Elliptic Curve Cryptograph
- Learning ECC -

Ji Yong-Hyeon

Department of Information Security, Cryptology, and Mathematics
College of Science and Technology
Kookmin University

March 12, 2024

Acknowledgements

Contents

Chapter 1

NIST P-256

Configuration

```
#ifdef _WIN32 // Windows-specific definitions
#include <windows.h>
#include <stdint.h>
/* ... */
typedef DWORD       u32;
typedef DWORDLONG   u64;
#elif defined(__linux__) // Linux-specific definitions
#include <stdint.h>
/* ... */
typedef uint8_t     u8;
typedef uint32_t    u32;
typedef uint64_t    u64;
#else
#error "Unsupported platform"
#endif
// Define this to force 32-bit mode in development
#define FORCE_32_BIT
// Simplified check for 32-bit or forced 32-bit mode
#if defined(FORCE_32_BIT) || !defined(_WIN64) && !defined(
   __x86_64__)
#define IS_32_BIT_ENV
#endif

#ifdef IS_32_BIT_ENV // 32-bit specific settings
#define ONE      0x1U
#define SIZE     8
typedef u32      word;
typedef word     field[SIZE];
#else // 64-bit specific settings
#define ONE      0x1ULL
#define SIZE     4
typedef u64      word;
typedef word     field[SIZE];
#endif
```

1.1 Data Representation

128-bit Hexa-string	0x77FDDC58464B01FC6606BC465BF5CBCB			
String Index	0 ··· 7	8 ··· 15	16 ··· 23	24 ··· 31
Split into Words	77FDDC58 data[3]	464B01FC data[2]	6606BC46 data[1]	5BF5CBCB data[0]
data[0]	(data[0] = '5'-'0';) -> (data[0] ≪= 4;) -> (data[0] |= 'B'-'0';) -> (data[0] ≪= 4;) -> (data[0] |= 'F'-'0';) -> ···			
	0x5 -> 0x50 -> 0x5B -> 0x5B0 -> ···			

```c
void stringToWord(word* wordArray, const char* hexString) {
    size_t length = strlen(hexString) / (SIZE == 8 ? 8 : 16);
    if (length != SIZE) {
        printf("Invaild 128-bit Hexa-string Length!\n");
        return;
    }
    for (size_t i = 0; i < length; i++)
#ifdef IS_32_BIT_ENV
        sscanf(&hexString[i * 8], "%8X",
            &wordArray[(length - 1) - i]);
#else
        sscanf(&hexString[i * 16], "%16lX",
            &wordArray[(length - 1) - i]);
#endif
}
```

```c
int main(void) {
    const char* string = "
        BD91C935C85617B079C6F2728B987CE4
        88BB17B4644D5F8B9C23AF955AB74663
    ";

    word data[SIZE];
    stringToWord(data, string);

    for (i32 i = SIZE-1; i >=0; i--)
#ifdef IS_32_BIT_ENV
        printf("%8X:", data[i]);
#else
        printf("%16lX:", data[i]);
#endif
    puts("");
}
```

```
BD91C935:C85617B0:79C6F272:8B987CE4:88BB17B4:644D5F8B:9C23AF95:5AB74663:
BD91C935C85617B0:79C6F2728B987CE4:88BB17B4644D5F8B:9C23AF955AB74663:
```

Example 1.1 (secp256r1). Consider $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

2^{256}	00000001	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
-2^{224}		00000001	00000000	00000000	00000000	00000000	00000000	00000000	00000000
$+2^{192}$			00000001	00000000	00000000	00000000	00000000	00000000	00000000
$+2^{96}$						00000001	00000000	00000000	00000000
-2^{0}									00000001
p		FFFFFFFF	00000001	00000000	00000000	00000000	FFFFFFFF	FFFFFFFF	FFFFFFFF
p_{inv}		00000000	FFFFFFFE	FFFFFFFF	FFFFFFFF	FFFFFFFF	00000000	00000000	00000001

```
#ifdef IS_32_BIT_ENV
static const field PRIME = {
    0xFFFFFFFFU, 0xFFFFFFFFU, 0xFFFFFFFFU, 0x00000000U,
    0x00000000U, 0x00000000U, 0x00000001U, 0xFFFFFFFFU
};
static const field PRIME_INVERSE = {
    0x00000001U, 0x00000000U, 0x00000000U, 0xFFFFFFFFU,
    0xFFFFFFFFU, 0xFFFFFFFFU, 0xFFFFFFFEU, 0x00000000U
};
#else
static const field PRIME = {
    0xFFFFFFFFFFFFFFFFU, 0x00000000FFFFFFFFU,
    0x0000000000000000U, 0xFFFFFFFF00000001U
};
static const field PRIME_INVERSE = {
    0x0000000000000000U, 0xFFFFFFFF00000000U,
    0xFFFFFFFFFFFFFFFFU, 0x00000000FFFFFFFFU
};
#endif
```

1.2 secp256r1

Define a prime number

$$
\begin{aligned}
p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
&= 2^{32*8} - 2^{32*7} + 2^{32*6} + 2^{32*3} - 2^{32*0}
\end{aligned}
$$

that is used in the context of cryptography, particularly in the construction of elliptic curves for cryptographic purposes. For prime p, let

$$
m = \lceil \log_2 p \rceil, \quad t = \left\lceil \frac{m}{W} \right\rceil .
$$

For example, for $p = 2^{256}$, we have $t = \left\lceil \frac{\lceil \log_2 2^{256} \rceil}{2^{32}} \right\rceil$. Note that

- $A = A[t-1] \parallel \cdots \parallel A[2] \parallel A[1] \parallel A[0]$
- $a = 2^{(t-1)W} A[t-1] + \cdots 2^{2W} A[2] + 2^{W} A[1] + A[0]$

sign	2^2	2^1	2^0	Decimal	One's Complement				Two's Complement			
0	0	0	0	+0	0	0	0	0	0	0	0	0
0	0	0	1	+1	0	0	0	1	0	0	0	1
0	0	1	0	+2	0	0	1	0	0	0	1	0
0	0	1	1	+3	0	0	1	1	0	0	1	1
0	1	0	0	+4	0	1	0	0	0	1	0	0
0	1	0	1	+5	0	1	0	1	0	1	0	1
0	1	1	0	+6	0	1	1	0	0	1	1	0
0	1	1	1	+7	0	1	1	1	0	1	1	1
1	0	0	0	-0	1	1	1	1	0	0	0	0
1	0	0	1	-1	1	1	1	0	1	1	1	1
1	0	1	0	-2	1	1	0	1	1	1	1	0
1	0	1	1	-3	1	1	0	0	1	1	0	1
1	1	0	0	-4	1	0	1	1	1	1	0	0
1	1	0	1	-5	1	0	1	0	1	0	1	1
1	1	1	0	-6	1	0	0	1	1	0	1	0
1	1	1	1	-7	1	0	0	0	1	0	0	1
				-8					1	0	0	0

```c
#ifdef _64BIT_SYSTEM
typedef u64 field_element[4]; // For 64-bit systems
#else
typedef u32 field_element[8]; // For 32-bit systems
#endif

// Example for modular addition (simplified)
void mod_add(field_element a, field_element b, field_element result) {
    uint64_t carry = 0;
    for (int i = 0; i < 4; ++i) { // Assuming 64-bit system
        uint64_t temp = (uint64_t)a[i] + b[i] + carry;
        result[i] = temp & 0xFFFFFFFFFFFFFFFF; // Keep only 64 bits
        carry = temp >> 64; // Carry for the next iteration
    }

    // Modular reduction if necessary
    if (carry || is_greater_or_equal(result, p256)) {
        // Subtract p256 if result >= p256
        subtract_p256(result);
    }
}

```

```
void subtract_p256(field_element x) {
    // This is a simplified version. In practice, you'd need to handle underflows.
    // Subtract (2^256 - 2^224 + 2^192 + 2^96 - 1)
    // In practice, implement this function based on the specific structure of p256
    // and considering the binary representation of the field elements.
}
```

1.3 Multi-Precision Addition

1.3.1 Theory for the Addition

Note. A positive integer $X \in \left[2^{w(n-1)}, 2^{wn}\right)$ is a n-word string. For example, let $w = 32$ and consider 4-word string

$x[3] \cdot 2^{w*3}$	$x[2] \cdot 2^{w*2}$	$x[1] \cdot 2^{w*1}$	$x[0] \cdot 2^{w*0}$

Then

Minimum	00000001	00000000	00000000	00000000	$= 2^{w*3}$
Maximum	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF	$= 2^{w*4} - 1$

Upper and Lower Bound of Addition

Proposition 1.1. *Let $w \in \{32, 64\}$ be a word. Let X and Y are n-word and m-word strings, respectively,* i.e.,

$$X = x[n-1] \parallel \cdots \parallel x[0] \in \left[2^{w(n-1)}, 2^{wn}\right) \quad \textit{with} \quad x[n-1] \neq 0$$
$$Y = y[m-1] \parallel \cdots \parallel y[0] \in \left[2^{w(m-1)}, 2^{mn}\right) \quad \textit{with} \quad y[m-1] \neq 0$$

Then

$$2^{w\cdot(\max(n,m)-1)} < X + Y < 2^{w\cdot(\max(m,n)+1)}.$$

Proof. Let $W := 2^w$. Then X and Y can be expressed as follows: $\begin{cases} X = xW^{n-1} + X' \\ Y = yW^{m-1} + Y' \end{cases}$ where

$$a, b \in (0, W), \quad A' \in [0, W^{n-1} - 1], \quad B' \in [0, W^{m-1} - 1].$$

Suppose that $n \geq m$ then

$$\begin{aligned} W^{n-1} \leq \max(A, B) < A + B &= (aW^{n-1} + A') + (bW^{m-1} + B') \\ &< (a+b)W^{n-1} + (W^{n-1} + W^{n-1}) \\ &= (a+b+2)W^{n-1} \\ &\leq ((W-1) + (W-1) + 2)W^{n-1} \\ &= 2W^n \leq W^{n+1}. \end{aligned}$$

Thus $W^{n-1} < A + B < W^{n+1}$. Here, $n = \max(n, m)$. □

Corollary 1.1.1.

$$\mathsf{len}_{\mathsf{word}}(X) = t = \mathsf{len}_{\mathsf{word}}(Y) \implies \mathsf{len}_{\mathsf{word}}(X + Y) \leq t + 1.$$

Single-Word Addition $x[i] + y[i]$

Proposition 1.2. *Let $x, y \in [0, 2^w)$ are single-words.*

(1) $\mathsf{len}_{\mathsf{word}}(x + y) \leq 2$.

(2) *(Division Theorem)*

$$x + y = q2^w + r = \left\lfloor \frac{x+y}{2^w} \right\rfloor \cdot 2^w + (x \boxplus y).$$

(3) *(carry)* $\left\lfloor \frac{x+y}{2^w} \right\rfloor \in \{0, 1\}$.

(4) $(\star)$ $x \boxplus y < x \quad \Leftrightarrow \quad \left\lfloor \frac{x+y}{2^w} \right\rfloor = 1$.

Note.

