انجمن رمز ایران
Iranian Society of Cryptology

Efficient Implementation of ECC

Raziyeh Salarifard
R_salarifard@sbu.ac.ir

Table of Content

ECC

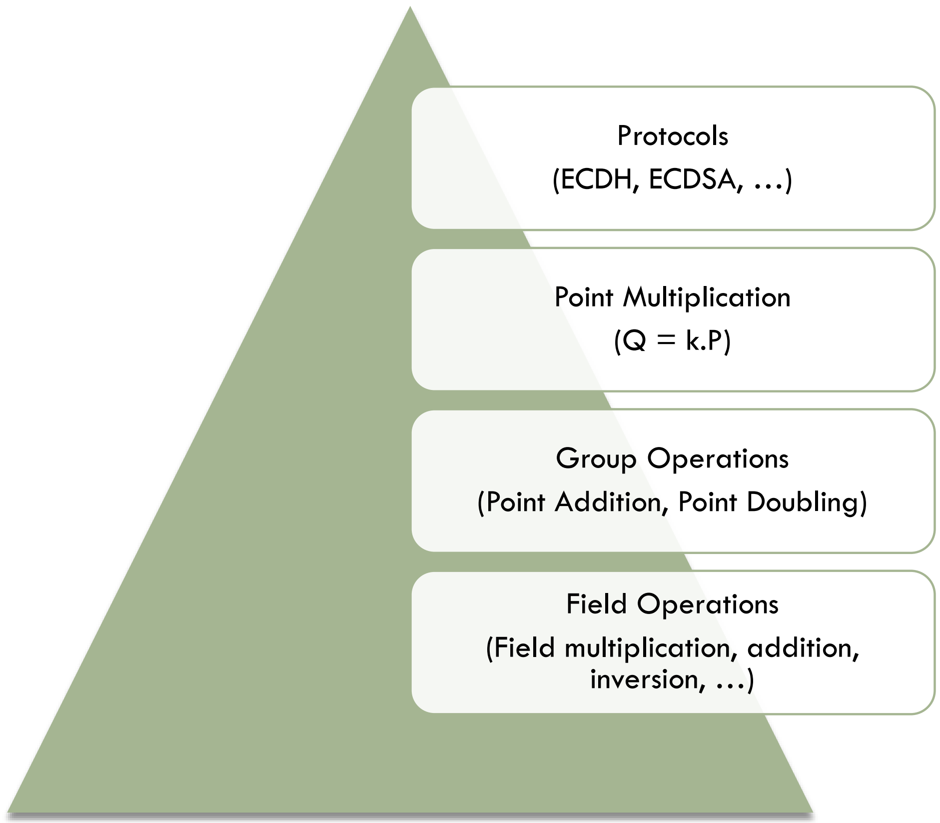
- Neal Koblitz and Victor S. Miller ,1985
- Wide use, 2004-2005

- ECC application
  - Digital Signature
  - Key Agreement
  - Random Number Generator

Why ECC?

- Elgamal
  - Pollard's Rho, Shank
    - 80 bits Security, 160
  - Index-calcules
    - 80 bits Security, 1024
- ECC
  - Pollard's Rho, Shank
    - 80 bits Security, 160
  - Index-calcules
    - Not working

ECC Operation



Protocols

- Elliptic Curve Diffie-Hellman Key Exchange (ECDH)

$a \leftarrow \{1, \ldots, \#E - 1\}$

Alice

$A = a.P$

$B = b.P$

$b \leftarrow \{1, \ldots, \#E - 1\}$

Bob

output:

$k_A = a.B = a.(bP)$

equal to:

$(a.b)P$

equal!

output:

$k_B = b.A = b.(aP)$

equal to:

$(b.a).P$

Point Multiplication

- Double and add
- Point addition and point Doubling
  - Group operation

```
N ← P
Q ← 0
for i from 0 to m do
    if d_i = 1 then
        Q ← point_add(Q, N)
    N ← point_double(N)
return Q
```

