Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation

Michael Hutter, Marc Joye, and Yannick Sierra

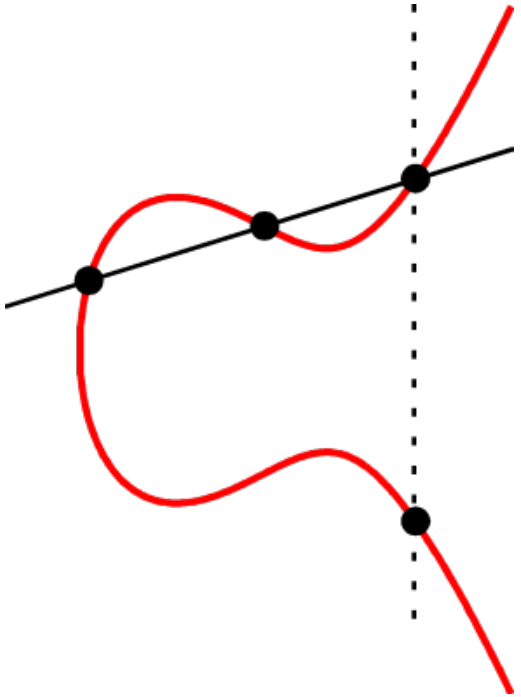
AFRICACRYPT 2011

5-7 July, Dakar, Senegal

Elliptic Curve Cryptography

- N. Koblitz and V. Miller 1985
- Scalar multiplication
  - Given a scalar *k* and a point *P* on *E*, then

    $Q = k[P] = \underbrace{P + P + \ldots + P}_{k \text{ times}}$



- ECDLP
  - Given *Q* and *P*, determine *k* from *P* and Q
- Low-resource primitive
  - Small key sizes
  - ECC 224 vs. RSA 2048

What is this talk about?

- New Formulae for ECC over GF*p*
  - Using efficient Co-*Z* technique
  - Based on Montgomery ladder
  - Memory-wise operations: out-of-place
  - Well suitable for low-resource designs

Outline

- Elliptic Curve Cryptography
  - Co-*Z* arithmetic
  - Related work
- Montgomery Ladder
  - Properties
  - Optimizations
- New Co-*Z* Formulae
  - Out-of-place property
  - Differential addition and doubling
- Discussion
  - Performance and security
  - Practical implementation
- Summary

Meloni's Co-*Z* Point Addition

- Idea: points share Z coordinate
- Let $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$, then $P_1 + P_2 = P_3$ is given by

$$\begin{cases} X_3 = D^2 - X_2C^2 - X_1C^2 \\ Y_3 = D(X_1C^2 - X_3) - Y_1C^3 \\ Z_3 = ZC \end{cases}$$

with $C = (X_2 - X_1)$ and $D = (Y_2 - Y_1)$

- Costs: $5M + 2S$

Meloni's Co-*Z* Point Addition

- Idea: points share *Z* coordinate
- Let $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$, then $P_1 + P_2 = P_3$ is given by

$$\begin{cases} X_3 = D^2 - X_2C^2 - X_1C^2 \\ Y_3 = D(X_1C^2 - X_3) - Y_1C^3 \\ Z_3 = ZC \end{cases}$$

  with $C = (X_2 - X_1)$ and $D = (Y_2 - Y_1)$
- Costs: $\underline{5M + 2S}$
- $P_1 \cong P_1' = (X_1C^2, Y_1C^3, ZC)$