$x \boxplus y < x$	`True`	carry $= 1$
	`False`	carry $= 0$

Single-Word Addition with Carry $x[i] \boxplus y[i] + \varepsilon$

Proposition 1.3. *Let $x, y \in [0, 2^w)$ and $\varepsilon \in \{0, 1\}$.*

(1) $\mathsf{len}_{\mathsf{word}}((x \boxplus y) + \varepsilon) \leq 2$.

(2) *(carry)* $\left\lfloor \frac{(x \boxplus y) + \varepsilon}{2^w} \right\rfloor \in \{0, 1\}$.

(3)

$$\left\lfloor \frac{x+y}{2^w} \right\rfloor = 1 \implies \left\lfloor \frac{(x \boxplus y) + \varepsilon}{2^w} \right\rfloor = 0.$$

Note.

$x \boxplus y < x$	`True`	carry $= 1$	$\Longrightarrow$
	`False`	carry $= 0$	

1.3.2 Practice for the Addition

Algorithm 1: Multi-Precision Addition

Input: $X, Y\ [0, 2^{wt}) \subseteq \mathbb{Z}$ `// (w,t) = (32,8) or (w,t) = (64,4) for secp256r1`

Output: (ε, Z) where $Z = (X + Y) \bmod 2^{wt}$ and $\varepsilon \in \{0, 1\}$ is carry bit

Function Addition_Core(ε, Z, X, Y):
- $(\varepsilon, z[0]) \leftarrow x[0] + y[0]$
- **for** $i = 1$ **to** $t - 1$ **do**
  - $(\varepsilon, z[i]) \leftarrow x[i] + y[i] + \varepsilon$
- **end**
- **return** $(\varepsilon, Z = z[t-1] \parallel z[t-2] \parallel \cdots \parallel z[0])$

end

Example 1.2.

X		0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
Y	+	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
ε	1	1	1	1	0
Z	0x00000001	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFE

Note (How to Compute Carry?)**.** content...

Algorithm 2: Addition in $\mathbb{F}_p$

Input: Modulus p, and integer $X, Y \in [0, p)$

Output: $Z = (X + Y) \bmod p$

- $(\varepsilon, Z) \leftarrow x[0] + y[0]$
- **for** $i = 1$ **to** $t - 1$ **do**
  - $(\varepsilon, z[i]) \leftarrow x[i] + y[i] + \varepsilon$
- **end**
- **return** $(\varepsilon, Z = z[t-1] \parallel z[t-2] \parallel \cdots \parallel z[0])$

Example 1.3.

ε	1	1	1	1	1	1	1	0	0
X		BD91C935	C85617B0	79C6F272	8B987CE4	88BB17B4	644D5F8B	9C23AF95	5AB74663
Y	+	4E272A73	41569559	F3E58053	BE961728	D67BF71E	FBA44BF2	83DAA7ED	9BF6DDA8
Z	1	0BB8F3A9	09ACAD0A	6DAC72C6	4A2E940D	5F370ED3	5FF1AB7E	1FFE5782	F6AE240B
$-p$		FFFFFFFF	00000001	00000000	00000000	00000000	FFFFFFFF	FFFFFFFF	FFFFFFFF
$Z - p$		0BB8F3AA	09ACAD09	6DAC72C6	4A2E940D	5F370ED2	5FF1AB7E	1FFE5782	F6AE240C

Note that

ε	0	1	1	1	1	0	0	0	0
Z	1	0BB8F3A9	09ACAD0A	6DAC72C6	4A2E940D	5F370ED3	5FF1AB7E	1FFE5782	F6AE240B
$+p_{\text{inv}}$		00000000	FFFFFFFE	FFFFFFFF	FFFFFFFF	FFFFFFFF	00000000	00000000	00000001
$Z + p_{\text{inv}}$	1	0BB8F3AA	09ACAD09	6DAC72C6	4A2E940D	5F370ED2	5FF1AB7E	1FFE5782	F6AE240C

1.4 Multi-Precision Subtraction

1.4.1 Theory for the Subtraction

Upper and Lower Bound of Subtraction

Proposition 1.4. *Let $w \in \{32, 64\}$ be a word. Let X and Y are n-word and m-word strings, respectively,* i.e.,

$$X = x[n-1] \parallel \cdots \parallel x[0] \in \left[2^{w(n-1)}, 2^{wn}\right) \quad \textit{with} \quad x[n-1] \neq 0$$
$$Y = y[m-1] \parallel \cdots \parallel y[0] \in \left[2^{w(m-1)}, 2^{wm}\right) \quad \textit{with} \quad y[m-1] \neq 0$$

Then, for $X \geq Y$,

$$0 \leq X - Y < X < 2^{wn}.$$

1.4.2 Practice for the Subtraction

Algorithm 3: Multi-Precision Subtraction

Input: $X, Y \left[0, 2^{wt}\right) \subseteq \mathbb{Z}$ // $(w,t) = (32,8)$ or $(w,t) = (64,4)$ for secp256r1
Output: (ε, Z) where $Z = (X - Y) \bmod 2^{wt}$ and $\varepsilon \in \{0,1\}$ is borrow bit

Function ADDITION_CORE(ε, Z, X, Y)**:**
  $(\varepsilon, z[0]) \leftarrow x[0] - y[0]$
  for $i = 1$ **to** $t-1$ **do**
    $(\varepsilon, z[i]) \leftarrow x[i] - y[i] - \varepsilon$
  end
  return $(\varepsilon, Z = z[t-1] \parallel z[t-2] \parallel \cdots \parallel z[0])$
end

Example 1.4.

X		0x00000000	0x00000000	0x00000000	0x00000000
$-Y$		0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
$-\varepsilon$	1	1	1	1	0
Z	0x00000001	0x00000000	0x00000000	0x00000000	0x00000001

1.5 Multi-Precision Multiplication

Note (Memory for the Multiplication). Let X and Y are n-word and m-word strings, i.e.,

$$X \in \left[2^{w(n-1)}, 2^{wn}\right), \quad Y \in \left[2^{w(m-1)}, 2^{wm}\right)$$

Then

$$\mathsf{len}_{\mathsf{word}}(XY) \in \{n+m-1, n+m\}.$$

Proof. Since

$$\begin{aligned} 2^{w(n-1)} \cdot 2^{w(m-1)} &\le X \cdot Y < 2^{wn} \cdot 2^{wm}, \\ 2^{w(n+m-2)} &\le XY < 2^{w(n+m)}, \end{aligned}$$

we have $XY \in \left[2^{w(n+m-2)}, 2^{w(n+m)}\right) = \left[2^{w(n+m-2)}, 2^{w(n+m-1)}\right) \cup \left[2^{w(n+m-1)}, 2^{w(n+m)}\right)$. Thus, either

$$\mathsf{len}_{\mathsf{word}}(XY) = n+m-1 \quad \text{or} \quad \mathsf{len}_{\mathsf{word}}(XY) = n+m.$$

□

Note (Single-Word Multiplication).

				570D:C5EE
×				4857:A994
	(C5EE*A994) $\ll$ 00/2	=		831C:8B98
+	(570D*A994) $\ll$ 32/2	=	0000:39A9	E884:0000
+	(4847*C5EE) $\ll$ 32/2	=	0000:37E1	D502:0000
+	(570D*4847) $\ll$ 32/1	=	1893:CC9B	0000:0000
			1899:AF03	9F82:8B98

Note that

$$u, v \in \left[0, 2^{w/2}\right) \implies uv \in \left[0, 2^{w}\right).$$

Let $x, y \in [0, 2^w)$ satisfy the followings:

$$x = x_1 2^{w/2} + x_0, \quad y = y_1 2^{w/2} + y_0$$

The product $xy \in \left[0, 2^{2w}\right)$ can be calculated using four $w/2$-bit integer multiplication operations:

		$x_1 : x_0$	
×		$y_1 : y_0$	
		x_0y_0	
	x_1y_0		
	y_1x_0		
x_1y_1			

$$\begin{aligned} xy &= (x_1 2^{w/2} + x_0)(y_1 2^{w/2} + y_0) \\ &= (x_1y_1)2^w + x_0y_0 + (x_1y_0 + y_1x_0)2^{w/2} \\ &= ((x_1y_2 \ll w) + x_0y_0) + ((x_1y_0 + y_1x_0) \ll w/2). \end{aligned}$$

Cost:

$$x \cdot y \in \left[0, 2^{2w}\right) \implies 4 \cdot \mathbf{M}_{w/2} + 3 \cdot \mathbf{A}_{2w}$$

$$XY \in \left[0, 2^{w(n+m)}\right) \implies (2n \cdot 2m) \cdot \mathbf{M}_{2/w} + (2n \cdot 2m - 1) \cdot \mathbf{A}_{(n+m)w}$$

Example 1.5.

(1) Let $w = 32$ then

$$4 \cdot \mathbf{M}_{16} + 3 \cdot \mathbf{A}_{64}.$$

(2) Let $w = 64$ then

$$4 \cdot \mathbf{M}_{32} + 3 \cdot \mathbf{A}_{128}.$$

Algorithm 4: Single-Word Multiplication $x[i] \cdot y[i]$

Input: $x, y \in [0, 2^w)$

Output: $z = xy \in \left[0, 2^{2 \cdot w}\right)$

1 **Function** Mul_Single(x, y):

2 $x_1, x_0 \leftarrow x_{[w:w/2]}, x_{[w/2:0]}$ `//` $x = x_1 \parallel x_0$

3 $y_1, y_0 \leftarrow y_{[w:w/2]}, y_{[w/2:0]}$ `//` $y = y_1 \parallel y_0$

4 $t_1, t_0 \leftarrow x_1 y_0, x_0 y_1$ `// Cross Mul.` $t_1, t_0 \in [0, 2^w)$

`/*` $x_1 y_0 + x_0 y_1 = t_1 2^w + t_0$, `where` $t_1 \in \{0, 1\}$ `is carry */`

5 $t_0 \leftarrow t_1 \boxplus t_0$

6 $t_1 \leftarrow t_0 < t_1$ `//` $t_1 \boxplus t_0 < t_1$

7 $z_1, z_0 \leftarrow x_1 y_1, x_0 y_0$ `//` $z_1, z_0 \in [0, 2^w)$

8 $t \leftarrow z_0$

9 $z_0 \leftarrow z_0 \boxplus (t_0 \ll w/2)$ `//` $z_0 = \left[x_0 y_0 + (x_1 y_0 + x_0 y_1) 2^{w/2}\right] \bmod 2^w$

10 $z_1 \leftarrow z_1 + (t_1 \ll w/2) + (t_0 \gg w/2) + (z_0 < t)$ `//` $z_1 \in [0, W)$

`/*` $z_1 = x_1 y_1 + (t_0 < t_1) 2^{w/2} + \left\lfloor t_0 / 2^{w/2} \right\rfloor +$ `(carry in line 9) */`

11 **return** $(z_1 \ll w) + z_0$ `//` $z \leftarrow z_1 \parallel z_0 \in \left[0, 2^{2w}\right)$

12 **end**

1.5.1 School-Book Multiplication

$$Z = XY = \left(\sum_{i=0}^{n-1} x_i 2^{iw}\right)\left(\sum_{j=0}^{m-1} y_j 2^{jw}\right) = \sum_{j=0}^{m-1}\left(\sum_{i=0}^{n-1}(x_i y_j)2^{w(i+j)}\right) \in \left[0, 2^{w(n+m)}\right).$$

Cost:

$$\left(\sum_{i=0}^{n-1} x_i 2^{iw}\right)\left(\sum_{j=0}^{m-1} y_j 2^{jw}\right) \implies (2n \cdot 2m) \cdot \mathbf{M}_{w/2} + (2n \cdot 2m - 1) \cdot \mathbf{A}_{(n+m)w}$$

Example 1.6. Let $x, y \in \left[2^{w(t-1)}, 2^{wt}\right)$.

