

Hardware implementation of ECC

The University of Electro-Communications

Kazuo SAKIYAMA

Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

Introduction

- Public-key Cryptography (PKC)
 - The most famous PKC is RSA and ECC
 - Used for key agreement (Diffie-Hellman), digital signatures, public-key encryption
- Implementation of public-key cryptosystem
 - Slower encryption/decryption (~Mbps) than symmetric key cryptography (AES: ~Gbps)
 - RSA: modular exponentiation, $c^d \bmod n$
 - ECC: point multiplication, kP on elliptic curve
 - Expensive!!

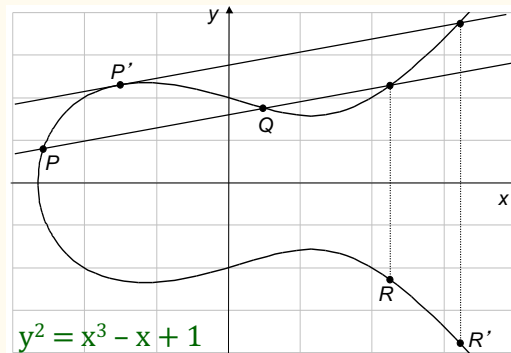
Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

2/50

Point operations on elliptic curve

- Point Addition
 $R = P + Q$
- Point Doubling
 $R' = 2P'$
Tangent line
- Scalar Multiplication
 kP
Binary algorithm



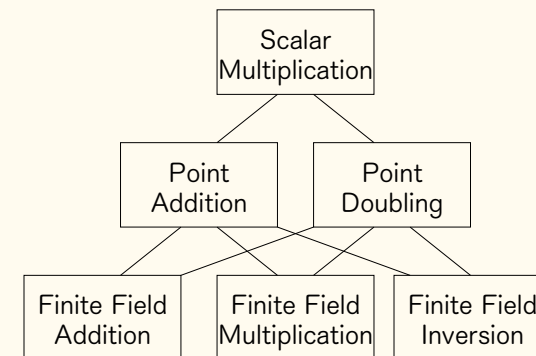
Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

3/50

Basic hierarchy for ECC

- Optimization is possible in multi-layer



Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

4/50

Series of modular operations



Type of Curve?
 GF(p), GF(2^m)?
 Security Level?
 Coordinate?
 SW/HW?
 FPGA/ASIC
 Technology?
 Size, Latency, Power?

(AB + C) mod p

Require: P = (X ₁ , Y ₁ , Z ₁), Q = (X ₂ , Y ₂ , Z ₂). Ensure: Q = Q + P.	Require: P = (X ₁ , Y ₁ , 1), Q = (X ₂ , Y ₂ , Z ₂). Ensure: Q = Q + P.	Require: Q = (X ₂ , Y ₂ , Z ₂). Ensure: Q = 2Q.
1: t ₁ = Z ₁ Z ₁ ; 2: t ₂ = X ₂ t ₁ ; 3: t ₃ = Z ₂ Z ₂ ; 4: t ₄ = X ₁ t ₃ + t ₂ ; 5: t ₅ = X ₁ t ₃ - t ₂ ; 6: t ₆ = t ₁ Z ₁ ; 7: t ₇ = Y ₂ t ₆ ; 8: t ₈ = t ₃ Z ₂ ; 9: t ₉ = t ₁ Y ₁ + Y ₂ ; 10: Y ₂ = t ₁ Y ₁ - Y ₂ ; 11: t ₁₀ = t ₂ t ₉ ; 12: t ₁₁ = t ₄ t ₉ ; 13: X ₂ = Y ₂ Y ₂ - t ₁ ; 14: t ₁₂ = -2X ₂ + t ₁ ; 15: t ₁₃ = t ₅ t ₉ ; 16: t ₁₄ = t ₃ t ₉ ; 17: t ₁₅ = t ₄ Y ₂ - t ₁ ; 18: Y ₂ = t ₃ /2; 19: Z ₂ = t ₂ Z ₂ ; 20: Z ₁ = Z ₁ Z ₂ ; 21: Return Q;	1: t ₁ = Z ₂ Z ₂ ; 2: t ₂ = X ₁ t ₁ + X ₂ ; 3: t ₃ = X ₁ t ₁ - X ₂ ; 4: t ₄ = t ₃ Z ₂ ; 5: t ₅ = t ₁ Y ₁ + Y ₂ ; 6: Y ₂ = t ₁ Y ₁ - Y ₂ ; 7: t ₆ = t ₃ t ₅ ; 8: t ₇ = t ₄ t ₅ ; 9: X ₂ = Y ₂ Y ₂ - t ₁ ; 10: t ₈ = -2X ₂ + t ₁ ; 11: t ₉ = t ₅ t ₅ ; 12: t ₁₀ = t ₇ t ₅ ; 13: t ₁₁ = t ₃ Y ₂ - t ₁ ; 14: Y ₂ = t ₃ /2; 15: Z ₂ = t ₂ Z ₂ ; 16: Return Q;	1: t ₁ = X ₂ X ₂ ; 2: t ₁ = 3t ₁ ; 3: t ₂ = Z ₂ Z ₂ ; 4: t ₂ = t ₂ t ₂ ; 5: t ₃ = t ₂ t ₂ + t ₁ ; 6: t ₁ = 2t ₁ ; 7: Z ₂ = Z ₂ t ₁ ; 8: t ₃ = t ₁ t ₁ ; 9: t ₄ = X ₂ t ₃ ; 10: X ₂ = 2t ₁ ; 11: X ₂ = t ₂ t ₂ - X ₂ ; 12: t ₁ = t ₁ t ₃ ; 13: t ₁ = t ₁ Y ₂ ; 14: t ₃ = t ₄ - X ₂ ; 15: Y ₂ = t ₂ t ₃ - t ₁ ; 16: Return Q;