Point Addition and Point Doubling

- Affine Coordinate
  - (x, y)
- $P = (x_1, y_1), Q = (x_2, y_2)$

$(x_3, y_3) = P+Q,\ P=Q:$

$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

$$x_3 = \lambda^2 - 2x_1$$

$$y_3 = (x_1 - x_3)\lambda - y_1$$

$(x_3, y_3) = P+Q,\ P \neq Q:$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = (x_1 - x_3)\lambda - y_1$$

Efficiency

- $\frac{Throughput}{Area} = \frac{M}{AT}$
  - M
    - Field size
  - AT = Area X Time

Efficiency in SW vs HW

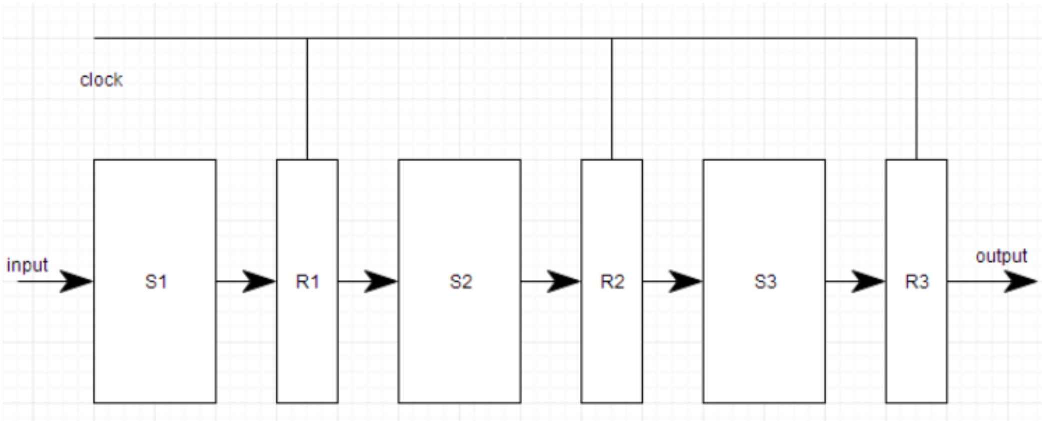
- Hardware
  - Lower area and time
- Software
  - Lower time
  - Fewer Instructions

Efficient Architecture (HW)

- Parallel
  - Area
  - Time
  - High-speed
- Serial
  - Area
  - Time
  - Low-complexity, lightweight
- Pipeline
  - Area ?
  - Time ?