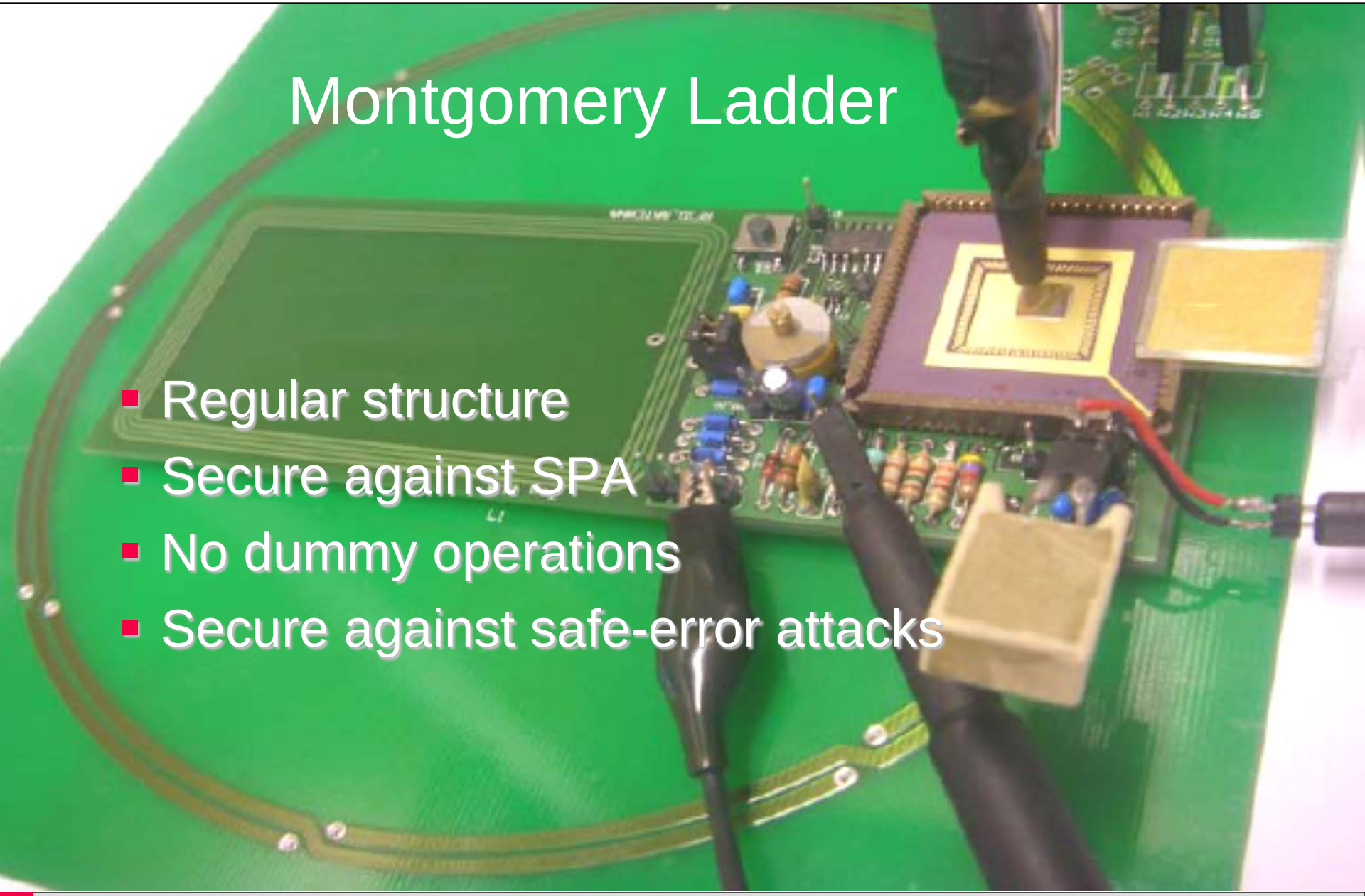
Montgomery Ladder

Input: $\boldsymbol{P} \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$, with $k_{n-1} \neq 0$
Output: $\boldsymbol{Q} = k\boldsymbol{P}$

1: $\boldsymbol{R_0} \leftarrow \boldsymbol{P}$; $\boldsymbol{R_1} \leftarrow 2\boldsymbol{P}$
2: **for** $i = n - 2$ downto 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** $\boldsymbol{R_0}$

- Nice Properties
  - Invariant $P = R_1 - R_0$
  - *X*-coord only = omit the *y* coordinate
  - Memory efficient

Montgomery Ladder



- Regular structure
- Secure against SPA
- No dummy operations
- Secure against safe-error attacks

X-Coord Only Formulae in Co-*Z*

- Consider *homogeneous* projective coordinates
- Let $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$, then