Kazuo SAKIYAMA

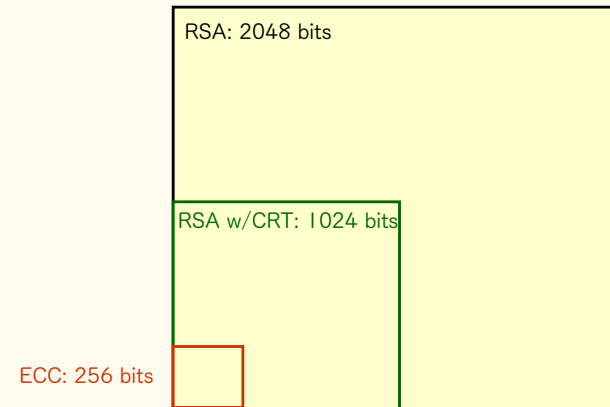
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

5/50

Data Size in Modular Operations



- Modular multiplication



Kazuo SAKIYAMA

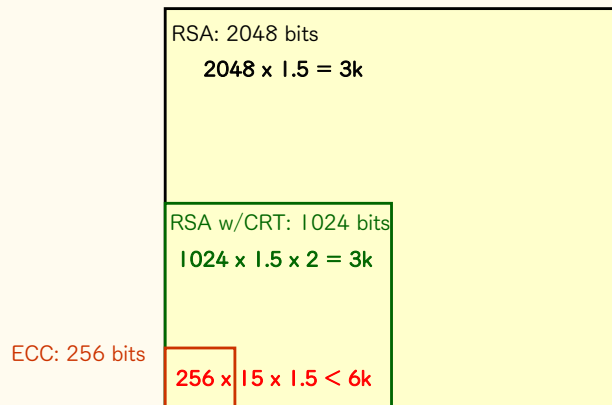
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

6/50

of Modular Multiplications



- RSA: c^d, ECC: kP



Kazuo SAKIYAMA

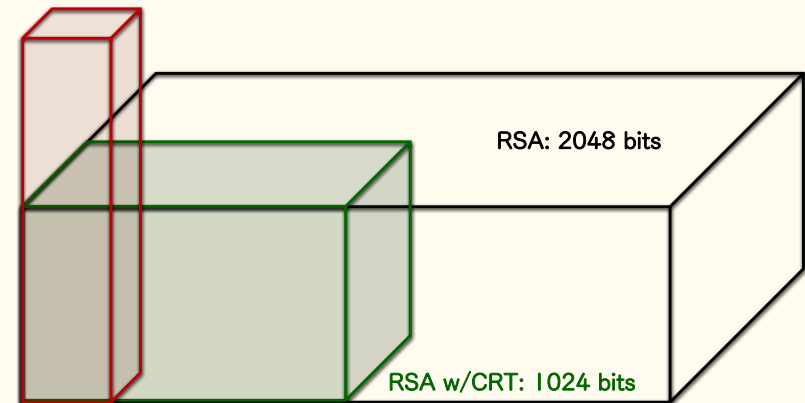
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

7/50

Total computational complexity



ECC: 256 bits



Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

8/50

Right-to-left & Left-to-right



```

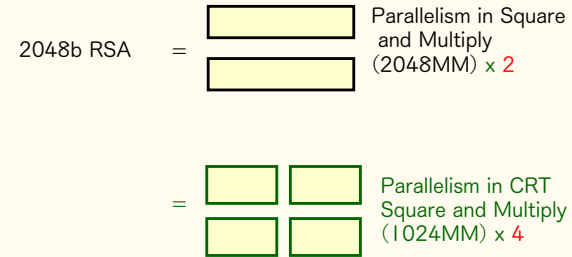
Left-to-right Binary Algorithm
Input: non-negative integers c, d = (d_{t-1}d_{t-2} \dots d_1d_0)_2.
Output: c^d mod n.
1: T \leftarrow 1
2: for i from t-1 down to 0 do
3:   T \leftarrow T^2 mod n
4:   if d_i = 1 then Dependency
5:     T \leftarrow cT mod n
6:   end if
7: end for
8: Return T

Right-to-left Binary Algorithm
Input: non-negative integers c, d = (d_{t-1}d_{t-2} \dots d_1d_0)_2.
Output: c^d mod n.
1: S \leftarrow 1, T \leftarrow c
2: for i from 0 to t-1 do
3:   if d_i = 1 then
4:     S \leftarrow ST mod n
5:   end if
6:   T \leftarrow T^2 mod n
7: end for
8: Return S
    
```

RSA: parallelism for high speed



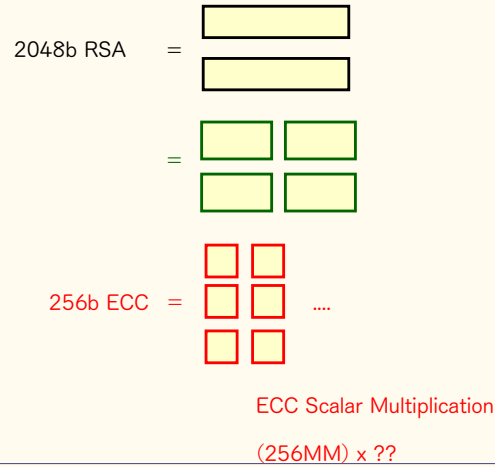
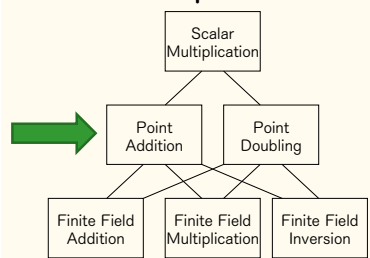
- CRT
- Binary algorithm



ECC: parallelism for high speed



- Point operations



Exercise I



- Q: Latency of 2048b CRT-RSA?
 - Latency of 1024b MM = 100ns
- A1: 100ns x 1024 x 1.5 x 2 = 300us
 - when using one MM
- A2: 100ns x 1024 x 1.5 = 150us
 - when using two MMs
- Squaring = MM

