ECC on small devices

Junfeng Fan

Katholieke Universiteit Leuven, Belgium

junfeng.fan@esat.kuleuven.be

- What is a small device?

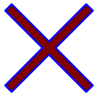
- What is a small device?

- What is a small device?

- What is a small device?

Trusted Platform Module

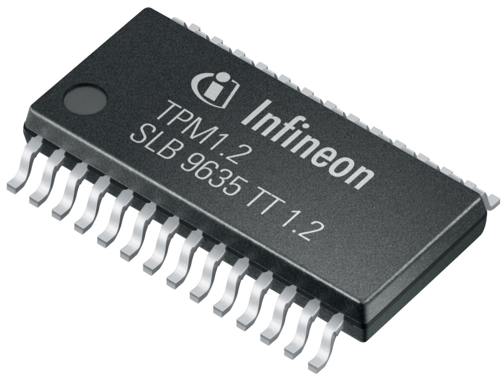
- What is a small device?



Trusted Platform Module

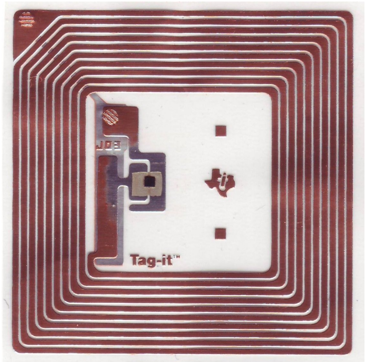
Credit Card

- What is a small device?



Trusted Platform Module

Credit Card

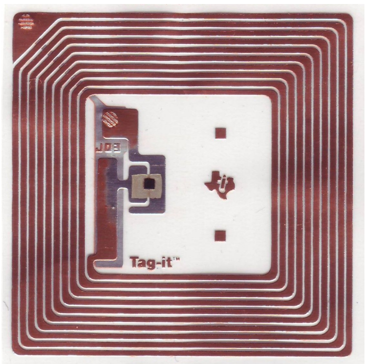


RFID Tag

- Why do we want ECC on small devices?

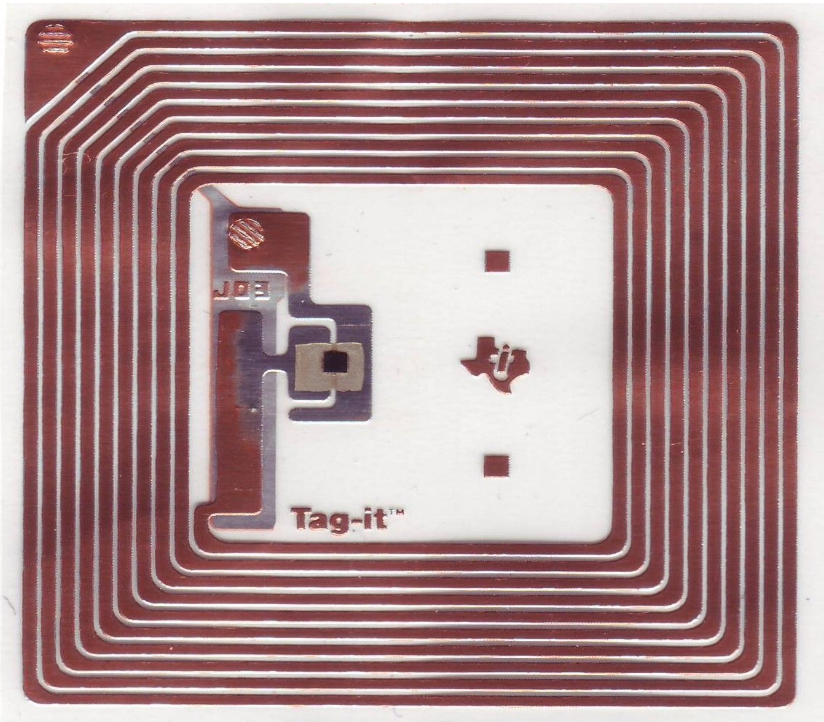
Trusted Platform Module

Credit Card

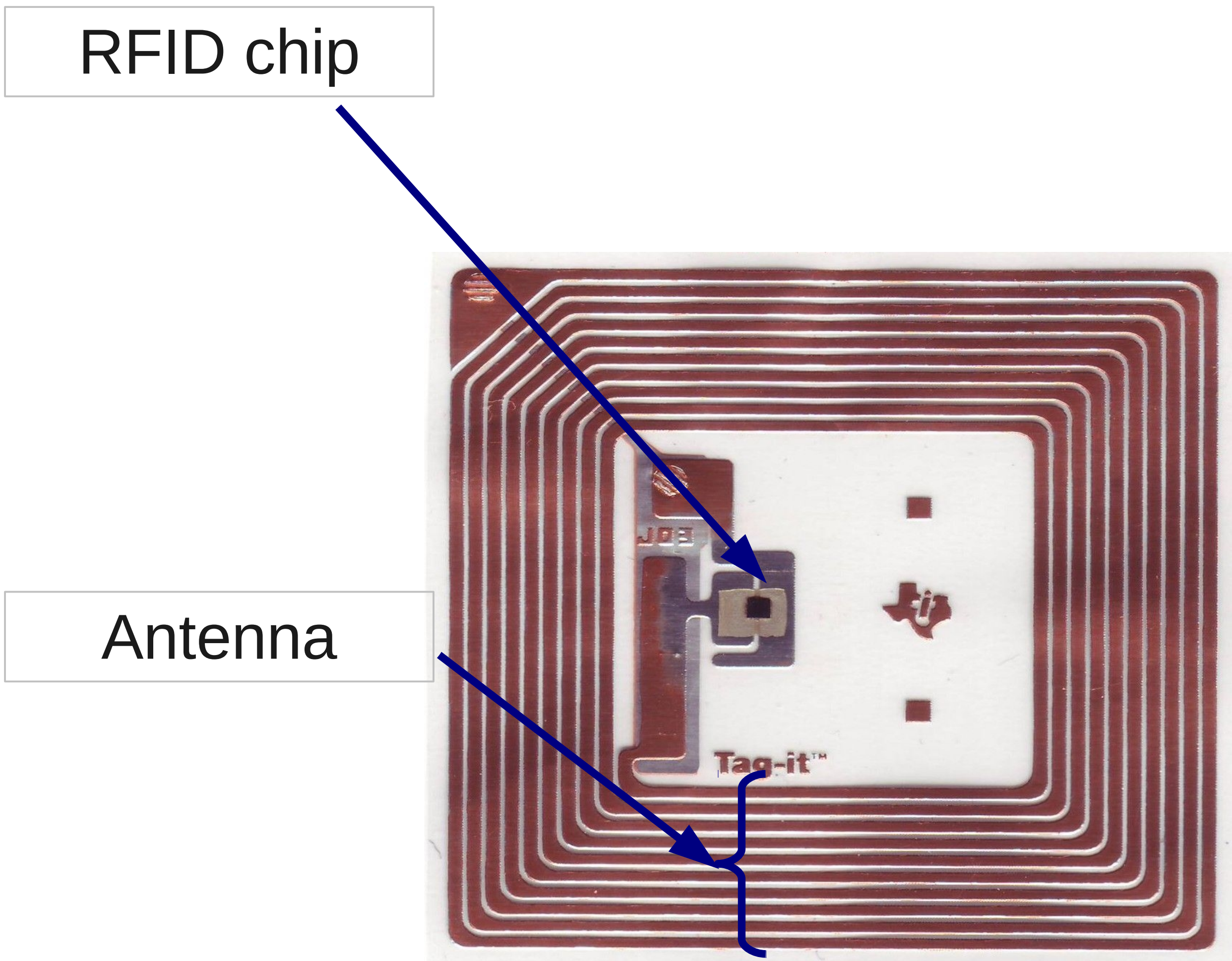


RFID Tag

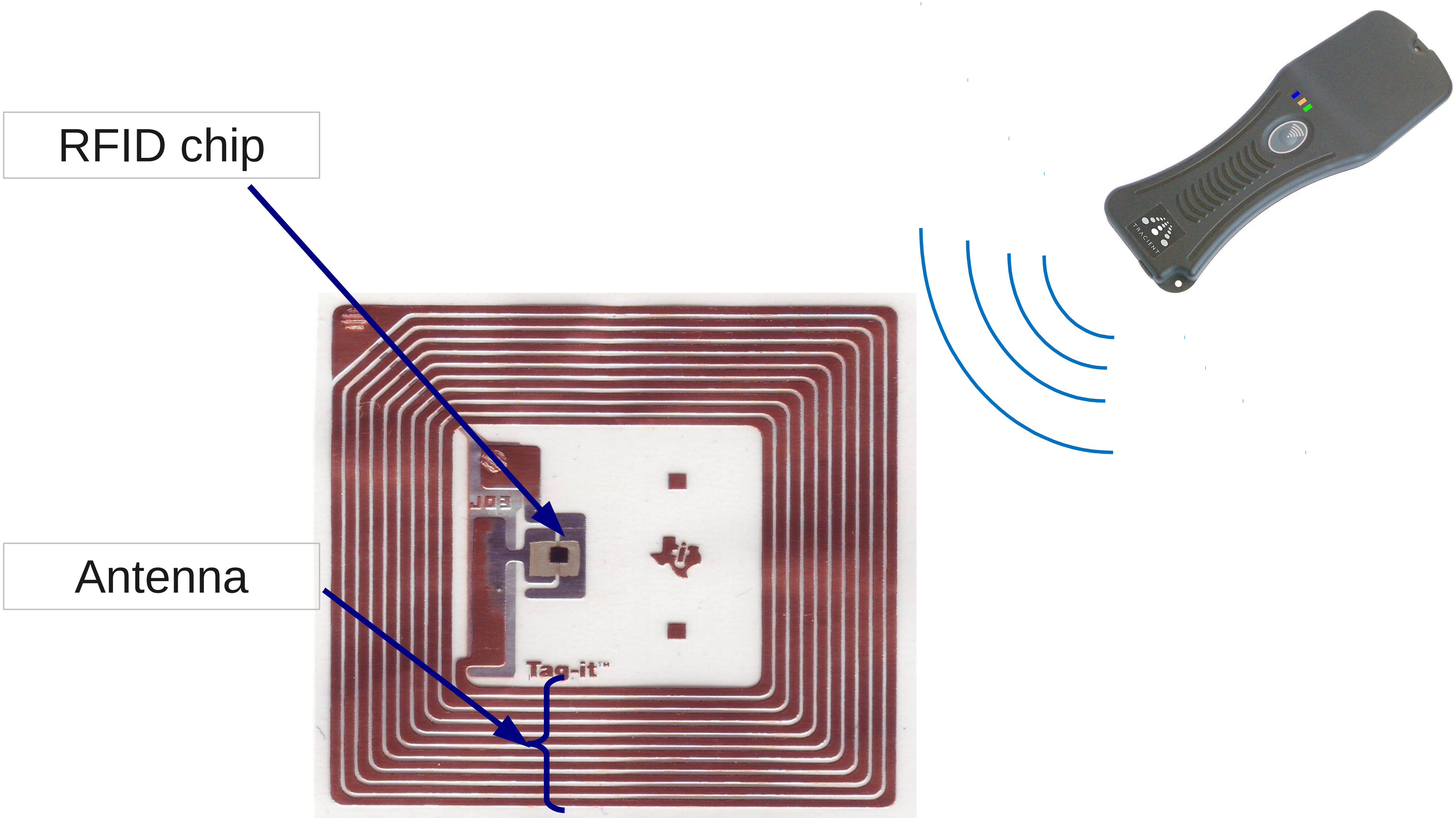
- Let's take RFID as an example...

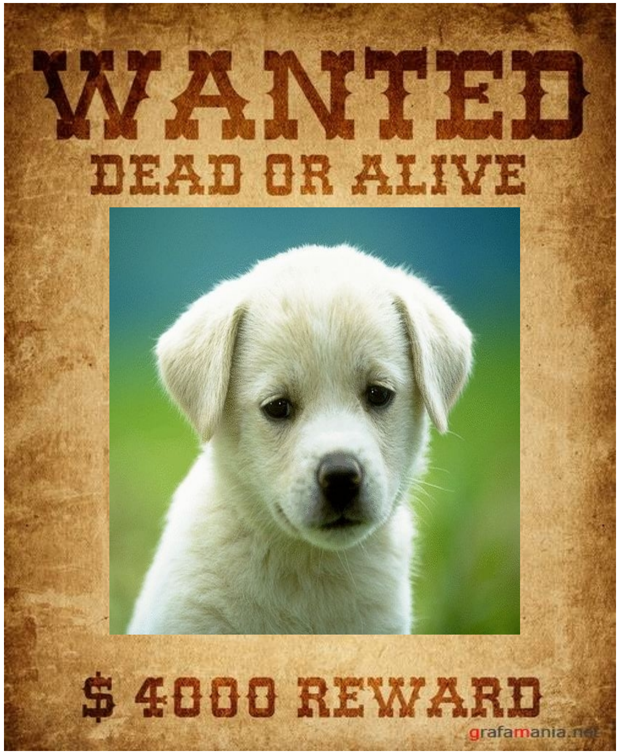


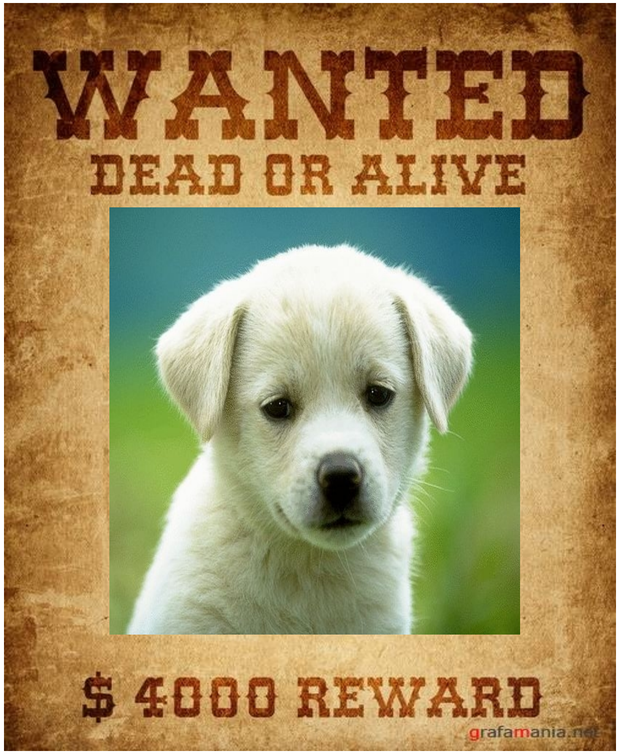
- Let's take RFID as an example...



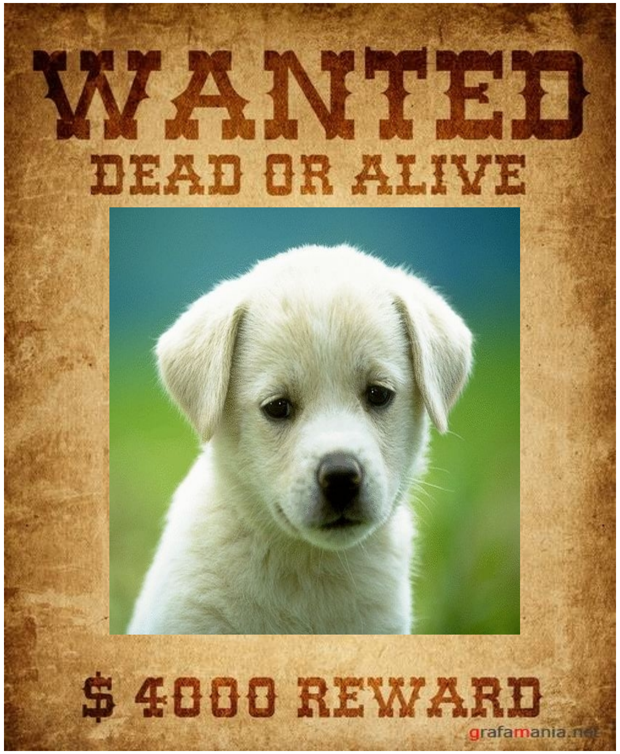
- Let's take RFID as an example...




WANTED
DEAD OR ALIVE
$ 4000 REWARD
grafamania.net


WANTED
DEAD OR ALIVE
$ 4000 REWARD
grafamania.net


EXPO
SHANHAI CHINA
中国2010年上海世博会
EXPO 2010SHANHAI CHINA
2010. 5. 1–10. 31
城市，让生活更美好
Better City, Better Life
平日 · 夜票
Sunday • Evening admission
¥90

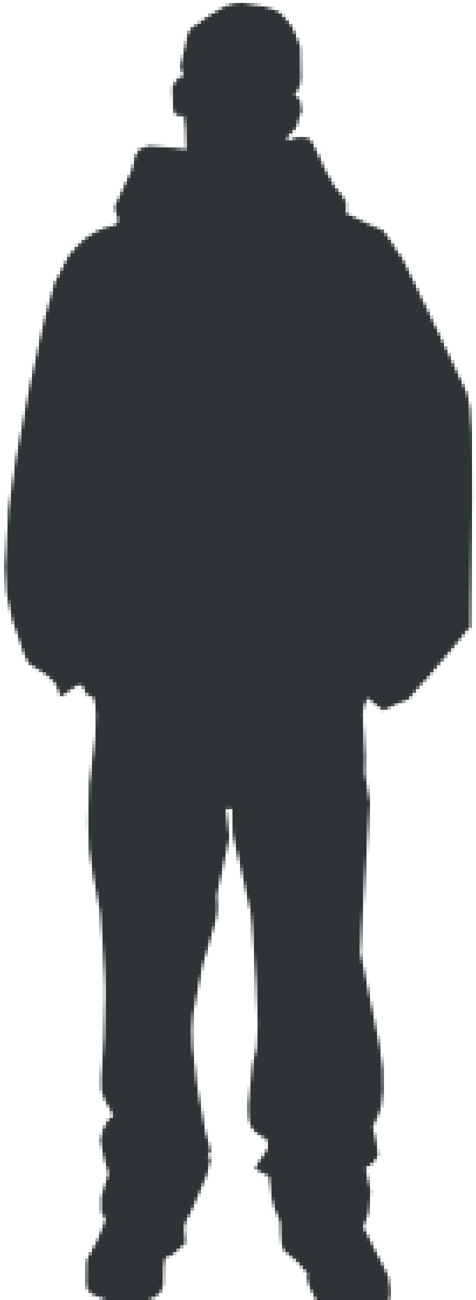

WANTED
DEAD OR ALIVE
$ 4000 REWARD
grafamania.net


中国2010年上海世博会
EXPO 2010SHANHAI CHINA
2010. 5. 1–10. 31
Better City, Better Life
EXPO
SHANHAI CHINA
平日・夜票
Sunday • Evening admission
¥90


かけこみはおやめください
投入口

- The problem is....privacy.

- The problem is....privacy.



- The problem is....privacy.

- The problem is....privacy.

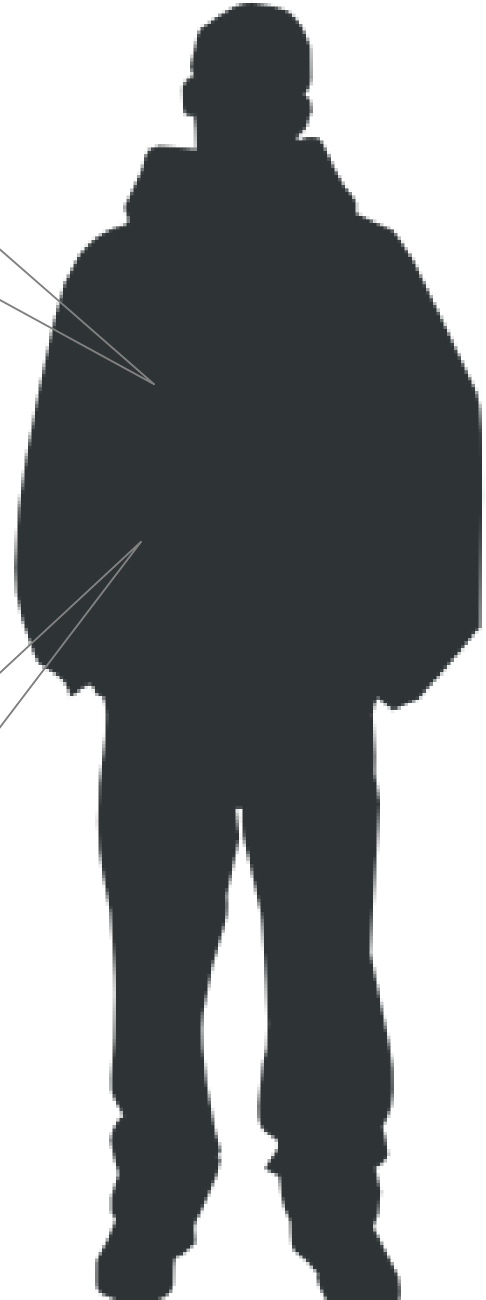
ID:

Thomas XXX
13.08,1976
Dengerland

- The problem is....privacy.

