Scalar Multiplication on Weierstraß Elliptic Curves From Co-Z Arithmetic

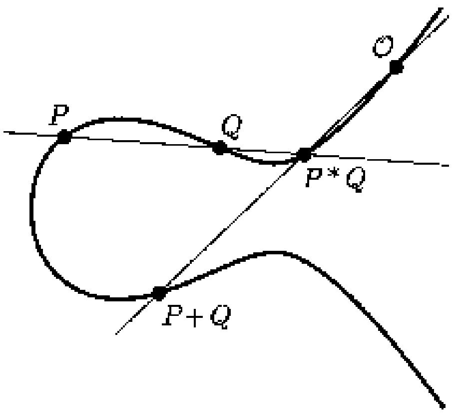
Arithmétique co-Z sur courbes elliptiques

Marc Joye

technicolor

Elliptic Curve Cryptography

- Invented [independently] by Neil Koblitz and Victor Miller in 1985



- Useful for key exchange, encryption and digital signature

Scalar Multiplication

Definition

Given scalar k and a point $\boldsymbol{P}$, compute $[k]\boldsymbol{P} = \underbrace{\boldsymbol{P} + \boldsymbol{P} + \cdots + \boldsymbol{P}}_{k \text{ times}}$

ECDLP Given $\boldsymbol{P}$ and $\boldsymbol{Q} = [k]\boldsymbol{P}$, recover k

- no subexponential algorithms are known to solve the ECDLP (in the *general* case)
- smaller key sizes can be used

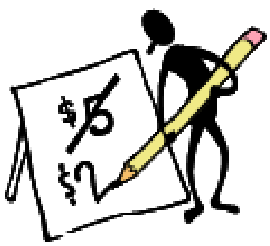
	Bit security				
	80	112	**128**	192	256
ECC	160	224	256	384	512
RSA	1024	2048	3072	8192	15360

This Talk

Goal

Implementation of the Montgomery ladder and of other [regular] binary ladders using efficient co-Z formulæ

- binary scalar multiplication algorithms
- suitable for memory-constrained devices



Outline

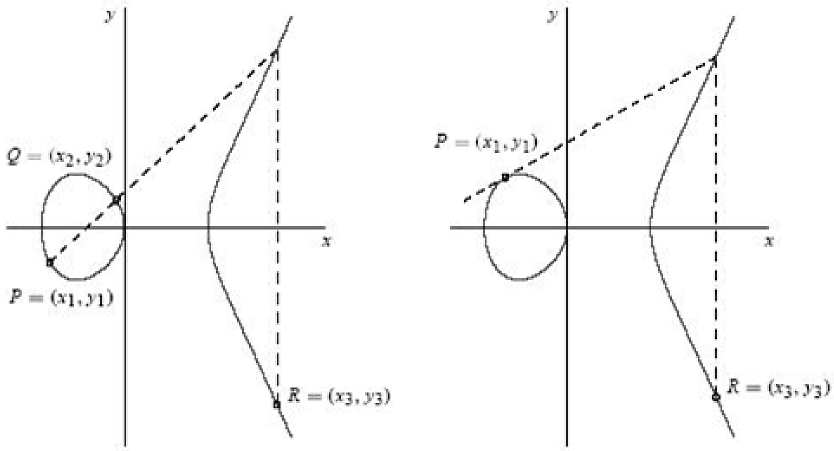
1. **Arithmetic on Elliptic Curves**
   - Jacobian coordinates
   - Co-*Z* point addition
2. **Binary Scalar Multiplication Algorithms**
3. **New Implementations**
   - Conjugate point addition
   - Binary ladders with co-*Z* trick
   - Enhanced algorithms
4. **Discussion**
   - Performance analysis
   - Security analysis
5. **Conclusion**

technicolor

Elliptic Curves

Weierstraß equation (affine coordinates)

Let $E : y^2 = x^3 + ax + b$ define over $\mathbb{F}_q$ ($char \neq 2, 3$) with discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$



(a) Addition: $P + Q = R$. (b) Doubling: $P + P = R$.