Exercise 2



- Q: Latency of 256b ECC?
 - Latency of 256b MM = 5ns
 - Point addition needs 12 MM
 - Point doubling needs 10 MM
- A1: $5\text{ns} \times 256 \times 16 = 160\mu\text{s}$
 - when using one MM
- A2: $160/2 = 80\mu\text{s}$
 - when using two MMs & **deg. parallelism = 2**

Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

13/50

Faster implementation



- Latency of 256 ECC can be less than 1us?
 - Short latency of 256b MM : e.g. **4x**
 - More parallelism : e.g. **4x**
 - Efficient point operation sequence : e.g. **4x**
- Limitation of
 - current CMOS technology
 - current PKC

Kazuo SAKIYAMA

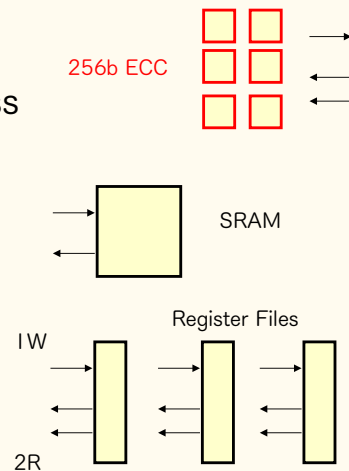
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

14/50

Yet Another Bottleneck



- Data coherence
 - Memory (register) access
 - SRAM
 - IW2R register
- Scheduling
 - Dynamic/Static



Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

15/50

Modular Multiplication



- Multiplication + Reduction
 - Barrett Reduction (1986): Focus on MSBs
 - Montgomery Reduction (1985): Focus on LSBs
- Based on n-bit multiplier ($n = 8, 16, 32, 64$)
 - E.g., CIOS [Koç, Acar, Kaliski Jr. 1996]
- Based on bit-serial multiplier
 - Multiplicand x n-bit [Großschädl 2001]

Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

16/50

Final reduction for prime field p



- $XYR^{-1} \bmod p$
- $T = (T + x_i Y + mp)/2 < 2p$

Input: k -bit integers X, Y with $0 \leq X, Y < p, R = 2^k$.

Output: $XYR^{-1} \bmod p$.

1: $T = (t_k, \dots, t_0)_2 \leftarrow 0$

2: **for** i from 0 to $k-1$ **do**

3: $m \leftarrow t_0 \oplus x_i \cdot y_0$

4: $T \leftarrow (T + x_i Y + mp)/2$

5: **end for**

6: **if** $T \geq p$ **then**

7: $T \leftarrow T - p$

8: **end if**

9: **return** T

Final Reduction
(Unnecessary for Binary fields)

Final reduction: $T = T - p < p$

Kazuo SAKIYAMA

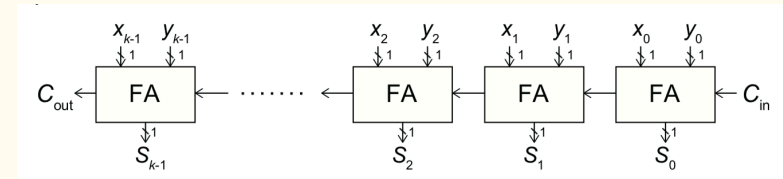
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

17/50

Ripple Carry Adder (RCA)



- Long carry chain
 - $s_i = x_i \oplus y_i \oplus z_i$
 - $c_{i+1} = \text{MAJ}(x_i, y_i, z_i)$
 $= (x_i \wedge y_i) \vee (y_i \wedge z_i) \vee (z_i \wedge x_i)$



Kazuo SAKIYAMA

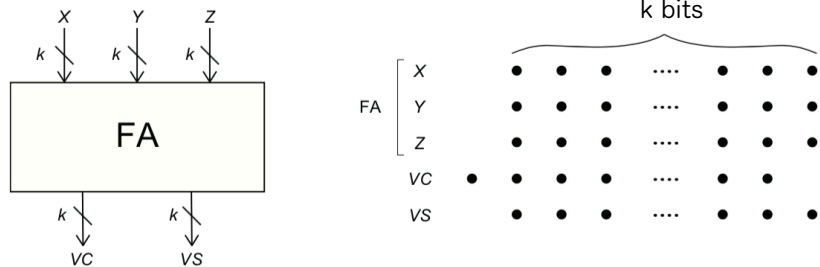
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

18/50

Carry Save Adder (CSA)



- 3-to-2 compression
 $- X + Y + Z = 2VC + VS$



Kazuo SAKIYAMA

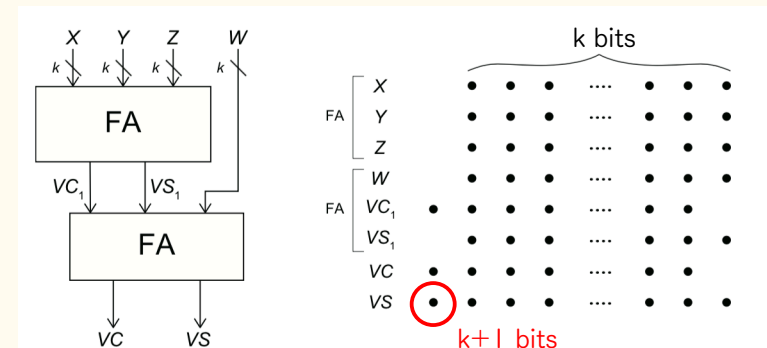
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

19/50

4-2 CSA



- Two FAs w/o carry chain
 $- X + Y + Z + W = 2VC + VS$



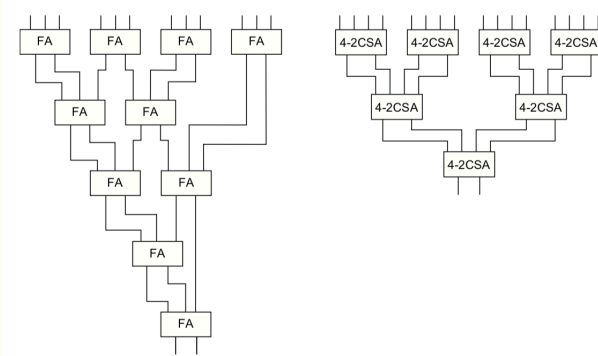
Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