$((w, t) = (32, 8))$

$$256 \cdot \mathbf{M}_{16} + 255 \cdot \mathbf{A}_{512}$$

$((w, t) = (64, 4))$

$$64 \cdot \mathbf{M}_{32} + 63 \cdot \mathbf{A}_{512}$$

Algorithm 5: School-Book Multiplication

Input: $X, Y \in \left[2^{w(t-1)}, 2^{wt}\right)$ `// (w,t) = (32,8) or (w,t) = (64,4) for secp256r1`

Output: $Z = X \cdot Y \in \left[2^{2w(t-1)}, 2^{wt+wt}\right)$

```
Z ← 0                                   // Initialization
for i = 0 to t − 1 // for X
do
    for j = 0 to t − 1 // for Y
    do
        T ← x_i y_j                      // T ∈ [0, 2^{2w})
        T ≪← w(i + j)    // ∈ [2^w, 2^{3w}), ..., [2^{w(t−1)}, 2^{w(t+1)}), ..., [2^{w(2t−2)}, 2^{2wt})
        Z ← Addition_Core(Z, T)
    end
end
return Z
```

<table>
<tr><td></td><td></td><td></td><td></td><td>x_3</td><td>x_2</td><td>x_1</td><td>x_0</td></tr>
<tr><td>$\times$</td><td></td><td></td><td></td><td>y_3</td><td>y_2</td><td>y_1</td><td>y_0</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="2">x_0y_0</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="2">x_0y_1</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2">x_0y_2</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td colspan="2">x_0y_3</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="2">x_1y_0</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2">x_1y_1</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td colspan="2">x_1y_2</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">x_1y_3</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="2">x_2y_0</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td colspan="2">x_2y_1</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">x_2y_2</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td colspan="2">x_2y_3</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td colspan="2">x_3y_0</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">x_3y_1</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td colspan="2">x_3y_2</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">x_3y_3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Cost:

$$\left(t^2\right) \cdot \mathbf{M}_w + (2t-1) \cdot \mathbf{A}_{2wt}$$

Example 1.7. Let $x, y \in \left[2^{w(t-1)}, 2^{wt}\right)$.

$((w,t) = (32,8))$

$$64 \cdot \mathbf{M}_{32} + 15 \cdot \mathbf{A}_{512}$$

$((w,t) = (64,4))$

$$16 \cdot \mathbf{M}_{64} + 7 \cdot \mathbf{A}_{512}$$

1.5.2 Product Scanning (Comba)

Let $n = 2p$ and $m = 2q$, and let

$$X = x_{2p-1} \parallel \cdots \parallel x_0 = \sum_{i=1}^{2p-1} x_i 2^{wi}, \quad Y = y_{2q-1} \parallel \cdots \parallel y_0 = \sum_{j=1}^{2q-1} y_j 2^{wj}$$

with $x_i, y_i \in [0, 2^w)$.

				x_3	x_2	x_1	x_0
$\times$				y_3	y_2	y_1	y_0
				x_2y_0		x_0y_0	
			x_3y_0		x_1y_0		
			x_2y_1		x_0y_1		
		x_3y_1		x_1y_1			
		x_2y_2		x_0y_2			
	x_3y_2		x_1y_2				
	x_2y_3		x_0y_3				
x_3y_3		x_1y_2					

$$\begin{aligned}
Z = XY = \left(\sum_{i=0}^{2p-1} x_i 2^{iw}\right)\left(\sum_{j=0}^{2q-1} y_j 2^{jw}\right) &= \sum_{j=0}^{2q-1}\left(\sum_{i=0}^{2p-1}(x_i y_j) 2^{w(i+j)}\right) \\
&= \sum_{j=0}^{2q-1}\left(\sum_{i=0}^{p-1}(x_{2i} y_j) 2^{w(2i+j)} + \sum_{i=0}^{p-1}(x_{2i+1} y_j) 2^{w(2i+1+j)}\right) \\
&= \sum_{j=0}^{2q-1}\left(\left(\sum_{i=0}^{p-1}(x_{2i} y_j) 2^{2iw}\right) 2^{wj} + \left(\sum_{i=0}^{p-1}(x_{2i+1} y_j) 2^{2iw}\right) 2^{w(j+1)}\right) \\
&= \sum_{j=0}^{2q-1}\left(\left(\sum_{i=0}^{p-1}(x_{2i} y_j) 2^{2iw}\right) + \left(\sum_{i=0}^{p-1}(x_{2i+1} y_j) 2^{2iw}\right) 2^{w}\right) 2^{wj}
\end{aligned}$$

Cost:

$$\left(\sum_{i=0}^{2p-1} x_i 2^{iw}\right)\left(\sum_{j=0}^{2q-1} y_j 2^{jw}\right) \implies (2p \cdot 2q) \cdot \mathbf{M}_w + (2q - 1) \cdot \mathbf{A}_{(2p+2q)w}$$

Example 1.8. Let $x, y \in \left[2^{w(t-1)}, 2^{wt}\right)$.

$((w, t) = (32, 8))$

$$64 \cdot \mathbf{M}_{32} + 15 \cdot \mathbf{A}_{512}$$

$((w, t) = (64, 4))$

$$16 \cdot \mathbf{M}_{64} + 3 \cdot \mathbf{A}_{512}$$

1.6 Multi-Precision Squaring

Note (Memory for the Squaring). Let X be a n-word strings, i.e., $X \in \left[2^{w(n-1)}, 2^{wn}\right)$. Then

$$\mathsf{len}_{\mathsf{word}}(X^2) \in \{2n-1, 2n\}.$$

Note (Single-Word Squaring)**.**

			`570D:C5EE`
×			`570D:C5EE`
	`(C5EE*C5EE)`$\ll 00/2$ =		`9908:2944`
+	`(570D*C5EE)`$\ll 32/2$ =	`0000:434D`	`EF16:0000`
+	`(570D*C5EE)`$\ll 32/2$ =	`0000:434D`	`EF16:0000`
+	`(570D*570D)`$\ll 32/1$ =	`1D99:D6A9`	`0000:0000`
		`1D9A:5D45`	`7734:2944`

The squaring $x^2 \in \left[0, 2^{2w}\right)$ can be calculated using four $w/2$-bit integer squaring operations:

$$\begin{array}{cccc}
 & & x_1 & : x_0 \\
\times & & x_1 & : x_0 \\
\hline
 & & \multicolumn{2}{c}{x_0^2} \\
 & \multicolumn{2}{c}{x_1x_0} & \\
 & \multicolumn{2}{c}{x_1x_0} & \\
\multicolumn{2}{c}{x_1^2} & &
\end{array}$$

$$\begin{aligned}
x^2 &= (x_1 2^{w/2} + x_0)^2 \\
&= x_1^2 2^w + x_0^2 + (2x_1x_0)2^{w/2} \\
&= ((x_1^2 \ll w) + x_0^2) + (x_1x_0 \ll (w/2+1)).
\end{aligned}$$

Cost:

$$x^2 \in \left[0, 2^{2w}\right) \implies 3 \cdot \mathbf{M}_{w/2} + 2 \cdot \mathbf{A}_{2w}$$

$$X^2 \in \left[0, 2^{w(2n)}\right) \implies (3n) \cdot \mathbf{M}_{2/w} + (3n-1) \cdot \mathbf{A}_{(n+m)w}$$

Example 1.9.

(1) Let $w = 32$ then

$$4 \cdot \mathbf{M}_{16} + 3 \cdot \mathbf{A}_{64}.$$

(2) Let $w = 64$ then

$$4 \cdot \mathbf{M}_{32} + 3 \cdot \mathbf{A}_{128}.$$

Algorithm 6: Single-Word Squaring $x[i]^2$

Input: $x \in [0, 2^w)$
Output: $x^2 \in \left[0, 2^{2 \cdot w}\right)$

Function SQU_SINGLE(x, y):
  $x_1, x_0 \leftarrow x_{[w:w/2]}, x_{[w/2:0]}$ // $x = x_1 \parallel x_0$
  $z_1, z_0 \leftarrow x_1^2, x_0^2$ // $z_i \in [0, 2^w)$
  $Z \leftarrow (z_1 \ll w) + z_0$ // $Z \in \left[0, 2^{2w}\right)$
  $T \leftarrow z_1 z_0$ // $T \in \left[0, 2^{2w}\right)$
  $T \ll\leftarrow (w/2 + 1)$
  $Z \leftarrow Z + T$
  return Z
end

1.7 Fast Reduction

$$\begin{aligned}
p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
0 &\equiv 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \pmod{p_{256}} \\
2^{256=32*8} &\equiv 2^{224} - 2^{192} - 2^{96} + 1 \pmod{p_{256}}
\end{aligned}$$

$$\begin{aligned}
2^{288=32*9} &\equiv 2^{256} - 2^{224} - 2^{128} + 2^{32} \pmod{p_{256}} \\
2^{288} &\equiv (2^{224} - 2^{192} - 2^{96} + 1) - 2^{224} - 2^{128} + 2^{32} \pmod{p_{256}} \\
2^{288} &\equiv -2^{192} - 2^{128} - 2^{96} + 2^{32} + 1 \pmod{p_{256}}
\end{aligned}$$

$$2^{320=32*10} \equiv -2^{224} - 2^{160} - 2^{128} + 2^{64} + 2^{32} \pmod{p_{256}}$$

$$\begin{aligned}
2^{352=32*11} &\equiv -2^{256} - 2^{192} - 2^{160} + 2^{96} + 2^{64} \pmod{p_{256}} \\
2^{352} &\equiv -(2^{224} - 2^{192} - 2^{96} + 1) - 2^{192} - 2^{160} + 2^{96} + 2^{64} \pmod{p_{256}} \\
2^{352} &\equiv -2^{224} - 2^{160} + 2 \cdot 2^{96} + 2^{64} - 1 \pmod{p_{256}}
\end{aligned}$$

$$\begin{aligned}
2^{384=32*12} &\equiv -2^{256} - 2^{192} + 2 \cdot 2^{128} + 2^{96} - 2^{32} \pmod{p_{256}} \\
2^{384} &\equiv -(2^{224} - 2^{192} - 2^{96} + 1) - 2^{192} + 2 \cdot 2^{128} + 2^{92} - 2^{32} \pmod{p_{256}} \\
2^{384} &\equiv -2^{224} + 2 \cdot 2^{128} + 2 \cdot 2^{96} - 2^{32} - 1 \pmod{p_{256}}
\end{aligned}$$