technicolor

Group Law

$$E(\mathbb{F}_q) = \{y^2 = x^3 + ax + b\} \cup \{\boldsymbol{O}\}$$

- Let $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2)$
- Group law
  - $\boldsymbol{P} + \boldsymbol{O} = \boldsymbol{O} + \boldsymbol{P} = \boldsymbol{P}$
  - $-\boldsymbol{P} = (x_1, -y_1)$
  - $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ where

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

$$\text{with } \lambda = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{[addition]} \\ \dfrac{3x_1^2 + a}{2y_1} & \text{[doubling]} \end{cases}$$

Jacobian Coordinates

- To avoid computing inverses in $\mathbb{F}_q$
  - affine point $(x, y) \rightarrow$ projective point $(X : Y : Z)$ such that $x = X/Z^2$ and $y = Y/Z^3$

Weierstraß equation (projective Jacobian coordinates)

Let $E : Y^2 = X^3 + aXZ^4 + bZ^6$ define over $\mathbb{F}_q$ (*char* $\neq 2, 3$) with discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$

- Point at infinity $\boldsymbol{O} = (1 : 1 : 0)$
- If $\boldsymbol{P} = (X_1 : Y_1 : Z_1) \in E$ then $-\boldsymbol{P} = (X_1 : -Y_1 : Z_1)$

Jacobian Point Doubling (1/2)

- Let $\boldsymbol{P} = (X_1 : Y_1 : Z_1) \in E \setminus \{\boldsymbol{O}\}$ with $\boldsymbol{P} \neq -\boldsymbol{P}$

Reminder: if $y_1 \neq 0$ then

$$2(x_1, y_1) = (x_3, y_3) = \left(\lambda^2 - 2x_1, (x_1 - x_3)\lambda - y_1\right)$$

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

- As $\boldsymbol{P} = \left(\frac{X_1}{Z_1^2} : \frac{Y_1}{Z_1^3} : 1\right)$ and $2\boldsymbol{P} = \left(\frac{X_3}{Z_3^2} : \frac{Y_3}{Z_3^3} : 1\right)$, we get
  - $\frac{X_3}{Z_3^2} = \left(\frac{3\left(\frac{X_1}{Z_1^2}\right)^2 + a}{2\frac{Y_1}{Z_1^3}}\right)^2 - 2\frac{X_1}{Z_1^2} = \frac{(3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2}{4Y_1^2Z_1^2}$
  - $\frac{Y_3}{Z_3^3} = \left(\frac{X_1}{Z_1^2} - \frac{X_3}{Z_3^2}\right)\frac{3\left(\frac{X_1}{Z_1^2}\right)^2 + a}{2\frac{Y_1}{Z_1^3}} - \frac{Y_1}{Z_1^3} = \cdots$

    $= \frac{(4X_1Y_1^2 - X_3)(3X_1^2 + aZ_1^4) - 8Y_1^4}{8Y_1^3Z_1^3}$

technicolor

Jacobian Point Doubling (2/2)

- Point doubling algorithm

Input $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ with $\boldsymbol{P} \neq -\boldsymbol{P}$
Output $2\boldsymbol{P} = (X_3 : Y_3 : Z_3)$

1. compute
$$\mathcal{X} \leftarrow X_1^2;\ \mathcal{Y} \leftarrow Y_1^2;\ \mathcal{Z} \leftarrow Z_1^2;$$
$$M \leftarrow 3\mathcal{X} + a\mathcal{Z}^2;\ T \leftarrow \mathcal{Y}^2;\ S \leftarrow 4X_1\mathcal{Y}$$
2. $X_3 \leftarrow M^2 - 2S$
3. $Y_3 \leftarrow M(S - X_3) - 8T$
4. $Z_3 \leftarrow 2Y_1Z_1$
5. return $(X_3 : Y_3 : Z_3)$

- Cost: 3M + 6S + 1c
  - or 1M + 8S + 1c by evaluating S and Z_3 as

$$\begin{cases} S \leftarrow 2[(X_1 + \mathcal{Y})^2 - \mathcal{X} - T] \\ Z_3 \leftarrow (Y_1 + Z_1)^2 - \mathcal{Y} - \mathcal{Z} \end{cases}$$

technicolor

Jacobian Point Addition (1/2)