  $x(P_1 + P_2) = (X_3, Z_3)$ is given by

  $$\begin{cases} X_3 = (2(X_1 + X_2)(X_1X_2 + aZ^2) + 4bZ^3 - x_DZU) \\ Z_3 = ZU \end{cases}$$

  → Costs: $5M + 2S$
- and $x(2P_2) = (X_4, Z_4)$ is given by

  $$\begin{cases} X_4 = ((X_2^2 - aZ^2)^2 - 8bZ^3X_2) \\ Z_4 = ZV \end{cases}$$

  → Costs: $4M + 3S$

  where $U = (X_1 - X_2)^2$ and $V = 4X_2(X_2^2 + aZ^2) + 4bZ^3$

Co-*Z* Update

- Co-Z update necessary → 3*M* additional

$$\begin{cases} X_1' = X_3 Z_4 \\ X_2' = X_4 Z_3 \\ Z' = Z_3 Z_4 \end{cases}$$

- Differential addition and doubling

  → Costs: 12*M* + 5*S*

Implicit Co-Z Update

- Let $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$, then
  x$(P_1 + P_2) = (X_3, Z_3)$ is given by
  $$\begin{cases} X_3 = V(2(X_1 + X_2)(X_1X_2 + aZ^2) + 4bZ^3 - x_DZU) \\ Z_3 = ZUV \end{cases}$$
- and x$(2P_2) = (X_4, Z_4)$ is given by
  $$\begin{cases} X_4 = U((X_2^2 - aZ^2)^2 - 8bZ^3X_2) \\ Z_4 = ZUV \end{cases}$$
  where $U = (X_1 - X_2)^2$ and $V = 4X_2(X_2^2 + aZ^2) + 4bZ^3$
- Costs: $10M + 4S$ $(+ 1M_a + 1M_b)$

Trick: Trade *M* against *S*

- Let $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$, then
  x$(P_1 + P_2) = (X_3, Z_3)$ is given by
  $$\begin{cases} X_3 = V(2(X_1 + X_2)(X_1X_2 + aZ^2) + 4bZ^3 - x_DZU) \\ Z_3 = ZUV \end{cases}$$
- and x$(2P_2) = (X_4, Z_4)$ is given by
  $$\begin{cases} X_4 = U((X_2^2 - aZ^2)^2 - 8bZ^3X_2) \\ Z_4 = ZUV \end{cases}$$
  where $U = (X_1 - X_2)^2$ and $V = 4X_2(X_2^2 + aZ^2) + 4bZ^3$
- $X_1X_2 = (X_1^2 + X_2^2 - (X_1 - X_2)^2)/2$

Implicit Co-Z Update

- Let $P_1 = (X_1, Y_1, Z)$ and $P_2 = (X_2, Y_2, Z)$, then $x(P_1 + P_2) = (X_3, Z_3)$ is given by

$$\begin{cases} X_3 = V((X_1 + X_2)(X_1^2 + X_2^2 - U + 2aZ^2) + 4bZ^3 - x_D ZU) \\ Z_3 = ZUV \end{cases}$$

- and $x(2P_2) = (X_4, Z_4)$ is given by

$$\begin{cases} X_4 = U((X_2^2 - aZ^2)^2 - 8bZ^3 X_2) \\ Z_4 = ZUV \end{cases}$$

where $U = (X_1 - X_2)^2$ and $V = 4X_2(X_2^2 + aZ^2) + 4bZ^3$

- Costs: $9M + 5S$ $(+ 1M_a + 1M_b)$

Explicit Formulae

Require: X_1, X_2, Z, x_D, a, $4b$
Ensure: X_1, X_2, Z

1:

1. $R_2 \leftarrow Z^2$	16. $X_1 \leftarrow X_1 - X_2$
2. $R_3 \leftarrow a \times R_2$	17. $X_2 \leftarrow X_2 + X_2$
3. $R_1 \leftarrow Z \times R_2$	18. $R_3 \leftarrow X_2 \times R_2$
4. $R_2 \leftarrow 4b \times R_1$	19. $R_4 \leftarrow R_4 - R_3$
5. $R_1 \leftarrow {X_2}^2$	20. $R_3 \leftarrow {X_1}^2$
6. $R_5 \leftarrow R_1 - R_3$	21. $R_1 \leftarrow R_1 - R_3$
7. $R_4 \leftarrow {R_5}^2$	22. $X_1 \leftarrow X_1 + X_2$
8. $R_1 \leftarrow R_1 + R_3$	23. $X_2 \leftarrow X_1 \times R_1$
9. $R_5 \leftarrow X_2 \times R_1$	24. $X_2 \leftarrow X_2 + R_2$
10. $R_5 \leftarrow R_5 + R_5$	25. $R_2 \leftarrow Z \times R_3$
11. $R_5 \leftarrow R_5 + R_5$	26. $Z \leftarrow x_D \times R_2$
12. $R_5 \leftarrow R_5 + R_2$	27. $X_2 \leftarrow X_2 - Z$
13. $R_1 \leftarrow R_1 + R_3$	28. $X_1 \leftarrow R_5 \times X_2$
14. $R_3 \leftarrow {X_1}^2$	29. $X_2 \leftarrow R_3 \times R_4$
15. $R_1 \leftarrow R_1 + R_3$	30. $Z \leftarrow R_2 \times R_5$

2: **return** (X_1, X_2, Z)

- 8 reg. Algorithm
  - $9M + 5S + 1M_a + 1M_{4b} + 14add$
- 7 reg. Algorithm (trade ~1M against 1 reg.)
  - $11M + 4S + 1M_a + 1M_{4b} + 14add$
- 10 reg. Algorithm
  - $10M + 5S + 13add$
- (X,Y,Z) Recovery
  - $8M + 2S$ (7 reg.)

Explicit Formulae - Performance

Method	Costs	M/bit[a]	M/bit[b]
Algorithm {10reg}	**10M + 5S + 13add**	**14**	**17.9**
Algorithm {8 reg}	**9M + 5S + 1M$_a$ + 1M$_{4b}$ + 14add**	**14**	**18.8**
Izu *et al.*	10M + 4S + 2M$_a$ + 1M$_b$ + 18add	14.2	20.8
Goundar *et al.*	8M + 7S + 3M$_a$ + 1M$_b$ + 18add	14.6	21.8
Algorithm {7 reg}	**11M + 4S + 1M$_a$ + 1M$_{4b}$ + 14add**	**15.2**	**20.0**
Fischer *et al.*	10M + 5S + 2M$_a$ + 2M$_b$ + 14add	16	21.4

[a] M$_b$ = 1M ; S = 0.8M ; 1M$_a$ ≃ 0 ; 1add ≃ 0 (negligible)
[b] M$_b$ = 1M ; S = 0.8M ; 1M$_a$ = 2add ; 1add = 0.3M