$$\begin{aligned}
2^{416=32*13} &\equiv -2^{256} + 2 \cdot 2^{160} + 2 \cdot 2^{128} - 2^{64} - 2^{32} \pmod{p_{256}} \\
2^{416} &\equiv -(2^{224} - 2^{192} - 2^{96} + 1) + 2 \cdot 2^{160} + 2 \cdot 2^{128} - 2^{64} - 2^{32} \pmod{p_{256}} \\
2^{416} &\equiv -2^{224} + 2^{192} + 2 \cdot 2^{160} + 2 \cdot 2^{128} + 2^{96} - 2^{64} - 2^{32} - 1 \pmod{p_{256}}
\end{aligned}$$

$$\begin{aligned}
2^{448=32*14} &\equiv -2^{256} + 2^{224} + 2 \cdot 2^{192} + 2 \cdot 2^{160} + 2^{128} - 2^{96} - 2^{64} - 2^{32} \pmod{p_{256}} \\
2^{448} &\equiv -(2^{224} - 2^{192} - 2^{96} + 1) + 2^{224} + 2 \cdot 2^{192} + 2 \cdot 2^{160} + 2^{128} - 2^{96} - 2^{64} - 2^{32} \pmod{p_{256}} \\
2^{448} &\equiv 3 \cdot 2^{192} + 2 \cdot 2^{160} + 2^{128} - 2^{64} - 2^{32} - 1 \pmod{p_{256}}
\end{aligned}$$

$$2^{480=32*15} \equiv 3 \cdot 2^{224} + 2 \cdot 2^{192} + 2^{160} - 2^{96} - 2^{64} - 2^{32} \pmod{p_{256}}$$

Consider $Z = z_{15} \parallel z_{14} \parallel \cdots \parallel z_0 = \sum_{i=0}^{15} z_i 2^{32i}$ with $Z \in \left[0, p_{256}^2\right)$.

$$
\begin{aligned}
z_8 \cdot 2^{256=32*8} &\equiv z_8 \cdot (+1 \cdot 2^{224} - 1 \cdot 2^{192} - 1 \cdot 2^{96} + 1) \\
z_9 \cdot 2^{288=32*9} &\equiv z_9 \cdot (- 1 \cdot 2^{192} - 1 \cdot 2^{128} - 1 \cdot 2^{96} + 2^{32} + 1) \\
z_{10} \cdot 2^{320=32*10} &\equiv z_{10} \cdot (-1 \cdot 2^{224} - 1 \cdot 2^{160} - 1 \cdot 2^{128} + 2^{64} + 2^{32}) \\
z_{11} \cdot 2^{352=32*11} &\equiv z_{11} \cdot (-1 \cdot 2^{224} - 1 \cdot 2^{160} + 2 \cdot 2^{96} + 2^{64} - 1) \\
z_{12} \cdot 2^{384=32*12} &\equiv z_{12} \cdot (-1 \cdot 2^{224} + 2 \cdot 2^{128} + 2 \cdot 2^{96} - 2^{32} - 1) \\
z_{13} \cdot 2^{416=32*13} &\equiv z_{13} \cdot (-1 \cdot 2^{224} + 1 \cdot 2^{192} + 2 \cdot 2^{160} + 2 \cdot 2^{128} + 1 \cdot 2^{96} - 2^{64} - 2^{32} - 1) \\
z_{14} \cdot 2^{448=32*14} &\equiv z_{14} \cdot (+ 3 \cdot 2^{192} + 2 \cdot 2^{160} + 1 \cdot 2^{128} - 2^{64} - 2^{32} - 1) \\
z_{15} \cdot 2^{480=32*15} &\equiv z_{15} \cdot (+3 \cdot 2^{224} + 2 \cdot 2^{192} + 1 \cdot 2^{160} - 1 \cdot 2^{96} - 2^{64} - 2^{32})
\end{aligned}
$$

	2^{224}	2^{192}	2^{160}	2^{128}	2^{96}	2^{64}	2^{32}	2^{0}
-	$+z_7$	$+z_6$	$+z_5$	$+z_4$	$+z_3$	$+z_2$	$+z_1$	$+z_0$
z_8	$+1$	-1				-1		$+1$
z_9		-1		-1	-1		$+1$	$+1$
z_{10}	-1		-1	-1		$+1$	$+1$	
z_{11}	-1		-1		$+2$	$+1$		-1
z_{12}	-1			$+2$	$+2$		-1	-1
z_{13}	-1	$+1$	$+2$	$+2$	$+1$	-1	-1	-1
z_{14}		$+3$	$+2$	$+1$		-1	-1	-1
z_{15}	$+3$	$+2$	$+1$		-1	-1	-1	

Algorithm 7: 32-bit Fast Reduction modulo $p_{256} = 2^{256} - 2^{224} + 2^{196} + 2^{96} - 1$

Input: $Z = z_{15} \parallel z_{14} \parallel \cdots \parallel z_0 = \sum_{i=0}^{15} z_i 2^{32i} \in \left[0, p_{256}^2\right)$

Output: $Z \bmod p_{256}$

$s_1 = z_{07} \parallel z_{06} \parallel z_{05} \parallel z_{04} \parallel z_{03} \parallel z_{02} \parallel z_{01} \parallel z_{00}$

$s_2 = z_{15} \parallel z_{14} \parallel z_{13} \parallel z_{12} \parallel z_{11} \parallel 0^{32} \parallel 0^{32} \parallel 0^{32}$

$s_3 = 0^{32} \parallel z_{15} \parallel z_{14} \parallel z_{13} \parallel z_{12} \parallel 0^{32} \parallel 0^{32} \parallel 0^{32}$

$s_4 = z_{15} \parallel z_{14} \parallel 0^{32} \parallel 0^{32} \parallel 0^{32} \parallel z_{10} \parallel z_{09} \parallel z_{08}$

$s_5 = z_{08} \parallel z_{13} \parallel z_{15} \parallel z_{14} \parallel z_{13} \parallel z_{11} \parallel z_{10} \parallel z_{09}$

$s_6 = z_{10} \parallel z_{08} \parallel 0^{32} \parallel 0^{32} \parallel 0^{32} \parallel z_{13} \parallel z_{12} \parallel z_{11}$

$s_7 = z_{11} \parallel z_{09} \parallel 0^{32} \parallel 0^{32} \parallel z_{15} \parallel z_{14} \parallel z_{13} \parallel z_{12}$

$s_8 = z_{12} \parallel 0^{32} \parallel z_{10} \parallel z_{09} \parallel z_{08} \parallel z_{15} \parallel z_{14} \parallel z_{13}$

$s_9 = z_{13} \parallel 0^{32} \parallel z_{11} \parallel z_{10} \parallel z_{09} \parallel 0^{32} \parallel z_{15} \parallel z_{14}$

return $(s_1 + 2s_2 + 2s_3 + s_4 + s_5 - s_6 - s_7 - s_8 - s_9 \bmod p_{256})$

Consider

$$Z = \zeta_7 \parallel \zeta_6 \parallel \cdots \parallel \zeta_0 = \sum_{j=0}^{7} \zeta_i 2^{64j}$$

with $Z \in \left[0, p_{256}^2\right)$.

$$\begin{aligned}
\zeta_4 \cdot 2^{256=64*4} &\equiv \zeta_4 \cdot (2^{224} - 2^{192} - 2^{96} + 1) \\
\zeta_5 \cdot 2^{320=64*5} &\equiv \zeta_5 \cdot (-2^{224} - 2^{160} - 2^{128} + 2^{64} + 2^{32}) \\
\zeta_6 \cdot 2^{384=64*6} &\equiv \zeta_6 \cdot (-2^{224} + 2 \cdot 2^{128} + 2 \cdot 2^{96} - 2^{32} - 1) \\
\zeta_7 \cdot 2^{448=64*7} &\equiv \zeta_7 \cdot (3 \cdot 2^{192} + 2 \cdot 2^{160} + 2^{128} - 2^{64} - 2^{32} - 1)
\end{aligned}$$

Algorithm 8: 64-bit Fast Reduction modulo $p_{256} = 2^{256} - 2^{224} + 2^{196} + 2^{96} - 1$

Input: $Z = \zeta_7 \parallel \zeta_6 \parallel \cdots \parallel \zeta_0 = \sum_{i=0}^{7} z_i 2^{64i} \in \left[0, p_{256}^2\right)$ with $\zeta_i = z_{2i+1} \parallel z_{2i}$

Output: $Z \bmod p_{256}$

$s_1 = \zeta_3 \parallel \zeta_2 \parallel \zeta_1 \parallel \zeta_0$

$s_2 = \zeta_7 \parallel \zeta_6 \parallel (\zeta_5 \mathbin{\&} \mathtt{0xF}^8\mathtt{0}^8) \parallel \mathtt{0}^{64}$

$s_3 = (\zeta_7 \gg 32) \parallel (((\zeta_7 \mathbin{\&} \mathtt{0xF}^8) \ll 32) \mid (\zeta_6 \gg 32)) \parallel (\zeta_6 \ll 32) \parallel \mathtt{0}^{64}$

$s_4 = \zeta_7 \parallel \mathtt{0}^{64} \parallel \zeta_5 \mathbin{\&} \mathtt{0x0}^8\mathtt{F}^8 \parallel \zeta_4$

$s_5 = (((\zeta_4 \mathbin{\&} \mathtt{0xF}^8) \ll 32) \mid (\zeta_6 \gg 32)) \parallel \zeta_7 \parallel (\zeta_6 \mathbin{\&} \mathtt{0xF}^8\mathtt{0}^8 \mid \zeta_5 \gg 32) \parallel ((\zeta_5 \ll 32) \mid (\zeta_4 \gg 32))$

$s_6 = ((\zeta_5 \ll 32) \mid (\zeta_4 \mathbin{\&} \mathtt{0xF}^8)) \parallel \mathtt{0}^{64} \parallel (\zeta_6 \gg 32) \parallel ((\zeta_6 \ll 32) \mid (\zeta_5 \gg 32))$

$s_7 = ((\zeta_5 \mathbin{\&} \mathtt{0xF}^8\mathtt{0}^8) \mid (\zeta_4 \gg 32)) \parallel \mathtt{0}^{64} \parallel \zeta_7 \parallel \zeta_6$

$s_8 = (\zeta_6 \ll 32) \parallel ((\zeta_5 \ll 32) \mid (\zeta_4 \gg 32)) \parallel ((\zeta_4 \ll 32) \mid (\zeta_7 \gg 32)) \parallel ((\zeta_7 \ll 32) \mid (\zeta_6 \gg 32))$

$s_9 = (\zeta_6 \mathbin{\&} \mathtt{0xF}^8\mathtt{0}^8) \parallel \zeta_5 \parallel (\zeta_4 \mathbin{\&} \mathtt{0xF}^8\mathtt{0}^8) \parallel \zeta_7$

return $(s_1 + 2s_2 + 2s_3 + s_4 + s_5 - s_6 - s_7 - s_8 - s_9 \bmod p_{256})$

1.8 Montgomery Reduction

1.8.1 Modular Multiplication without Trivial Division

1. **Residue Classes and Modular Arithmetic:**

   A residue class modulo $N > 1$ is a set, N-residue,

   $$[a]_N := \mathbb{Z}/N\mathbb{Z} = \{a + kN : k \in \mathbb{Z}\} .$$

   The complete set of distinct residue classes modulo N forms a complete residue system:

   $$\left\{[0]_N, [1]_N, [2]_N, \ldots, [N-1]_N\right\} = \{0, 1, 2, \ldots, N-1\} .$$

2. **Choice of Radix R:**

   The radix R is chosen such that

   - $\gcd(R, N) = 1$, and
   - $R = 2^k > N$ for some integer k due to binary computational efficiency.