- Let $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2) \in E \setminus \{\boldsymbol{O}\}$ with $\boldsymbol{P} \neq \pm \boldsymbol{Q}$

Reminder: if $x_1 \neq x_2$ then
$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3) = (\lambda^2 - x_1 - x_2, (x_1 - x_3)\lambda - y_1)$$
where $\lambda = \frac{y_1 - y_2}{x_1 - x_2}$

- As $\boldsymbol{P} + \boldsymbol{Q} = \left(\frac{X_3}{Z_3{}^2} : \frac{Y_3}{Z_3{}^3} : 1\right)$, we get
  - $$\frac{X_3}{Z_3{}^2} = \cdots = \frac{(Y_1 Z_2{}^3 - Y_2 Z_1{}^3)^2 - (X_1 Z_2{}^2 + X_2 Z_1{}^2)(X_1 Z_2{}^2 - X_2 Z_1{}^2)^2}{[Z_1 Z_2 (X_1 Z_2{}^2 - X_2 Z_1{}^2)]^2}$$
  - $$\frac{Y_3}{Z_3{}^3} = \cdots = \frac{(Y_1 Z_2{}^3 - Y_2 Z_1{}^3)[X_1 Z_2{}^2 (X_1 Z_2{}^2 - X_2 Z_1{}^2)^2 - X_3] - Y_1 Z_2{}^3 (X_1 Z_2{}^2 - X_2 Z_1{}^2)^3}{[Z_1 Z_2 (X_1 Z_2{}^2 - X_2 Z_1{}^2)]^3}$$

technicolor

Jacobian Point Addition (2/2)

- Point addition algorithm

Input $\boldsymbol{P} = (X_1 : Y_1 : Z_1)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z_2)$ with $\boldsymbol{P} \neq \pm \boldsymbol{Q}$ and $\boldsymbol{P}, \boldsymbol{Q} \neq \boldsymbol{O}$

Output $\boldsymbol{P} + \boldsymbol{Q} = (X_3 : Y_3 : Z_3)$

1. compute $\mathcal{Z}_1 \leftarrow Z_1{}^2;\ \mathcal{Z}_2 \leftarrow Z_2{}^2;$
   $U_1 \leftarrow X_1 \mathcal{Z}_2;\ U_2 \leftarrow X_2 \mathcal{Z}_1;\ H \leftarrow U_1 - U_2;$
   $S_1 \leftarrow Y_1 \mathcal{Z}_2 Z_2;\ S_2 \leftarrow Y_2 \mathcal{Z}_1 Z_1;\ R \leftarrow S_1 - S_2;$
   $\mathcal{H} \leftarrow H^2;\ G \leftarrow \mathcal{H} H;\ V \leftarrow U_1 \mathcal{H}$
2. $X_3 \leftarrow R^2 + G - 2V$
3. $Y_3 \leftarrow R(V - X_3) - S_1 G$
4. $Z_3 \leftarrow Z_1 Z_2 H$
5. return $(X_3 : Y_3 : Z_3)$

- Cost: $12M + 4S$
  - or $11M + 5S$ by evaluating $2Z_1Z_2 = (Z_1 + Z_2)^2 - \mathcal{Z}_1 - \mathcal{Z}_2$ and "rescaling" X_3 and Y_3 accordingly

technicolor

Co-Z Point Addition (ZADD)

- Introduced by Meloni [WAIFI 2007]
- Addition of two distinct points with the same Z-coordinate
  - scalar multiplication algorithms are confined to
    - Euclidean addition chains
    - Zeckendorf's representation