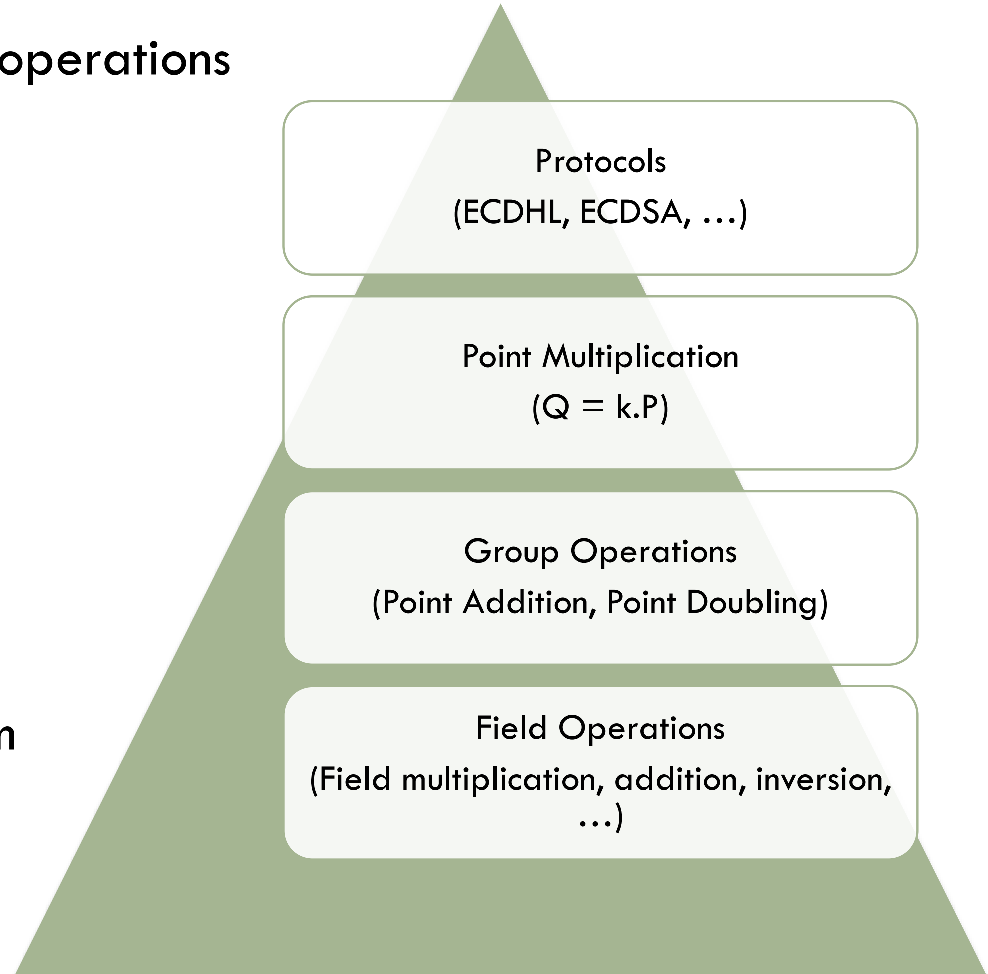
Pipeline Architectures

- Pipeline
  - Area
    - Number of stages
      - Registers
      - Area
    - In comparison with parallel
  - Time
    - CPD
    - #CCs
      - Utilization



Efficient Point Multiplication

- Designing efficient field operations
  - HW
    - Lower area and time
  - SW
    - Fewer instructions
- Field operations
  - scheduling
    - Utilization
    - HW
  - Coordinate systems
    - HW, SW
- Lightweight PM algorithm
  - Fewer group operations
    - HW, SW

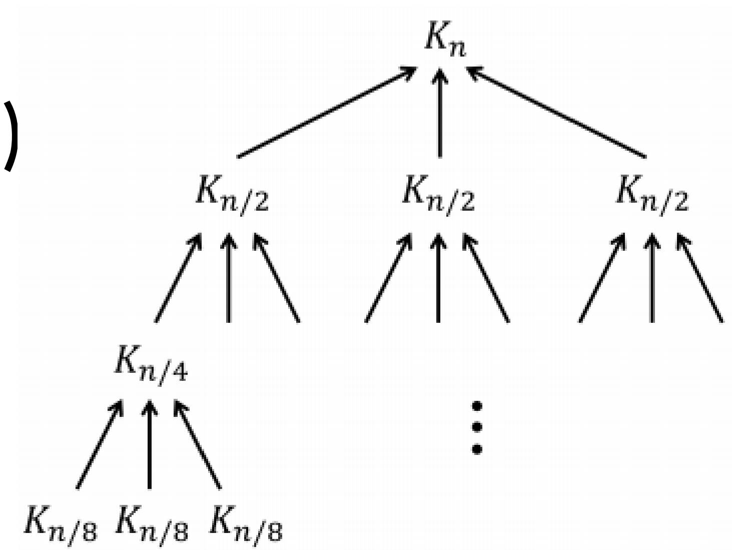


Efficient Field Operations

- Field Multiplication
  - Most effective operation
    - Inversion
      - Fermat's little theorem
      - $a^{-1} \equiv a^{m-2} \pmod{m}$
    - Addition, subtraction
  - Binary extension field
    - Legacy use
    - Squaring
  - Prime field
    - Squaring ⇔ multiplication

Karatsuba-ofman Multiplication (KOM)

- Karatsuba , 1960, Divide and conquer
- K ways, M steps : 2-ways, 3-steps
- $O(n^2) \rightarrow O(n^{\log 3}) \rightarrow O(n^{1.58})$



KOM (cont.)