3. **Bézout's Identity and Inverses:**

   Bézout's identity states that for any non-zero integers a and b, there exist integers x and y such that:

   $$ax + by = \gcd(a, b).$$

   Since $\gcd(R, N) = 1$, $\exists R^{-1}, N'$ s.t.

   $$RR^{-1} - NN' = 1.$$

   Here,

   $$\begin{aligned} RR^{-1} &\equiv 1 \pmod{N}, \\ N(-N') &\equiv 1 \pmod{R}. \end{aligned}$$

4. **Computation of Modular Inverses and Fast Computation:**

   Consider an integer

   $$T \in [0, RN) .$$

   We are interested in computing

   $$TR^{-1} \bmod N.$$

   The objective in computational terms is to map values from the standard residue system to a system that enables faster computation. For any integer $i \in [0, N)$ we use i to represent the residue class

   $$[iR^{-1}]_N = \left\{iR^{-1} + kN : k \in \mathbb{Z}\right\} .$$

Define an N-residue to be a residue class modulo N. Select a radix R coprime to N (possibly the machine word size or a power thereof) such that $R > N$ and such that computations modulo R are inexpensive to process. Let R^{-1} and N' be integers satisfying $0 < R^{-1} < N$ and $0 < N' < R$ and $RR^{-1} - NN' = 1$.

For $0 \leq i < N$, let i represent the residue class containing iR^{-1} mod N. This is a complete residue system. The rationale behind this selection is our ability to quickly compute TR^{-1} mod N from T if $0 \leq T < RN$.

Algorithm 9: Montgomery Reduction: $\mathsf{MontRed}(T)$ with $T \in [0, RN)$

/* $2^{wt^{-1}}$ satisfies $2^{wt}2^{wt^{-1}} \equiv 1 \pmod N$ */
/* N^{-1} satisfies $NN^{-1} \equiv 1 \pmod{2^{wt}}$ */
/* N' satisfies $N' = 2^{wt} - N^{-1}$ */
Input: $(2^{wt}) \bmod N$ with the pre-computed values 2^{wt}, N'
Output: $\mathsf{MontRed}(x) = x2^{wt^{-1}} \bmod N$

Function *MontRed(T)*:
  $s \leftarrow (T \bmod R) \cdot N' \bmod R$ // $s \in [0, R)$
  $t \leftarrow (T + s \cdot N)/R$ // $t \in [0, 2N)$
  if $t \geq N$ **then**
    $t \leftarrow t - N$
  end
  return t
end

Correctness for MontRed. We observe that

$$
\begin{aligned}
s &= (T \bmod R) \cdot N' \bmod R, \\
s &\equiv (T \bmod R) \cdot N' \pmod R \\
&\implies \\
sN &\equiv (T \bmod R) \cdot N'N \pmod R, \\
&\equiv (T \bmod R)(-N^{-1})N \pmod R, \\
&\equiv -T \pmod R \\
&\implies \\
R &\mid sN - (-T) = sN + T \\
&\implies \\
&\frac{T + sN}{R} \in \mathbb{Z}.
\end{aligned}
$$

Also,

$$
\begin{aligned}
tR &\equiv T \pmod N, \\
t &\equiv TR^{-1} \pmod N
\end{aligned}
$$

and

$$
\begin{aligned}
&0 \leq T + sN < RN + RN, \\
&0 \leq \frac{T + sN}{R} < \frac{2RN}{R}, \\
&0 \leq t < 2N.
\end{aligned}
$$

□

1.8.2 Implementation

Algorithm 10: Montgomery Reduction on P-256

/* $2^{256}R^{-1} \equiv 1 \pmod{p_{256}}$ */
/* $N(-N') \equiv 1 \pmod{2^{256}}$ */

Input: Modulus $N = p_{256} = n_t \parallel n_{t-1} \parallel \cdots \parallel n_0$ with $\gcd(N, 2^{wt=256}) = 1$, $R = 2^{wt=256}$, $N' = 1$ and $T = \tau_{2t-1} \parallel \tau_{2t-2} \parallel \tau_0 < p_{256} \cdot 2^{256} \in \{0, 1\}^{512}$
Output: $TR^{-1} \bmod p_{256}$

Function MontRed(T):
  $s \leftarrow (T \bmod R) \cdot N' \bmod R$ // $s \in [0, R)$
  $t \leftarrow (T + s \cdot N)/R$ // $t \in [0, 2N)$
  if $t \geq N$ **then**
    $t \leftarrow t - N$
  end
  return t
end

1.8.3 Algebraic Relationships between Montgomery and Standard Domains

- **Standard (Integer) Domain:** In this domain, elements (integers) are represented in their standard form, i.e., any integer a within the range $0 \leq a < p$, where p is a prime defining the modulus of the arithmetic field.

- **Montgomery Domain:** In the Montgomery domain, elements are transformed into a scaled representation. An integer a in the standard domain is mapped to $\widetilde{a} = aR \mod p$ in the Montgomery domain, where R is a power of 2, typically chosen as the smallest power of 2 greater than or equal to p, such that $R > p$.

Algebraic Relationships

1. **Transformation to Montgomery Domain:** The transformation from the standard domain to the Montgomery domain involves scaling the integer by a factor of R modulo p. If a is an integer in the standard domain, its Montgomery form $\widetilde{a}$ is computed as $\widetilde{a} = aR \mod p$.

2. **Montgomery Multiplication:** In the Montgomery domain, the multiplication of two integers $\widetilde{a}$ and $\widetilde{b}$ is defined as $\widetilde{c} = \widetilde{a} \cdot \widetilde{b} \cdot R^{-1} \mod p$, where R^{-1} is the modular inverse of R modulo p.

3. **Conversion Back to Standard Domain:** To convert an element $\widetilde{a}$ from the Montgomery domain back to the standard domain, we compute $a = \widetilde{a} \cdot R^{-1} \mod p$.

$$
\mathbb{Z}_N \underset{f^{-1}(u)=uR^{-1} \bmod N}{\overset{f(x)=xR \bmod N}{\rightleftarrows}} \mathbb{Z}_N
$$

$$
\begin{aligned}
a &\longmapsto \tilde{a} = aR \bmod N \\
b &\longmapsto \tilde{b} = bR \bmod N \\
ab \bmod N &\longmapsto (\tilde{a}\tilde{b}R^{-1}) \bmod N
\end{aligned}
$$

In this diagram: - f represents the transformation function from the standard domain to the Montgomery domain.
- f^{-1} represents the inverse transformation from the Montgomery domain back to the standard domain.
- a, b are elements of the standard domain $\mathbb{F}_p$.
- $\widetilde{a}, \widetilde{b}$ are their respective images in the Montgomery domain.

Mathematical Properties

- **Identity and Inversion:** The Montgomery representation of 1 (the multiplicative identity in the standard domain) is $R \mod p$. Conversely, the standard representation of the Montgomery multiplicative identity is computed by $R^{-1} \mod p$.

- **Consistency:** Operations performed in the Montgomery domain are consistent with those in the standard domain when converted back.
- **Efficiency:** The main advantage of Montgomery reduction is that it allows for modular multiplication without direct modular division by p.

Diagrammatic Representation

The algebraic relationship between the standard domain and the Montgomery domain can be visualized as follows:

$$a \in \text{Standard Domain} \underset{\text{Reversion}}{\overset{\text{Transformation}}{\times}} \bar{a} = aR \mod p \in \text{Montgomery Domain}$$

1.8.4 Montgomery Reduction

Montgomery Reduction transforms the problem of modular multiplication into a more efficiently computable form by redefining the multiplication operation in terms of alternative representations of the numbers involved. Specifically, it introduces a mapping function based on a chosen constant R, which is co-prime to the modulus m and typically a power of 2 for computational efficiency.

Example 1.10. Compute

$$5 \cdot 7 \bmod 17$$

Solution. Let $N = 17$ and $R = 2^5 = 32 > 17$. Then

- $R^{-1} = 8$ since $R^{-1}R = 256 \equiv 1 \pmod{17}$;
- $N^{-1} = 17$ since $N^{-1}N = 289 \equiv 1 \pmod{32}$;
- $N' = R - N^{-1} = 15$.

Step 1: Transformation to Montgomery Space

We transform $a = 5$ and $b = 7$ into Montgomery space:

$$\begin{aligned} \tilde{a} &= 5 \cdot R \bmod 17 \\ &= 5 \cdot 32 \bmod 17 = 7, \\ \tilde{b} &= 7 \cdot R \bmod 17 \\ &= 7 \cdot 32 \bmod 17 = 3. \end{aligned}$$

Here, $\tilde{a}$ and $\tilde{b}$ are the Montgomery representations of a and b, respectively.

Step 2: Montgomery Multiplication

1. *Compute the product:*

$$\tilde{a} \times \tilde{b} = 21.$$

Note that R^{-1} satisfies $RR^{-1} \equiv 1 \pmod{11}$:

$$\begin{aligned} R^{-1}R = R^{-1} \cdot 2^4 &\equiv 1 \pmod{11} \\ R^{-1}R = R^{-1} \cdot 5 &\equiv 1 \pmod{11} \quad \because 16 \equiv 5 \pmod{11} \\ R^{-1} &\equiv 9 \pmod{11} \quad \because 5 \cdot 9 \equiv 1 \pmod{11} \end{aligned}$$

We multiply X' and Y' but the result is in Montgomery form: for $Z' = X'Y'$,

$$Z'R^{-1} \bmod 11 = 8 \cdot 9 \bmod 11 = 6.$$

Step 3: Conversion Back from Montgomery Space

□

The Montgomery Representation and the inverse Montgomery Transformation

Definition 1.1. The Montgomery representation of a finite field element $x \in [0, N)$,

$$\begin{array}{rcrcl} M & : & \mathbb{Z}_N & \longrightarrow & \mathbb{Z}_N \\ & & x & \longmapsto & (xR) \bmod N \end{array},$$

is a mapping from the standard representation to the Montgomery domain, where $R = 2^k$ for some k, a constant such that $R \geq m \gcd(R, m) = 1$.
The inverse Montgomery transformation,

$$\begin{array}{rcrcl} M^{-1} & : & \mathbb{Z}_N & \longrightarrow & \mathbb{Z}_N \\ & & u & \longmapsto & (uR^{-1}) \bmod N \end{array},$$

is a mapping back form the Montgomery domain to the standard representation, where R^{-1} is the modular inverse of R modulo m, i.e., $RR^{-1} \equiv 1 \pmod{m}$.