Co-Z point addition

Let $\boldsymbol{P} = (X_1 : Y_1 : Z)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z)$. Then $\boldsymbol{P} + \boldsymbol{Q} = (X_3 : Y_3 : Z_3)$ where

$$X_3 = D - W_1 - W_2, \quad Y_3 = (Y_1 - Y_2)(W_1 - X_3) - A_1, \quad Z_3 = Z(X_1 - X_2)$$

with $A_1 = Y_1(W_1 - W_2)$, $W_1 = X_1C$, $W_2 = X_2C$, $C = (X_1 - X_2)^2$ and $D = (Y_1 - Y_2)^2$

- Cost of ZADD: $5\mathrm{M} + 2\mathrm{S}$

Co-Z Point Addition with Update (ZADDU)

- Main advantage of Meloni's addition

Equivalent representation of $\boldsymbol{P}$

Evaluation of $\boldsymbol{R} = \mathrm{ZADD}(\boldsymbol{P}, \boldsymbol{Q})$ yields for free

$$\boldsymbol{P}' = (X_1(X_1 - X_2)^2 : Y_1(X_1 - X_2)^3 : Z_3) = (W_1 : A_1 : Z_3) \sim \boldsymbol{P}$$

that is, $Z(\boldsymbol{P}') = Z(\boldsymbol{R})$

- Notation: $(\boldsymbol{R}, \boldsymbol{P}') = \mathrm{ZADDU}(\boldsymbol{P}, \boldsymbol{Q})$
- Cost of ZADDU: $5\mathrm{M} + 2\mathrm{S}$

Left-to-Right Methods

Algorithm 1 Left-to-right binary method

Input: $P \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$
Output: $Q = kP$

1: $R_0 \leftarrow O$; $R_1 \leftarrow P$
2: **for** $i = n - 1$ down to 0 **do**
3: $\quad R_0 \leftarrow 2R_0$
4: $\quad$ **if** $(k_i = 1)$ **then** $R_0 \leftarrow R_0 + R_1$
5: **end for**
6: **return** R_0

Algorithm 2 Montgomery ladder

Input: $P \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$
Output: $Q = kP$

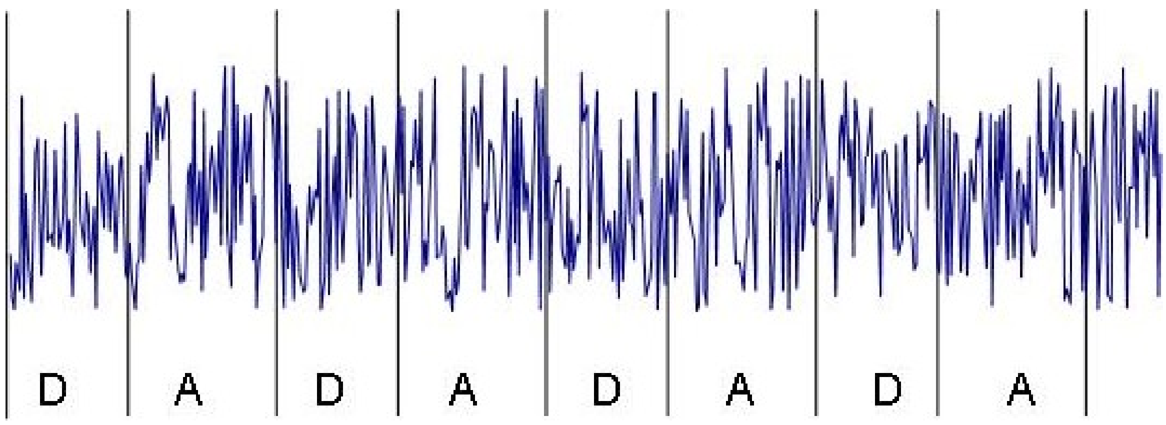
1: $R_0 \leftarrow O$; $R_1 \leftarrow P$
2: **for** $i = n - 1$ down to 0 **do**
3: $\quad b \leftarrow k_i$; $R_{1-b} \leftarrow R_{1-b} + R_b$
4: $\quad R_b \leftarrow 2R_b$
5: **end for**
6: **return** R_0

- Subject to SPA-type attacks
- Inserting dummy addition prevents SPA
  - subject to safe-error attacks