ID:

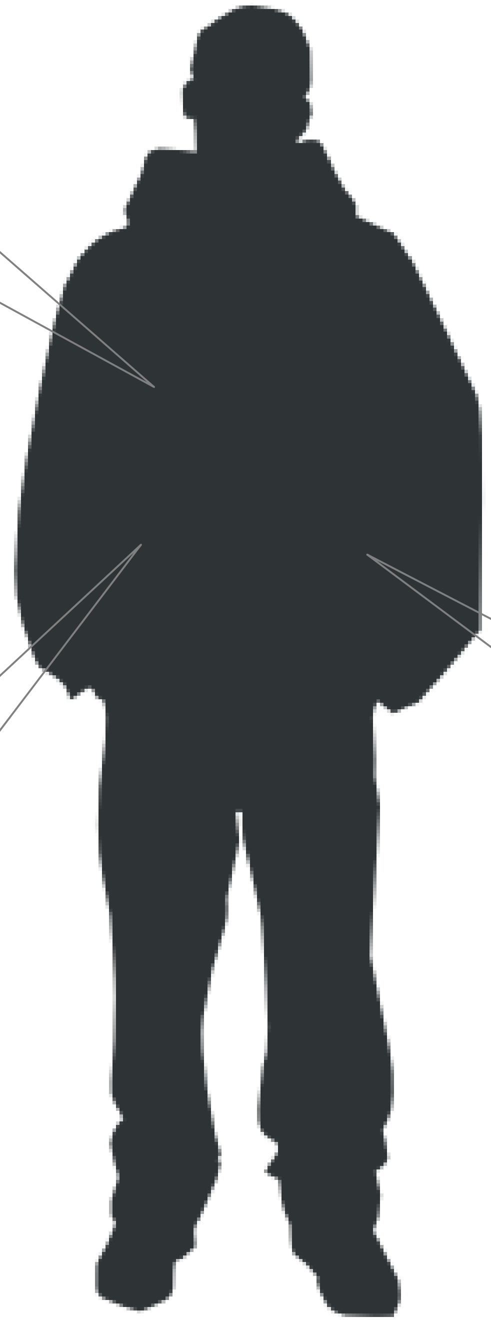
Thomas XXX
13.08,1976
Dengerland

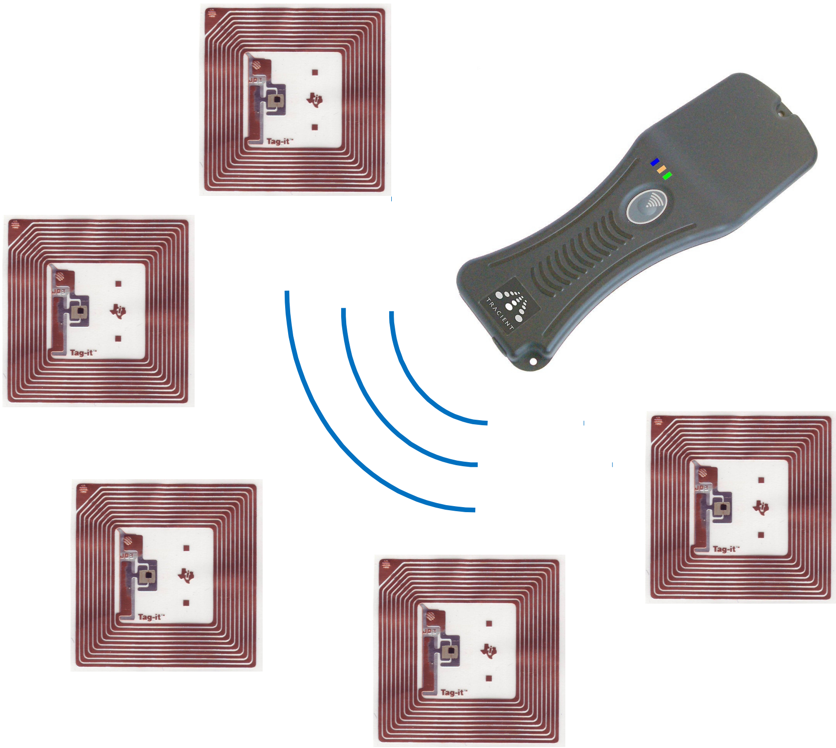


- The problem is....privacy.

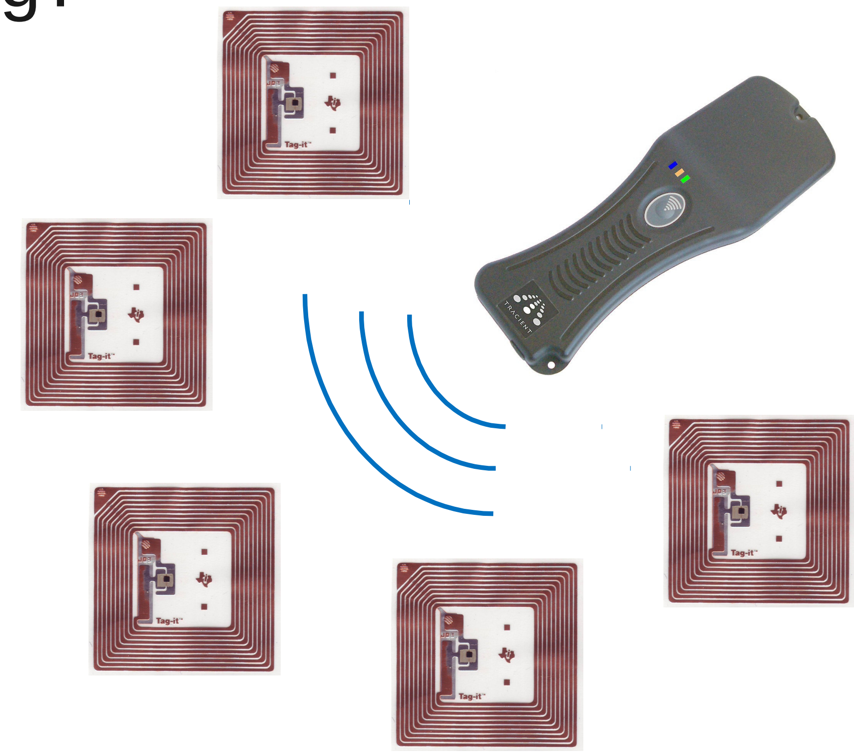
ID:

Thomas XXX
13.08,1976
Dengerland

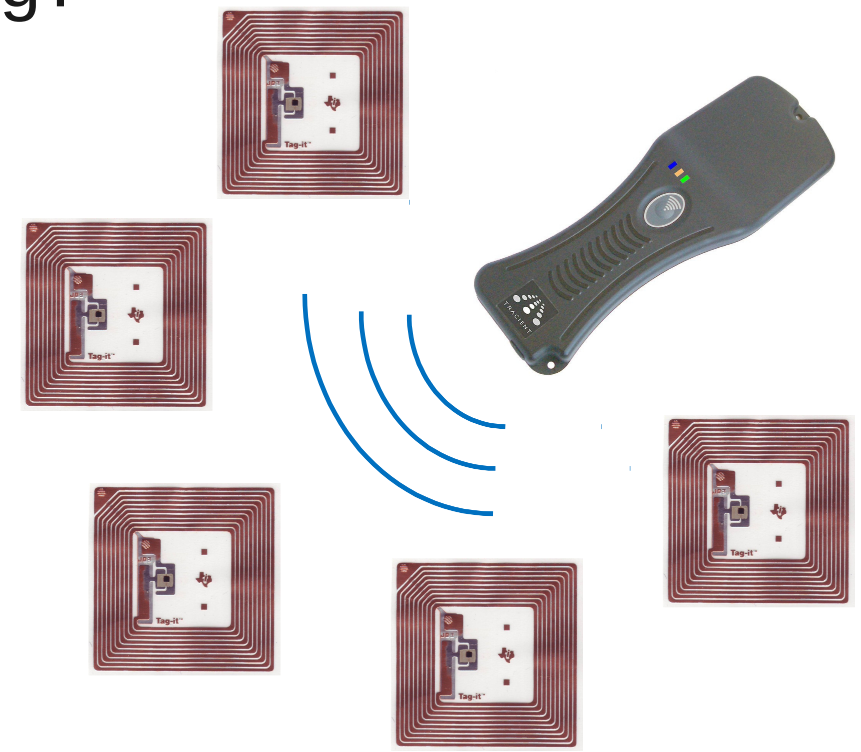



Tag-it

- What makes a good RFID tag?

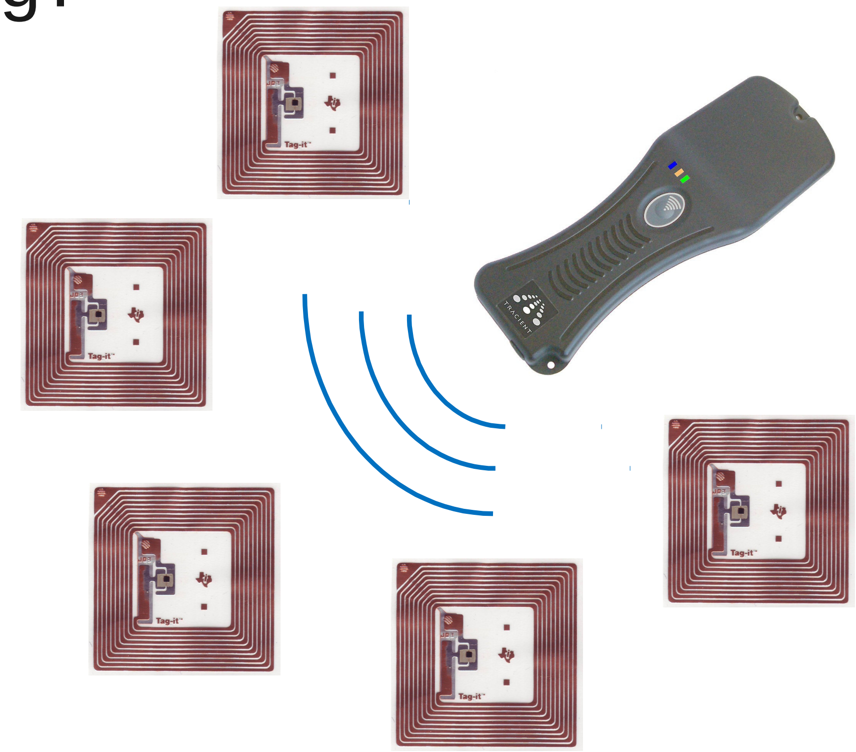


- What makes a good RFID tag?



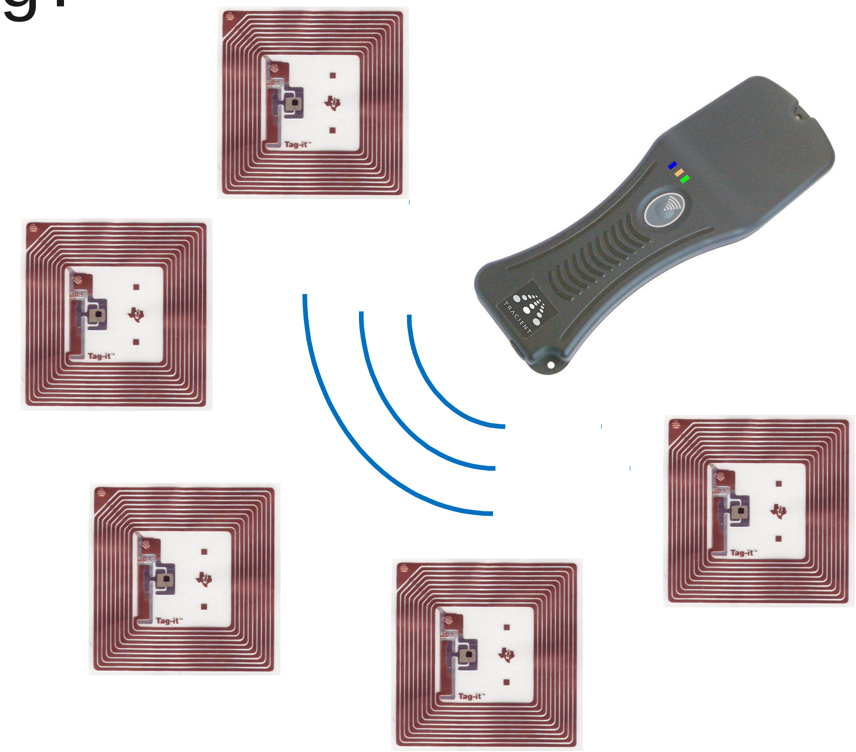
- It works!

- What makes a good RFID tag?



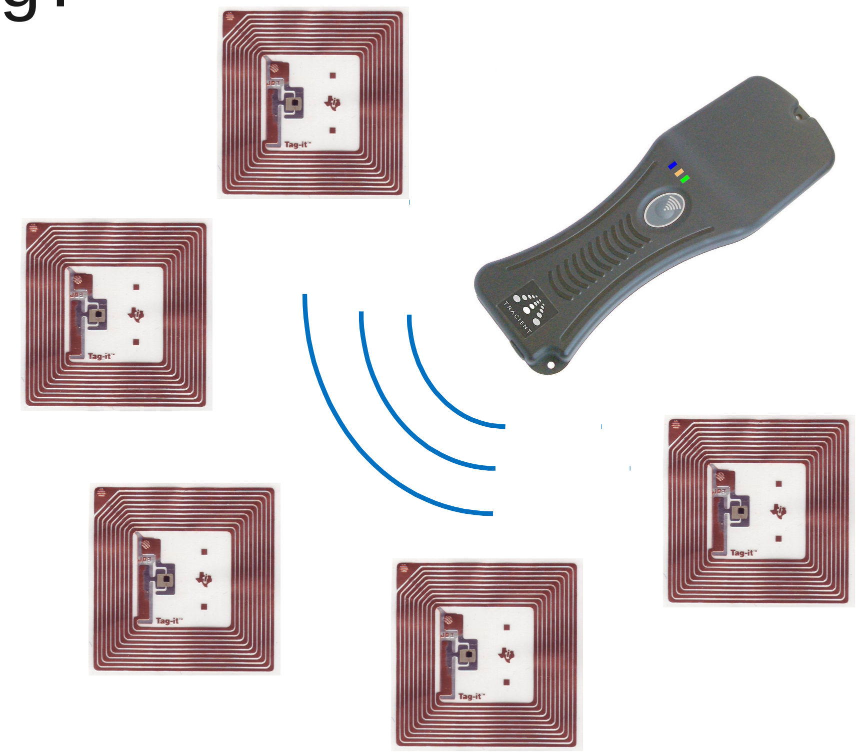
- It works!
- It's cheap.

- What makes a good RFID tag?



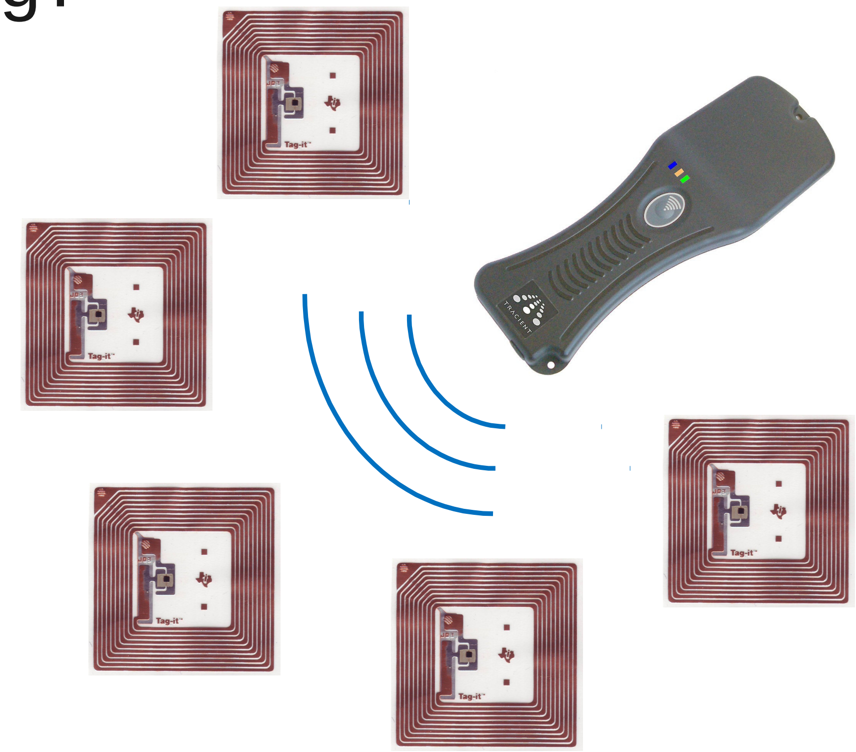
- It works!
- It's cheap.
- It's secure.

- What makes a good RFID tag?



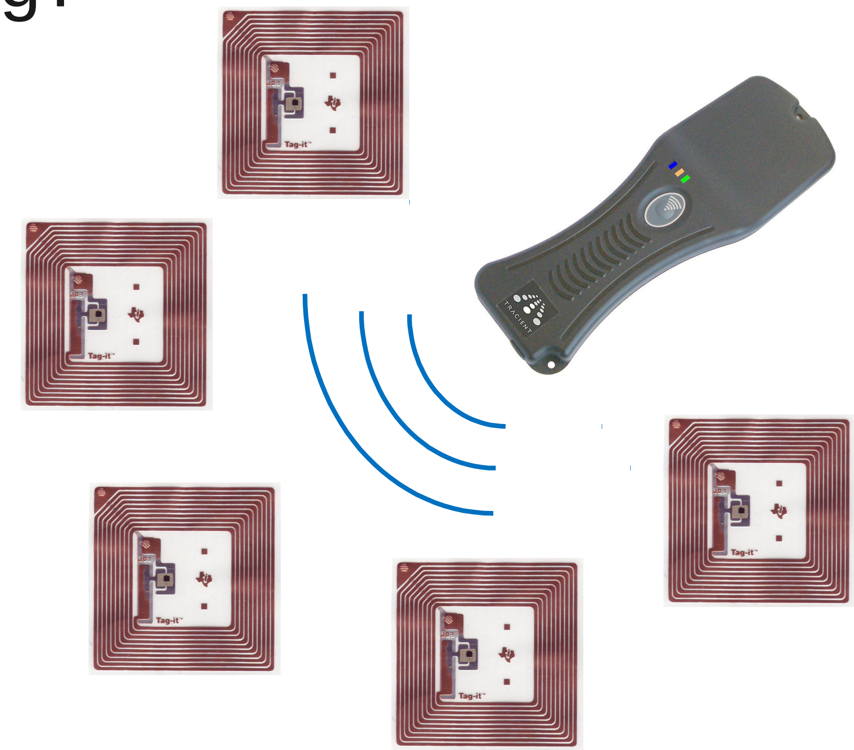
- It works!
- It's cheap.
- It's secure.
- It's untraceable.

- What makes a good RFID tag?



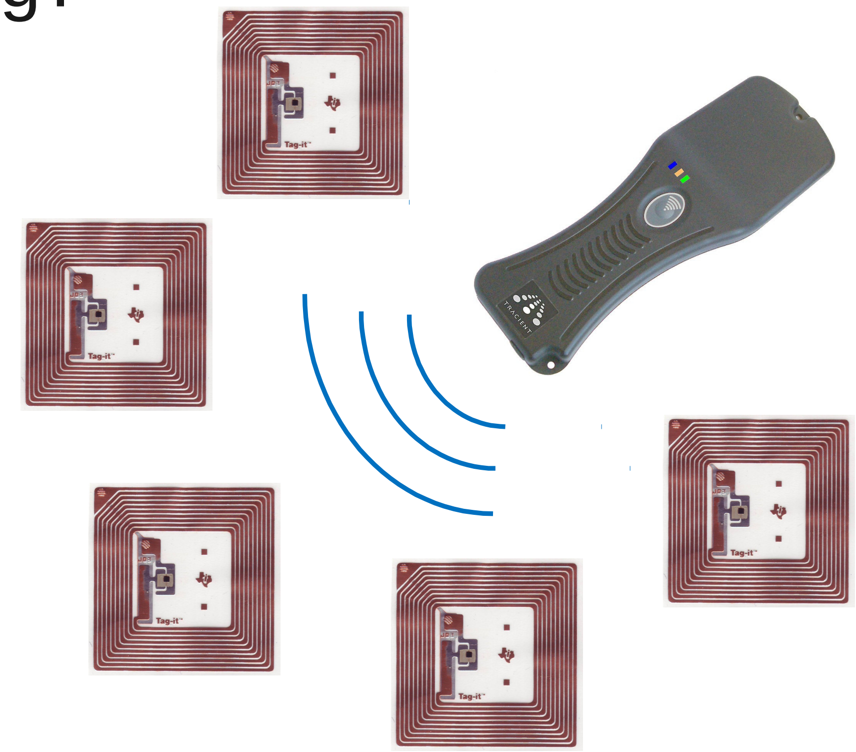
- It works!
- It's cheap.
- It's secure.
- It's untraceable.
- It's scalable.

- What makes a good RFID tag?



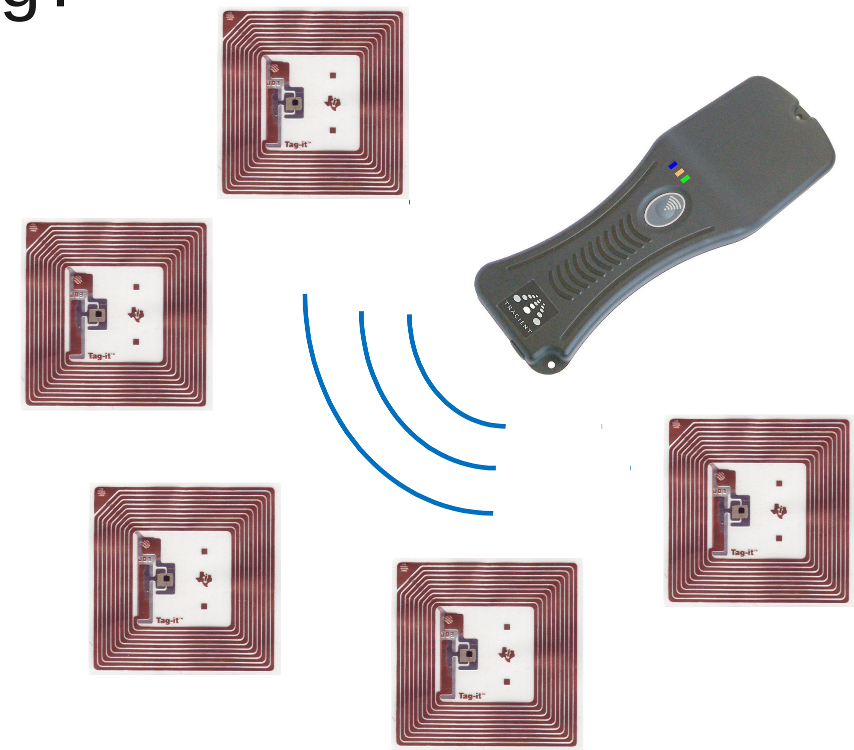
- It works!
- It's cheap.
- It's secure.
- It's untraceable.
- It's scalable.
- It's fast.

- What makes a good RFID tag?