- 2-ways, 1-step
- 256 bits inputs

$$
\begin{aligned}
A.B &= (a_h 2^{128} + a_l)(b_h 2^{128} + b_l) \\
&= a_h b_h 2^{256} + (a_l b_h + a_h b_l) 2^{128} + a_l b_l \\
&= a_h b_h 2^{256} + [(a_h + a_l)(b_h + b_l) - a_h b_h - a_1 b_l] 2^{128} + a_l b_l.
\end{aligned}
$$

KOM (cont.)

- Finding best ways and steps
- Efficient architecture
  - Pipeline architecture
- Consider your prime!

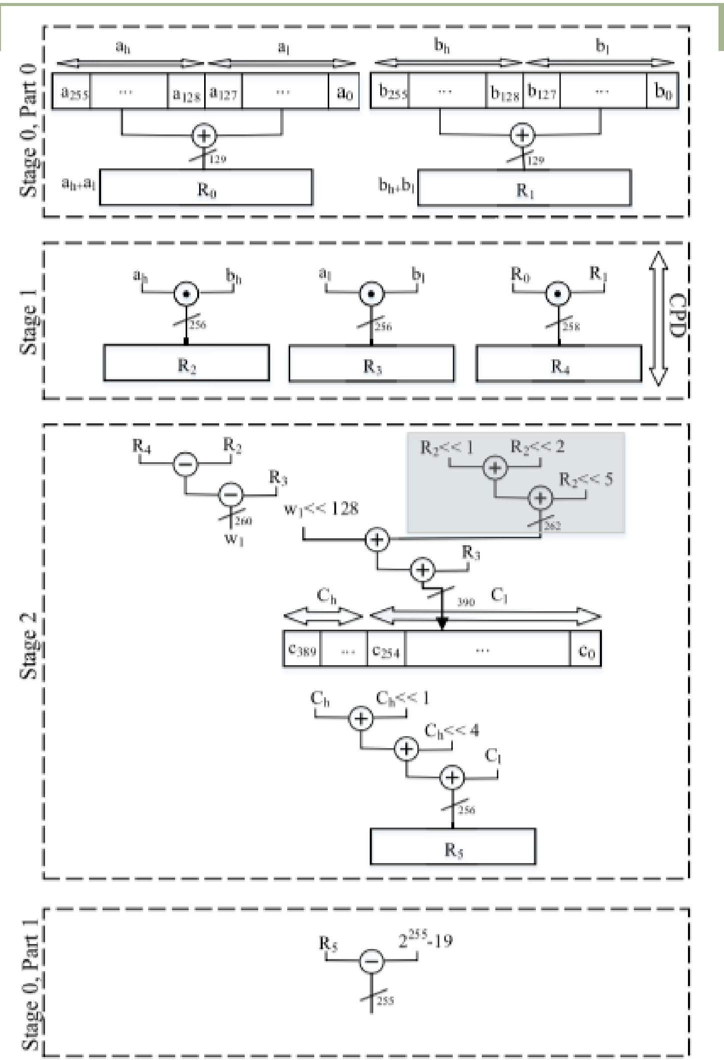
KOM (cont.)