- Regular structure, no dummy operations
- Naturally resistant against SPA and safe-error attacks
- 2 registers

technicolor

Power Trace



$\Longrightarrow$ $d = \ldots$

technicolor

Right-to-Left Methods

Algorithm 3 Right-to-left binary method

Input: $P \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$
Output: $Q = kP$

$R_0 \leftarrow O$; $R_1 \leftarrow P$
for $i = 0$ to $n-1$ **do**
  if $(k_i = 1)$ **then** $R_0 \leftarrow R_0 + R_1$
  $R_1 \leftarrow 2R_1$
end for
return R_0

- Idem left-to-right method

  (SPA-type attacks, safe-error attacks)

Algorithm 4 Joye's double-add

Input: $P \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$
Output: $Q = kP$

$R_0 \leftarrow O$; $R_1 \leftarrow P$
for $i = 0$ to $n-1$ **do**
  $b \leftarrow k_i$
  $R_{1-b} \leftarrow 2R_{1-b} + R_b$
end for
return R_0

- Idem Montgomery ladder

  (regular structure, no dummy operations, naturally resistant against SPA and safe-error attacks, 2 registers)

Signed-Digit Methods

- Any group of w bits $00\ldots01 \equiv 1\bar{1}\bar{1}\ldots\bar{1}$ (where $\bar{1} = -1$)
  - ZSD expansion of an [odd] integer k, $k = \sum_{i=0}^{n-1} \kappa_i 2^i$, with

$$\kappa_{n-1} = 1 \quad \text{and} \quad \kappa_i = (-1)^{1+k_{i+1}} \text{ for } n-2 \geqslant i \geqslant 0$$

Algorithm 5 Left-to-right signed-digit method

Input: $P \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$ with $k_0 = 1$
Output: $Q = kP$

$R_0 \leftarrow P$; $R_1 \leftarrow P$
for $i = n-1$ down to 1 **do**
  $\kappa \leftarrow (-1)^{1+k_i}$
  $R_0 \leftarrow 2R_0 + (\kappa)R_1$
end for
return R_0

Algorithm 6 Right-to-left signed-digit method

Input: $P \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$ with $k_0 = 1$
Output: $Q = kP$

$R_0 \leftarrow O$; $R_1 \leftarrow P$
for $i = 1$ to $n-1$ **do**
  $\kappa \leftarrow (-1)^{1+k_i}$; $R_0 \leftarrow R_0 + (\kappa)R_1$
  $R_1 \leftarrow 2R_1$
end for
$R_0 \leftarrow R_0 + R_1$
return R_0

- Idem Montgomery ladder

Conjugate co-Z Point Addition (ZADDC)

- New co-Z point operation
  - using caching techniques

Conjugate co-Z point addition

From $-\boldsymbol{Q} = (X_2 : -Y_2 : Z_2)$, evaluation of $\boldsymbol{R} = \text{ZADD}(\boldsymbol{P}, \boldsymbol{Q})$ allows one to get $\boldsymbol{S} := \boldsymbol{P} - \boldsymbol{Q} = (\overline{X_3}, \overline{Y_3}, Z_3)$ where

$$\overline{X_3} = (Y_1 + Y_2)^2 - W_1 - W_2, \quad \overline{Y_3} = (Y_1 + Y_2)(W_1 - \overline{X_3})$$

with an additional cost of $1M + 1S$

- Notation: $(\boldsymbol{P} + \boldsymbol{Q}, \boldsymbol{P} - \boldsymbol{Q}) = \text{ZADDC}(\boldsymbol{P}, \boldsymbol{Q})$
- Total cost of ZADDC: $6M + 3S$

technicolor

Left-to-Right Binary Ladder With co-Z Trick

Algorithm 7 Montgomery ladder with co-Z formulæ

Input: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$ with $k_{n-1} = 1$
Output: $\boldsymbol{Q} = k\boldsymbol{P}$