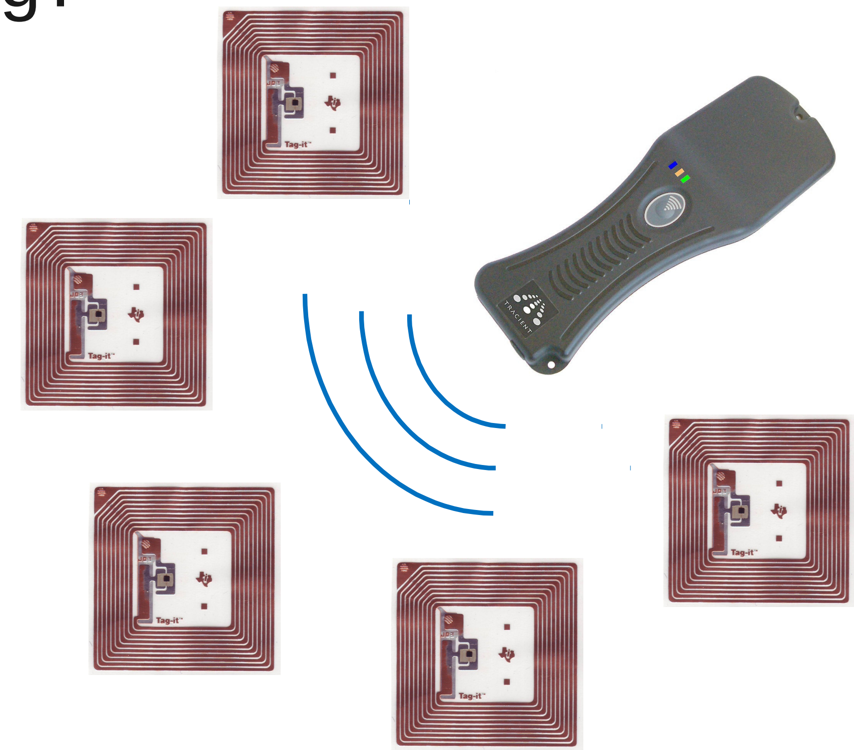
- It works!
- It's cheap. → Small area
- It's secure.
- It's untraceable.
- It's scalable.
- It's fast.

➢ What makes a good RFID tag?



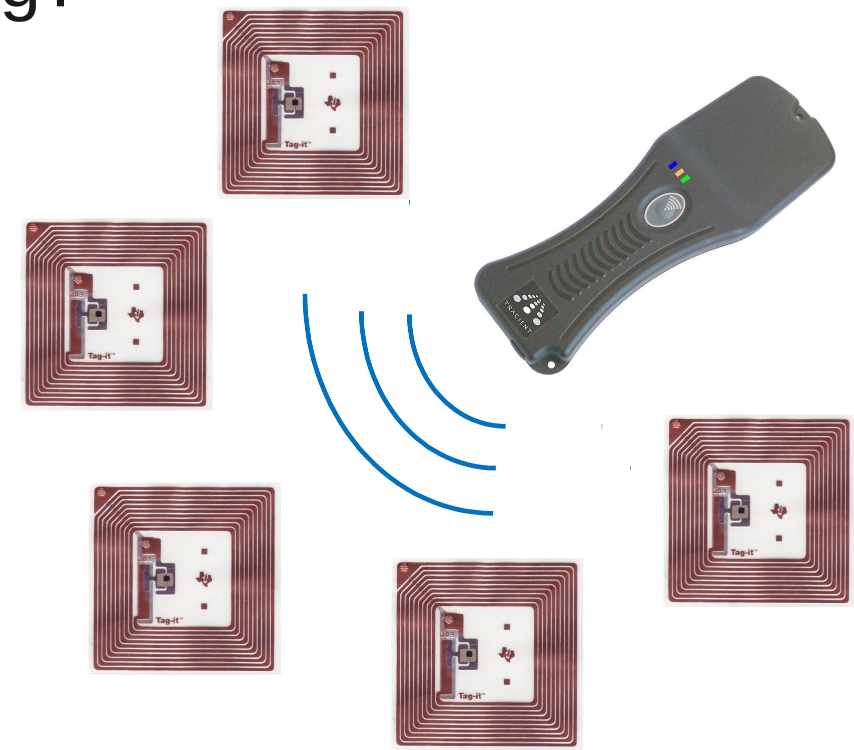
- It works!
- It's cheap. → Small area
- It's secure. → Crypto
- It's untraceable.
- It's scalable.
- It's fast.

- What makes a good RFID tag?



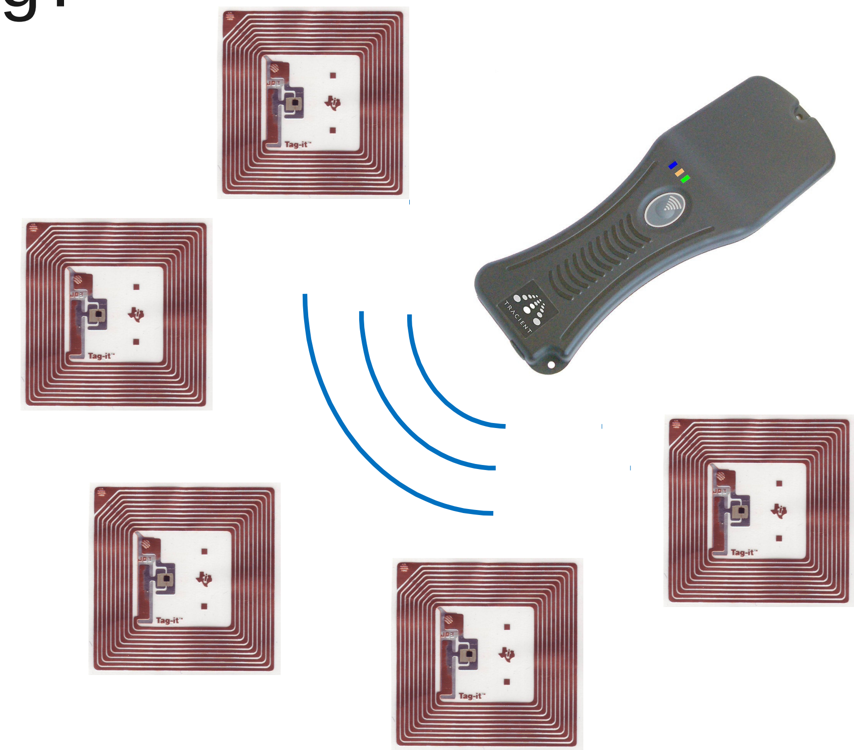
- It works!
- It's cheap. → Small area
- It's secure. → Crypto
- It's untraceable. → PKC
- It's scalable. → PKC
- It's fast.

- What makes a good RFID tag?



- It works!
- It's cheap. → Small area
- It's secure. → Crypto
- It's untraceable. → PKC
- It's scalable. → PKC
- It's fast. → lightweight

- What makes a good RFID tag?



- It works!
- It's cheap. → Small area
- It's secure. → Crypto
- It's untraceable. → PKC
- It's scalable. → PKC
- It's fast. → lightweight

Small area, Crypto, PKC, lightweight → **ECC?**

The Schnorr Protocol [Schnorr'89]

- Tag's private key: x
- Tag's public key : $\mathbf{X}(=[-x]\mathbf{P})$

Reader (Verifier)		**Tag (Prover)**
		r_1=TRNG( )
	← $\mathbf{R}_1$	$\mathbf{R}_1 = [r_1]\mathbf{P}$
r_2 = TRNG( )	r_2 →	
If $[v]\mathbf{P} + [r_2]\mathbf{X} == \mathbf{R}_1$ Then accept	← v	$v = xr_2 + r_1 \bmod n$

The Schnorr Protocol [Schnorr'89]

- Tag's private key: x
- Tag's public key : $\mathbf{X}(=[-x]\mathbf{P})$

Reader (Verifier)		**Tag (Prover)**
		r_1=TRNG( )
	$\mathbf{R}_1$ ←	$\mathbf{R}_1 = [r_1]\mathbf{P}$
r_2 = TRNG( )	→ r_2	
If $[v]\mathbf{P} + [r_2]\mathbf{X} == \mathbf{R}_1$ Then accept	v ←	$v = xr_2 + r_1 \bmod n$

Tracing Attack: $([v]\mathbf{P} - \mathbf{R}_1)r_2^{-1} = [x]\mathbf{P} = -\mathbf{X}$

The Vaudenay Protocol [Vaudenay'07]

- Reader's private key : K_S, K_M
- Reader's public key : K_P
- Tag's ID: ID, K=F_{K_M}(ID)

Reader (Verifier)		Tag (Prover)
a=TRNG( )	a →	
	← c	c = $\mathbf{Enc}_{K_P}(ID \| K \| a)$
$ID \| K \| a'$ = $\mathbf{Dec}_{K_S}(c)$ If $a == a'$ K == F_{K_M}(ID) Then accept ID		

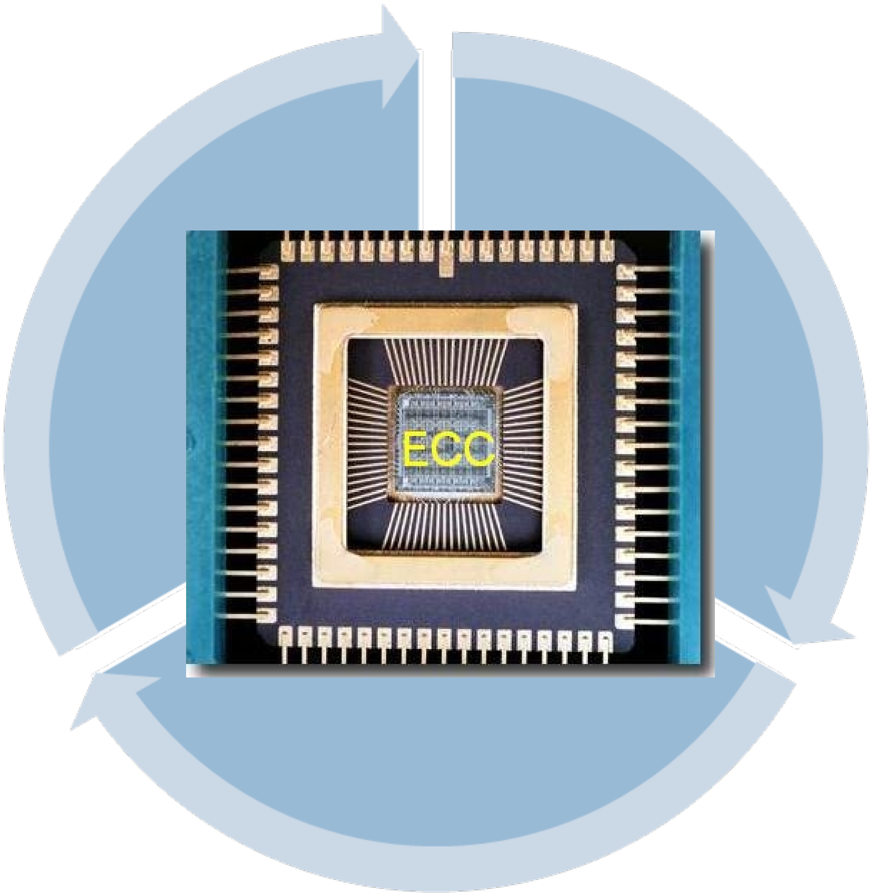
➢ The Vaudenay Protocol [Vaudenay'07]

- Reader's private key : K_S, K_M
- Reader's public key : K_P
- Tag's ID: ID, $K=F_{K_M}(ID)$

Reader (Verifier)		**Tag (Prover)**
a=TRNG( )	a →	
	← c	c = $\mathbf{Enc}_{K_P}(ID\|\|K\|\|a)$
$ID\|\|K\|\|a' = \mathbf{Dec}_{K_S}(c)$ If $a == a'$ $K == F_{K_M}(ID)$ Then accept ID		

If the PKC in use is **IND-CPA-secure**, then the above RFID scheme is **narrow-strong** private.

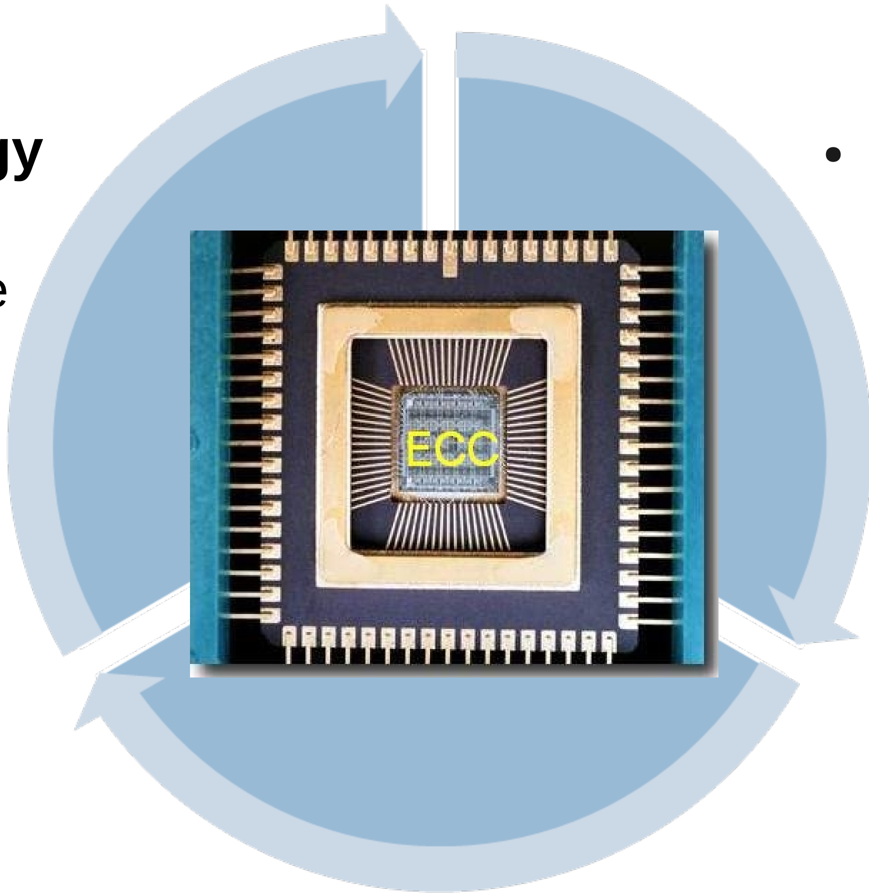
- An ECC processor for RFID tags



- An ECC processor for RFID tags

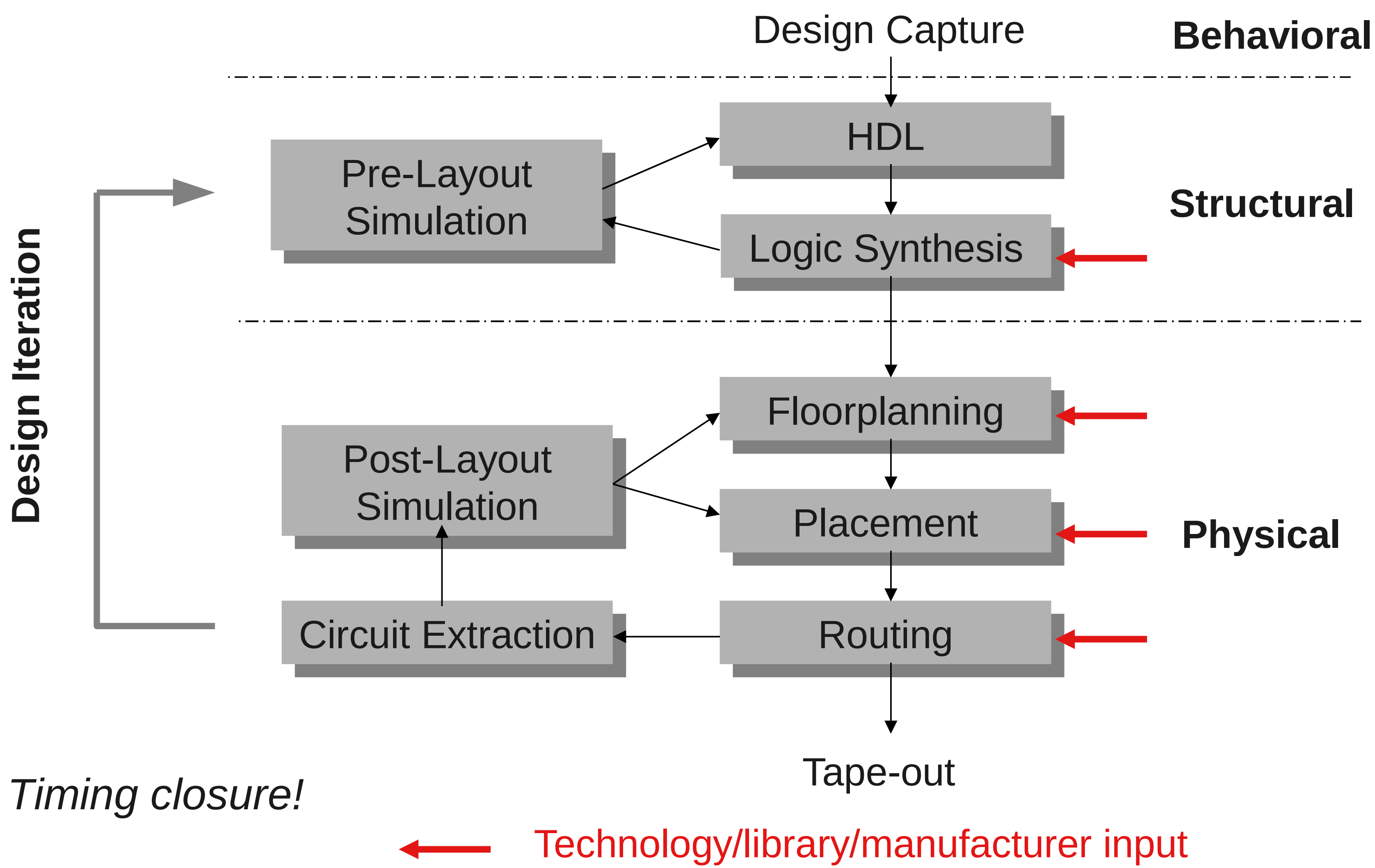
- **Area & Energy**
  - Smaller ALU
  - Less storage

- **Physical Security**
  - Side-channel analysis
  - Fault analysis



- **Performance**
  - Fast field arithmetic
  - Fast group operations

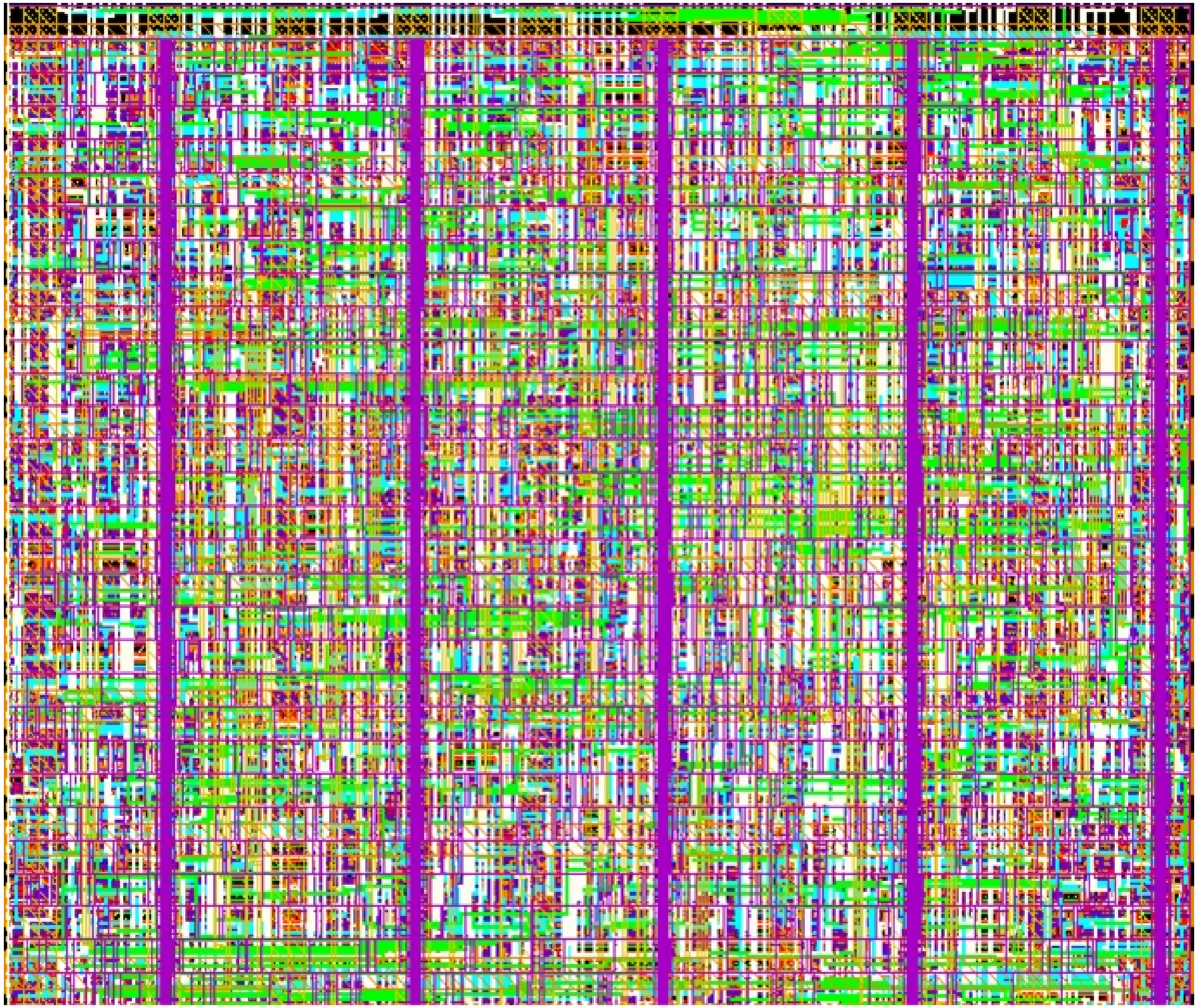
➢ Hardware design flow



Timing closure!