- Prime = 2^{255}-19 => 2^{256}=38
- Reduces the output digits
  - 512 bits to 390 bits
- Reduces the reduction algorithm steps
- Increases a multiply by 38
  - 38 = 100110
  - 3 shifts
  - 2 additions

$$
\begin{aligned}
A.B &= (a_h 2^{128} + a_l)(b_h 2^{128} + b_l) \\
&= a_h b_h 2^{256} + (a_l b_h + a_h b_l) 2^{128} + a_l b_l \\
&= a_h b_h 2^{256} + [(a_h + a_l)(b_h + b_l) - a_h b_h - a_1 b_l] 2^{128} \\
&\quad + a_l b_l.
\end{aligned}
$$

$$
\begin{aligned}
A.B &= (a_h 2^{128} + a_l)(b_h 2^{128} + b_l) \\
&= a_h b_h 2^{256} + (a_l b_h + a_h b_l) 2^{128} + a_l b_l \\
&= a_h b_h 38 + [(a_h + a_l)(b_h + b_l) - a_h b_h - a_1 b_l] 2^{128} \\
&\quad + a_l b_l.
\end{aligned}
$$

KOM (cont.)



Stage 0, Part 0
a_h
a_l
b_h
b_l
a_{255} ... a_{128} a_{127} ... a_0
b_{255} ... b_{128} b_{127} ... b_0
129
a_h+a_l R_0
b_h+b_l R_1
Stage 1
a_h b_h 256 R_2
a_l b_l 256 R_3
R_0 R_1 258 R_4
CPD
Stage 2
R_4 R_2 R_3 260 w_1
R_2<< 1 R_2<< 2 R_2<< 5 262
w_1<< 128
R_3
390
C_h C_l
c_{389} ... c_{254} ... c_0
C_h C_h<< 1 C_h<< 4 C_l
256
R_5
Stage 0, Part 1
R_5 2^{255}-19
255

Efficient Group Operation

- Coordinate System choosing
  - Affine
    - (x,y)
  - Projective, Jacobin, etc.
    - (X,Y,Z)
  - Montgomery-Ladder point multiplication
    - X-coordinate
- Field operations scheduling
  - Iterations overlapping
  - Using pipeline architecture

Efficient Group Operation (cont.)

- Curve 25519
  - GF(2^{255}-19)
- Montgomery PM
  - X-coordinate
- Montgomery Ladder step
  - 18 field operation
    - 4 addition
    - 4 subtraction
    - 10 multiplication
  - Dependency

Algorithm 3 Single Curve265519 Montgomery ladder step

Require: $(X_1, X_2, Z_2, X_3, Z_3)$.
Ensure: New (X_2, Z_2, X_3, Z_3) for next step.

$T_1 \leftarrow X_2 + Z_2$
$T_2 \leftarrow X_2 - Z_2$
$T_3 \leftarrow X_3 + Z_3$
$T_4 \leftarrow X_3 - Z_3$
$T_7 \leftarrow (T_2)^2$
$T_6 \leftarrow (T_1)^2$
$T_5 \leftarrow T_6 - T_7$
$T_9 \leftarrow T_3.T_2$
$T_8 \leftarrow T_4.T_1$
$X_3 \leftarrow T_8 + T_9$
$Z_3 \leftarrow T_8 - T_9$
$X_3 \leftarrow X_3^2$
$Z_3 \leftarrow Z_3^2$
$Z_3 \leftarrow Z_3.X_1$
$X_2 \leftarrow T_6.T_7$
$Z_2 \leftarrow 121666.T_5$
$Z_2 \leftarrow Z_2 + T_7$
$Z_2 \leftarrow Z_2.T_5$
return (X_2, Z_2, X_3, Z_3).

Efficient Group Operation (cont.)