The Montgomery Reduction

Definition 1.2. Define a mapping **Montgomery Reduction** MontRed : $\mathbb{Z}_{NR} \rightarrow \mathbb{Z}_N$ as follows:

$$\mathsf{MontRed}(x) := xR^{-1} \bmod N,$$

for $x \in [0, NR)$, where

(i) $R = 2^{wt} > N$

(ii) $\gcd(R = 2^{wt}, N) = 1$.

- Let $\mathcal{M} : \mathbb{F}_p \longrightarrow \widetilde{\mathbb{F}}_p$ be the Montgomery transformation function, where $\mathcal{M}(a) = aR \mod p$ transforms an element from the standard domain $\mathbb{F}_p$ to the Montgomery domain $\widetilde{\mathbb{F}}_p$.
- Let $\mathcal{M}^{-1} : \widetilde{\mathbb{F}}_p \longrightarrow \mathbb{F}_p$ be the inverse Montgomery transformation function, where $\mathcal{M}^{-1}(\widetilde{a}) = \widetilde{a}R^{-1} \mod p$ transforms an element back from the Montgomery domain $\widetilde{\mathbb{F}}_p$ to the standard domain $\mathbb{F}_p$.

1.9 Montgomery Reduction

Montgomery From

Definition 1.3. Consider a element of finite field $\mathbb{F}_N$:

$$x \in [0, N)\,.$$

Montgomery representation is defined as:

$$[x] := (xR) \bmod N\,,$$

where

(i) $R = 2^{wt} > N$

(ii) $\gcd(R = 2^{wt}, N) = 1$.

Montgomery Reduction

Definition 1.4. Define a mapping **Montgomery Reduction** $\mathsf{MontRed} : \mathbb{Z}_{NR} \rightarrow \mathbb{Z}_N$ as follows:

$$\mathsf{MontRed}(u) := uR^{-1} \bmod N\,,$$

for $u \in \left\{[u] : u \in \mathbb{F}_N\right\}$.

Remark 1.1.

- $\mathsf{MontRed}(xR^2 \bmod N) = [x]$
- $\mathsf{MontRed}([x]) = x$

Algorithm 11: Montgomery Reduction: $\mathsf{MontRed}(x) = x2^{wt^{-1}} \bmod N$

/* $2^{wt^{-1}}$ satisfies $2^{wt}2^{wt^{-1}} \equiv 1 \pmod N$ */
/* N^{-1} satisfies $NN^{-1} \equiv 1 \pmod{2^{wt}}$ */
/* N' satisfies $N' = 2^{wt} - N^{-1}$ */
Input: $(x2^{wt}) \bmod N$ with the pre-computed values $2^{wt}, N'$
Output: $\mathsf{MontRed}(x) = x2^{wt^{-1}} \bmod N$

$s \leftarrow (x \bmod 2^{wt}) \cdot N' \bmod 2^{wt}$ // $s \leftarrow (x \wedge 1^{wt}) \cdot N^{-1} \bmod 2^{wt}$
$t \leftarrow (x + s \cdot N)/2^{wt}$ // $t \leftarrow (x + sN) \gg wt$ and then $t \in [0, 2N)$
if $t \geq N$ **then**
  $t \leftarrow t - N$
end
return t

Correctness for Montogmery Reduction.

$$
\begin{aligned}
s &= (x \bmod 2^{wt}) \cdot N' \bmod 2^{wt}, \\
s &\equiv (x \bmod 2^{wt}) \cdot N' \pmod{2^{wt}} \\
&\implies \\
sN &\equiv (x \bmod 2^{wt}) \cdot N'N \pmod{2^{wt}}, \\
&\equiv (x \bmod 2^{wt})(-N^{-1})N \pmod{2^{wt}}, \\
&\equiv -x \pmod{2^{wt}} \\
&\implies \\
&2^{wt} \mid sN - (-x) = sN + x \\
&\implies \\
&\frac{x + sN}{2^{wt}} \in \mathbb{Z}.
\end{aligned}
$$

□

Chapter 2

Elliptic Curve Theory

2.1 A Puzzle of Squares and Pyramids

Consider the following question:

> "What is the number of balls that may be piled as a square pyramid and also re-arranged into a square array?"

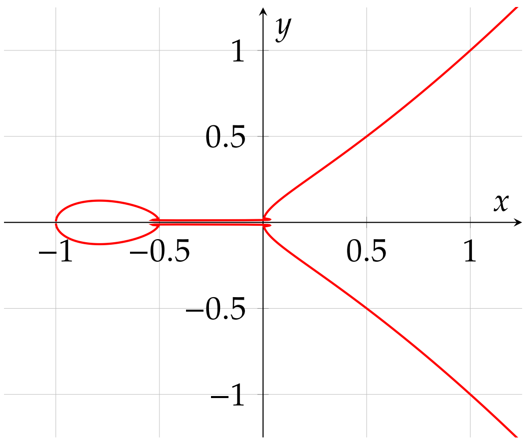
To address this, let x be the height of the pyramid. The number of balls in a pyramid of height x is given by:

$$1^2 + 2^2 + 3^2 + \ldots + x^2 = \frac{x(x+1)(2x+1)}{6}$$

We seek a configuration where this sum also forms a perfect square, i.e.,

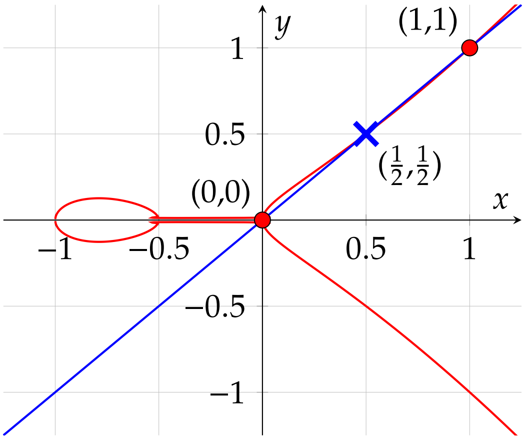
$$y^2 = \frac{x(x+1)(2x+1)}{6}$$

This equation forms the basis of our puzzle, intertwining the concepts of geometric and numeric squares.



2.1.1 Diophantus' Approach

We consider a set of known points to produce new points. The trivial solutions $(0,0)$ and $(1,1)$ fit the equation of the line $y = x$.



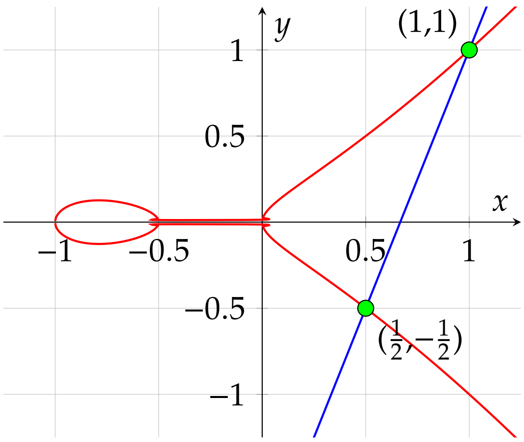
Intersecting this line with the curve described by our pyramid problem, we rearrange terms:

$$\begin{aligned}\frac{x(x+1)(2x+1)}{6} &= x^2,\\ (x^2+x)(2x+1) &= 6x^2,\\ 2x^3+x^2+2x^2+x &= 6x^2,\\ x(2x^2-3x+1) &= 0,\\ x(x-1)(2x-1) &= 0.\end{aligned}$$

We find that $x = \frac{1}{2}$ is a solution, implying $y = \frac{1}{2}$. The symmetry of the curve also yields $\left(\frac{1}{2}, -\frac{1}{2}\right)$ as another solution.

2.1.2 Extending Diophantus' Method

Consider the line through $\left(\frac{1}{2}, -\frac{1}{2}\right)$ and $(1,1)$, which implies $y = 3x - 2$.



Intersecting this with our curve, we derive:

$$x^3 - \frac{51}{2}x^2 + \ldots = 0 \tag{2.1}$$

This leads to the solutions $x = 24$ and $y = 70$, demonstrating the power of algebraic manipulation and geometric insight.

2.2 Why is it called an Elliptic Curve?

The term "elliptic curve" has its roots in the quest to measure the circumference of an ellipse. Consider the trigonometric function $y = \sin w$. The inverse function, $w(y) = \sin^{-1} y$, is expressed as an integral:

$$w(y) = \sin^{-1} y = \int_0^y \frac{1}{\sqrt{1-t^2}}\, dt$$

This integral is foundational in understanding the link between elliptic curves and elliptic integrals.

2.2.1 Abel's Insight

Niels Henrik Abel, a prominent mathematician, extended this concept. Starting with $y = \sin w$, Abel explored the inverse functions of elliptic integrals, uncovering their double periodicity. He defined the function:

$$F(w) = \int_0^w \frac{dz}{\sqrt{(1-z^2)(1-k^2z^2)}}$$

Abel's work laid the groundwork for understanding the complex nature of elliptic curves.

2.2.2 The Geometry of an Ellipse

An ellipse is defined by the equation $x^2/a^2 + y^2/b^2 = 1$. This simple equation belies the complexity of calculating its arc length.

2.2.3 The Arc Length of an Ellipse

Defining $k^2 = 1 - \frac{b^2}{a^2}$ and changing variables $x \to ax$, we express the arc length of an ellipse as:

$$a \int_{-1}^{1} \sqrt{\frac{1-k^2x^2}{1-x^2}}\, dx = a \int_{-1}^{1} \frac{1-k^2x^2}{\sqrt{(1-x^2)(1-k^2x^2)}}\, dx$$

This leads to the following representation of the arc length:

$$\textbf{Arc Length} = a \int_{-1}^{1} \frac{1-k^2x^2}{y}\, dx \quad \text{with} \quad y^2 = (1-x^2)(1-k^2x).$$

2.2.4 Connecting to an Elliptic Curve

The ellipse's arc length calculation brings us to a critical realization. An elliptic integral is generally expressed as:

$$\int R(x, y)\, dx$$

This integral, deeply connected to the geometry of ellipses, underpins the theory of elliptic curves.

Double Periodicity in Elliptic Curves

Elliptic curves are intimately connected with the study of complex tori, which can be represented through the use of doubly periodic functions. A fundamental example of such a function is the Weierstrass $\wp$ function, defined by a lattice Λ in the complex plane.