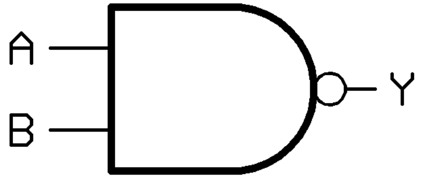
Slides courtesy: Prof. Ingrid Verbauwhede

- Layout of an integrated circuit



- Area
    - Gate Equivalent (GE): equivalent of NAND gates

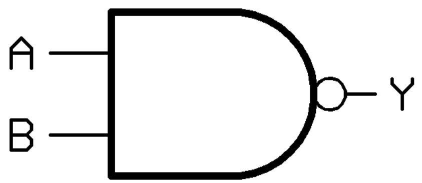
- Area
  - Gate Equivalent (GE): equivalent of NAND gates



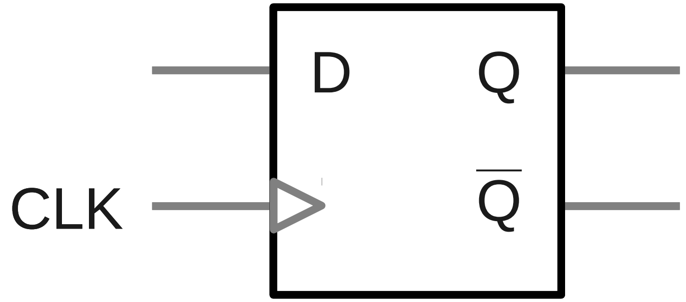
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Area

- Gate Equivalent (GE): equivalent of NAND gates



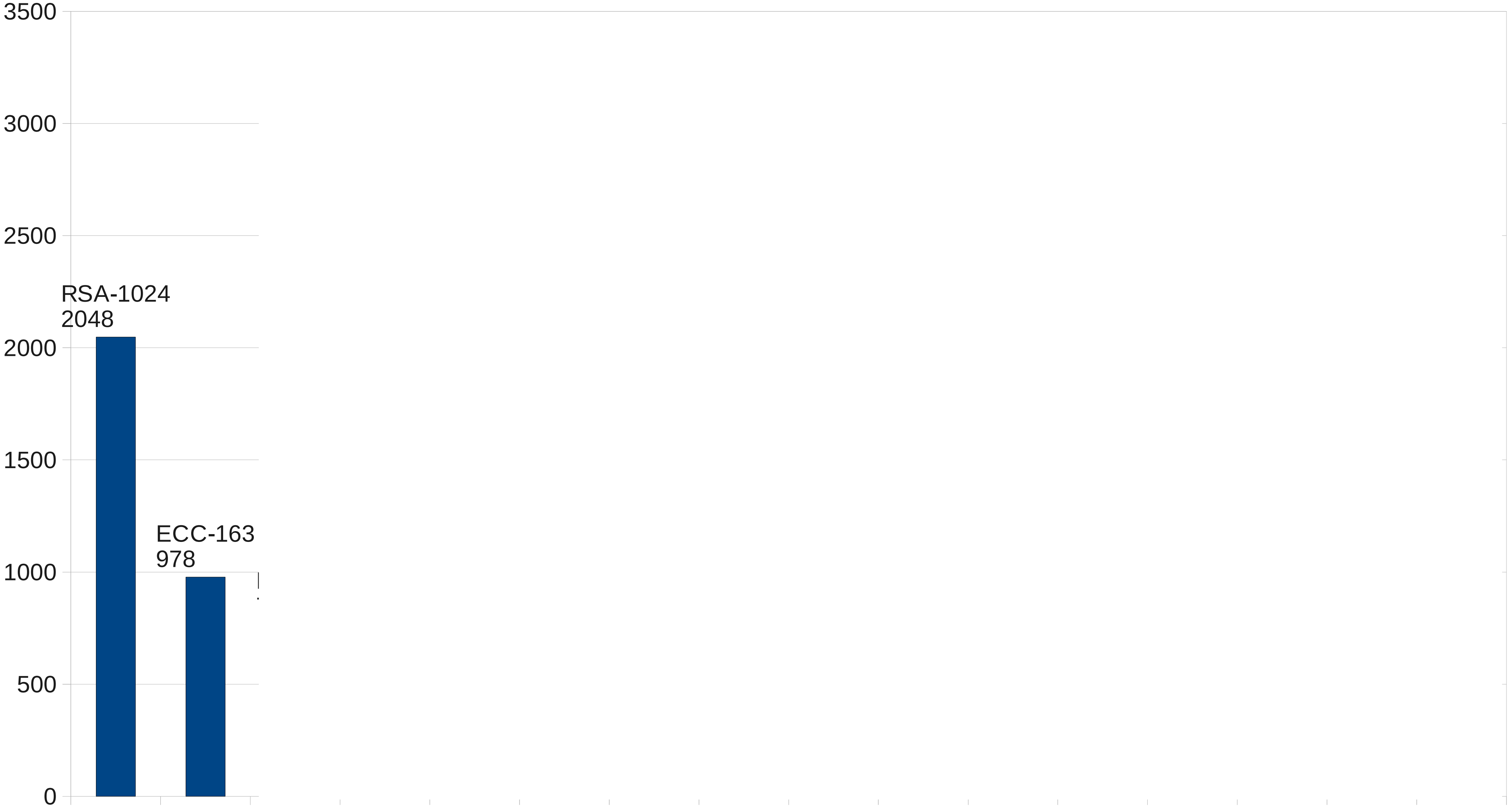
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



D Flip-Flop (≈ 6 GE)

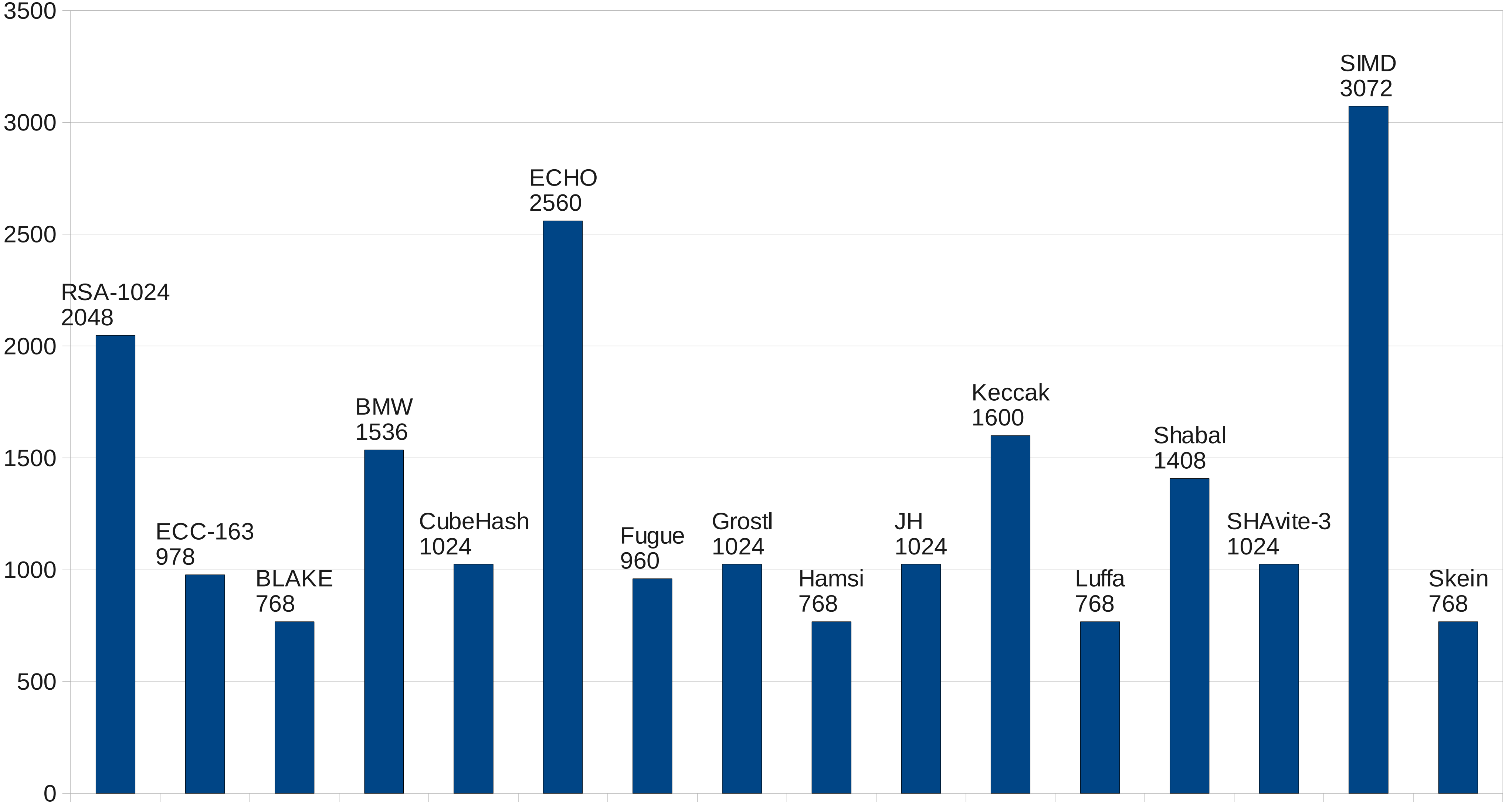
CLK	Q	$\overline{Q}$
rising edge	D	$\overline{D}$
falling edge	Q	$\overline{Q}$

- Memory requirement



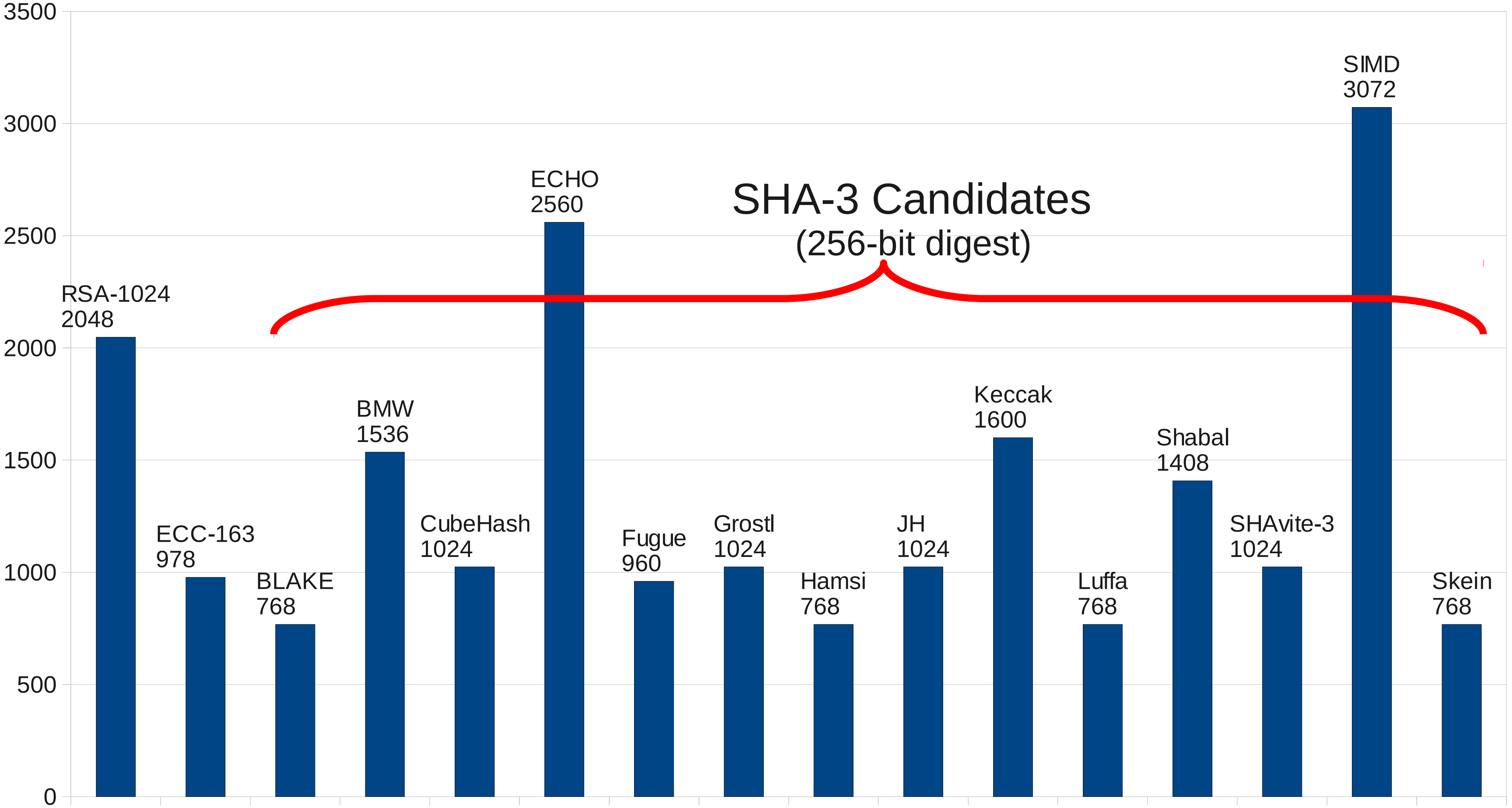
* Ideguchi *et al*, 2009

➢ Memory requirement



* Ideguchi *et al*, 2009

- Memory requirement



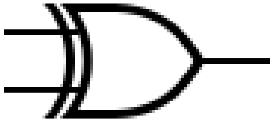
* Ideguchi *et al*, 2009

- Let's make an ECC processor

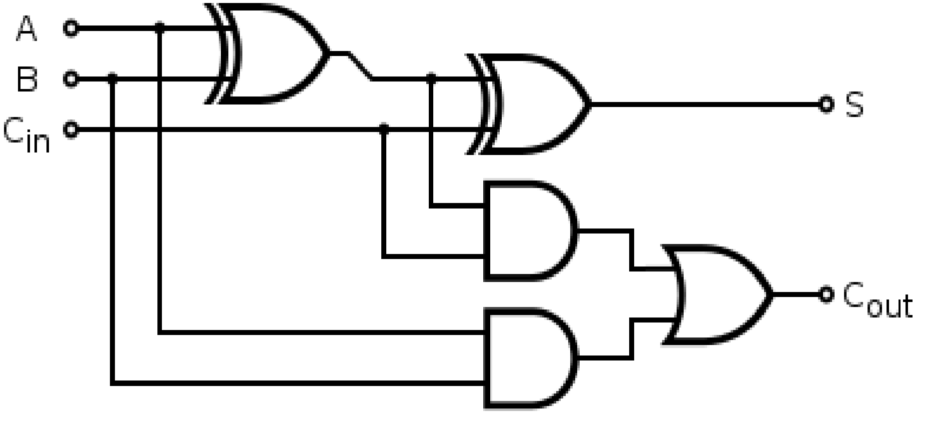
  - Binary fields v.s. Prime fields
  - Security level
  - Coorinate systems
  - Representation of field elements
  - Architecture
  - Physical security properties

➢ $\mathbf{F}_{2^m}$ v.s. $\mathbf{F}_p$

- Use binary fields instead of prime fields
  - No carry bits, smaller and faster ALU

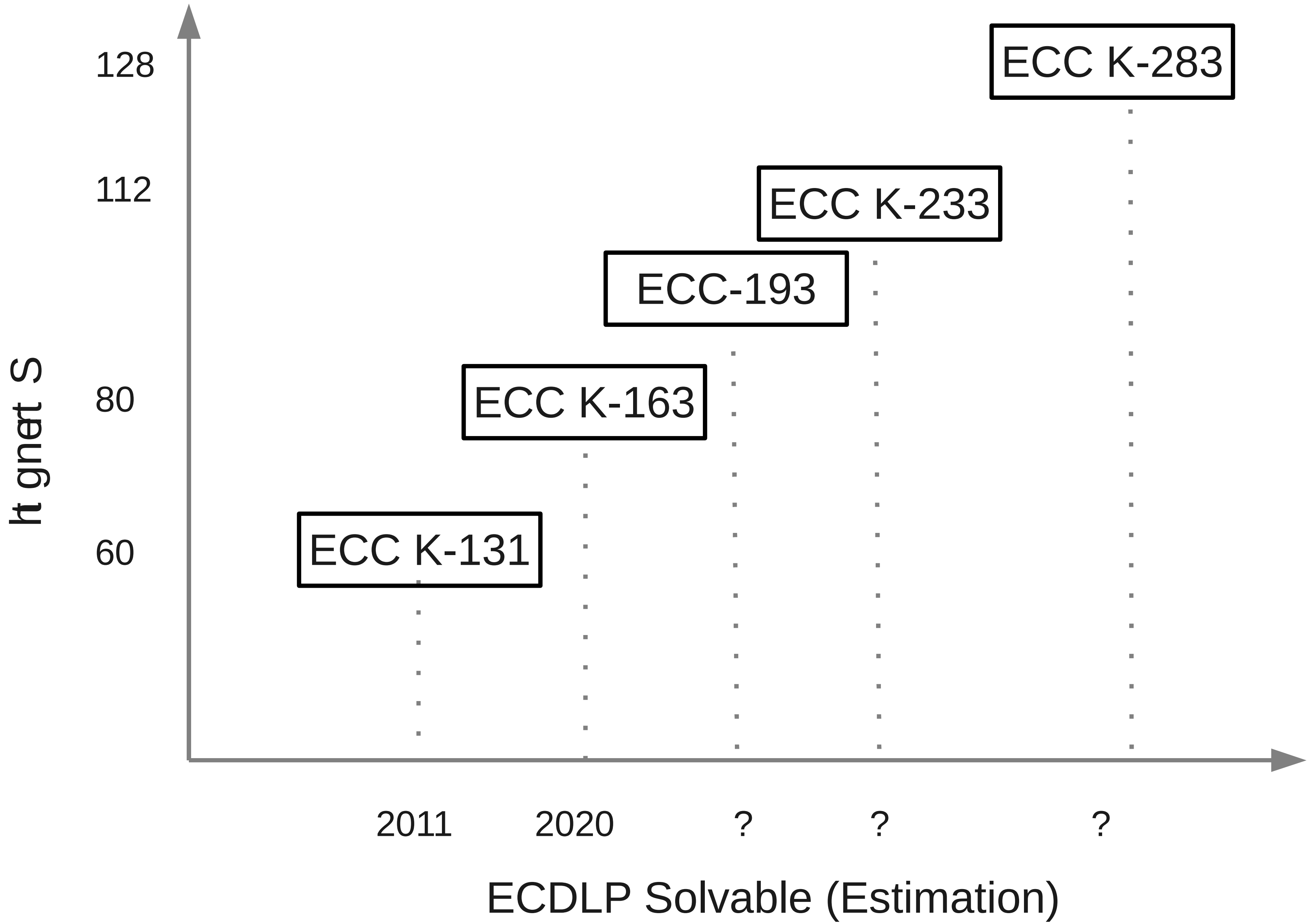


1-bit Add in GF(2^m)

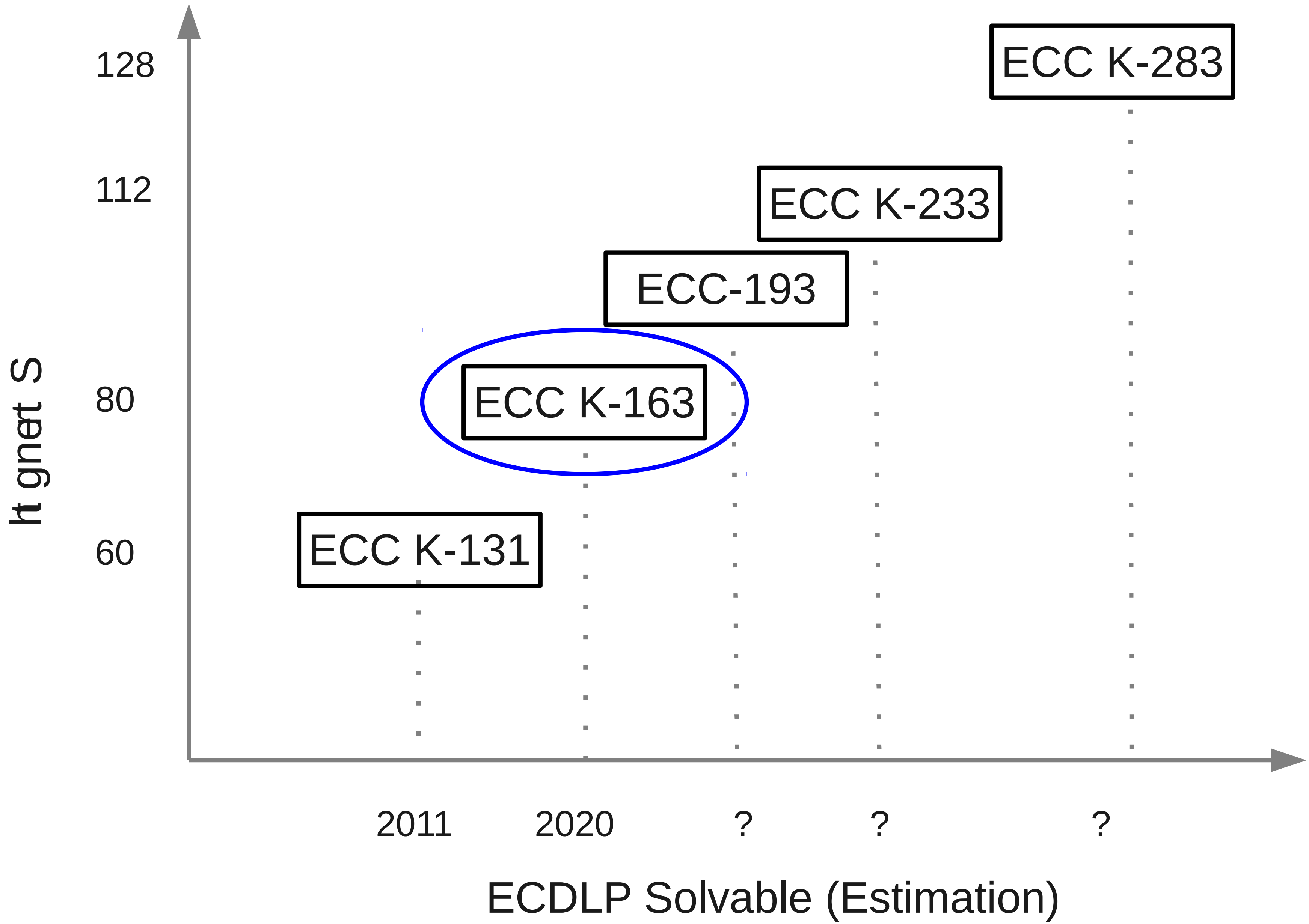


1-bit full-adder

- Security level



- Security level



- Coordinate systems

Coordinates	Point Representation	Inversion	Point Multiplication
Affine	**P1**=(x_1, y_1) **P2**=(x_2, y_2)	Each key bit	-
Projective	**P1**=(X_1, Y_1, Z_1) **P2**=(X_2, Y_2, Z_2)	One	-
López-Dahab (Affine)	**P1**=(x_1) **P2**=(x_2)	Each key bit	Montgomery Ladder (**P2** = **P1** + **P**)
López-Dahab (Projective)	**P1**=(X_1, Z_1) **P2**=(X_2, Z_2)	One	
* W-coordinate (Affine)	**P1**=(w_1) **P2**=(w_2)	Each key bit	
* W-coordinate (Projective)	**P1**=(W_1, Z_1) **P2**=(W_2, Z_2)	One	