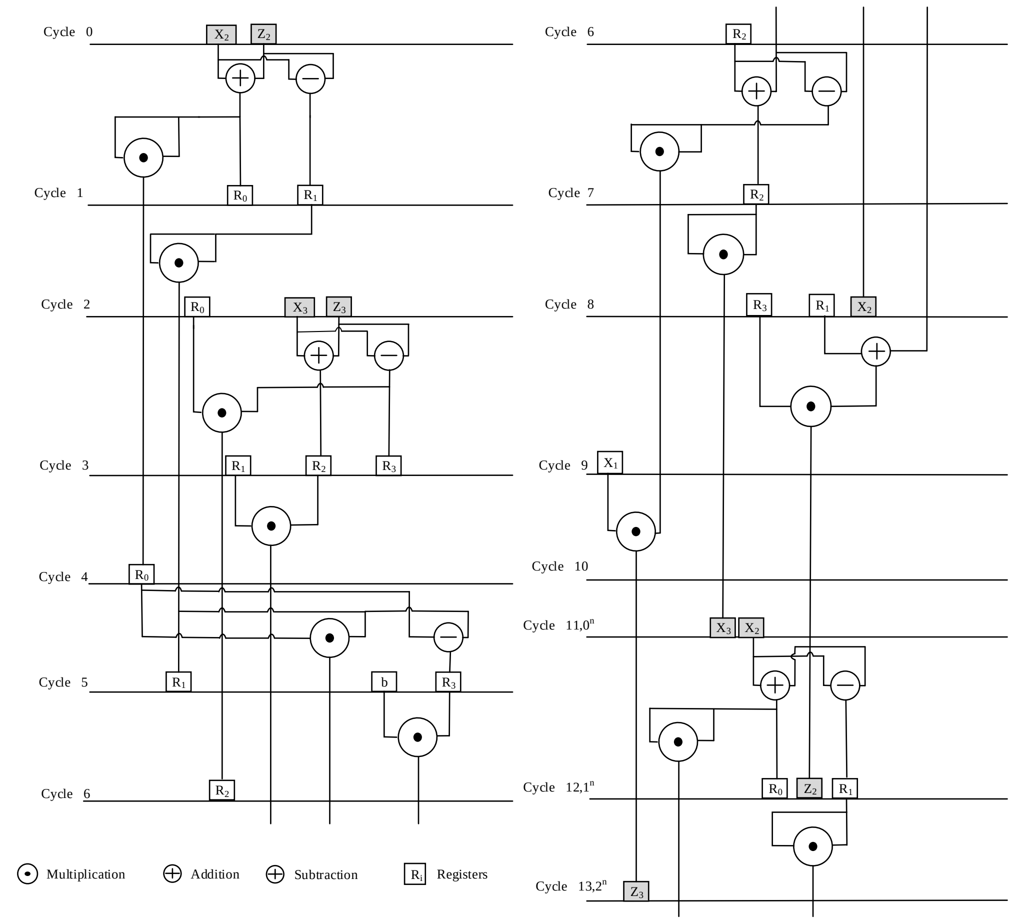
- The most critical decisions
  - How many field multiplications
    - Efficiency
      - Area and time balance
  - How many pipeline stages
    - Average number of independent multiplication
      - Utilization
    - Time
      - CPD
      - Latency

Efficient Group Operation (cont.)

- Best related work scheduling [Koppermann, MICPRO'17]
  - 32 step
  - 2 multiplication
    - 1 addition
    - 1 subtraction

Efficient Group Operation (cont.)

- [Salarifard et al, 2019]
  - 11 step
  - 1 multiplication
    - 1 addition
    - 1 subtraction



Lightweight PM algorithms

- Scalar representation
  - Non-adjacent Form (NAF) representation
    - Decreasing Hamming Weight
- Windowing methods
  - Pre-computation
  - Fewer Point Addition
- Montgomery Ladder
  - X-coordinates

Double and Add Point Multiplication

Input: Elliptic curve E, an elliptic curve point P and a scalar d in binary representation such that $d = d_n \cdots d_0$.

Output: $T = dP\ (mod\ n)$

```
T ← P
    for i ← n − 1 downto 0 do
        T ← T + T mod n
        if d_i = 1 then
            T ← T + P mod n
        end if
    end for
return T
```

Non-adjacent Form (NAF)

- Decreasing Hamming weight
- Removing two equal adjacent non-zero bits

$$
\begin{array}{ccccccccccccc}
 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
 & & & & & & \downarrow & & & & & & \\
 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & -1 \\
 & & & & & & \downarrow & & & & & & \\
 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 \\
 & & & & & & \downarrow & & & & & & \\
1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1
\end{array}
$$

NAF-based Point Multiplication

Input: Elliptic curve E, an elliptic curve point P and a scalar d in binary representation such that $d = d_n \cdots d_0$.

