$	$\times 1.13$	+0	+1
Randomized start point	$\times 1$	$\times 2^{-7.3}$	$\times 2^{-7.3}$	$\times 1$	$\times 1$	+0	+0
Exponent splitting	$\times 1$	$\times 2^{-160}$	$\times 2^{-160}$	$\times 2$	$\times 2+1$	+1	+2
Randomized addressing (RA)	$\times 1$	$\times 1$	$\times 2^{-160}$	$\times 1$	$\times 1$	+0	+2

FUJITSU

THE POSSIBILITIES ARE INFINITE

Choosing the best combination of the countermeasures

■ Case1 (Slower encryption)

“Binary method (from MSB)” +
“Add-and-double-always” +
“RPC” + “Randomized Exponent”

$$(AR_s, AR_d, AR_a) = (0, 2^{-180}, 2^{-180})$$
$$(D, A) = (180, 180)$$
$$(R_p, R_s) = (3, 2)$$

■ Case2 (Larger memory size)

“Montgomery Ladder” +
“RPC” + “RA”

$$(AR_s, AR_d, AR_a) = (0, 2^{-160}, 2^{-160})$$
$$(D, A) = (160, 160)$$
$$(R_p, R_s) = (4, 3)$$

■ Case3 (Best combination)

“Binary method (from MSB)” +
“Add-and-double-always” +
“RPC” + RA

$$(AR_s, AR_d, AR_a) = (0, 2^{-160}, 2^{-160})$$
$$(D, A) = (160, 160)$$
$$(R_p, R_s) = (3, 3)$$

Choosing the best combination of the countermeasures

■ Case1 (Slower encryption)

“Binary method (from MSB)” +
“Add-and-double-always” +
“RPC” + “Randomized Exponent”

$$\begin{aligned}(AR_s, AR_d, AR_a) &= (0, 2^{-180}, 2^{-180}) \\ (D, A) &= (180, 180) \\ (R_p, R_s) &= (3, 2)\end{aligned}$$

■ Case2 (Larger memory size)

“Montgomery Ladder” +
“RPC” + “RA”

$$\begin{aligned}(AR_s, AR_d, AR_a) &= (0, 2^{-160}, 2^{-160}) \\ (D, A) &= (160, 160) \\ (R_p, R_s) &= (4, 3)\end{aligned}$$

■ Case3 (Best combination)

“Binary method (from MSB)” +
“Add-and-double-always” +
“RPC” + RA

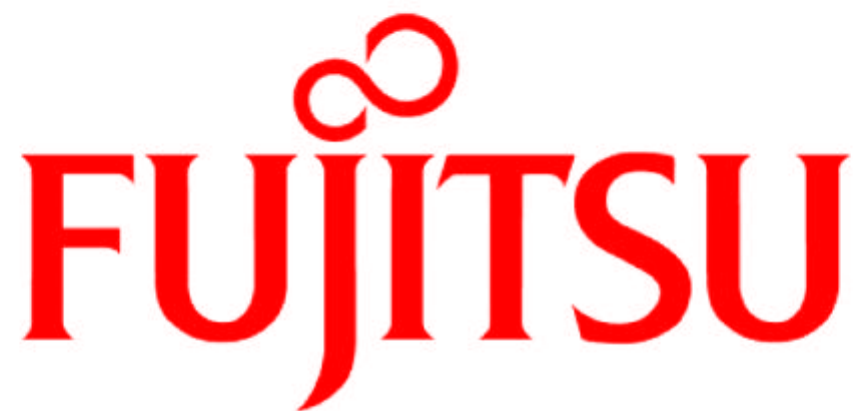
$$\begin{aligned}(AR_s, AR_d, AR_a) &= (0, 2^{-160}, 2^{-160}) \\ (D, A) &= (160, 160) \\ (R_p, R_s) &= (3, 3)\end{aligned}$$

Remarks and Summary of our Criteria

- **Enables to choose the best combination of the countermeasures within the system requirement**
- **With our criteria, RA is estimated to be the best solution against ADPA**
- **Security against RPA can be also evaluated, by expanding the basic data table**

Conclusions

- We proposed an ADPA countermeasure (RA), which
 - involves no overhead
 - and implemented easily with simple software code.
- We showed its security against ADPA experimentally
- We proposed an evaluation criteria of the countermeasures, which
 - enables to chose an optimal countermeasures within the system requirement
 - can be applied to other countermeasures and analysis than those in our paper
- By our criteria, RA is the best solution to the ADPA attack