* Binary Edwards Curve only

- Coordinate systems

Coordinates	Point Representation	Inversion	Point Multiplication
Affine	$\mathbf{P_1}=(x_1, y_1)$ $\mathbf{P_2}=(x_2, y_2)$	Each key bit	-
Projective	$\mathbf{P_1}=(X_1, Y_1, Z_1)$ $\mathbf{P_2}=(X_2, Y_2, Z_2)$	One	-
López-Dahab (Affine)	$\mathbf{P_1}=(x_1)$ $\mathbf{P_2}=(x_2)$	Each key bit	Montgomery Ladder ($\mathbf{P_2} = \mathbf{P_1} + \mathbf{P}$)
López-Dahab (Projective)	$\mathbf{P_1}=(X_1, Z_1)$ $\mathbf{P_2}=(X_2, Z_2)$	One	
* W-coordinate (Affine)	$\mathbf{P_1}=(w_1)$ $\mathbf{P_2}=(w_2)$	Each key bit	
* W-coordinate (Projective)	$\mathbf{P_1}=(W_1, Z_1)$ $\mathbf{P_2}=(W_2, Z_2)$	One	

* Binary Edwards Curve only

- Count the number of registers

Algorithm 1: Montgomery Powering Ladder

Input: $k=\{1, k_{t-1},..,k_0\}$ and point **P**

Output: $[k]$**P**

1: $P_1 \leftarrow P$, $P_2 \leftarrow [2]P$

2: for $i=t-1$ to 0 do

3: if $k_i=1$ then

$P_1 \leftarrow P_1 + P_2$, $P_2 \leftarrow [2]P_2$

else

$P_2 \leftarrow P_1 + P_2$, $P_1 \leftarrow [2]P_1$

4: end for

Return P_1

Point Addition: $(X_1, Z_1) + (X_2, Z_2)$	**Point Doubling:** $2(X_1, Z_1)$
$T_1 \leftarrow x_0$	$T_1 \leftarrow c$
$X_1 \leftarrow X_1 \cdot X_2$	$X_1 \leftarrow X_1^2$
$Z_1 \leftarrow Z_1 \cdot X_2$	$Z_1 \leftarrow Z_1^2$
$T_2 \leftarrow X_1 \cdot Z_1$	$T_1 \leftarrow Z_1 \cdot T_1$
$Z_1 \leftarrow X_1 + Z_1$	$Z_1 \leftarrow X_1 \cdot Z_1$
$Z_1 \leftarrow Z_1^2$	$T_1 \leftarrow T_1^2$
$X_1 \leftarrow T_1 \cdot Z_1$	$X_1 \leftarrow X_1^2$
$X_1 \leftarrow X_1 + T_2$	$X_1 \leftarrow X_1 + T_1$
Register: 7 **Mul. : 4** **Sqr. : 1**	**Register: 3** **Mul. : 2** **Sqr. : 4**

Common-Z trick (7 --> 6)

- 7 registers in total:

  (x_0, X_1, Z_1, X_2, Z_2, T_1, T_2)

- Further reduction:

  (x_0, X_1, X_2, Z, T_1, T_2)

  $X_1 \leftarrow X_1 \cdot Z_2$

  $X_2 \leftarrow X_2 \cdot Z_1$

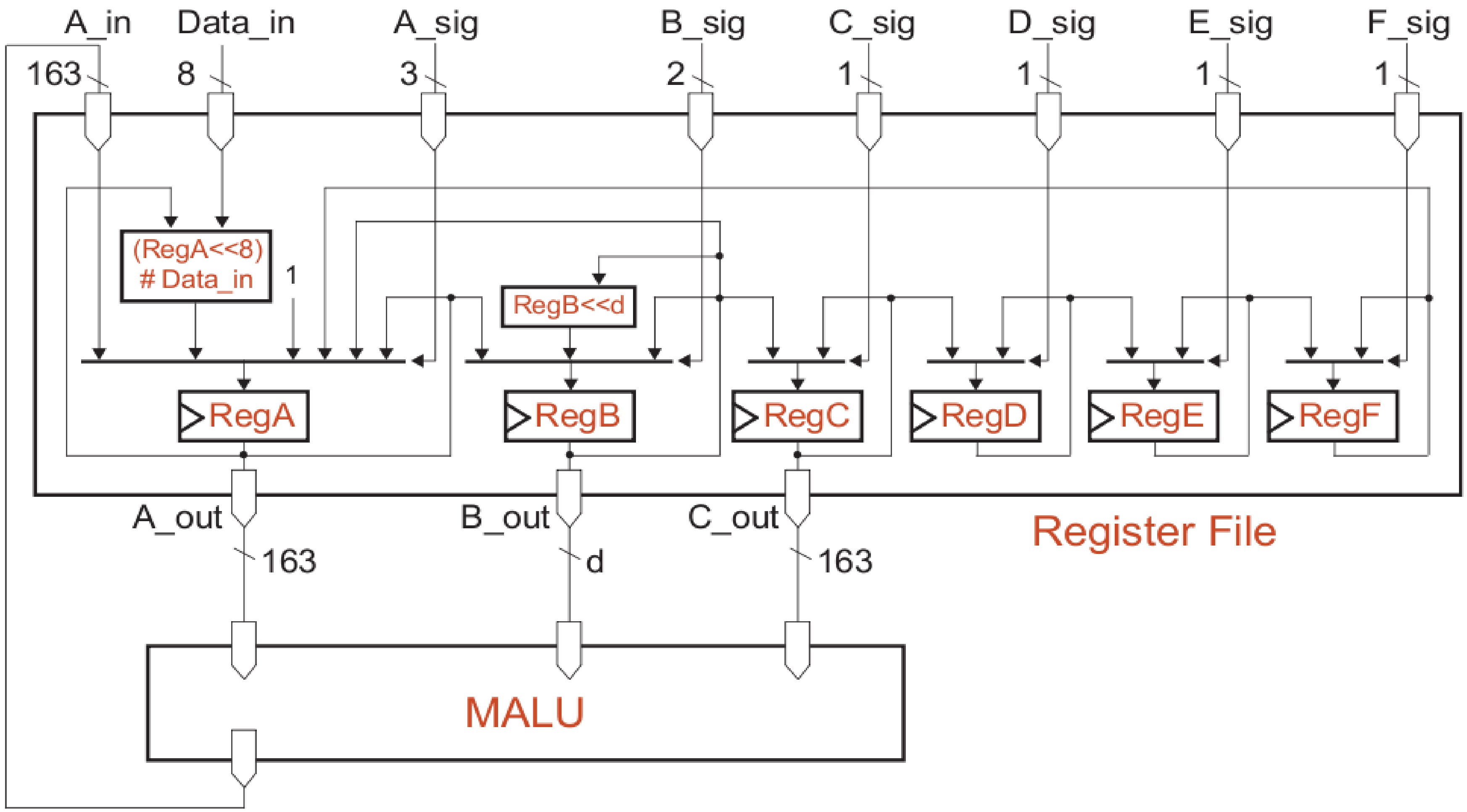
  $Z \leftarrow Z_1 \cdot Z_2$

- Cost for one iteration:

  $6M+5S \rightarrow 7M+4S$

Point Addition: $(X_1, Z_1) + (X_2, Z_2)$	**Point Doubling:** $2(X_1, Z_1)$
$T_1 \leftarrow x_0$	$T_1 \leftarrow c$
$X_1 \leftarrow X_1 \cdot X_2$	$X_1 \leftarrow X_1^2$
$Z_1 \leftarrow Z_1 \cdot X_2$	$Z_1 \leftarrow Z_1^2$
$T_2 \leftarrow X_1 \cdot Z_1$	$T_1 \leftarrow Z_1 \cdot T_1$
$Z_1 \leftarrow X_1 + Z_1$	$Z_1 \leftarrow X_1 \cdot Z_1$
$Z_1 \leftarrow Z_1^2$	$T_1 \leftarrow T_1^2$
$X_1 \leftarrow T_1 \cdot Z_1$	$X_1 \leftarrow X_1^2$
$X_1 \leftarrow X_1 + T_2$	$X_1 \leftarrow X_1 + T_1$
Register: 7	**Register: 3**
Mul. : 4	**Mul. : 2**
Sqr. : 1	**Sqr. : 4**

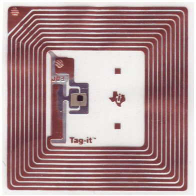
- Circular-shift register file



Slides courtesy: Yongki Lee

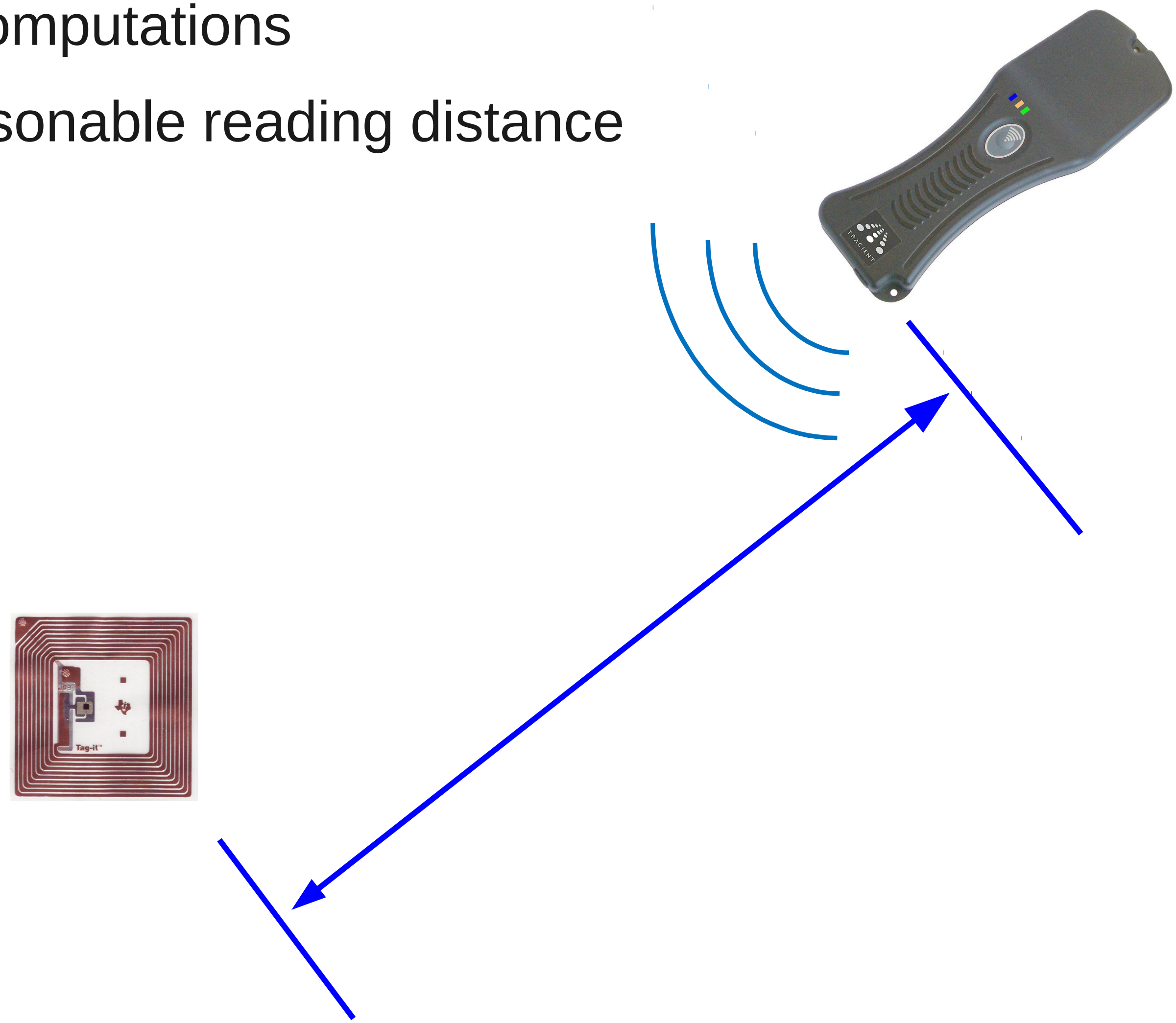
- Power & Energy

- Power & Energy
    - To support the computations

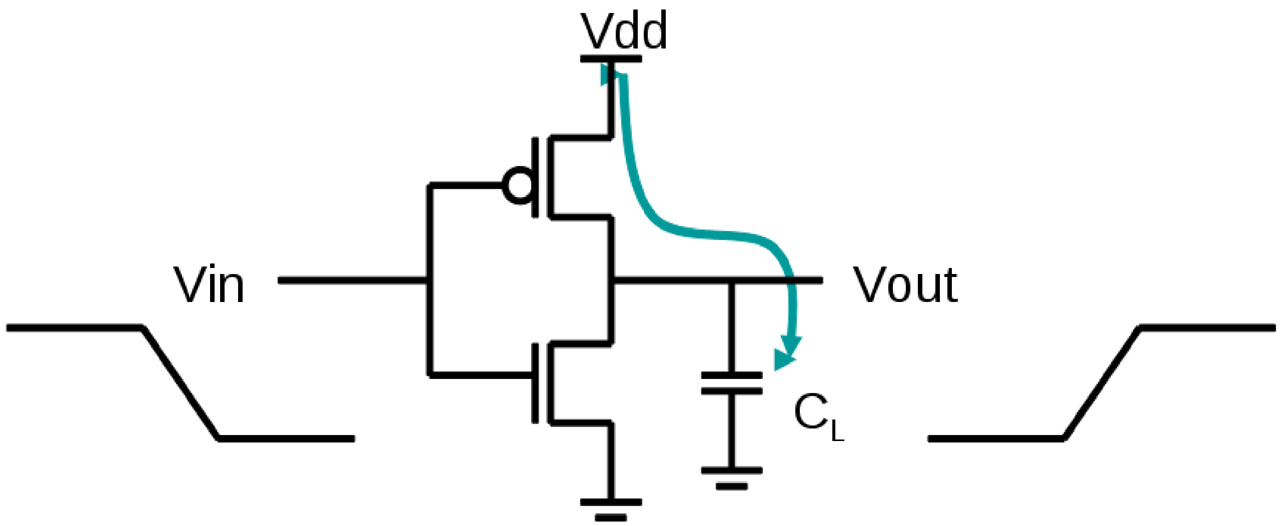


- Power & Energy

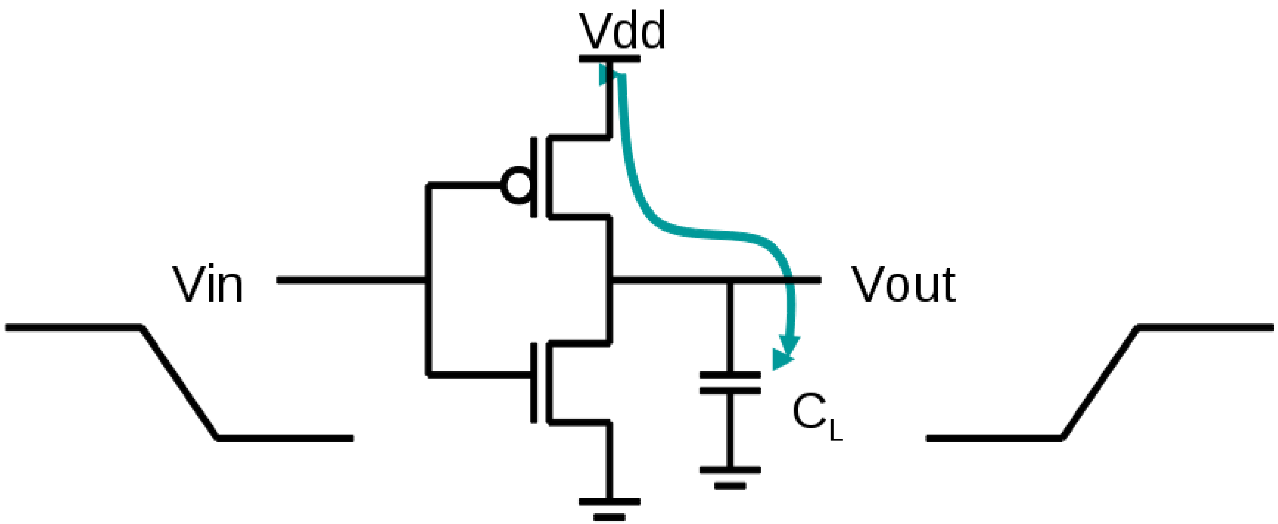
  - To support the computations
  - To support a reasonable reading distance



- Power & Energy



- Power & Energy



$$\underline{P_d} = \underline{\alpha}\ \underline{C}\ \underline{V^2} \underline{f}$$

Dynamic Power — Switch Activity — Output capacitance — Vdd — Clock Frequency

- A bit-serial multiplier

Input: $A(x) = \{a_{m-1}, a_{m-2} \ldots a_1, a_0\}$,

$B(x) = \{b_{m-1}, b_{m-2} \ldots b_1, b_0\}$,

and $P(x) = \{1, p_{m-1} \ldots p_1, 1\}$

Output: $C(x) = A(x)B(x)$ mod $P(x)$

1: $C(x) \leftarrow 0$;

2: **for** $i = m-1$ to 0 **do**

3: $C(x) \leftarrow xC(x) + b_i A(x)$;

$C(x) \leftarrow C(x)$ mod $P(x)$;

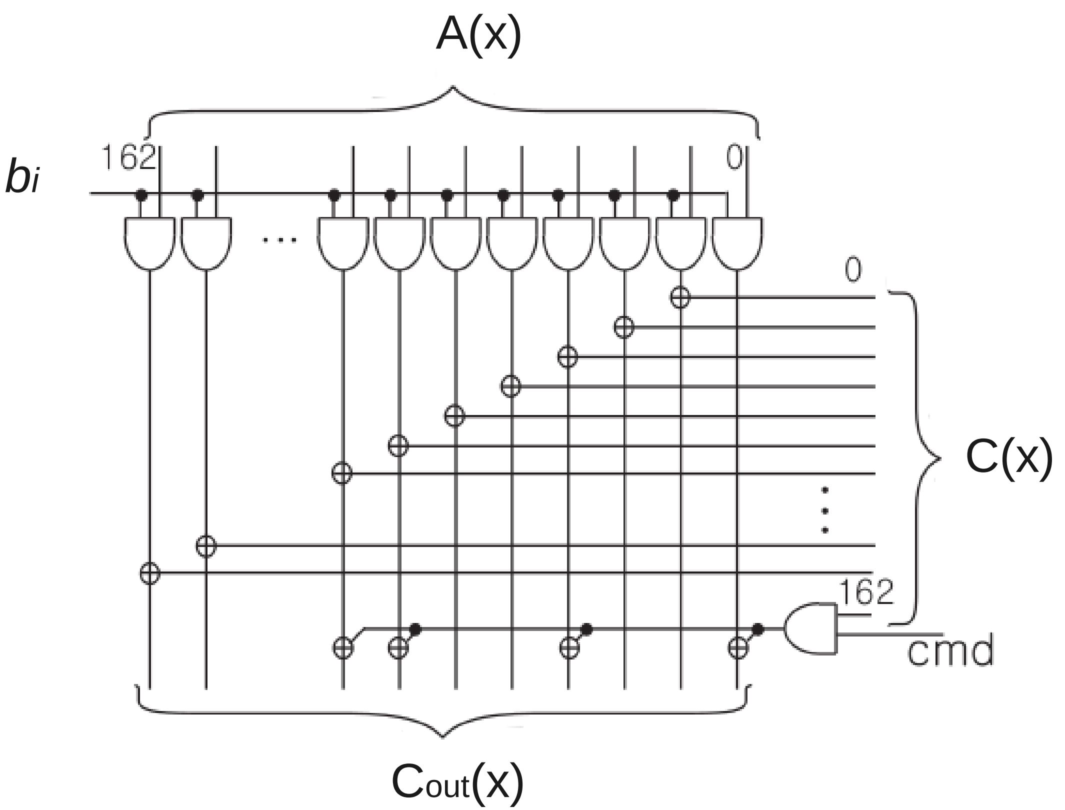
4: ***end for***

Return: $C(x)$

A bit-serial multiplier

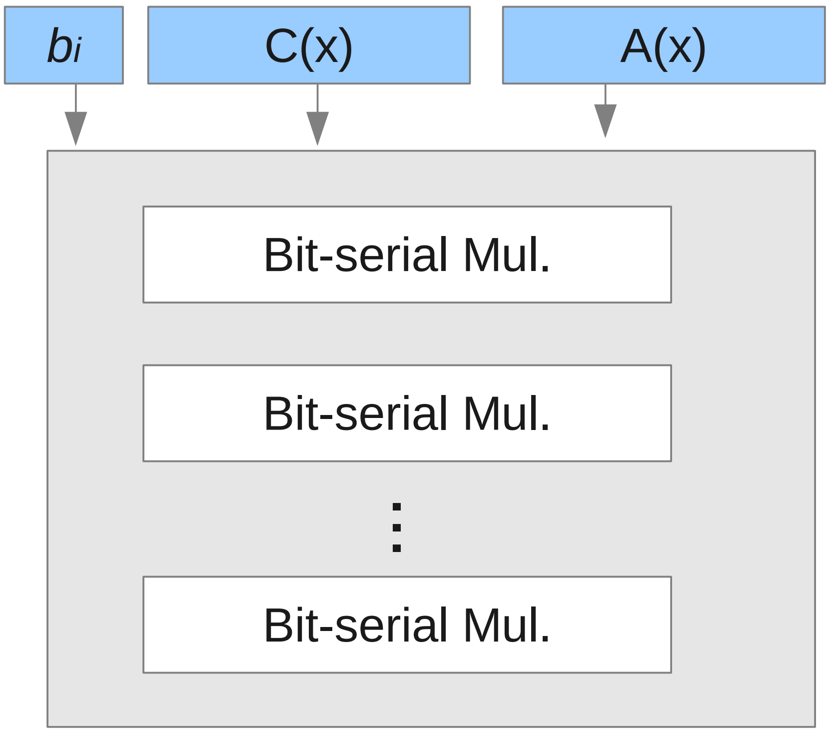
Input: $A(x)=\{a_{m-1},a_{m-2}\ldots a_1,a_0\}$,
$B(x)=\{b_{m-1},b_{m-2}\ldots b_1,b_0\}$,
and $P(x)=\{1,p_{m-1}\ldots p_1,1\}$
Output: $C(x) = A(x)B(x) \bmod P(x)$