20/50

Adder Tree

- 16 operands
 - Latency ~ 5 or 6 FAs (RCA = 16FAs)



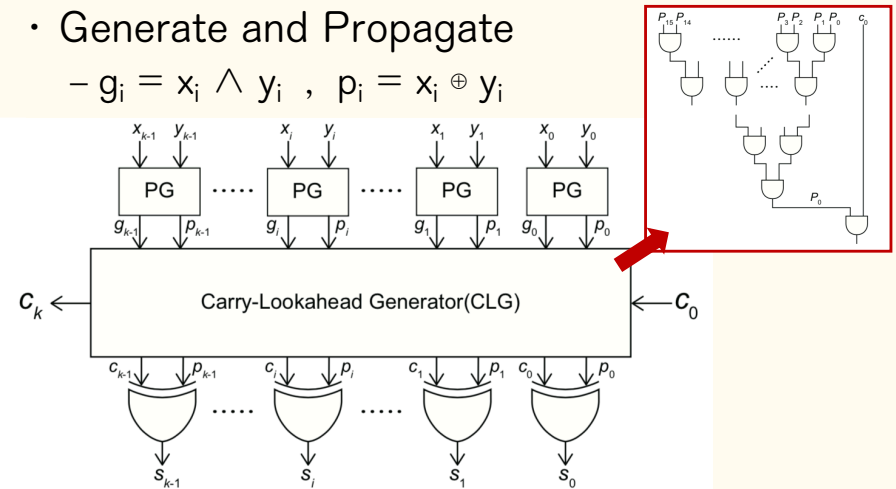
Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

21/50

Carry Lookahead Adder (CLA)

- Generate and Propagate
 - $g_i = x_i \wedge y_i$, $p_i = x_i \oplus y_i$



Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

22/50

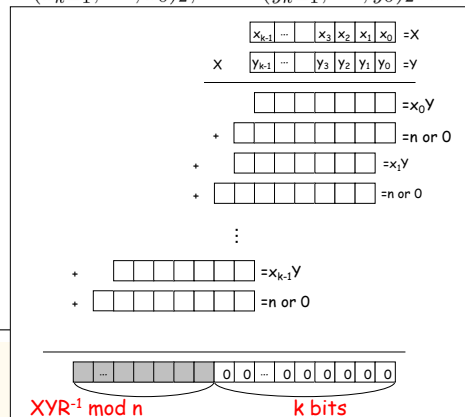
Bit-serial Montgomery MM

Bit-serial Montgomery Modular Multiplication

Input: integers $n = (n_{k-1}, \dots, 1)_2$, $X = (x_{k-1}, \dots, x_0)_2$, $Y = (y_{k-1}, \dots, y_0)_2$ with $0 \leq X, Y < n$.

Output: $XYR^{-1} \bmod n$.

- 1: $T_0 \leftarrow 0$
- 2: **for** i from 0 to $k-1$ **do**
- 3: $q_i \leftarrow t_0 \oplus (x_0 \wedge y_i)$
- 4: $T \leftarrow (T + q_i n + y_i X) / 2$
- 5: **end for**
- 6: **if** $T \geq n$ **then**
- 7: $T \leftarrow T - n$
- 8: **end if**
- 9: **return** T



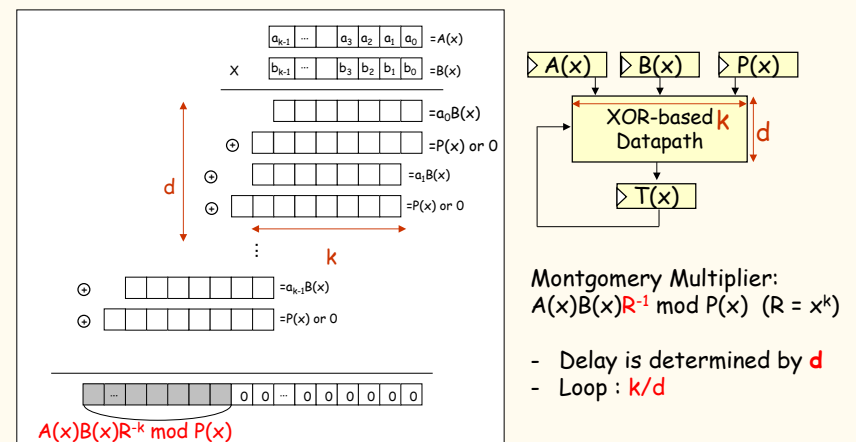
Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

23/50

Modular Reduction from LSB

- LSB-first bit-serial multiplier (binary fields)



Kazuo SAKIYAMA

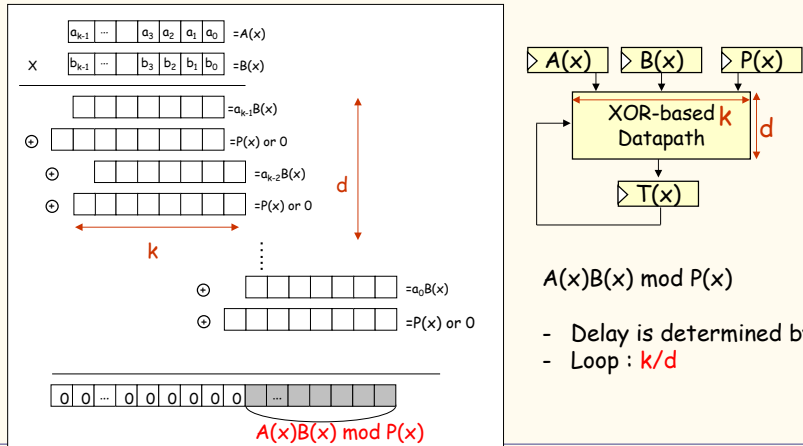
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

24/50

Modular Reduction from MSB



- MSB-first bit-serial multiplier (binary field)



Kazuo SAKIYAMA

Final reduction for prime field p



- $XYR^{-1} \bmod p$
- $T = (T + x_i Y + mp)/2 < 2p$

Input: k -bit integers X, Y with $0 \leq X, Y < p, R = 2^k$.