1: $\boldsymbol{R_0} \leftarrow \boldsymbol{O}$; $\boldsymbol{R_1} \leftarrow \boldsymbol{P}$
2: **for** $i = n - 1$ down to 0 **do**
3: $\quad b \leftarrow k_i$; $\boldsymbol{R_{1-b}} \leftarrow \boldsymbol{R_{1-b}} + \boldsymbol{R_b}$
4: $\quad \boldsymbol{R_b} \leftarrow 2\boldsymbol{R_b}$
5: **end for**
6: **return** $\boldsymbol{R_0}$

$(2\boldsymbol{P}, \boldsymbol{P'}) = \text{DBLU}(\boldsymbol{P})$ where $\boldsymbol{P'} \sim \boldsymbol{P}$ and $\text{Z}(\boldsymbol{P'}) = \text{Z}(2\boldsymbol{P})$
$\boldsymbol{T} \leftarrow \boldsymbol{R_b} - \boldsymbol{R_{1-b}}$
$\boldsymbol{R_{1-b}} \leftarrow \boldsymbol{R_b} + \boldsymbol{R_{1-b}}$; $\boldsymbol{R_b} \leftarrow \boldsymbol{R_{1-b}} + \boldsymbol{T}$ $(= 2\boldsymbol{R_b})$

technicolor

Right-to-Left Binary Ladder With co-Z Trick

Algorithm 8 Joye's double-add with co-Z formulæ

Input: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$ with $k_0 = 1$
Output: $\boldsymbol{Q} = k\boldsymbol{P}$

1: $\boldsymbol{R_0} \leftarrow \boldsymbol{P}$; $\boldsymbol{R_1} \leftarrow \boldsymbol{P}$
2: **for** $i = 1$ to $n-1$ **do**
3: $\quad b \leftarrow k_i$; $\boldsymbol{T} \leftarrow \boldsymbol{R_{1-b}} + \boldsymbol{R_b}$
4: $\quad \boldsymbol{R_{1-b}} \leftarrow \boldsymbol{T} + \boldsymbol{R_{1-b}}$
5: **end for**
6: **return** $\boldsymbol{R_0}$

$\boldsymbol{R_0}$ and $\boldsymbol{R_1}$ now have the same Z-coordinate but are not different (!)
$\Longrightarrow$ start for-loop at $i = 2$
$(3\boldsymbol{P}, \boldsymbol{P'}) = \mathrm{TPLU}(\boldsymbol{P})$ where $\boldsymbol{P'} \sim \boldsymbol{P}$ and $\mathrm{Z}(\boldsymbol{P'}) = \mathrm{Z}(3\boldsymbol{P})$

Left-to-Right Signed-Digit Ladder With co-Z Trick

Algorithm 9 Left-to-right signed-digit algorithm with co-Z formulæ

Input: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}_{\geqslant 3}$ with $k_0 = k_{n-1} = 1$
Output: $\boldsymbol{Q} = k\boldsymbol{P}$

1: $(\boldsymbol{R_0}, \boldsymbol{R_1}) \leftarrow \mathrm{TPLU}(\boldsymbol{P})$
2: **for** $i = n-2$ to 1 **do**
3: $\quad \kappa \leftarrow (-1)^{1+k_i}$
4: $\quad (\boldsymbol{R_1}, \boldsymbol{R_0}) \leftarrow \mathrm{ZADDU}(\boldsymbol{R_0}, (\kappa)\boldsymbol{R_1})$
5: $\quad (\boldsymbol{R_0}, \boldsymbol{R_1}) \leftarrow \mathrm{ZADDC}(\boldsymbol{R_1}, \boldsymbol{R_0})$; $\boldsymbol{R_1} \leftarrow (\kappa)\boldsymbol{R_1}$
6: **end for**
7: **return** $\boldsymbol{R_0}$

- Cost per bit: $(5\mathrm{M} + 2\mathrm{S}) + (6\mathrm{M} + 3\mathrm{S}) = 11\mathrm{M} + 5\mathrm{S}$

Right-to-Left Signed-Digit Ladder With co-Z Trick

Algorithm 10 Right-to-left signed-digit algorithm with co-Z formulæ

Input: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}_{\geqslant 3}$ with $k_0 = 1$
Output: $\boldsymbol{Q} = k\boldsymbol{P}$