The Weierstrass $\wp$ Function

Given a lattice $\Lambda \subset \mathbb{C}$, the Weierstrass $\wp$ function is defined as:

$$\wp(z; \Lambda) = \frac{1}{z^2} + \sum_{\omega \in \Lambda \setminus \{0\}} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right). \tag{2.2}$$

This function is even, $\wp(-z) = \wp(z)$, and exhibits double periodicity with respect to the lattice Λ, meaning:

$$\wp(z+\omega) = \wp(z) \quad \text{for all } \omega \in \Lambda. \tag{2.3}$$

Elliptic Curves and the $\wp$ Function

An elliptic curve can be associated with the Weierstrass $\wp$ function. Specifically, an elliptic curve over $\mathbb{C}$ can be described in the Weierstrass form:

$$y^2 = 4x^3 - g_2 x - g_3, \tag{2.4}$$

where g_2 and g_3 are constants derived from the lattice Λ. The coordinates (x, y) on the elliptic curve correspond to the values of the Weierstrass $\wp$ function and its derivative:

$$x = \wp(z; \Lambda), \quad y = \wp'(z; \Lambda). \tag{2.5}$$

Double Periodicity

Two linearly independent periods.

$$\phi(z + w_1) = \phi(z + w_2) = \phi(z) \text{ for all complex number } z.$$

It satisfies

$$[\phi'(z)]^2 = 4\phi(z)^3 - 60G_4\phi(z) - 140G_6$$

- So for $x = \phi(z)$ and $y = \phi'(z)$
- $y^2 = 4x^3 - 60G_3x - 140G_6$

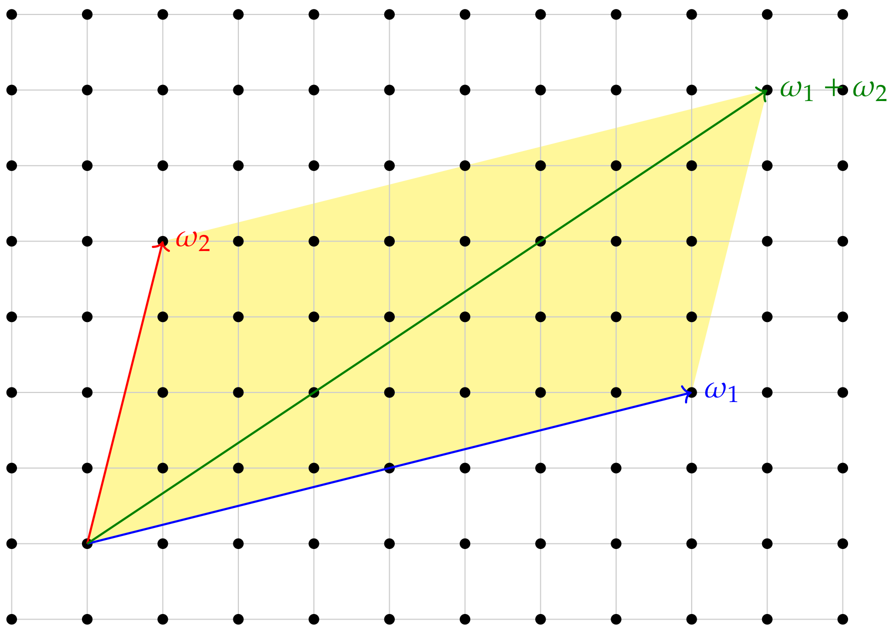
Elliptic Functions and Elliptic Curves

The $\wp$-function and its derivative satisfy an algebraic relation

$$\wp'(z)^2 = \wp(z)^3 + A\wp(z) + B$$

The double periodicity means that it is a function on the quotient space $\mathbb{C}/\Lambda$, where Λ is the lattice

$$\Lambda = \{n_1\omega_1 + n_2\omega_2 : n_1, n_2 \in \mathbb{Z}\}.$$



The lattice L is generated by ω_1 and ω_2
in the quotient space $\mathbb{C}/L$.

Elliptic Functions and Elliptic Curves

Elliptic functions and elliptic curves are fundamental objects in complex analysis and algebraic geometry, respectively. They are interconnected through the Weierstrass $\wp$ function and its properties.

Weierstrass Elliptic Functions

The Weierstrass elliptic functions are defined with respect to a lattice $\Lambda \subset \mathbb{C}$. The Weierstrass $\wp$ function, a key example of an elliptic function, is defined as:

$$\wp(z;\Lambda) = \frac{1}{z^2} + \sum_{\omega\in\Lambda\setminus\{0\}} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2}\right). \tag{2.6}$$

This function is doubly periodic and meromorphic with poles of order two at lattice points.

Elliptic Curves and the Weierstrass $\wp$ Function

An elliptic curve can be described as a set of points satisfying a cubic equation in two variables. Over the complex numbers, this curve can be associated with the Weierstrass $\wp$ function.

An elliptic curve in Weierstrass form is given by:

$$y^2 = 4x^3 - g_2 x - g_3, \tag{2.7}$$

where g_2 and g_3 are constants determined by the lattice Λ. The function $\wp$ and its derivative relate to the curve as follows:

$$x = \wp(z; \Lambda), \tag{2.8}$$
$$y = \wp'(z; \Lambda). \tag{2.9}$$

This establishes a correspondence between points on the complex torus $\mathbb{C}/\Lambda$ and points on the elliptic curve.

Properties of Elliptic Curves

Elliptic curves have several important properties:

- They form a group under a geometrically defined addition operation.
- The addition operation on the curve corresponds to the addition of points in the complex plane modulo the lattice Λ.
- Elliptic curves over finite fields have applications in number theory and cryptography.

The Complex Points on an Elliptic Curve

The ϕ-function gives a complex analytic isomorphism

$$\frac{\mathbb{C}}{L} = \left(\phi(z), \phi'(z)\right) \rightarrow E(\mathbb{C})$$

with the notation that $\mathbb{C}$ is the complex numbers, L is a lattice, and $E(\mathbb{C})$ is the set of complex points on an elliptic curve.

Thus the points of E with coordinates in the complex numbers $\mathbb{C}$ form a *torus*, that is, the surface of a donut.

$X^2 + Y^2 = C$

- Let $x = a + b\sqrt{-1}$, $y = c + d\sqrt{-1}$.
- The solution over complex numbers is a surface, in fact topologically sphere.
- If unbelievable, check out level curves.
- Furthermore, it has group structure.

$$(a + b\sqrt{-1})(c + d\sqrt{-1}) \text{ becomes } ac - bd + (ad + bc)\sqrt{-1}$$

Why is it called Torus?

- Complex Tori

$$y^2 = x(x^2 - 1)$$

- If we introduce *points at infinity* and the *complex numbers*, we can argue that the graph is a torus.

Why Elliptic Curve?

- Discrete Logarithm Problem
- Given a finite group G with two of its elements a and b.
- Find an integer x such that, $a^x = b$ if it exists.
- Example: Non-zero elements of some finite field.

Better groups?

For a finite field F,

$$GL_2(F) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mid ad - bc \neq 0,\ a, b, c, d \in F \right\}$$

- The Times(London) Jan. 1999 An Irish schoolgirl Sarah Flannery used matrices as an alternative to RSA. Her algorithm is far faster than the RSA and equally secure.

- The Art of Computer Programming by Donald Knuth

How about this group?

- $F = \mathbb{Z}/17\mathbb{Z} = \mathbb{Z}$ (mod 17)
- $6^2 = 36 \equiv 2$ mod 17
- 6 behaves like $\sqrt{2}$

$X^2 - 2Y^2 = 1$
$(3 + 2\sqrt{2})(3 - 2\sqrt{2}) = 1$
$(3 + 12)(3 - 12) = -36 \equiv 1$ mod 17

- Let $G = \{(x, y) \mid x^2 - 2y^2 = 1$ over $\mathbb{F}\}$ The operation on G is defined as:

$$(x_1, y_1) \cdot (x_2, y_2) =$$

$$\left(x_1 + \sqrt{2}y_1\right)\left(x_2 + \sqrt{2}y_2\right) =$$

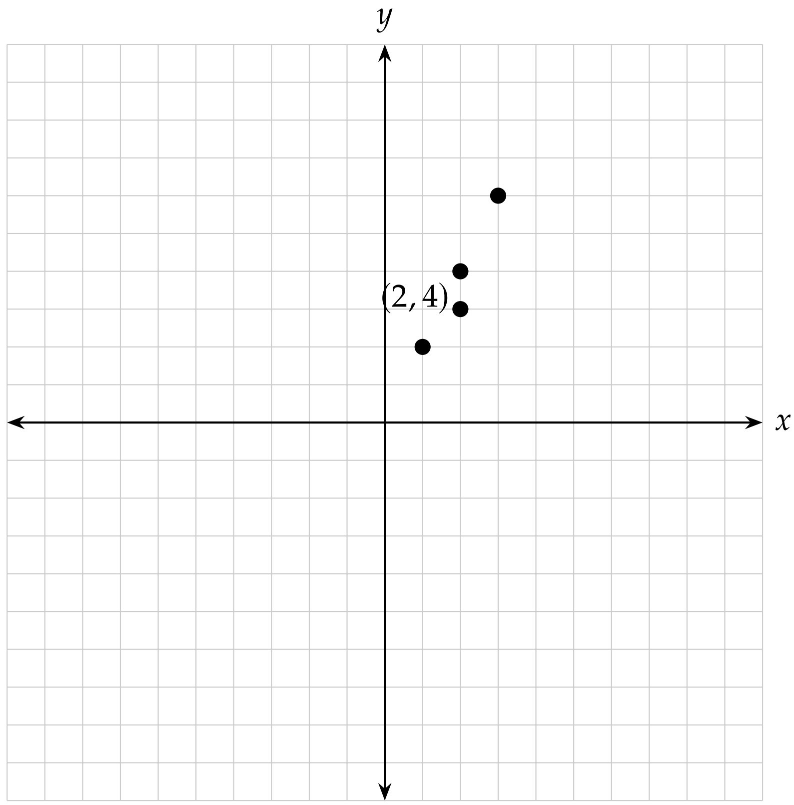
$$= \left(x_1x_2 + 2y_1y_2\right) + \sqrt{2}\left(x_1y_2 + x_2y_1\right)$$

$$(x_1, y_1) \cdot (x_2, y_2) =$$

$$= \left(x_1x_2 + 2y_1y_2\right) x_1y_2 + x_2y_1$$

Why Elliptic Curve?

- DLP (Discrete Logarithm Problem) on finite field can be solved faster than we thought!
- by "index calculus"
- To protect against this attack...
- Elliptic curves!


y
x
(2, 4)

Chapter 3

Elliptic Curves in Cryptography

- Elliptic Curve (EC) cryptography were first proposed in 1985 independently by Neal Koblitz and Victor Miller.
- The **discrete logarithm** problem on elliptic curve groups is believed to be more difficult than the corresponding problem in the multiplicative group of non-zero elements of the underlying finite field.

On finite fields

Consider $y^2 \equiv x^3 + 2x + 3 \pmod 5$

$$
\begin{aligned}
& x = 0,\ y^2 = 3 && \text{no solution} \pmod 5 \\
& x = 1,\ y^2 = 6 \equiv 1, && y = 1, 4 \pmod 5 \\
& x = 2,\ y^2 = 15 \equiv 0, && y = 0 \pmod 5 \\
& x = 3,\ y^2 = 36 \equiv 1, && y = 1, 4 \pmod 5 \\
& x = 4,\ y^2 = 75 \equiv 0, && y = 0 \pmod 5
\end{aligned}
$$

Then points on the elliptic curve are $(1,1)$, $(1,4)$, $(2,0)$, $(3,1)$, $(3,4)$, $(4,0)$ and the point at infinity. Denote it by O.