Output: $T = dP\ (mod\ n)$

```
T ← P
    for i ← n − 1 downto 0 do
        T ← T + T mod n
        if d_i = 1 then
            T ← T + P mod n
        else if d_i = −1
            T ← T − P mod n
        end if
    end for
return T
```

Windowing Method Point Multiplication

Input: Elliptic curve E, an elliptic curve point P ,Window width w, $d = [\frac{l}{w}]$ and a scalar $k = (k_{d-1} \cdots k_0)_{2^w}$.

Output: $Q = kP \ (mod\ n)$

1:$P_0 \leftarrow P$

2: Precompute: **for** $i \ \leftarrow \ 1\, upto 2^{w-1}$ **do:** $P_i = P_{i-1} + P$

3: $Q \leftarrow 0$

4: **for** $i \ \leftarrow d - 1\, downto\ 0$ **do**

5: $Q \leftarrow 2^w Q$

6: $Q \leftarrow Q + P_{kd}$

7: **return** Q

Montgomery Ladder Point Multiplication

Input: Elliptic curve E, an elliptic curve point P and a scalar d in binary representation such that $d = d_n \cdots d_0$.

Output: $dP\ (mod\ n)$

```
P_1 ← P and P_2 ← 2P
    for i ← n − 1 downto 0 do
        if d_i = 0 then
            P_1 ← 2P_1 and P_2 ← P_1 + P_2
        else
            P_1 ← P_1 + P_2 and P_2 ← 2P_2
        end if
    end for
return P_1
```

Montgomery Ladder PM (cont.)

- X-coordinate

$$(x, y) \rightarrow \{(X, Z) | Z \neq 0, X = x.Z\}$$

- Point Addition

$$X_3 = 4(XX' - ZZ'),$$
$$Z_3 = 4(XZ' - ZX')x_p$$

- Point Doubling

$$X_2 = (X^2 - Z^2)^2 = (X - Z)^2(X + Z)^2,$$
$$Z_2 = 4XZ(X^2 + AXZ + Z^2),$$

Evaluation

- SW
  - Software code (C, C++, Java, etc)
  - Time
- HW
  - HDL
    - Verilog
    - VHDL
  - FPGA
  - ASIC

Hardware Evaluation (FPGA)

- Xilinx ISE
- Device (Zynq, Vertix, etc.)
  - Related work
- Slice, DSP, CPD, #CCs
- AT = Area X Time
  - Area in FPGA?
    - DSPs
    - Slices
    - One DSP = 100 Slices [Salarifard et al, TCAS-I, 2019].

Hardware Evaluation (ASIC)

- Technology
  - 32 nm, 45 nm, 65 nm , etc.
- Standard Cell Libraries
  - Nangate, TSMC, UMC, etc.
  - Not available in web!
- Estimation
  - Gate (NAND, NOR)
  - CPD
    - Gate delay
  - Delay
    - Number of maximum FFs from input to output
- Implementation
  - Design Compiler
    - Area, CPD, Power, Energy, etc.
  - No free, Vmware (CentOS).

Conclusion

- Efficiency
  - Area and Time
  - Pipeline
- Efficient Point Multiplication
  - Efficient Field Operations
  - Efficient Group Operation
  - Lightweight Point Multiplication algorithm
- Evaluation
  - SW
    - Time
  - HW
    - FPGA and ASIC

References

- Koc, Cetin Kaya. "Cryptographic Engineering." Springer, 2011.
- Hankerson, Darrel, Alfred J. Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.
- Salarifard, Raziyeh, and Siavash Bayat-Sarmadi. "An efficient low-latency point-multiplication over curve25519." *IEEE Transactions on Circuits and Systems I: Regular Papers* 66.10 (2019): 3854-3862.

Any question?

R_salarifard@sbu.ac.ir