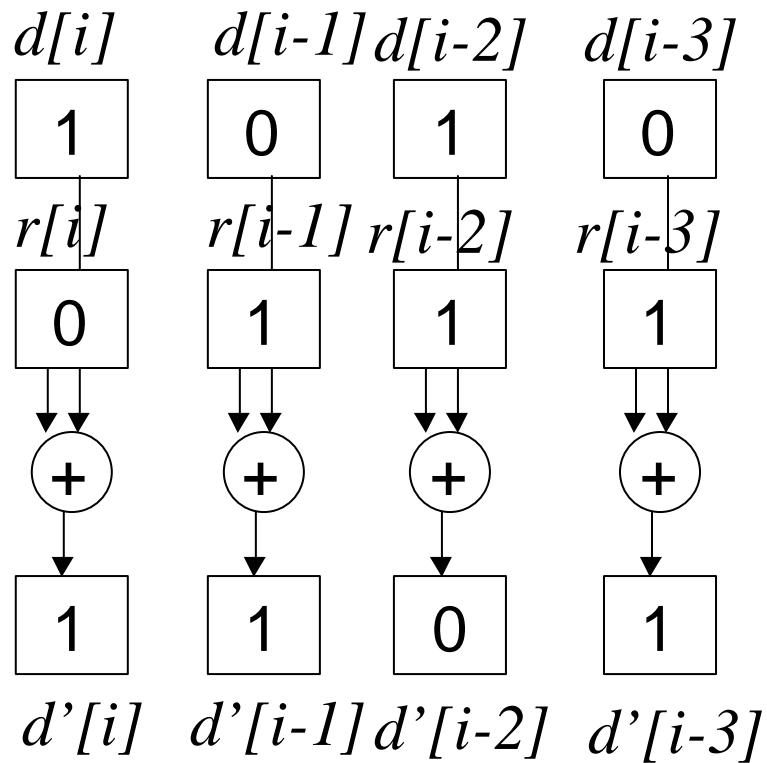


FUJITSU

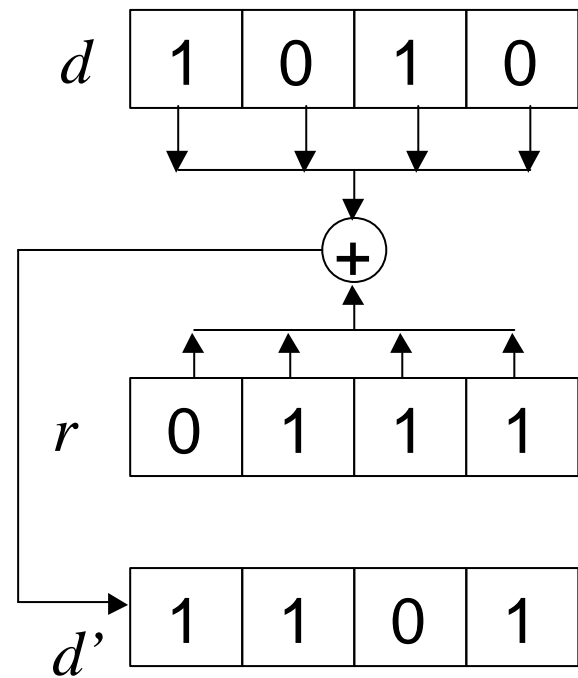
THE POSSIBILITIES ARE INFINITE

Basic Idea of More Efficient Software Implementation

Single-bit RA



Multiple-bit RA



More Efficient Software Implementation

- Plurality of address-bits are randomized at once
→ Effective in software implementation

Single-bit RA

INPUT: $d; P$
OUTPUT: dP

```
1:  $P' = \text{RPC}(P)$ ,  $Q[r[m-1]] = P'$ 
2:  $Q[1-r[m-1]] = \text{ECDBL}(Q[r[m-1]])$ 
3: for  $i=m-2$  downto 0 {
4:  $Q[2] = \text{ECDBL}(Q[d[i] \hat{\wedge} r[i+1]])$ 
5:  $Q[1] = \text{ECADD}(Q[0], Q[1])$ 
6:  $Q[0] = Q[2-(d[i] \hat{\wedge} r[i])]$ 
7:  $Q[1] = Q[1+(d[i] \hat{\wedge} r[i])]$ 
8: }
9: return  $\text{invRPC}(Q[r[0]])$ 
```

Multiple-bit RA

INPUT: $d; P$
OUTPUT: dP

```
1:  $P' = \text{RPC}(P)$ ,  $Q[r[m-1]] = P'$ 
2:  $d' = d \hat{\wedge} (r \gg 1)$ ,  $d'' = d \hat{\wedge} r$ 
3:  $Q[1-r[m-1]] = \text{ECDBL}(Q[r[m-1]])$ 
4: for  $i=m-2$  downto 0 {
5:  $Q[2] = \text{ECDBL}(Q[d'[i]])$ 
6:  $Q[1] = \text{ECADD}(Q[0], Q[1])$ 
7:  $Q[0] = Q[2-d''[i]]$ 
8:  $Q[1] = Q[1+d''[i]]$ 
9: }
10: return  $\text{invRPC}(Q[r[0]])$ 
```