Notation

- $GF(q)$ or $\mathbb{F}_q$: finite field with q elements, typically, $q = p$ where p is prime, or 2^m.
- $E(\mathbb{F}_q)$: elliptic curve over $\mathbb{F}_q$.
- (x, y): point on $E(\mathbb{F}_q)$.
- O: point at infinity.

Definition of Elliptic curves

- An elliptic curve over a field K is a non-singular cubic curve in two variables, $f(x, y) = 0$ with a rational point (which may be a point at infinity).
- The field K is usually taken to be the complex numbers, reals, rationals, algebraic extensions of rationals, p-adic numbers, or a *finite field*.
- Elliptic curves groups for cryptography are examined with the underlying fields of $\mathbb{F}_p$ (where $p > 3$ is a prime) and $\mathbb{F}_{2^m}$ (a binary representation with 2^m elements).

EC

An *elliptic curve* is a plane curve defined by an equation of the form, when characteristic is neither 2 nor 3, and ... What the hell?

$$y^2 = x^3 + ax + b$$

Hmm...

- $x^3 + y^3 + 1 = 0$ is a cubic curve...?
- Let $x = u + v$, $y = u - v$.
- Then $(u + v)^3 + (u - v)^3 + 1 = 0$.
- This simplifies to $2u^3 + 6uv^2 + 1 = 0$.
- Which leads to $6(v/u)^2 = -(1/u)^3 - 2$.
- So $X = -6/u$, $Y = 36v/u$.
- Hence $Y^2 = X^3 - 432$.

Weierstrass Equation

A two-variable equation $F(x, y) = 0$, forms a curve in the plane.

The generalized Weierstrass Equation of elliptic curves:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Quadratic Equation

- $x^2 + ax + b = 0$
- $x = t - \frac{a}{2}$
- $t^2 - \frac{a^2}{4} - 4b = 0$

Cubic Equation

- $x^3 + ax^2 + bx + c = 0$
- $x = t - \frac{a}{3}$
- $t^3 + pt + q = 0$
- $p = b - \frac{a^2}{3}$
- $q = c + \frac{2a^3}{27} - \frac{ab}{3}$

Field Characteristics

- If characteristic field is not 2:

$$\left(v + \frac{a_1 x}{2} + \frac{a_3}{2}\right)^2 = x^3 + \left(\frac{a_1^2}{4} + a_2\right) x^2 + a_4 x + \left(\frac{a_1 a_3}{4} + a_6\right)$$

$$\Rightarrow y_1^2 = x^3 + a_2' x^2 + a_4' x + a_6'$$

- If characteristics of field is neither 2 nor 3:

$$x_1 = x + \frac{a_2}{3}$$

$$\Rightarrow y_1^2 = x_1^3 + \Delta x + B$$

Discriminant

- Discriminant of $x^2 + bx + c$ is $b^2 - 4c$
- $b^2 - 4c$ is non-zero $\Leftrightarrow$ no double roots
- Discriminant of $x^3 + ax + b$ is $-4a^3 - 27b^2$
- $-4a^3 - 27b^2$ is non-zero $\Leftrightarrow$ no double roots

j-invariant

- Define j of this elliptic curve E as $j(E)/1728 = 4a^3/(4a^3 + 27b^2)$
- If we change $x = m^2 x, y = m^3 y$, get $\tilde{E}$:
- then $j(E) = j(\tilde{E})$
- j-value fixes E

$$y^2 = x^3 + ax + b$$

j-invariant

- If we change $x = m^2 x, y = m^3 y$, get $\tilde{E}$:
- then $j(E) = j(\tilde{E})$
- Why not something like $x = mx + ny^2 + s$?
- It has to keep the point at infinity and keep the form $y^2 = x^3 + ax + b$

Points on the Elliptic Curve (EC)

- Elliptic Curve over field L
- $E(L) = \{\infty\} \cup \{(x, y) \in L \times L \mid y^2 + \ldots = x^3 + \ldots\}$
- It is useful to add the point at infinity.

Group Law

- A group law may be defined where the sum of two points is the reflection across the x-axis of the third point on the same line
- Chords and tangents

The Abelian Group

Given two points P, Q on E, there is a third point, denoted by $P + Q$ on $\bar{E}$, and the following relations hold for all P, Q, R in E.

- $P + Q = Q + P$ (commutativity)
- $(P + Q) + R = P + (Q + R)$ (associativity)
- $P + O = O + P = P$ (existence of an identity element)
- there exists $(-P)$ such that $(-P) + P = O$ (existence of inverses)

Associativity

- $(P + Q) + R = P + (Q + R)$
- Associativity is non-trivial.
- It gives Pascal's theorem and Pappus's theorem.

Elliptic Curve Picture

- Consider elliptic curve $E : y^2 = x^3 - x + 1$
- If P_1 and P_2 are on E, we can define $P_3 = P_1 + P_2$ as shown in the picture.

Doubling of a point

- Let $P = Q$
- $2y_1 \frac{dy}{dx} = 3x_1^2 + a$
- $m = \frac{dy}{dx} = \frac{3x_1^2+a}{2y_1}$
- If $y_1 \neq 0$ (since then $P_1 + P_2 = \infty$):
  - $0 = x^3 - m^2x^2 + \ldots$
  - $x_3 = m^2 - 2x_1, y_3 = m(x_1 - x_3) - y_1$
- What happens when $P_2 = \infty = O$?

Sum of two points

Define for two points $P(x_1, y_1)$ and $Q(x_2, y_2)$ in the Elliptic curve:

$$\lambda = \begin{cases} \frac{y_2-y_1}{x_2-x_1} & \text{for } x_1 \neq x_2 \\ \frac{3x_1^2+a}{2y_1} & \text{for } x_1 = x_2 \end{cases}$$

Then $P + Q$ is given by $R(x_3, y_3)$:

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_3 - x_1) + y_1$$

What is -P?

- $y^2 = x^3 + ax + b$
- $P = (x_1, y_1)$
- What is -P? Is -P = $(x_1, -y_1)$?
- Yes. But this works only for $y^2 = x^3 + ax + b$.
- For $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$
- -P = $(x_1, -a_1x_1 - a_3 - y_1)$

Motivation

- over $\mathbb{F}_3$
- $Y^2Z + 2XYZ + YZ^2 = X^3 - XZ^2 + 7Z^3$ has a solution (1,2,1).
- Note that (0,1,0) is a solution.
- Important Point 1: We do not say (0,0,0) is a solution of the Weierstrass equation.

Homogeneous vs Affine

- Important Point 2: We treat $(1, 2, 1) \sim (2, 1, 2)$, i.e., consider them to be identical and call it a point of the curve given by the Weierstrass equation.

- $\frac{5^2}{13^2} + \frac{12^2}{13^2} = \frac{13^2}{13^2}$

- $\frac{10^2}{26^2} + \frac{24^2}{26^2} = \frac{26^2}{26^2}$

- $X^2 + Y^2 = Z^2$ implies $\left(\frac{X}{Z}\right)^2 + \left(\frac{Y}{Z}\right)^2 = 1$

Projective Co-ordinates

- Two-dimensional projective space P_K^2 over K is given by the equivalence classes of triples (x, y, z) with x, y, z in K and at least one of x, y, z non-zero.

- Two triples (x_1, y_1, z_1) and (x_2, y_2, z_2) are said to be equivalent if there exists a non-zero element λ in K, such that:
$$(x_1, y_1, z_1) = (\lambda x_2, \lambda y_2, \lambda z_2)$$

- The equivalence class depends only on the ratios and hence is denoted by $(x : y : z)$.

Singularity

- For an elliptic curve $y^2 = f(x)$, define $F(x, y) = y^2 - F(x)$. A singularity of the EC is a point (x_0, y_0) such that:
  - $\frac{\partial F}{\partial x}(x_0, y_0) = \frac{\partial F}{\partial y}(x_0, y_0) = 0$
  - or, $2y_0 = -f'(x_0) = 0$
  - or, $f(x_0) = f'(x_0)$
  - Therefore, f has a double root.

Singularity

- $y^2 = x^2(x - 1)$ double roots $x = 0$

- Let $x - 1 = s^2$

- $y^2 = (s^2 + 1)^2(s^2)$

- Hence $x = s^2 + 1, y = s(s^2 + 1)$

If singular, then

- K = a field
- $K(x, y) = K(t)$
- For $y^2 = x^2(x-1)$, $x = s^2 + 1$, $y = s(s^2 + 1)$
- For $y^2 = x^3$, $y = t^3$, $x = t^2$
- For an elliptic curve, $K(x, y)$ is never $K(t)$.

Projective Form

- $E : Y^2Z + a_1XYZ + a_3Y^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$
- has a point $(0, 1, 0)$, point at infinity, denoted by $\mathcal{O}$.

Elliptic Curves in Characteristic 2

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

- If a_1 is not 0, this reduces to the form:

$$y^2 + xy = x^3 + Ax^2 + B$$

- If a_1 is 0, the reduced form is:

$$y^2 + a_3y = x^3 + Bx + C$$

- Note that the form cannot be:

$$y^2 = x^3 + Ax + B$$

EC over Finite Fields

- An elliptic curve may be defined over any finite field GF(q).
- For $GF(2^m)$, the curve has a different form:

$$y^2 + xy = x^3 + ax^2 + b$$

- where b is not 0.
- Addition formulae are similar to those over the reals.

Terminology

- Order of point P is the smallest integer r such that $[r]P = \mathcal{O}$.
- Order of the curve is the number of points of $E(\mathbb{F})$, denoted by $\#E(\mathbb{F})$.

Group Properties

- Let $\#E(\mathbb{F}_q)$ denote the number of points on an elliptic curve $E(\mathbb{F}_q)$, including $\mathcal{O}$.
- Hasse bound: $\#E(\mathbb{F}_q) = q + 1 - t$, where $|t| < 2\sqrt{q}$.
- The group of points is either cyclic or a product of two cyclic groups.

So it's an Abelian Group...

- Group homomorphism? Isogeny, isogenous.
- Endomorphism, isomorphic.
- Examples of endomorphisms are:
  - $[2] : E \longrightarrow E, P \mapsto [2]P$
  - $[n] : E \longrightarrow E, P \mapsto [n]P$

Non-trivial Isogeny

- $E : y^2 = x^3 - x$
- $[i = \sqrt{-1}] : (x, y) \mapsto (-x, iy)$
- $[i = \sqrt{-1}]^2 = [i][i] = (-1) : (x, y) \mapsto (x, -y), P \mapsto -P$
- here $i^2 = -1$
- $6^2 = -1 \mod 37$
- Called complex multiplication.

Frobenius Map

- GF(q), $q = p^k$
- $F : \mathrm{GF}(q) \longrightarrow \mathrm{GF}(q)$
- $F(x) = x^p$ for any x
- F is an isomorphism of GF(q). So F defines an isogeny for any elliptic curve over GF(q).

E[n]

- For any group G, any natural number n, $G[n] = \{g | g^n = 1\}$.
- $E[n] = \{P | [n]P = \mathcal{O}\}$.

Bibliography

[1] Peter L. Montgomery, "Modular Multiplication Without Trial Division," *Math. Computation*, vol. 44, pp. 519–521, 1985.