Output: $XYR^{-1} \bmod p$.

```

1:  $T = (t_k, \dots, t_0)_2 \leftarrow 0$ 
2: for  $i$  from 0 to  $k-1$  do
3:    $m \leftarrow t_0 \oplus x_i \cdot y_0$ 
4:    $T \leftarrow (T + x_i Y + mp)/2$ 
5: end for
6: if  $T \geq p$  then
7:    $T \leftarrow T - p$ 
8: end if
9: return  $T$ 
    
```

Final Reduction

Final reduction: $T = T - p < p$

Kazuo SAKIYAMA

Towards no final reduction



- $XYR^{-1} \bmod p$
- $T = (T + x_i Y + mp)/2 < 3p$

Input: $(k+1)$ -bit integers, X, Y with $0 \leq X, Y < 2p, R = 2^{k+2}$.

Output: $XYR^{-1} \bmod p$.

```

1:  $T = (t_{k+1}, \dots, t_0)_2 \leftarrow 0$ 
2: for  $i$  from 0 to  $k$  do
3:    $m \leftarrow t_0 \oplus x_i \cdot y_0$ 
4:    $T \leftarrow (T + x_i Y + mp)/2$ 
5: end for
6:  $T \leftarrow (T + mp)/2$ 
7: return  $T$ 
    
```

Loop + 1

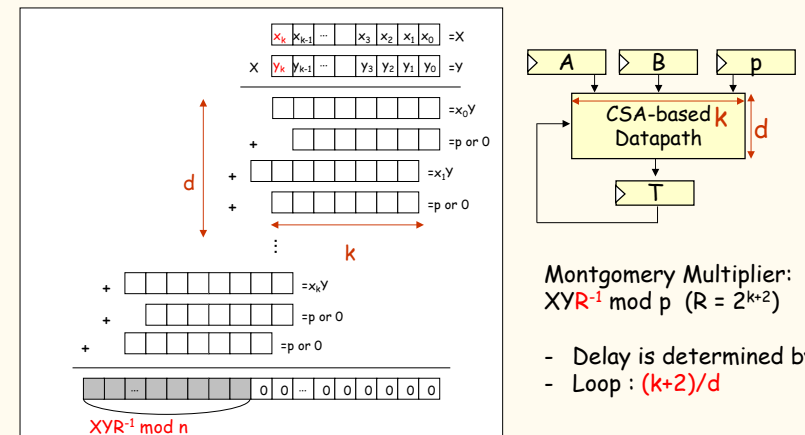
$T = (T + mp)/2 < 2p$

Kazuo SAKIYAMA

Modular Reduction from LSB



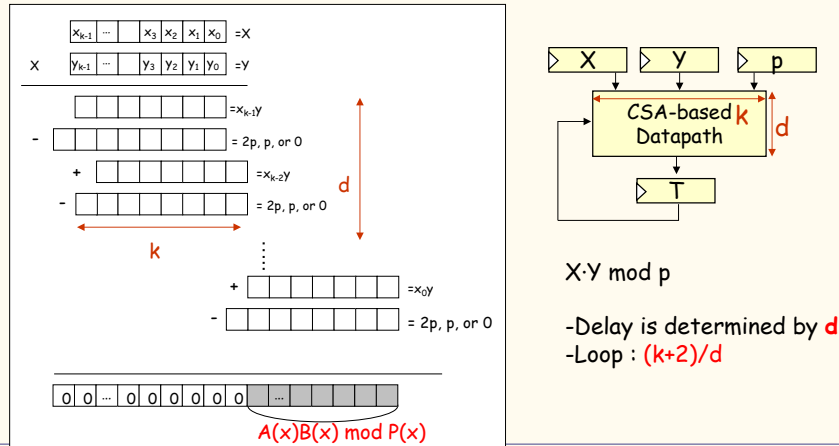
- LSB-first bit-serial multiplier (prime field)



Montgomery Multiplier:
 $XYR^{-1} \bmod p$ ($R = 2^{k+2}$)

Kazuo SAKIYAMA

- MSB-first bit-serial multiplier (prime field)



Kazuo SAKIYAMA

- $XY \bmod p$
- Intermediate value, $T = (2T + x_i Y) < 3p$

Input: k -bit integers X, Y with $0 \leq X, Y < p$.

Output: $XY \bmod p$.

```

1:  $T = (t_{k+1}, \dots, t_0)_2 \leftarrow 0$ 
2: for  $i$  from  $k-1$  to  $0$  do
3:    $T \leftarrow (2T + x_i Y)$ 
4:    $q \leftarrow \lfloor T/p \rfloor$ 
5:    $T \leftarrow (T - qp)$ 
6: end for
7: return  $T$ 
    
```

- Division is escaped with precomputation

[Knezevic, Vercauteren, Verbauwhede, 2010]

Kazuo SAKIYAMA

- $T/p = T/2^{k-1} * 2^{k-1}/p$
- S4 : $q = T/2^{k-1}$ shift operation (wiring)
- S5 : $T = (T - q * 2^{k-1}/p)$ precomputation

Input: k -bit integers X, Y with $0 \leq X, Y < p$.