$\kappa \leftarrow (-1)^{1+k_1}$; $(\boldsymbol{R_1}, \boldsymbol{R_0}) \leftarrow \mathrm{DBLU}(\boldsymbol{P})$; $\boldsymbol{R_0} \leftarrow (\kappa)\boldsymbol{R_0}$
for $i = 2$ to $n-1$ **do**
    $\kappa \leftarrow (-1)^{1+k_i}$
    $(\boldsymbol{R_0}, \boldsymbol{R_1}) \leftarrow \mathrm{ZADDC}((\kappa)\boldsymbol{R_1}, \boldsymbol{R_0})$
    $(\boldsymbol{R_1}, \boldsymbol{R_0}) \leftarrow \mathrm{ZADDU}(\boldsymbol{R_0}, \boldsymbol{R_1})$; $\boldsymbol{R_1} \leftarrow (\kappa)\boldsymbol{R_1}$
end for
$\boldsymbol{R_0} \leftarrow \mathrm{ZADD}(\boldsymbol{R_0}, \boldsymbol{R_1})$
return $\boldsymbol{R_0}$

- Cost per bit: $(6\mathrm{M} + 3\mathrm{S}) + (5\mathrm{M} + 2\mathrm{S}) = 11\mathrm{M} + 5\mathrm{S}$

Combined Double-Add Operation

- Point doubling-addition evaluates: $\boldsymbol{R} \leftarrow 2\boldsymbol{P} + \boldsymbol{Q}$
  - $\boldsymbol{T} \leftarrow \boldsymbol{P} + \boldsymbol{Q}$ followed by $\begin{cases} \boldsymbol{R} \leftarrow \boldsymbol{T} + \boldsymbol{P} \\ \boldsymbol{Q} \leftarrow \boldsymbol{T} - \boldsymbol{P} \end{cases}$
  - $(\boldsymbol{T}, \boldsymbol{P}) \leftarrow \mathrm{ZADDU}(\boldsymbol{P}, \boldsymbol{Q})$; $(\boldsymbol{R}, \boldsymbol{Q}) \leftarrow \mathrm{ZADDC}(\boldsymbol{T}, \boldsymbol{P})$
  - cost: $11\mathrm{M} + 5\mathrm{S}$
- Combined operation

Co-Z point doubling-addition with update

$$(\boldsymbol{R}, \boldsymbol{Q}) \leftarrow \mathrm{ZDAU}(\boldsymbol{P}, \boldsymbol{Q})$$

  - trades 2M against 2S
  - cost: $9\mathrm{M} + 7\mathrm{S}$

Application

Algorithm 11 Joye's double-add with co-Z formulæ

Input: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \dots, k_0)_2 \in \mathbb{N}$ with $k_0 = 1$
Output: $\boldsymbol{Q} = k\boldsymbol{P}$

```
b ← k_1; R_b ← P; (R_{1-b}, R_b) ← TPLU(R_b)
for i = 2 to n − 1 do
    b ← k_i
    (R_{1-b}, R_b) ← ZDAU(R_{1-b}, R_b)
end for
return R_0
```

- Cost per bit: $9\mathrm{M} + 7\mathrm{S}$
- (Similar savings apply to Montgomery ladder and right-to-left signed-digit algorithm)

technicolor

(X, Y)-only Operations

Co-Z point addition

Let $\boldsymbol{P} = (X_1 : Y_1 : Z)$ and $\boldsymbol{Q} = (X_2 : Y_2 : Z)$. Then $\boldsymbol{P} + \boldsymbol{Q} = (X_3 : Y_3 : Z_3)$ where

$$X_3 = D - W_1 - W_2, \quad Y_3 = (Y_1 - Y_2)(W_1 - X_3) - A_1, \quad Z_3 = Z(X_1 - X_2)$$

with $A_1 = Y_1(W_1 - W_2)$, $W_1 = X_1 C$, $W_2 = X_2 C$, $C = (X_1 - X_2)^2$ and $D = (Y_1 - Y_2)^2$