- Nice Property:
  - All operations are performed out-of-place

In-Place vs. Out-of-Place Formulae

- Consider multi-precision multiplication

  $R_m \leftarrow R_1 \bullet R_2$ $\qquad R_1, R_2 \in GF(p)$

  $R_1 \leftarrow R_m \pmod p$

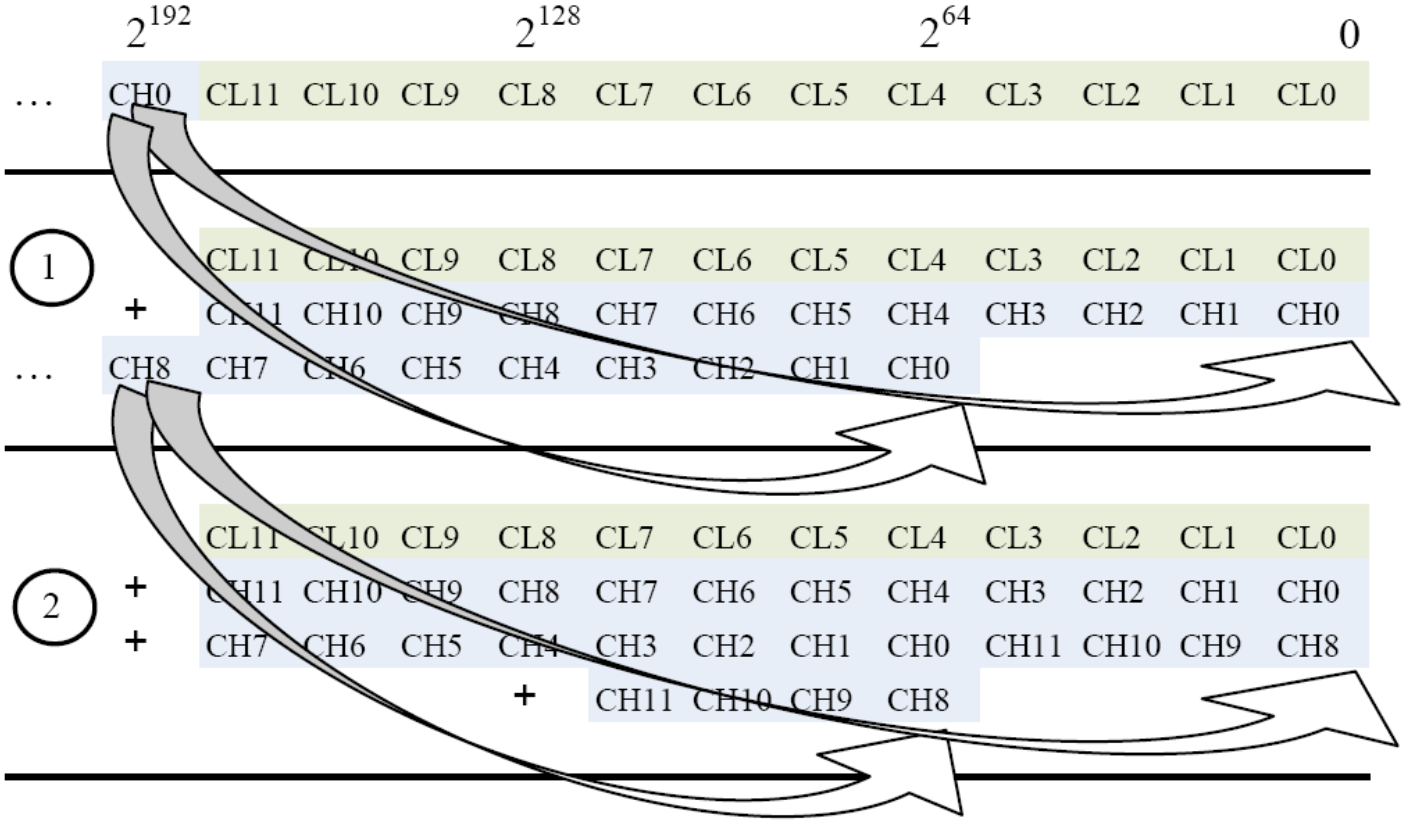
- Memory-wise optimization: interleaved reduction
  - Alternate between multiplication and reduction
  - Common technique to save memory
  - In-place update:

    $R_1 \leftarrow R_1 \bullet R_2 \pmod p$ $\qquad$ + buffer to keep operand R1

- No buffer needed for out-of-place operations
  - $R_3 \leftarrow R_1 \bullet R_2 \pmod p$ 

An Example

- Consider 192-bit multi-precision multiplication with interleaved reduction (fast NIST P-192 reduction)



Memory Requirements

Method	Working registers	In-place memory	Constants	Total
Algorithm {7 reg}	**7 reg.**	-	$\{x_D, a, 4b\}$	**10 reg.**
Izu *et al.*	7 reg.	+1 reg.	$\{x_D, a, b\}$	11 reg.
Goundar *et al.*	7 reg.	+1 reg.	$\{x_D, a, b\}$	11 reg.
Algorithm {8 reg}	**8 reg.**	-	$\{x_D, a, 4b\}$	**11 reg.**
Fischer *et al.*	8 reg.	+1 reg.	$\{x_D, a, 4b\}$	12 reg.
Algorithm {10 reg}	**10 reg.**	-	-	**10 reg.**