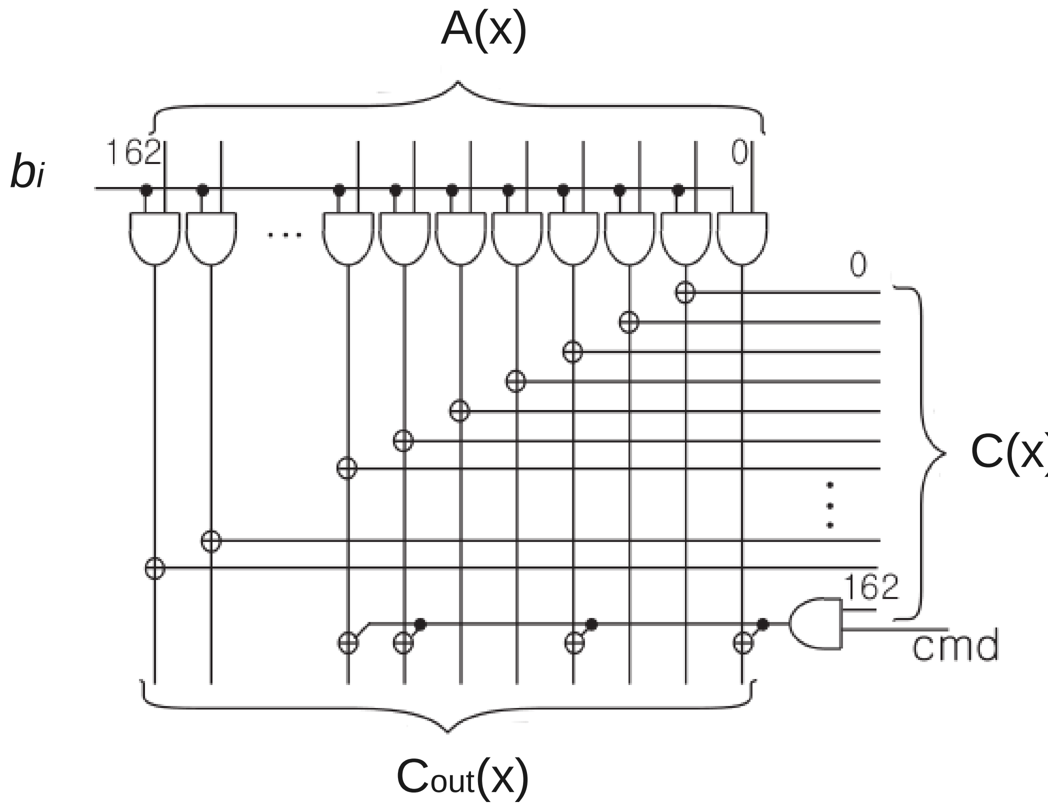
1: $C(x) \leftarrow 0$;
2: **for** $i = m-1$ to 0 **do**
3: $C(x) \leftarrow xC(x) + b_i A(x)$;
$C(x) \leftarrow C(x) \bmod P(x)$;
4: ***end for***
Return: $C(x)$



Bit-serial multiplier
[Delay: ≈ m cycles]

- Power & Energy





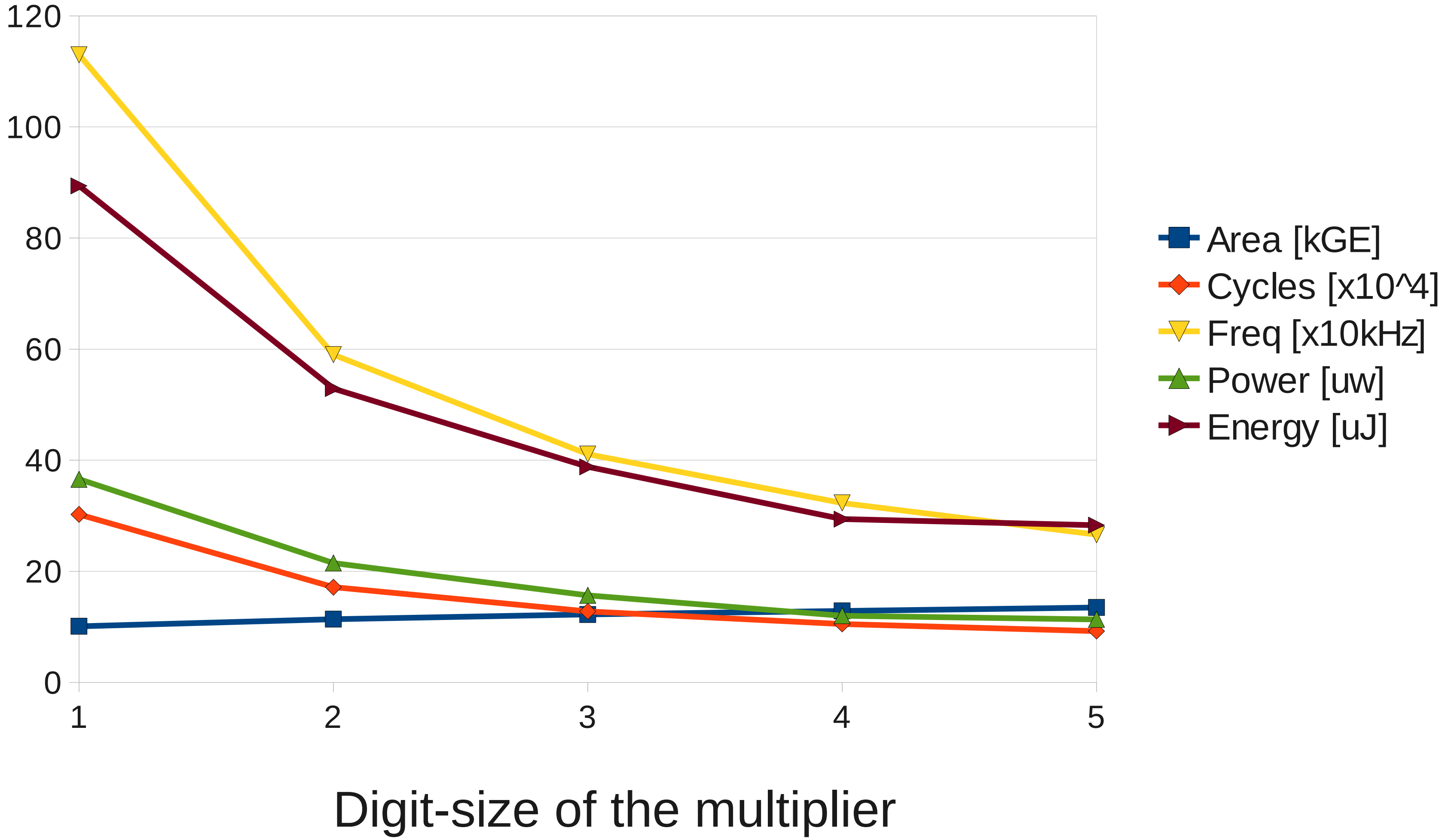
Digit-serial Multiplier
[Delay: ≈ m/d cycles]

Bit-serial multiplier
[Delay: ≈ m cycles]

- Power & Energy
    - Target : One point multiplication within 0.25s

Power & Energy

- Target : One point multiplication within 0.25s



- Physical attacks

- Physical attacks

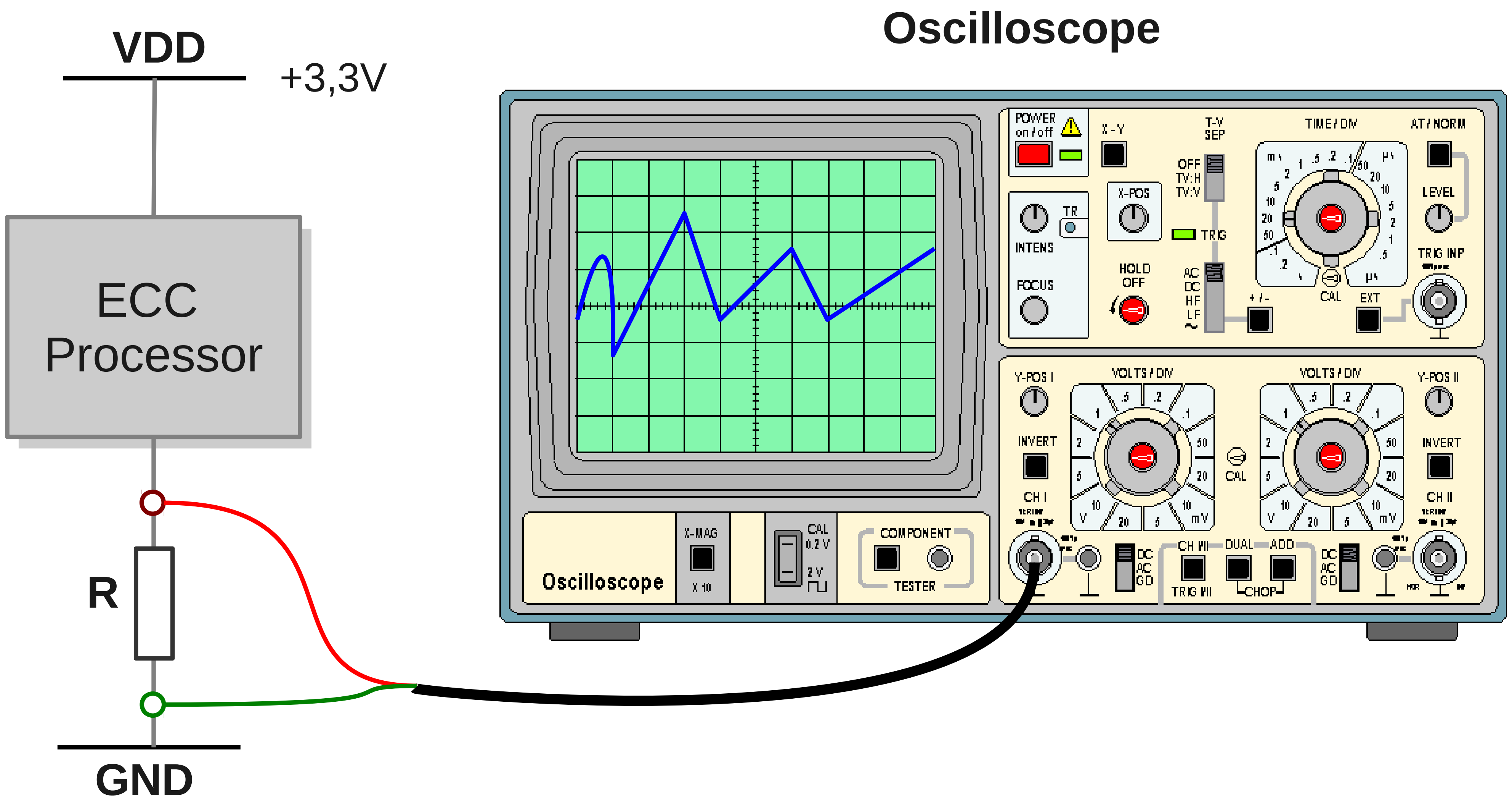
Side-Channel Analysis

- Physical attacks

Side-Channel Analysis

Fault Analysis

- Power analysis



Simple power analysis

$k = (k_{l-1}, k_{l-2}, ..., k_0)$

Left-to-right binary method for point multiplication

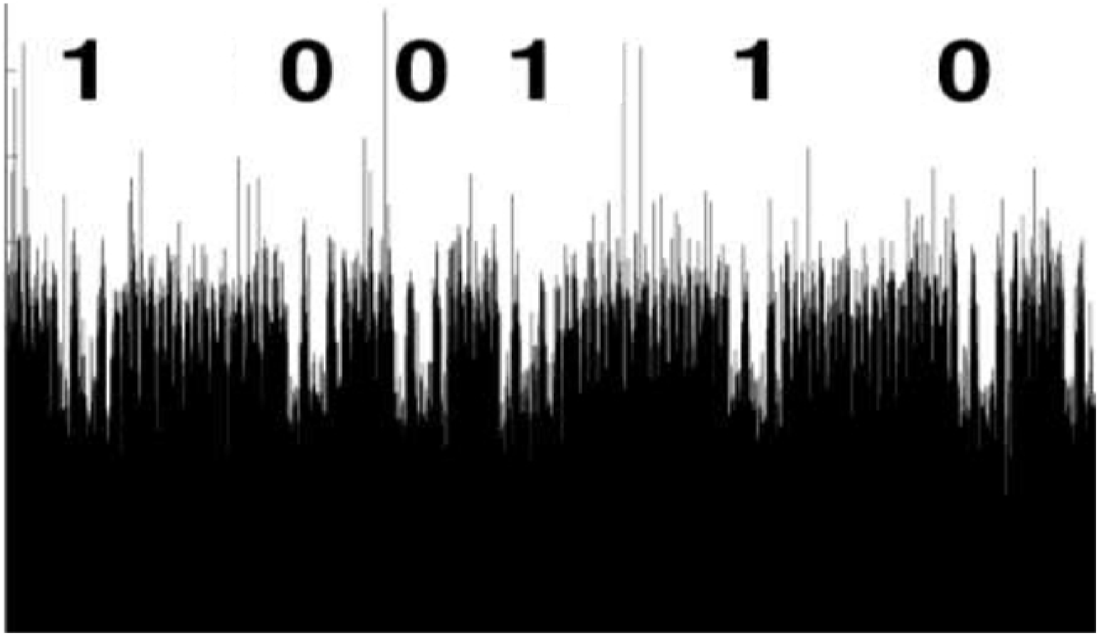
```
R ← O
for i=l-1 downto 0 do
    R ← [2]R
    if ki = 1 then
        R ← R + P
    end if
end for
```

- Simple power analysis

$k = (k_{l-1}, k_{l-2}, ..., k_0)$

Left-to-right binary method for point multiplication

```
R ← O
for i=l-1 downto 0 do
    R ← [2]R
    if ki = 1 then
        R ← R + P
    end if
end for
```



- Montgomery Ladder?

Algorithm 1: Montgomery Powering Ladder

Input: $k=\{1, k_{t-1},..,k_0\}$ and point **P**

Output: $[k]\mathbf{P}$

1: $\mathbf{P_1} \leftarrow \mathbf{P}$, $\mathbf{P_2} \leftarrow [2]\mathbf{P}$

2: for $i=t-1$ to 0 do

3: if $k_i=1$ then

$\mathbf{P_1} \leftarrow \mathbf{P_1} + \mathbf{P_2}$, $\mathbf{P_2} \leftarrow [2]\mathbf{P_2}$

else

$\mathbf{P_2} \leftarrow \mathbf{P_1} + \mathbf{P_2}$, $\mathbf{P_1} \leftarrow [2]\mathbf{P_1}$

4: end for

Return P_1

- Montgomery Ladder?

Algorithm 1: Montgomery Powering Ladder

Input: $k=\{1, k_{t-1},..,k_0\}$ and point **P**

Output: $[k]$**P**

1: $\mathbf{P}_1 \leftarrow \mathbf{P}$, $\mathbf{P}_2 \leftarrow [2]\mathbf{P}$

2: for $i=t-1$ to 0 do

3: if $k_i=1$ then

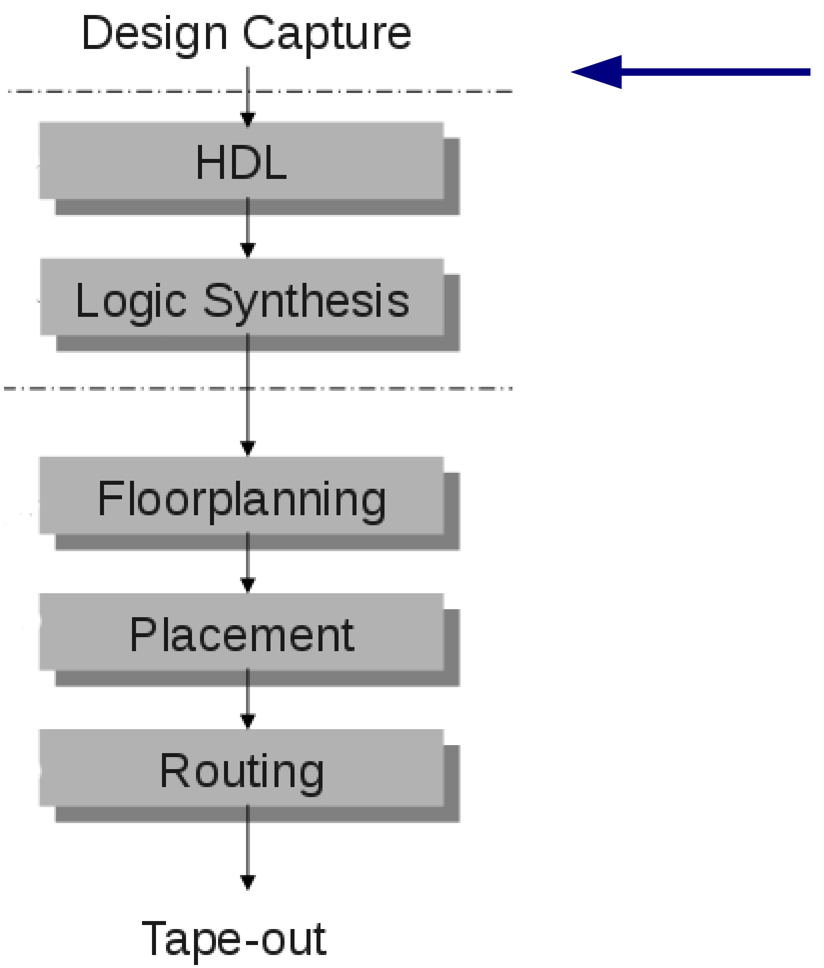
$\mathbf{P}_1 \leftarrow \mathbf{P}_1 + \mathbf{P}_2$, $\mathbf{P}_2 \leftarrow [2]\mathbf{P}_2$

else

$\mathbf{P}_2 \leftarrow \mathbf{P}_1 + \mathbf{P}_2$, $\mathbf{P}_1 \leftarrow [2]\mathbf{P}_1$

4: end for

Return P_1



- Montgomery Ladder?

Algorithm 1: Montgomery Powering Ladder

Input: $k=\{1, k_{t-1},..,k_0\}$ and point **P**

Output: $[k]\mathbf{P}$

1: $\mathbf{P_1} \leftarrow \mathbf{P}$, $\mathbf{P_2} \leftarrow [2]\mathbf{P}$

2: for $i=t-1$ to 0 do

3: if $k_i=1$ then

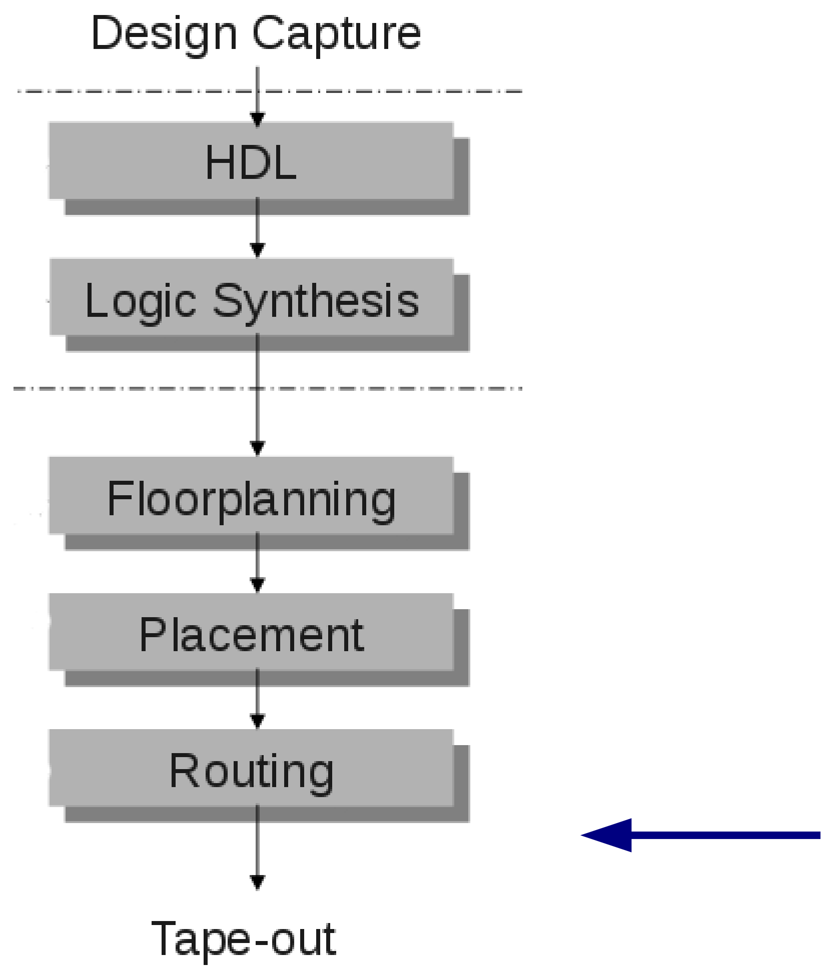
$\mathbf{P_1} \leftarrow \mathbf{P_1} + \mathbf{P_2}$, $\mathbf{P_2} \leftarrow [2]\mathbf{P_2}$

else

$\mathbf{P_2} \leftarrow \mathbf{P_1} + \mathbf{P_2}$, $\mathbf{P_1} \leftarrow [2]\mathbf{P_1}$