Output: $XY \bmod p$.

```

1:  $T = (t_{k+1}, \dots, t_0)_2 \leftarrow 0$ 
2: for  $i$  from  $k-1$  to  $0$  do
3:    $T \leftarrow (2T + x_i Y)$ 
4:    $q \leftarrow \lfloor T/p \rfloor$ 
5:    $T \leftarrow (T - qp)$ 
6: end for
7: return  $T$ 
    
```

- But $q = 0, 1, 2 \dots$

Kazuo SAKIYAMA

- Both MSB/LSB Modular Multiplication [Batina et. al., 2004; Kaihara and Takagi, 2005]

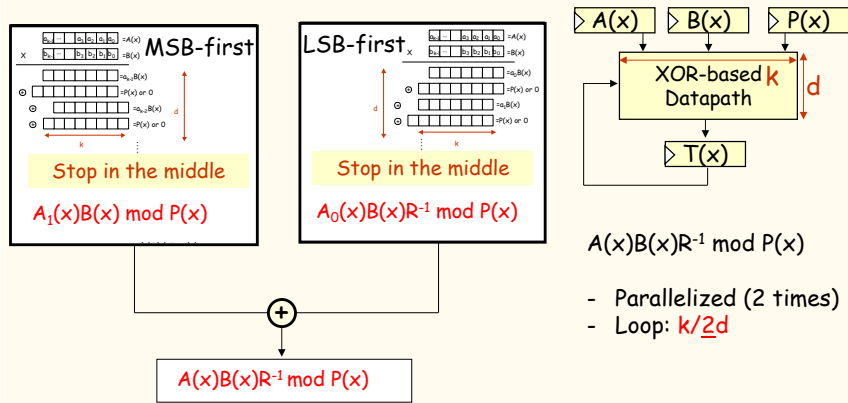
- Combine MSB-first and LSB-first MM

$$\begin{aligned}
 & A(x)B(x)R^{-1} \bmod P(x) \\
 &= (A_1(x)R + A_0)B(x)R^{-1} \bmod P(x) \\
 &= A_1(x)B(x) \bmod P(x) \text{ MSB-first} \\
 &\quad + A_0(x)B(x)R^{-1} \bmod P(x) \text{ LSB-first}
 \end{aligned}$$

- $R = x^{k/2}$

Kazuo SAKIYAMA

- Bipartite bit-serial multiplier (binary field)



- Parallelism in Modular Multiplication
- Applicable to prime field (original proposal)
 - Need to take care (final) reductions
- Almost double performance, less than double area (F/Fs are shared)

- Booth recoding (radix-4)
 - Partial products x_iY
 - Escape $3Y$ nor $-3Y$
 - Half # of operands (at worst)
 - CSA + CLA

x_{2i+1}	x_{2i}	x_{2i-1}	z_i
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2 (= $\bar{2}$)
1	0	1	-1 (= $\bar{1}$)
1	1	0	-1 (= $\bar{1}$)
1	1	1	0

- Also derived by $2X - X (=X)$
 - Ex) $X = (00\ 1111\ 0011\ 1001)_2 = (10\bar{1}\ 10\bar{2}1)_4$

$2X =$	0	0	1	1	1	1	0	0	1	1	1	0	0	1	0	
$- X =$			0	0	1	1	1	1	0	0	1	1	1	0	0	1
$Z =$			1	0	$\bar{1}$	1	0	$\bar{2}$								1

Multiplier-Based Montgomery Modular Multiplication

Input: integers $n = (n_{r-1}, \dots, n_0)_{2^m}$, $X = (x_{r-1}, \dots, x_0)_{2^m}$, $Y = (y_{r-1}, \dots, y_0)_{2^m}$ with $m = \lceil k/r \rceil$, $0 \leq X, Y < n$, $\gcd(n_0, 2) = 1$.

Output: $XYR^{-1} \bmod n$, $RR^{-1} - nm' = 1$, $R = 2^k$, $0 < R^{-1} < n$, $0 < n' < R$.

- $T \leftarrow 0$
- for i from 0 to $m-1$ do
- $q_i \leftarrow (T + x_0 y_i) n' \bmod 2^k$ **Multiplications**
- $T \leftarrow (T + q_i n + y_i X) / 2^k$
- end for
- if $T < n$ then
- $T \leftarrow T - n$
- end if
- Return T

$X = (\text{f3 d7 2f 88}), Y = (\text{f6 c6 ba 98}), n = (\text{f9 81 89 2b})$

i	q_i	T	\leftarrow	$T + q_i n + y_i X$
Initial	-	(00 00 00 00)		
0	c0	(01 3e 3a ba 01)	\leftarrow	(01 3e 3a ba 01 00)
1	c5	(00 ee 8d 65 8a)	\leftarrow	(00 ee 8d 65 8a 00)
2	6a	(01 37 7f 16 e7)	\leftarrow	(01 37 7f 16 e7 00)
3	f3	(01 d8 4c 17 69)	\leftarrow	(01 d8 4c 17 69 00)
-	-	(00 de ca 8e 3e)		-

Architecture for ECC coprocessor

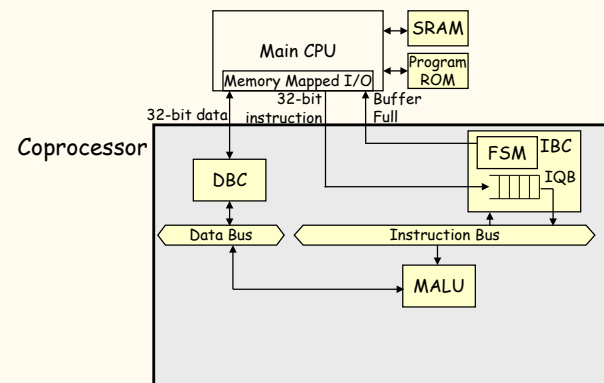


- HW/SW co-design
 - CPU + coprocessor for ECC
- Speed performance depends on architecture
 - Trade-off between cost and performance
 - Local dedicated memory for high performance
- Instruction-set extension, ASIC, etc.