- ZADDU and ZADDC do not involve the Z-coord. of the input points for calculating the X- and Y-outputs
- Computation of $\boldsymbol{Q} = k\boldsymbol{P}$ using X- and Y-coordinates only
  - Z-coordinate of output point $\boldsymbol{Q}$ is recovered at the end of the algorithm
    - possible with the Montgomery ladder and the zero-less signed-digit left-to-right algorithm

technicolor

Performance: Point Addition Formulæ

Operation	Notation	# regs.	Cost
Point addition:			
– co-Z addition with update	ZADDU	6	$5M + 2S$
– (X, Y)-only co-Z add. with update	ZADDU′	5	$4M + 2S$
– conjugate co-Z addition	ZADDC	7	$6M + 3S$
– (X, Y)-only conjugate co-Z addition	ZADDC′	6	$5M + 3S$

- Comparison
  - ZADDU is $\approx$ 56% more efficient w.r.t. general addition ($11M + 5S$)
  - ZADDC is $\approx$ 50% more efficient w.r.t. general conjugate addition ($12M + 6S$)
- At most 7 field registers are required for implementation

technicolor

Performance: Point Doubling-Addition Formulæ

Operation	Notation	# regs.	Cost
Point doubling-addition:			
– co-Z DA with update	ZDAU	8	$9M + 7S$
– (X, Y)-only co-Z DA with update	ZDAU′	6	$8M + 6S$

- Comparison
  - co-Z doubling-addition formula with update (ZDAU):
    - $\approx$ 25% more efficient w.r.t. general doubling-addition ($13M + 8S$)
    - $\approx$ 12% more efficient w.r.t. mixed doubling-addition ($11M + 7S$)
- At most 8 field registers are required for implementation
- (Similar performance for co-Z conjugate-addition-addition with update — ZACAU: ZADDC followed by ZADDU)

technicolor

Performance: Scalar Multiplication

Algorithm	Main op.	# regs.	Total cost
Joye's double-add:			
– basic version	DA	10	$n(13M + 8S) + 1I + 3M + 1S$
– co-Z version	ZDAU	8	$n(9M + 7S) + 1I - 9M - 6S$
Co-Z signed-digit alg.	ZACAU	8	$n(9M + 7S) + 1I - 9M - 6S$
Montgomery ladder:			
– basic version		8	$n(12M + 13S) + 1I + 3M + 1S$
– X-only version		7	$n(9M + 7S) + 1I + 14M + 3S^{\dagger}$
– co-Z version	ZACAU′	6	$n(8M + 6S) + 1I + 1M$
Co-Z signed-digit alg.	ZDAU′	6	$n(8M + 6S) + 1I - 5M - 4S$

† assuming that multiplications by a have negligible cost

- Comparison
  - co-Z versions are always faster
  - co-Z versions require less memory
  - cost is independent of the curve parameters

technicolor

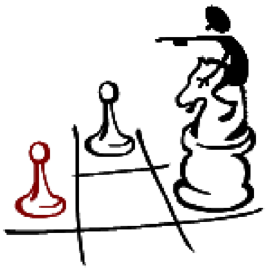
Security Analysis

- Proposed co-Z implementations are built on highly regular scalar multiplication algorithms
  - inherit similar security features
  - naturally resistant against
    - SPA-type attacks
    - safe-error attacks
- Can be combined with existing DPA-type countermeasures
- Output complete point representation
  - possible to check redundant relations
    - e.g., output point belongs to the curve
  - useful feature against (regular) fault attacks

technicolor

Summary

- Efficient co-Z conjugate point addition formula (as well as other companion co-Z formulæ)
- New strategies for evaluating scalar multiplications on elliptic curves using co-Z arithmetic
  - suitable for memory constrained devices
  - nicely combine with certain binary ladders



technicolor

Acknowledgments

- This is a joint work with
  - Raveen R. Goundar
  - Atsuko Miyaji
  - Matthieu Rivain
  - Alexandre Venelli

Full version

J. Cryptographic Engineering 1(2):161-176, 2011

technicolor