Hardware Random Recoding
Redundant Representations of Numbers, Side Channel Analysis, Elliptic Curve Cryptography

Thomas Chabrier, Danuta Pamula, Arnaud Tisserand

IRISA Laboratory, CAIRN Research Team
Plan

Context

Redundant Representations

Proposed Solution and Implementation Results

Conclusion and Future Prospects
Context

Elliptic curve cryptography (ECC):

- considered finite field: $\mathbb{F}_p$ with $p$ a large prime (160–600 bits)
- simplified Weierstrass equation:
  
  $$y^2 = x^3 + ax + b$$

  where $a, b \in \mathbb{F}_p^2$ and

  $$\Delta = -16(4a^3 + 27b^2) \neq 0$$

Hardware implementation issues:

- performance: speed, area, low power/energy consumption
- security: protection against side channel attacks

ECC Scalar Multiplication $[k]P$

- scalar multiplication: $[k]P = P + P + \ldots + P$ with $k \in \mathbb{N}$

right to left and left to right binary "double and add" algorithms to compute $[k]P$:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>$Q \leftarrow \infty$</td>
</tr>
<tr>
<td>2:</td>
<td>for $i$ from 0 to $t-1$ do</td>
</tr>
<tr>
<td>3:</td>
<td>if $k_i = 1$ then $Q \leftarrow Q + P$ ADD</td>
</tr>
<tr>
<td>4:</td>
<td>$P \leftarrow 2P$ DBL</td>
</tr>
</tbody>
</table>

avg. cost: $(n-1) \cdot DBL$ and $\frac{n}{2} \cdot ADD$

- non adjacent form (NAF):

$$k = \sum_{i=0}^{l-1} k_i 2^i$$

where $k_i \in \{\overline{1}, 0, 1\}$ \quad $k_i k_{i+1} = 0$

$k = 267 = (1 0 0 0 0 0 1 0 1 1) \_2$

$\overline{2} = \overline{\overline{1} 0 0 0 1 0 \overline{1} 0 \overline{1}} \_2 = \text{NAF}$

$\overline{3} = \overline{1 0 0 0 0 1 0 0 3} \_2 = \text{NAF}$

avg. cost: $(n-1) \cdot DBL$ and $\frac{n}{w+1} \cdot ADD$

Notation: $\overline{d} \leftrightarrow -d$
Side Channel Analysis

- measure some external parameters on running device in order to deduce internal secret informations

Side Channel Analysis for ECC

- in ECC: **identify point additions and point doublings** operations in order to **deduce the key value** in $[k]P$

**Typical countermeasures:**
- **resistant algorithms** (double and add always, Montgomery ladder, insert dummy operations, . . .) $\rightarrow$ regular behavior
- **unified formulae**
- **randomization of the scalar**
  - Coron countermeasure (first): $k' = k + r|E(\mathbb{F}_p)|$
  - random recoding with **DBNS** and signed digit representations
- **randomization of the base point**
- **isomorphism randomization of the curve**
ECC Processor

- functional units (FU): $\pm, \times, 1/x$ for $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$, key recoding
- memory: register file + internal registers in the FUs
- control: operations ($E$ and $\mathbb{F}_q$ levels) schedule
DBNS: Double-Based Number System

\[ k = \sum_{i=0}^{n-1} k_i 2^{a_i} 3^{b_i} \quad \text{with } k_i \in \{-1, 1\}, \ a_i, b_i \geq 0 \]

The double-base chain approach:

- representations of integers in two coprime bases \((2, 3)\)
- extremely redundant and sparse number system

Example: 127 has 783 different representations:
\[ 127 = 2^2 3^3 + 2^1 3^2 + 2^0 3^0 = 2^2 3^3 + 2^4 3^0 + 2^0 3^1 = \ldots \]

Strictly chained DBNS representation (ref. [1]):

- compute \([k]P \iff \text{Need } a_0 \geq \ldots \geq a_{n-1} \text{ and } b_0 \geq \ldots \geq b_{n-1}\)
- cost: \((n - 1) \cdot ADD + a_0 \cdot DBL + b_0 \cdot TPL\)

Random Recoding Rules

We focus on 4 recodings:

1. **Reduction**

   - **1 + 2** \(\xrightarrow{\text{expansion}}\) **3**
   
   \[
   2^{i+1}3^{j-1} + 2^i3^{j-1} = 2^i3^j \quad [R_1]
   
   2^{i-1}3^{j+1} - 2^{i-1}3^j = 2^i3^j \quad [R_2]
   
   2^{i-2}3^{j+1} + 2^{i-2}3^j = 2^i3^j \quad [R_3]
   
   2^{i+2}3^{j-1} - 2^i3^{j-1} = 2^i3^j \quad [R_4]
   
2. **Reduction**

   - **1 + 2^3** \(\xrightarrow{\text{expansion}}\) **3^2**
   
   \[
   2^{i+3}3^{j-2} + 2^i3^{j-2} = 2^i3^j \quad [R_5]
   
   2^{i-3}3^{j+2} - 2^{i-3}3^j = 2^i3^j \quad [R_6]
   
3. **Reduction**

   - **1 + 1** \(\xrightarrow{\text{expansion}}\) **2**
   
   \[
   2^{i+1}3^j - 2^i3^j = 2^i3^j \quad [R_7]
   
   2^{i-1}3^j + 2^{i-1}3^j = 2^i3^j \quad [R_8]
   
Rules have to respect decreasing exponents

Random applications of the rules
Example of Some Possible DBNS Recodings for $k = 140400$

\[
\begin{align*}
[140400]P & = [2^43^3]([2^3]([2^0]P - P) + P) \\
        & = [2^43^3]([2^3]P + P) \\
        & = [2^43^3]([2^3][2^0]P - P) + P
\end{align*}
\]
Binary Signed-Digit Representation

\[ k = \sum_{i=0}^{n} k_i 2^i \quad \text{with} \quad k_i \in \{\bar{1}, 0, 1\} \]

Example of some BSD representations for \( k = 11 \):

\[
\begin{align*}
(01011)_{BSU} &= 2^3 + 2^1 + 2^0 \\
(011\bar{1}1)_{BSU} &= 2^3 + 2^2 - 2^1 + 2^0 \\
& \vdots
\end{align*}
\]

Number of BSD representations: \( \lambda(k, n) \) (ref. [2])

\[
\begin{align*}
\lambda(149, 9) &= 50 \\
\lambda(1365, 12) &= 233 \\
\lambda(87381, 17) &= 4181
\end{align*}
\]

Recoding Rules for Randomization

Recoding rules: $01 \Leftrightarrow 1\bar{1}$ and $0\bar{1} \Leftrightarrow \bar{1}1$

Random recoding approach:

- left–to–right or right–to–left algorithm
- serial scanning of all digits of $k$
- random bits $r = (r_2, r_1, r_0)$

Compute a random signed-digit representation of

$$ k = (0k_{n-1} \cdots k_0)_2: $$

1:  for $i$ from 1 to $n-1$ do
2:      if $r_2 = 1$ then
3:          if $r_1 = 1$ then $(k_{i+1}, k_i) \leftarrow f(k_{i+1}, k_i)$
4:          if $r_0 = 1$ then $(k_i, k_{i-1}) \leftarrow f(k_i, k_{i-1})$
5:      else
6:         if $r_0 = 1$ then $(k_i, k_{i-1}) \leftarrow f(k_i, k_{i-1})$
7:         if $r_1 = 1$ then $(k_{i+1}, k_i) \leftarrow f(k_{i+1}, k_i)$
8:     return $k$
Recoding Example for $k = 11 = (01011)_2$

Problem: this representation may have too many 1s
Solution: reduction of the Hamming weight in order to improve scalar multiplication
Width–$w$ Signed-Digit

$$k = \sum_{i=0}^{n} k_i 2^i$$

with $k_i \in \{0, \pm 1, \pm 3, \ldots, \pm (2^w - 1)\}$

- maximum 1 digit $\neq 0$ in $w$ consecutive digits

Example of width–$w$ signed digit representations for $k = 11$:

- $w = 2$
  - $(01003)_{SD2}$
  - $(0030\bar{1})_{SD2}$

- $w = 3$
  - $(01003)_{SD3}$
  - $(1000\bar{5})_{SD3}$

- precomputations: $[2^i - 1]P$ for $i$ from 2 to $w$

- average cost: $(n - 1) \cdot DBL$ and $\frac{n}{w+1} \cdot ADD$

⇒ less representations: $3 = 011 = 1\bar{1}1 = 10\bar{1}$
Cost Comparison

<table>
<thead>
<tr>
<th>Curve Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ADD_J + A$</td>
<td>$8[m] + 3[s]$</td>
</tr>
<tr>
<td>$\alpha$-DBL $J$</td>
<td>$4\alpha[m] + (4\alpha + 2)[s]$</td>
</tr>
<tr>
<td>$\alpha$-TPL $J$</td>
<td>$(11\alpha - 1)[m] + (4\alpha + 2)[s]$</td>
</tr>
</tbody>
</table>

assumption in $\mathbb{F}_p$: 1 square $\approx 0.8$ multiplication

cost $[k]P$ with:

- SD2: $1500[m] + 1575[s] \approx 2760[m]$
- SD3: $1354[m] + 1524[s] \approx 2573[m]$
- SD4: $1284[m] + 1494[s] \approx 2479[m]$
- DBNS recoding: $1752[m] + 930[s] \approx 2496[m]$
Circuit-Level Representations of Signed-Digits

2 implementation versions:

**SM** (Sign Magnitude) and **OH** (One Hot)

For $w = 2$, the digit set is $\{\bar{3}, \bar{1}, 0, 1, 3\}$, and two circuit-level codings have been used:

\[
\begin{array}{c}
\overline{\text{Benefit}}: \text{constant number of transitions for } 0 \rightarrow 1 \text{ and } 1 \rightarrow 0 \\
\overline{\text{Cost}}: \text{larger area and memory} \\
\overline{\text{Remark}}: \text{same approach for } w = 3
\end{array}
\]
### Implementation Results - SM Version

ISE version 12.4
standard efforts for synthesis and P&R
Virtex 5 XC5VLX50T FPGA

<table>
<thead>
<tr>
<th>n</th>
<th>w</th>
<th>optimization goal</th>
<th># registers</th>
<th># LUTs</th>
<th>max. freq. [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>2</td>
<td>area</td>
<td>451</td>
<td>2497</td>
<td>182</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>1604</td>
<td>2970</td>
<td>222</td>
</tr>
<tr>
<td>192</td>
<td>3</td>
<td>area</td>
<td>457</td>
<td>2704</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>1803</td>
<td>3251</td>
<td>212</td>
</tr>
<tr>
<td>224</td>
<td>2</td>
<td>area</td>
<td>515</td>
<td>2924</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>1860</td>
<td>3081</td>
<td>179</td>
</tr>
<tr>
<td>224</td>
<td>3</td>
<td>area</td>
<td>521</td>
<td>3128</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>2093</td>
<td>3653</td>
<td>195</td>
</tr>
</tbody>
</table>
### Implementation Results - OH Version

ISE version 12.4
standard efforts for synthesis and P&R
Virtex 5 XC5VLX50T FPGA

<table>
<thead>
<tr>
<th>n</th>
<th>w</th>
<th>optimization goal</th>
<th># registers</th>
<th># LUTs</th>
<th>max. freq. [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>192</td>
<td>2</td>
<td>area</td>
<td>838</td>
<td>2976</td>
<td>182</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>2186</td>
<td>3606</td>
<td>195</td>
</tr>
<tr>
<td>192</td>
<td>3</td>
<td>area</td>
<td>847</td>
<td>3215</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>2971</td>
<td>4215</td>
<td>170</td>
</tr>
<tr>
<td>224</td>
<td>2</td>
<td>area</td>
<td>966</td>
<td>3434</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>2538</td>
<td>3874</td>
<td>179</td>
</tr>
<tr>
<td>224</td>
<td>3</td>
<td>area</td>
<td>975</td>
<td>3670</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td></td>
<td>speed</td>
<td>3450</td>
<td>4489</td>
<td>187</td>
</tr>
</tbody>
</table>
Conclusion

- use redundant representations of numbers
- random recoding
- hardware implementation with low overhead

Future prospects:

- integration in the ECC processor
- physical robustness evaluation
References