ECC coprocessor (I)



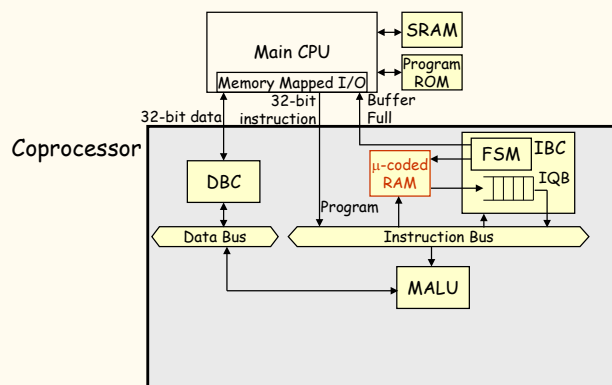
TYPE I: Smallest implementation (baseline)



ECC coprocessor (II)



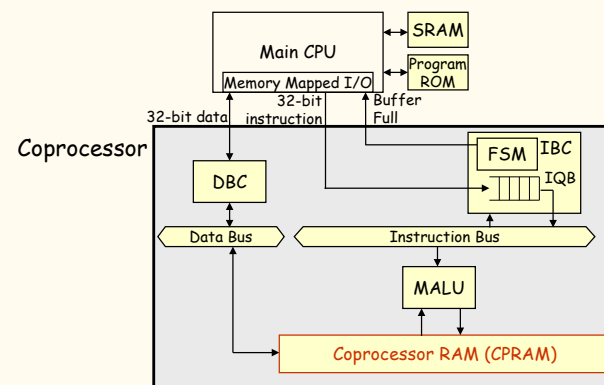
TYPE II: TYPE I + m-coded RAM



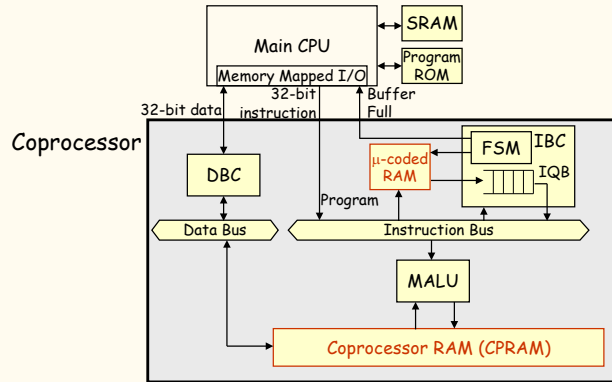
ECC coprocessor (III)



TYPE III: TYPE I + CPRAM



TYPE IV: TYPE I + CPRAM & m-code RAM

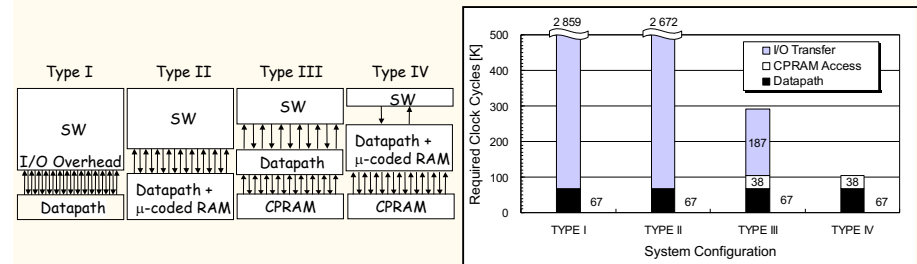


Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

41/50

- Performance for HECC over $GF(2^{83})$
 - The number of cycles is reduced by CPRAM
 - Micro-coded RAM can avoid instruction stalls



Kazuo SAKIYAMA

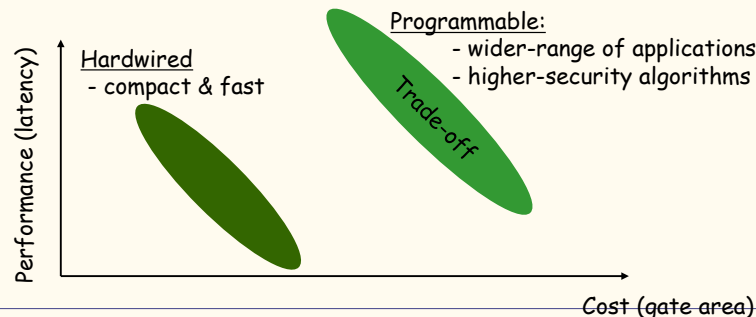
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

42/50

Cost, Performance and Security Trade-offs

Programmable VS hardwired parameters

- ▣ Curve parameters
- ▣ Computational sequence: Coordinates, operation form.
- ▣ Prime p , irreducible polynomial $P(x)$
- ▣ Field sizes



Kazuo SAKIYAMA

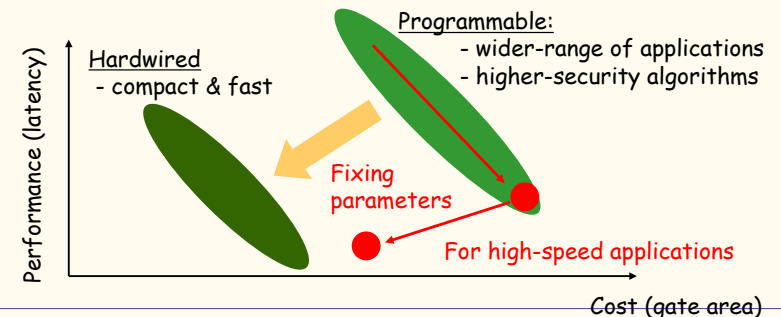
Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

43/50

Cost, Performance and Security Trade-offs

Programmable VS hardwired parameters

- ▣ Curve parameters
- ▣ Computational sequence: Coordinates, operation form.
- ▣ Prime p , irreducible polynomial $P(x)$
- ▣ Field sizes