Implementation

Input: $P \in E(\mathbb{F}_q)$ and $k = (k_{n-1}, \ldots, k_0)_2 \in \mathbb{N}$, with $k_{n-1} \neq 0$
Output: $Q = kP$

1: $\{X_1, X_2, Z\} \leftarrow \texttt{AddDblCoZ}(\{0, \lambda x_P, \lambda\})$
2: $X_1 \leftarrow x_P \cdot Z$
3: **for** $i = n - 2$ downto 0 **do**
4: $\quad b \leftarrow k_i$;
5: $\quad \{X_{2-b}, X_{1+b}, Z\} \leftarrow \texttt{AddDblCoZ}(\{X_{2-b}, X_{1+b}, Z\})$
6: **end for**
7: $\{X, Y, Z\} \leftarrow \texttt{RecoverFullCoordinatesCoZ}(\{X_1, X_2, Z\})$
8: $Z(Y^2 - bZ^2) \stackrel{?}{=} X(X^2 + aZ^2)$
9: **return** $\{X, Y, Z\}$

Practical Implementation

ECC formulae have been integrated in an RFID tag
- Implements ECDSA
- Authentication through digital signatures

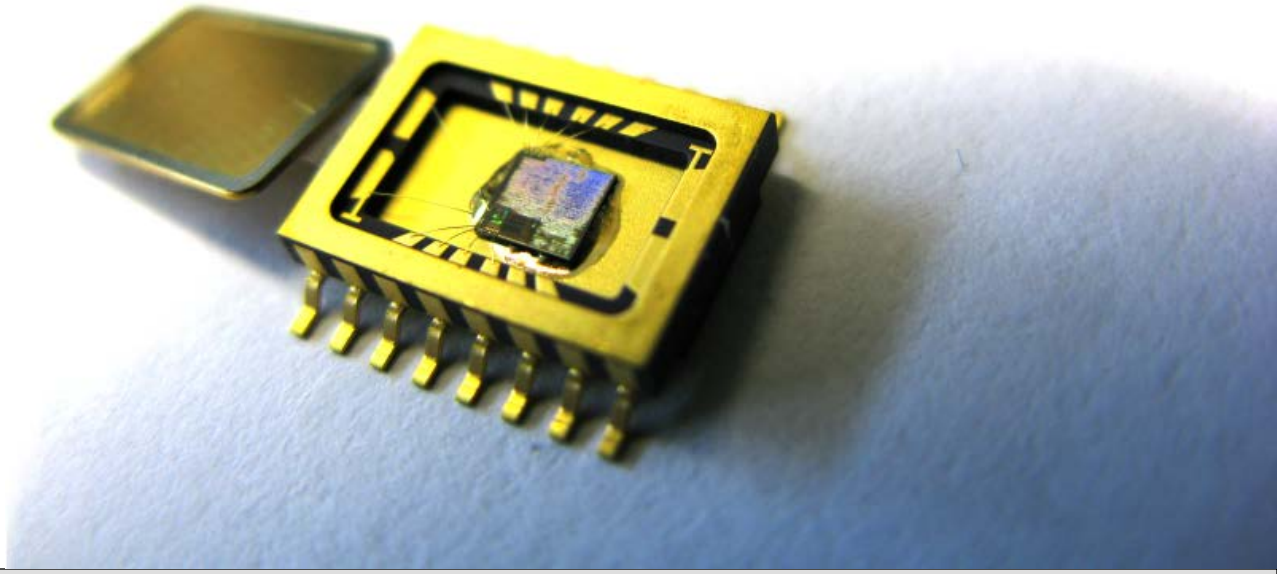
Anti-Counterfeiting
- Step toward preventing illicit copying intellectual property and goods

Touch & Verify
- Using NFC-enabled mobile phones, e.g. Nokia 6212

Summary

- Efficient Co-*Z* formulae
  - Require 7, 8, or 10 registers (out-of-place)
  - Montgomery-ladder based
  - Very fast → 9*M* + 5*S*

- Suitable for low-resource devices



Thanks for your attention!

Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation

Michael Hutter

IAIK – Graz University of Technology
michael.hutter@iaik.tugraz.at
www.iaik.tugraz.at