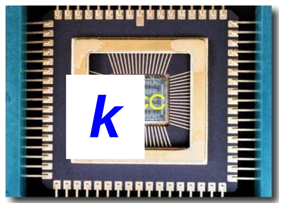
4: end for

Return P_1



- Differential power analysis

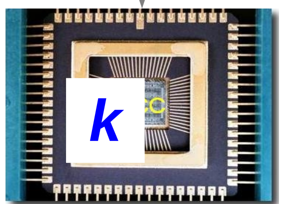
- Differential power analysis



Power Model

➢ Differential power analysis

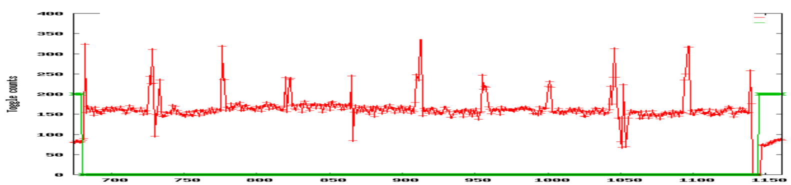
P_1, P_2, …, P_n



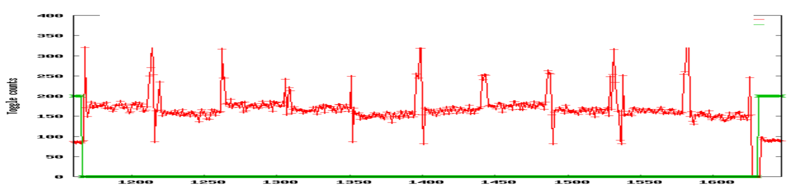
[*k*]P_1, [*k*]P_2, …,[*k*]P_n

Power Model

[*k*]P_1

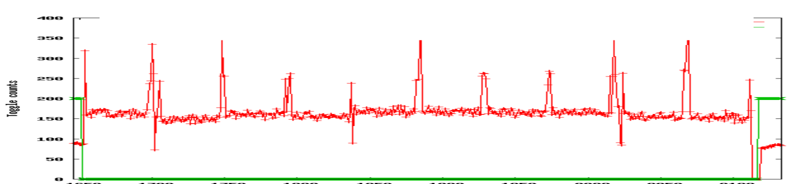


[*k*]P_2



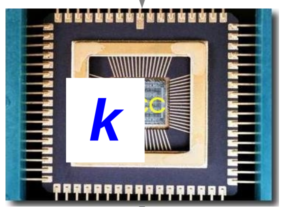
⋮ ⋮

[*k*]P_n



➢ Differential power analysis

P_1, P_2, …, P_n

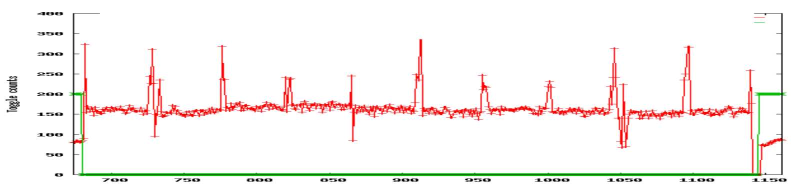


Key guess $k=k'$

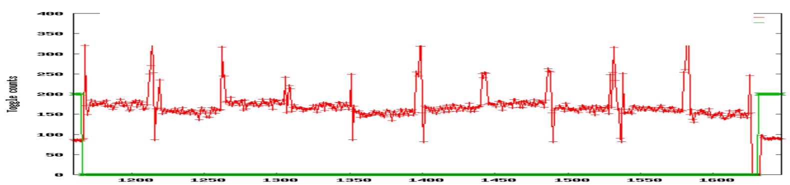
Power Model

[k]P_1, [k]P_2, …,[k]P_n

[k]P_1

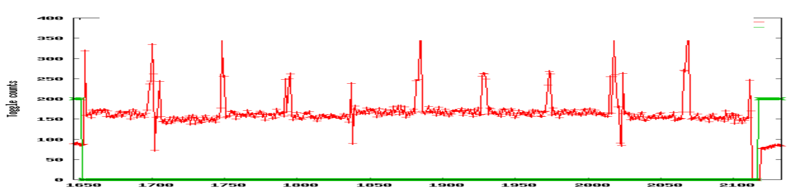


[k]P_2



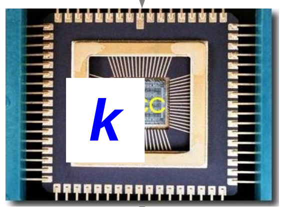
⋮ ⋮

[k]P_n



- Differential power analysis

$P_1, P_2, \ldots, P_n$



$[k]P_1, [k]P_2, \ldots, [k]P_n$

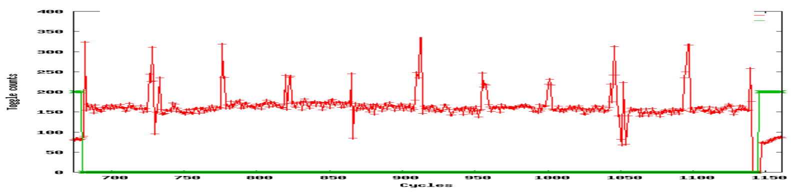
Key guess $k=k'$

Power Model

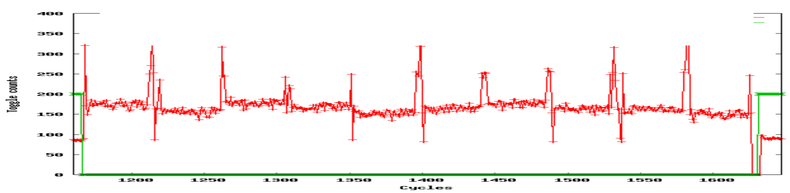
$P_1, P_2, \ldots, P_n$

$[k']P_1, [k']P_2, \ldots, [k']P_n$

$[k]P_1$

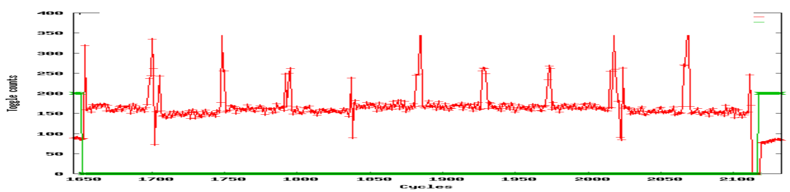


$[k]P_2$

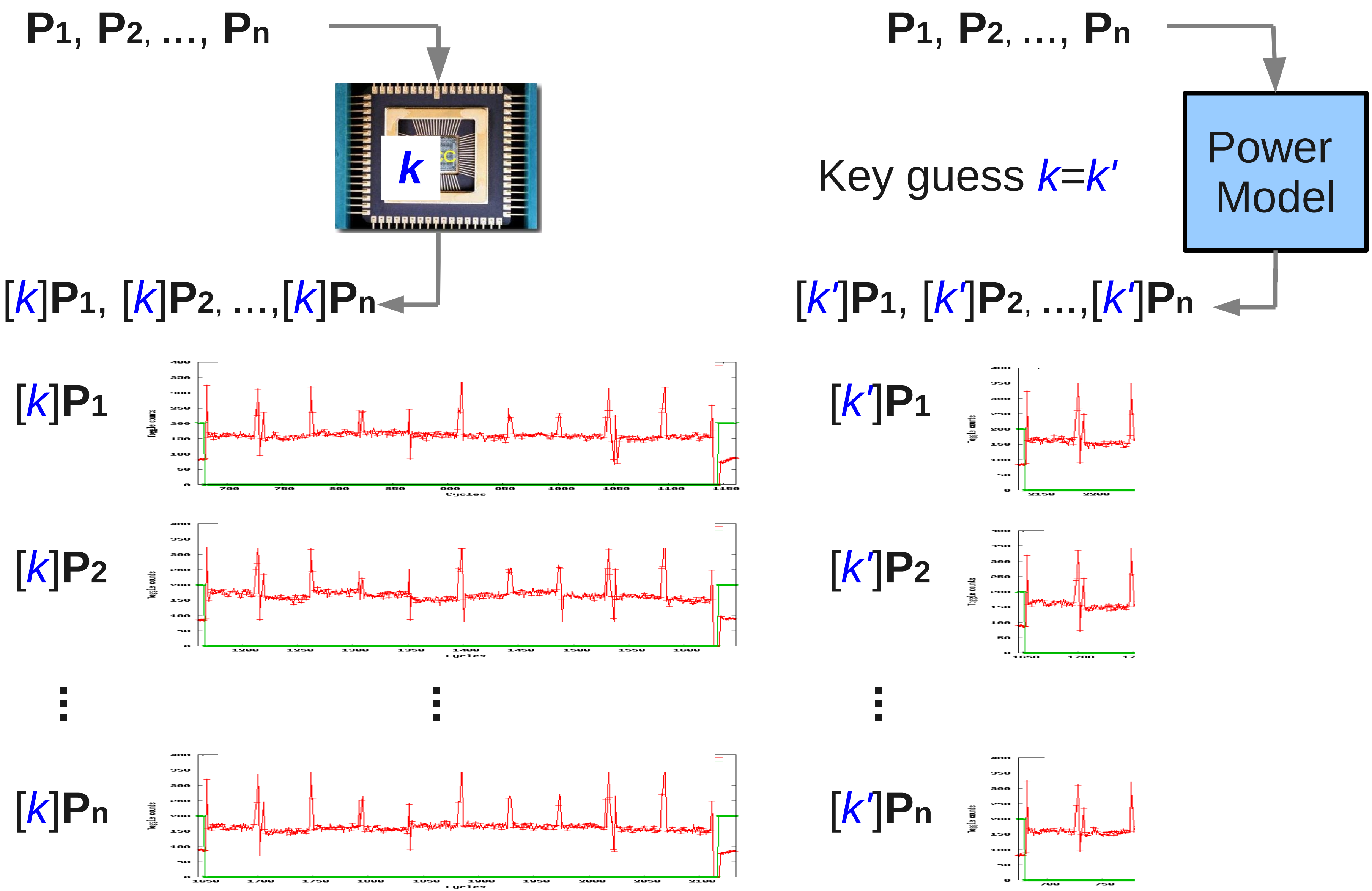


⋮ ⋮

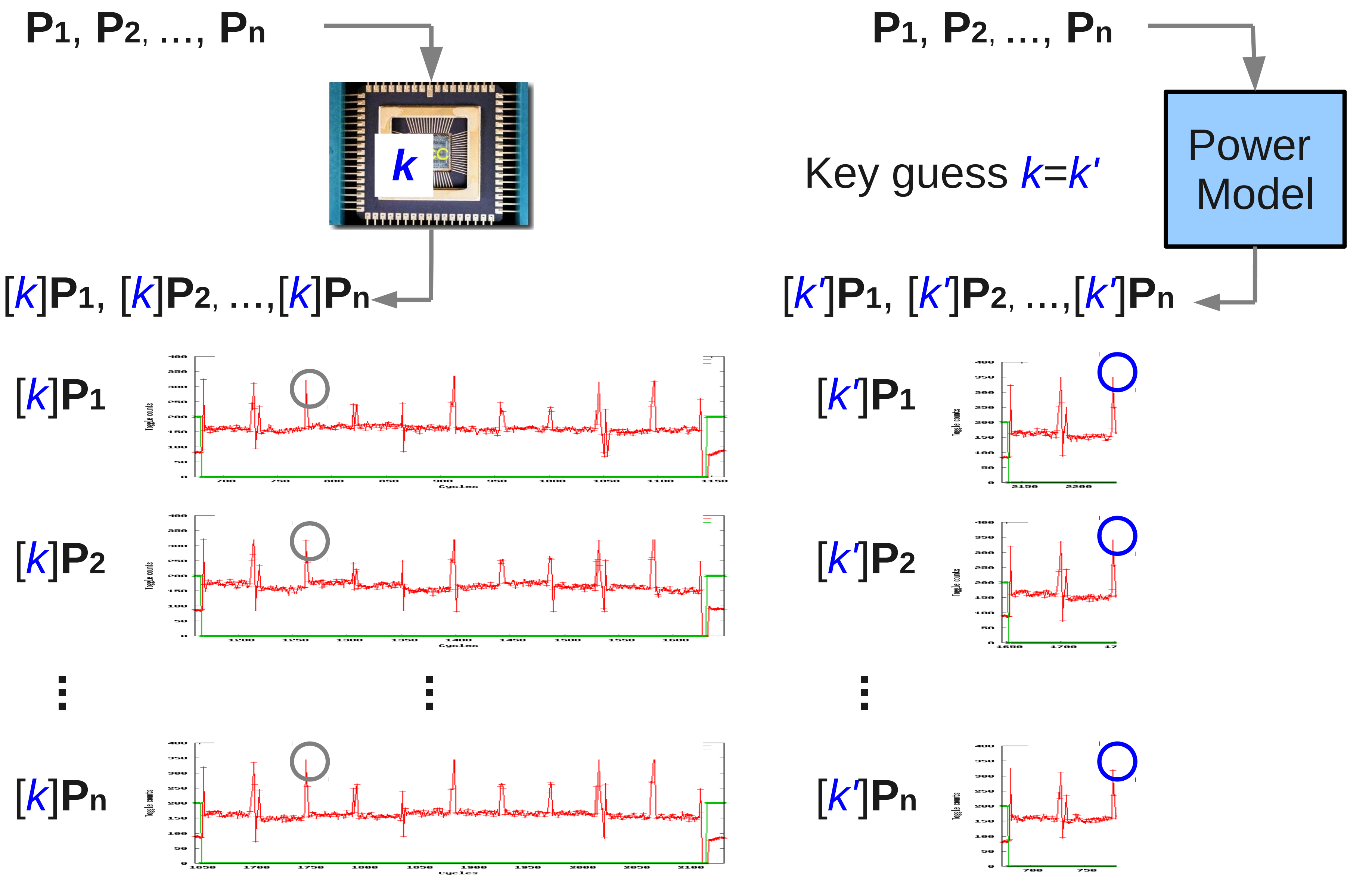
$[k]P_n$



- Differential power analysis



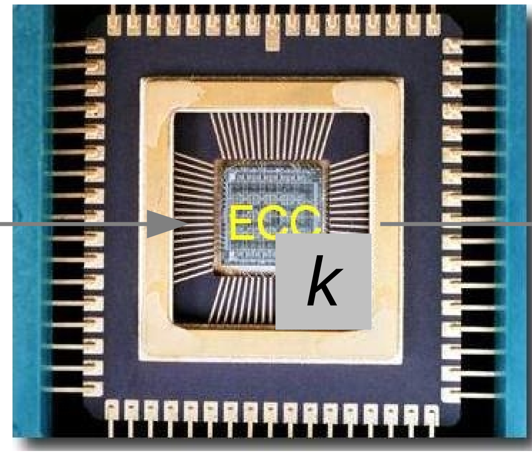
- Differential power analysis



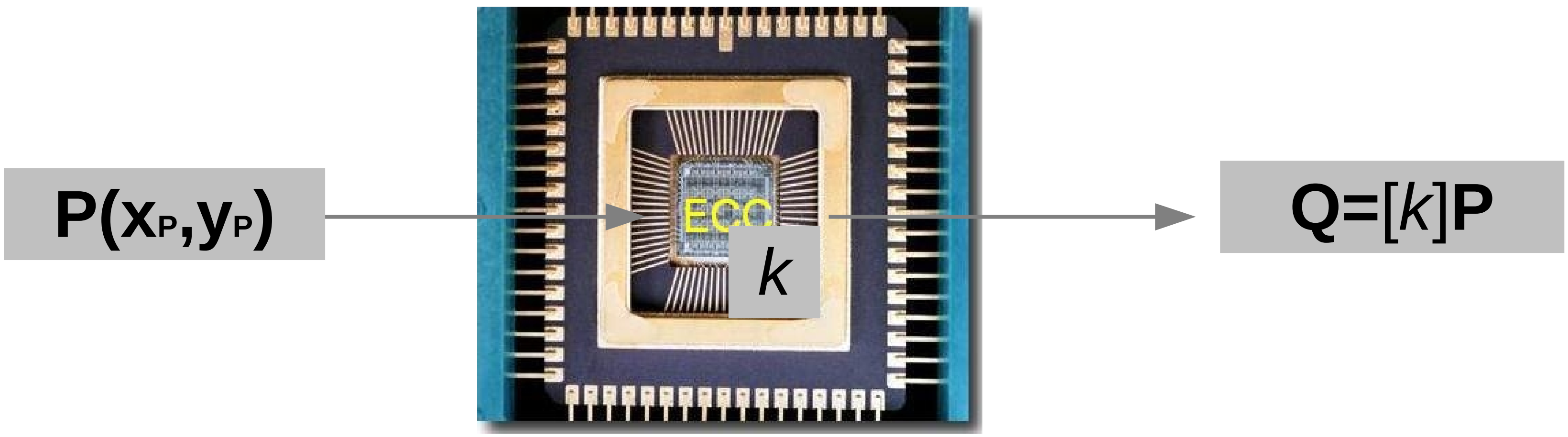
- Fault analysis

- Fault analysis

- Fault analysis (weak curve) [Biehl+'00]

P(x_P,y_P) →  → **Q=[*k*]P**

- Fault analysis (weak curve) [Biehl+'00]

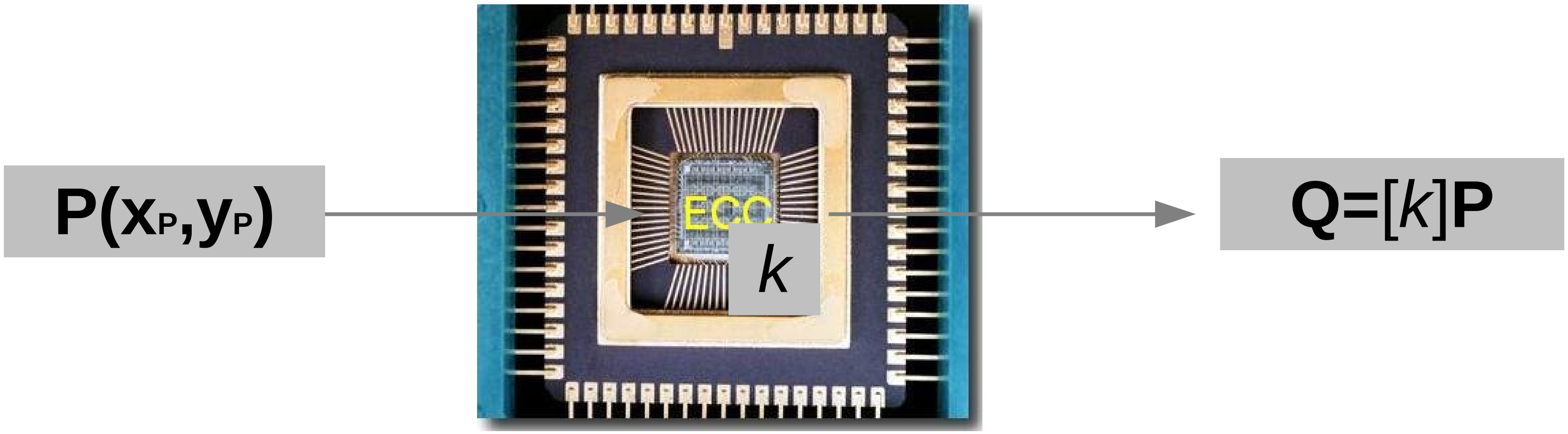


- The specified curve is:

$$\mathbf{E} : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6,$$

and **P**($\mathbf{x_P}$,$\mathbf{y_P}$) is on **E.**

➢ Fault analysis (weak curve) [Biehl+'00]



- The specified curve is:

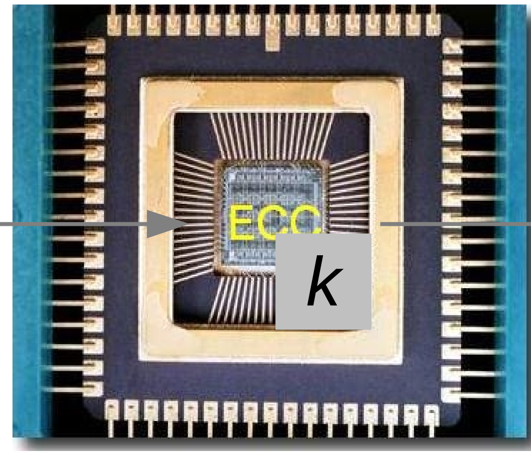
$$\mathbf{E} : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6,$$

and **P**($\mathbf{x_P}$,$\mathbf{y_P}$) is on **E.**

- Inject a fault: P($\mathbf{x_P}$,$\mathbf{y_P}$) → P'($\mathbf{x_P}$,$\mathbf{y'_P}$),

$$\mathbf{E'} : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a'_6,$$

- Fault analysis (weak curve) [Biehl+'00]

P(x_P,y_P) → 

→ **Q=[*k*]P**

- The specified curve is:

$$\mathbf{E} : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6,$$

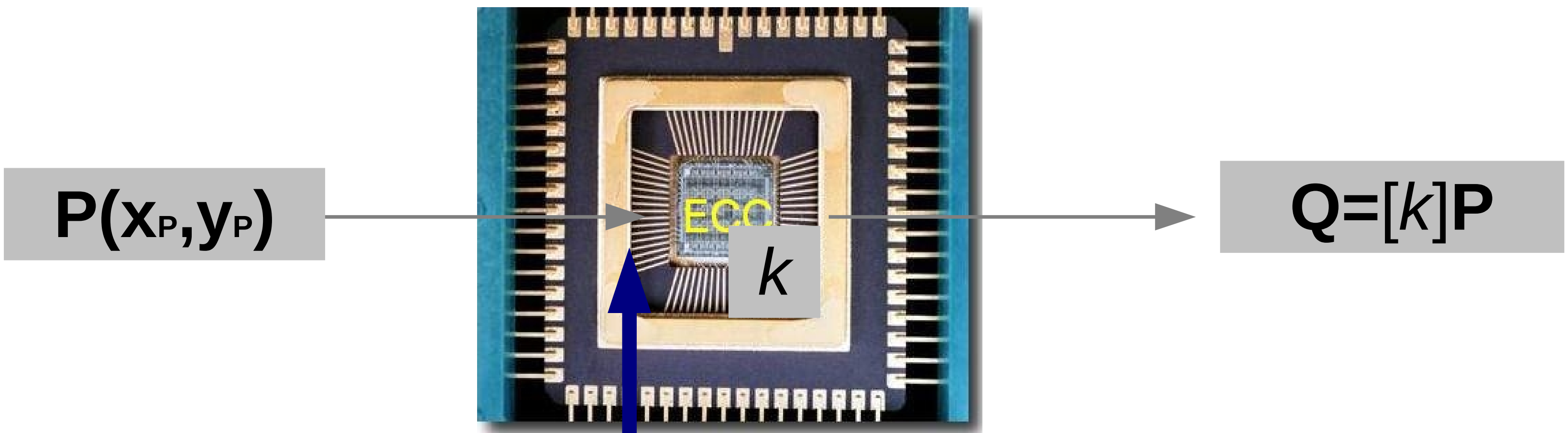
and **P**($\mathbf{x_P}$,$\mathbf{y_P}$) is on **E.**

- Inject a fault: **P**($\mathbf{x_P}$,$\mathbf{y_P}$) → **P'**($\mathbf{x_P}$,$\mathbf{y'_P}$),

$$\mathbf{E'} : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a'_6,$$

Not used for PA/PD

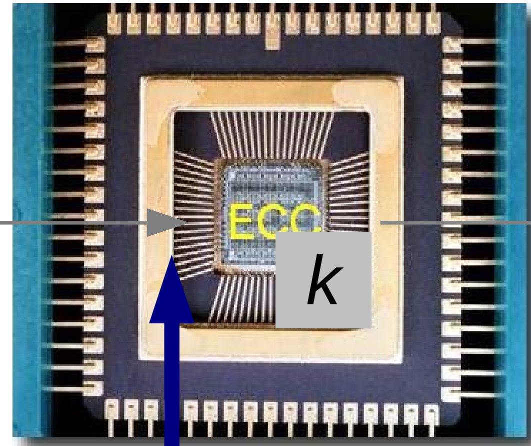
➢Point validation



PV:
Before the point multiplication :
1, check the integrity of curve **E**.
2, check if **P** is on the curve or not.