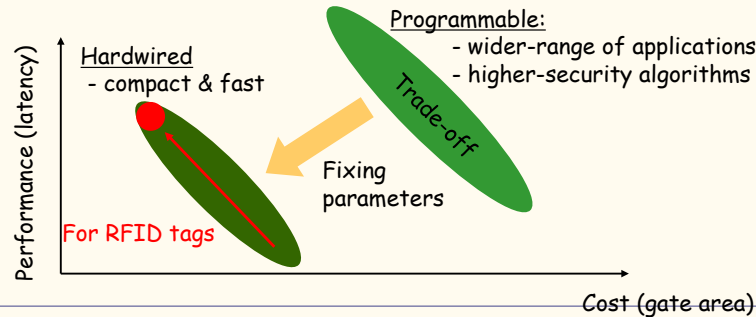
Kazuo SAKIYAMA

Autumn School, 22nd Workshop on Elliptic Curve Cryptography 17/11/2018

44/50

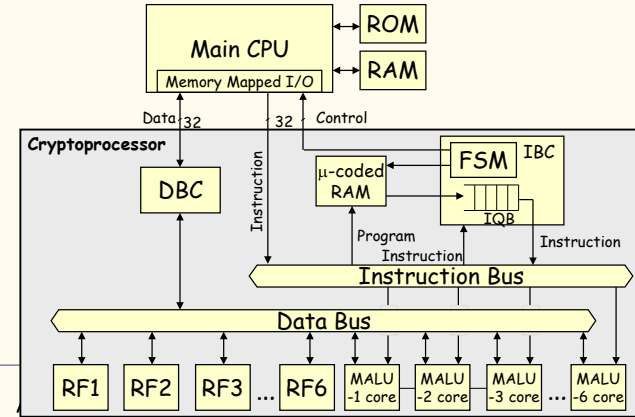
Programmable VS hardwired parameters

- ▣ Curve parameters
- ▣ Computational sequence: Coordinates, operation form.
- ▣ Prime p , irreducible polynomial $P(x)$
- ▣ Field sizes



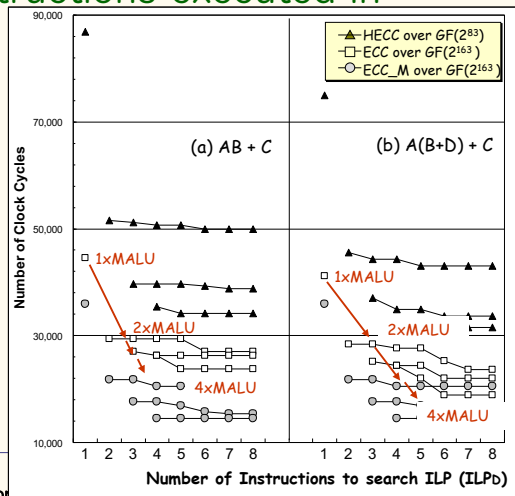
Programmable high-speed coprocessor

- ▣ ECC / HECC over $GF(2^m)$
- ▣ Six MALU cores (MALU)



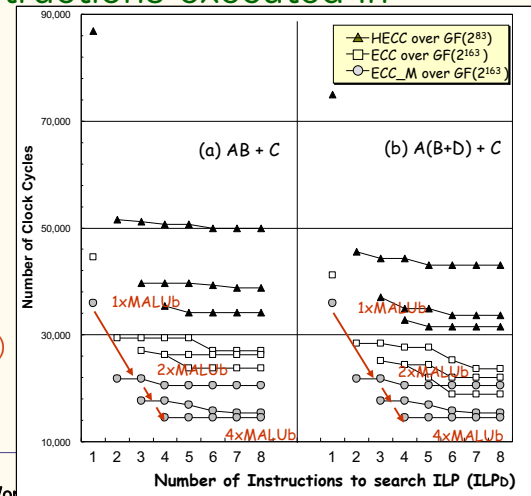
How many instructions executed in parallel?

- ▣ Operation form
 $AB + C$ and
 $A(B+C) + D$
- ▣ Check the data dependencies for ECC (and HECC)



How many instructions executed in parallel?

- ▣ Operation form
 $AB + C$ and
 $A(B+C) + D$
- ▣ Check the data dependencies for ECC (and HECC)

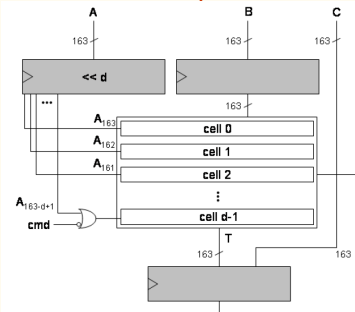


Low-power MALU for RFID tags

- ▣ Based on MALU
- ▣ Cost and performance for ECC and HECC
- ▣ Power estimation with 0.13-um CMOS library

Resource sharing

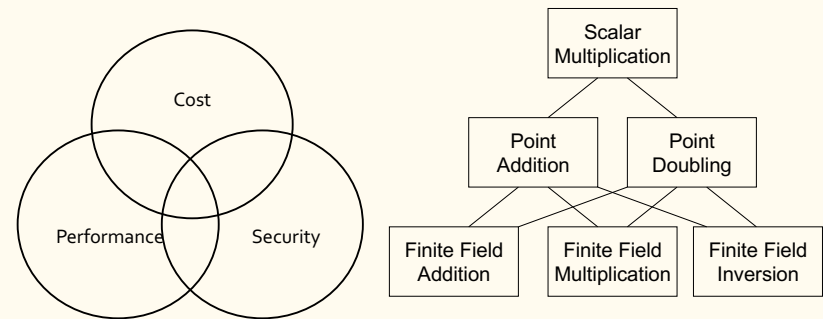
- ▣ Modular addition and multiplication use the same hardware (cell)



Kazuo SAKIYAMA

Trade-offs in ECC hardware

- ▣ Parallel processing
- ▣ Operation forms: $AB + C$ or $A(B+D) + C$
- ▣ Cost, performance, security



Kazuo SAKIYAMA