➢Point validation

$P(x_P,y_P)$



$Q=[k]P$

PV:
Before the point multiplication :
1, check the integrity of curve **E**.
2, check if **P** is on the curve or not.

But:
Can the adversary inject faults after the validation step?

➢ Fault analysis (twist curve) [Fouque+'08]

- Consider a curve defined on $\mathbf{F}_p$:

$$\mathbf{E} : y^2 z = x^3 + a\, xz^2 + bz^3.$$

y coordinates is not needed for Montgomery ladder.

Fault analysis (twist curve) [Fouque+'08]

- Consider a curve defined on $\mathbf{F}_p$:

$$\mathbf{E} : y^2z = x^3 + a\,xz^2 + bz^3.$$

y coordinates is not needed for Montgomery ladder.

- The twist of **E**:
  - $\mathbf{E'} : \varepsilon y^2z = x^3 + a\,xz^2 + bz^3,$

where ε is quadratic non-residue in $\mathbf{F}_p$.

- Let ($\mathbf{X_P}$, **-**) be a point on **E**, then a random fault on $\mathbf{X_P}$ may lead to a point on **E'** with a probability of 1/2.

Fault analysis (twist curve) [Fouque+'08]

- Consider a curve defined on $\mathbf{F}_p$:

$$\mathbf{E} : y^2 z = x^3 + a\, xz^2 + bz^3.$$

y coordinates is not needed for Montgomery ladder.

- The twist of **E**:
  - $\mathbf{E'} : \varepsilon y^2 z = x^3 + a\, xz^2 + bz^3,$

where ε is quadratic non-residue in $\mathbf{F}_p$.

- Let $(\mathbf{x}_P, -)$ be a point on **E**, then a random fault on $\mathbf{x}_P$ may lead to a point on **E'** with a probability of 1/2.

So, it is necessary to perform PV after point multiplication.

Fault analysis (twist curve) [Fouque+'08]

- Consider a curve defined on $\mathbf{F}_p$:

$$\mathbf{E} : y^2z = x^3 + a\,xz^2 + bz^3.$$

y coordinates is not needed for Montgomery ladder.

- The twist of **E**:
  - $\mathbf{E'} : \varepsilon y^2z = x^3 + a\,xz^2 + bz^3,$

where ε is quadratic non-residue in $\mathbf{F}_p$.

- Let ($\mathbf{x_P}$, **-**) be a point on **E**, then a random fault on $\mathbf{x_P}$ may lead to a point on **E'** with a probability of 1/2.

So, it is necessary to perform PV after point multiplication.

But:
Can the adversary inject faults before the validation step?

√: Effective **x**: Attacked **-**: Not related **H**: helps the attack
?: Not clear or not published *****: Implementation dependent

	Passive attacks						Active attacks						
							Safe-error		Weak curve			Differential	
	SPA TA	Temp-late	DPA	Doubl. Attack	RPA ZPA	Carry based	M type	C type	Invalid Point	Invalid curve	Twist curve	Sign change	Diff. Fault
Indistinguishable PA/PD	√	-	-	?	-	-	-	-	-	-	-	-	-
Double-add-always	√	-	-	x	-	-	-	H	-	-	-	-	-
Montgomery ladder $\perp$	√	-	-	x	?	-	√*	-	-	-	H	√	-
Montgomery ladder $\top$	√	-	-	x	x	-	√*	-	-	-	√	-	-
Random key splitting	-	?	√	?	√	x	-	-	-	-	?	?	?
Scalar randomization	-	x	x	x	√	x	-	-	-	-	-	?	?
Base point blinding	-	x	x	x	√	-	-	-	?	*?	-	-	?
Randomized proj. coord.	-	√	√	?	x	-	-	-	-	-	-	-	?
Randomized EC Iso.	-	?	√	?	x	-	-	-	-	-	-	-	?
Randomized Field Iso.	-	?	√	?	x	-	-	-	-	-	-	-	?
Point validity check	-	-	-	-	-	-	-	H	√	?	√$\top$	H	√
Curve integrity check	-	-	-	-	-	-	-	-	-	√	-	-	
Coherence check	-	-	-	-	-	-	-	H	-	?	-	√*	√

- Attacking points

- Tag's private key: x
- Tag's public key : $\mathbf{X}(=[-x]\mathbf{P})$

Reader (Verifier)		**Tag (Prover)**
		r_1=TRNG( )
	$\mathbf{R_1}$ ←	$\mathbf{R_1} = [r_1]\mathbf{P}$
r_2 = TRNG( )	r_2 →	
If $[v]\mathbf{P} + [r_2]\mathbf{X} == \mathbf{R_1}$ Then accept	v ←	$v = xr_2 + r_1 \bmod n$

The Schnorr Protocol

Attacking points

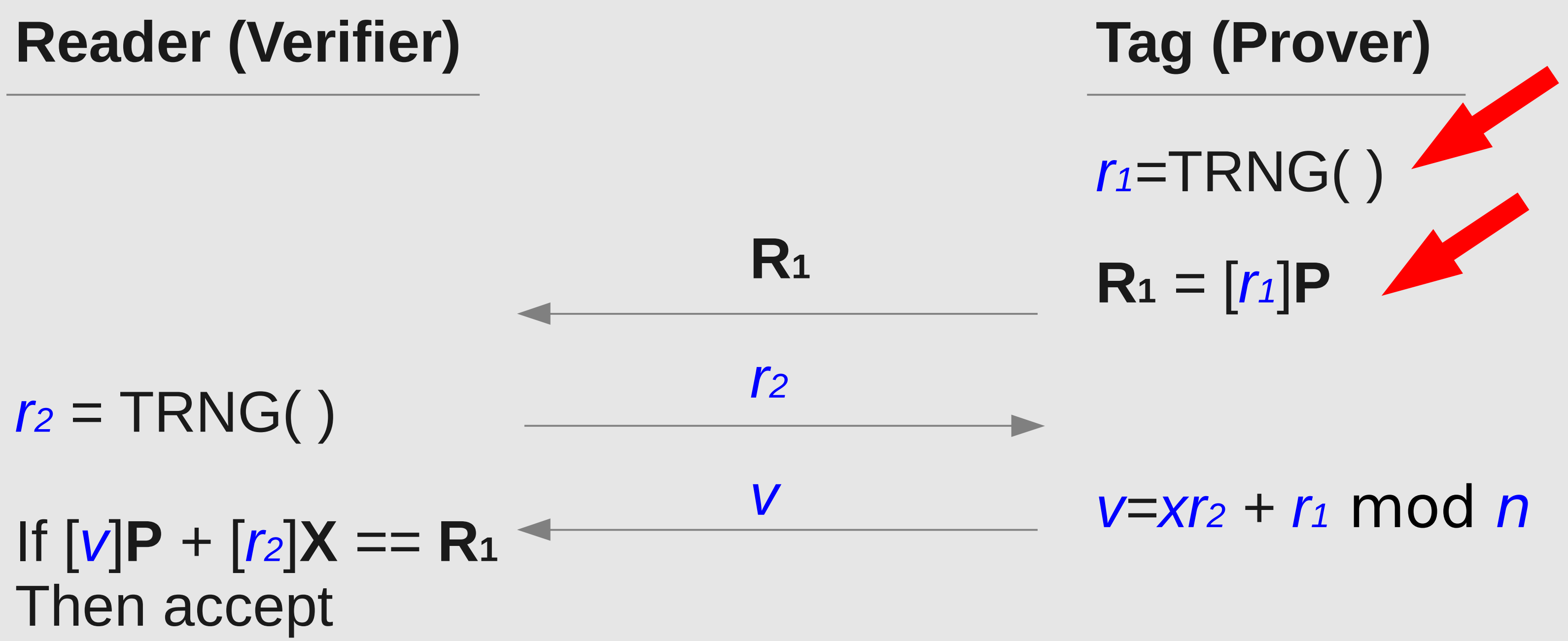
- Tag's private key: x
- Tag's public key : $\mathbf{X}(=[-x]\mathbf{P})$

Reader (Verifier)		**Tag (Prover)**
		r_1=TRNG( )
	$\mathbf{R}_1$ ←	$\mathbf{R}_1 = [r_1]\mathbf{P}$
r_2 = TRNG( )	r_2 →	
If $[v]\mathbf{P} + [r_2]\mathbf{X} == \mathbf{R}_1$ Then accept	v ←	$v = xr_2 + r_1 \bmod n$

The Schnorr Protocol

Attacking points

- Tag's private key: x
- Tag's public key : $\mathbf{X}(=[-x]\mathbf{P})$



The Schnorr Protocol

➢ Attacking points

- Tag's private key: x
- Tag's public key : $\mathbf{X}(=[-x]\mathbf{P})$

Reader (Verifier)

Tag (Prover)

r_1=TRNG( )

$\mathbf{R}_1$

$\mathbf{R}_1 = [r_1]\mathbf{P}$

r_2

r_2 = TRNG( )

v

$v = xr_2 + r_1 \bmod n$

If $[v]\mathbf{P} + [r_2]\mathbf{X} == \mathbf{R}_1$
Then accept

The Schnorr Protocol

- Ideally, it would be nice to have...

- Ideally, it would be nice to have...

P=(x,±1) or **P**=(x)

- Ideally, it would be nice to have...

P=(x,±1) or **P**=(x)

+

No inversions involved in scalar multiplication

- Ideally, it would be nice to have...

P=(x,±1) or **P**=(x)

+

No inversions involved in scalar multiplication

+

It has no weak twists

- Ideally, it would be nice to have...

P=(x,±1) or **P**=(x)

+

No inversions involved in scalar multiplication

+

It has no weak twists

+

A random (n-bit) fault on curve parameters is not likely to hit a weak curve

- Ideally, it would be nice to have...

P=(x,±1) or **P**=(x)

+

No inversions involved in scalar multiplication

+

It has no weak twists

+

A random (n-bit) fault on curve parameters is not likely to hit a weak curve

+

The protocol has minimum attacking points

- Ideally, it would be nice to have...

P=(x,±1) or **P**=(x)

+

No inversions involved in scalar multiplication

+

It has no weak twists

+

A random (n-bit) fault on curve parameters is not likely to hit a weak curve

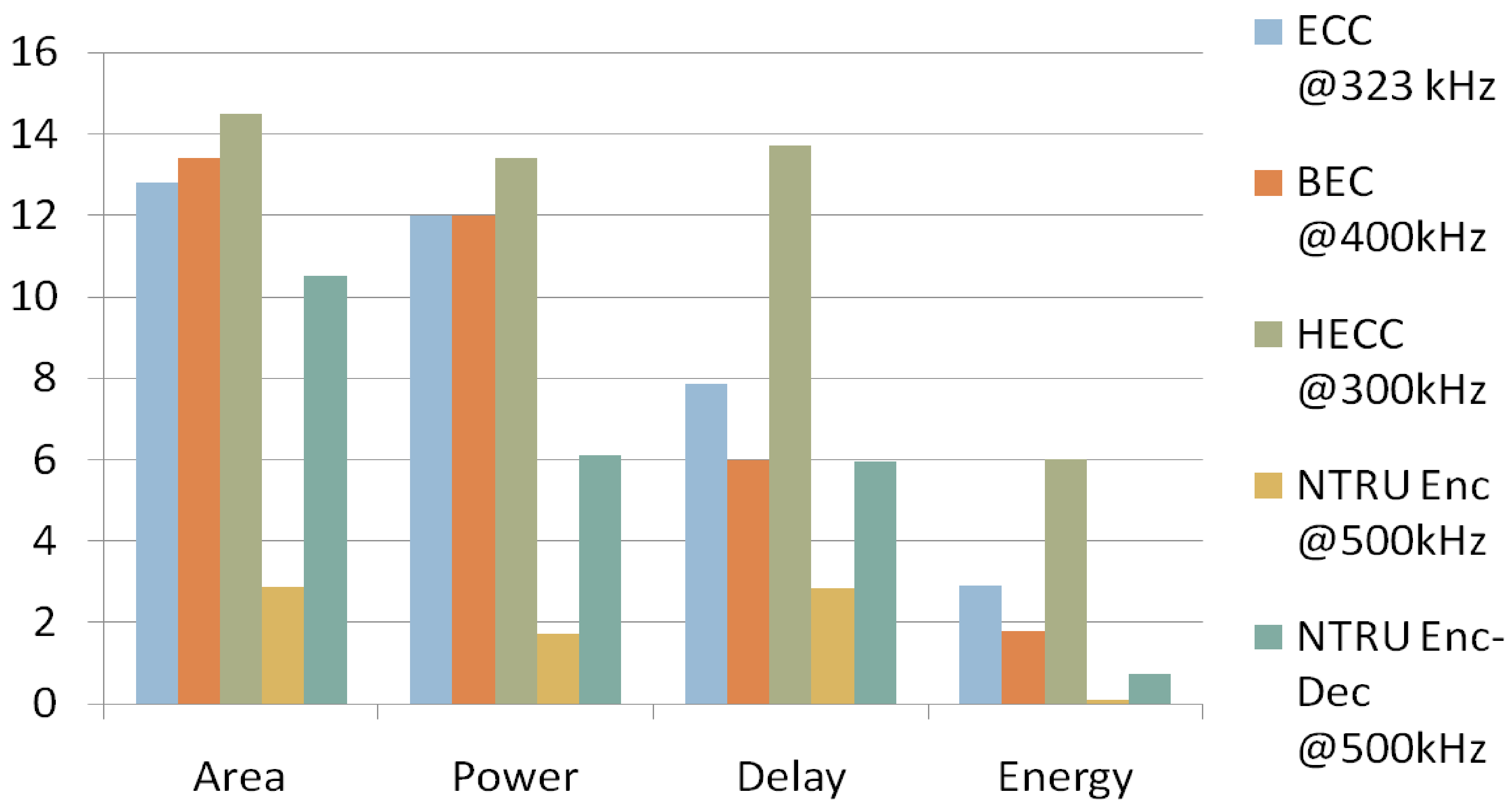
+

The protocol has minimum attacking points

+

Lightweight countermeasures

➢ Comparison

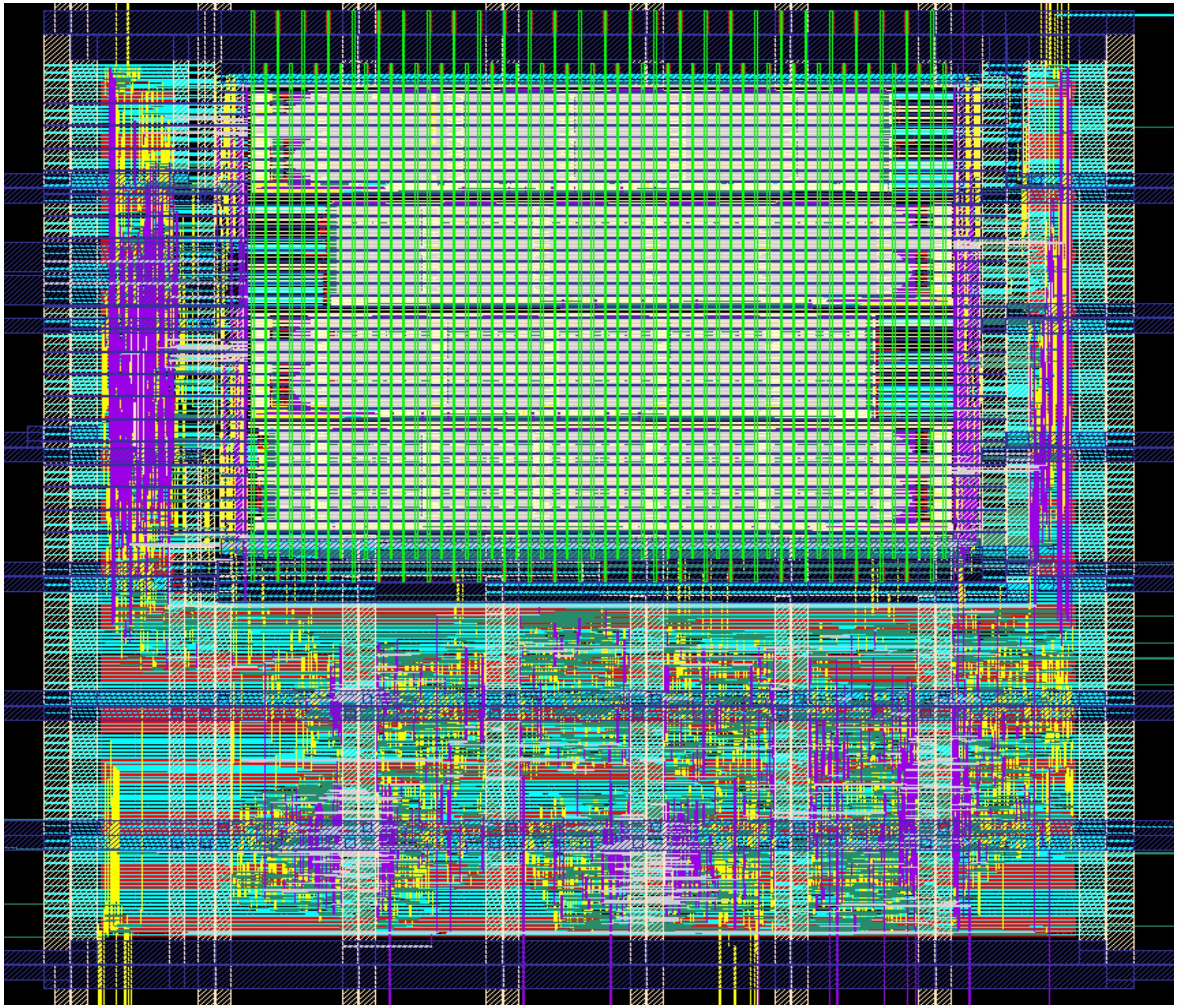


[LBV’08]
[KFV'10]
[FBV’08]
[ABFV’08]

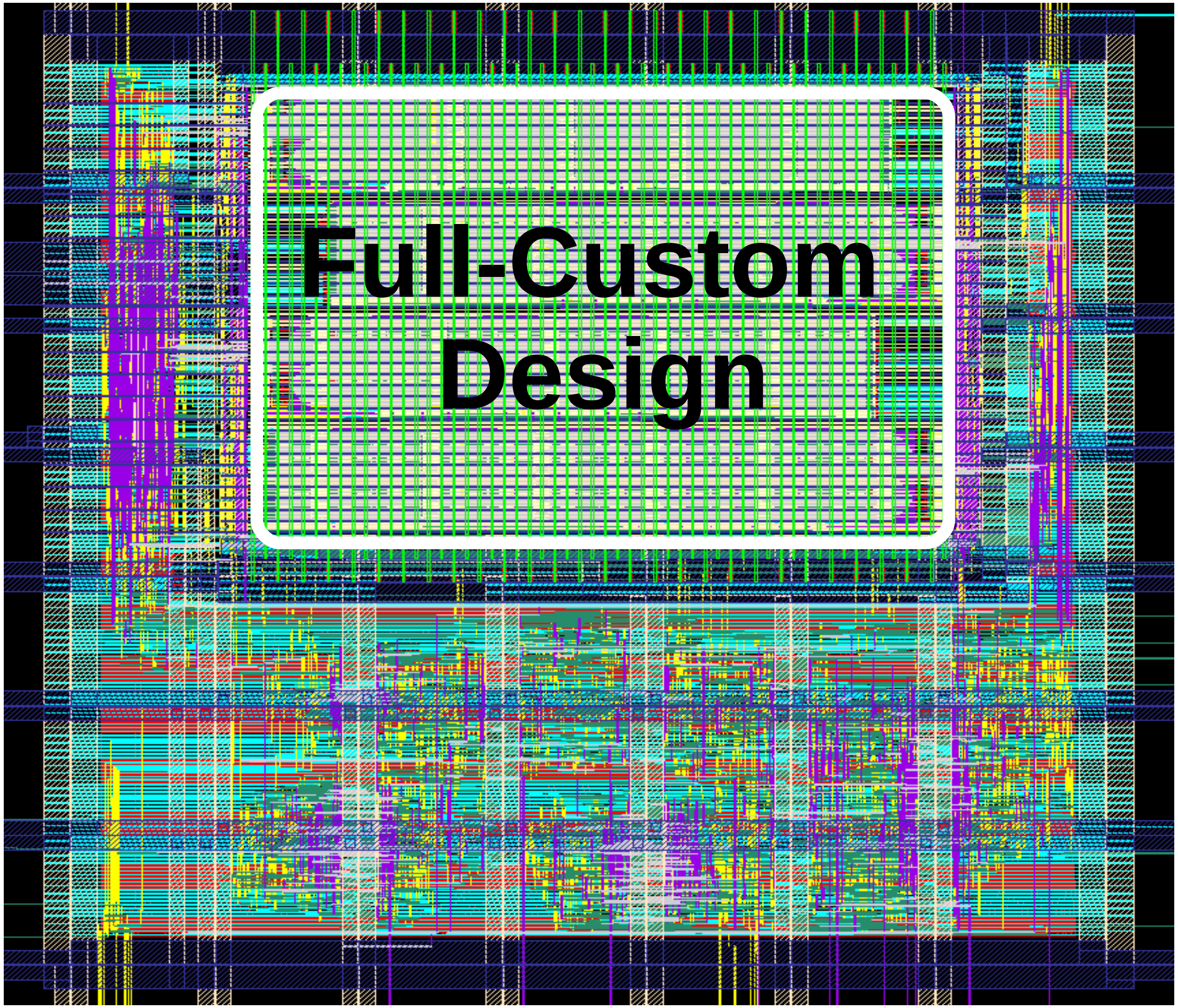
[kGates] **[uW]** **[10^4 Cycles]** **[uJ]**

* ECC/BEC over GF(2^{163})
* HECC over GF(2^{83})
* NTRU parameter: {N=167, q=128, p=3}

- An ECC processor for RFID (Expected in Nov, 2010)



- An ECC processor for RFID (Expected in Nov, 2010)



Thanks for your attention.