Elliptic Curve Scalar Multiplication, Side-Channel Attacks and Counter-measures

Jean-Marc ROBERT

Team DALI/LIRMM, Université de Perpignan, France

28 November 2014

Work funded by ANR PAVOIS 12 BS02 002 02 project
Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis

3. Conclusion
Outline

1 Problematic
   • Cryptographic Protocols: what about the Group?
   • Elliptic Curve Point Operations
   • Elliptic Curve Scalar Multiplication

2 Side-channel Attacks
   • First Attack: Simple Power Analysis
   • How to thwart SPA?
   • Second Attack: Differential Power Analysis
   • How to thwart DPA?
   • Synthesis

3 Conclusion
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group \((G, +, O)\) and a group generator \(P\).

- Alice

\[
\text{Computes } A = a \cdot P
\]

\[
\text{sends } A
\]

- Bob

\[
\text{Computes } B = b \cdot P
\]

\[
\text{sends } B
\]

\[
\text{Computes } K = a \cdot B = b \cdot A
\]

\[
\text{Shared Secret Key } K = a \cdot b \cdot P
\]

The main operation is the scalar multiplication \(a \cdot P\).
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group \((G, +, \mathcal{O})\) and a group generator \(P\).

- Alice
  
  \[
  a \leftarrow \text{random()}
  \]
  
  Computes \(A = a \cdot P\)

- Bob
  
  \[
  b \leftarrow \text{random()}
  \]
  
  Computes \(B = b \cdot P\)

Shared Secret Key \(K = a \cdot b \cdot P\).
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group \((G, +, \mathcal{O})\) and a group generator \(P\).

- Alice
  
  \[ a \leftarrow \text{random()} \]
  
  Computes \( A = a \cdot P \)

- Bob
  
  \[ b \leftarrow \text{random()} \]
  
  Computes \( B = b \cdot P \)

\( B \rightarrow \) Alice

\( A \leftarrow \) Bob

Shared Secret Key \( K = a \cdot b \cdot P \)
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group \((G, +, O)\) and a group generator \(P\).

Alice
\[
\begin{align*}
a & \leftarrow \text{random()} \\
\text{Computes } A &= a \cdot P \\
\text{Computes } K &= a \cdot B
\end{align*}
\]

Bob
\[
\begin{align*}
b & \leftarrow \text{random()} \\
\text{Computes } B &= b \cdot P \\
\text{Computes } K &= b \cdot A
\end{align*}
\]
Diffie-Hellmann key exchange protocol

Alice and Bob agree on an Abelian group \((G, +, O)\) and a group generator \(P\).

- Alice
  - \(a \leftarrow \text{random}()\)
  - Computes \(A = a \cdot P\)
  - Computes \(K = a \cdot B\)

- Bob
  - \(b \leftarrow \text{random}()\)
  - Sends \(B\) to Alice
  - Computes \(B = b \cdot P\)
  - Sends \(A\) to Alice
  - Computes \(K = b \cdot A\)

Shared Secret Key \(K = a \cdot b \cdot P\)

\(\rightarrow\) The main operation is the scalar multiplication \(a \cdot P\).
## Multiplicative groups vs Abelian (additive) groups

<table>
<thead>
<tr>
<th></th>
<th>Multiplicative group $(\mathcal{G}, \times, 1)$</th>
<th>Abelian group $(\mathcal{G}, +, 0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group operation</strong></td>
<td>$a \times b$</td>
<td>$P + Q$</td>
</tr>
<tr>
<td></td>
<td>$a^2$</td>
<td>$[2] \cdot P$</td>
</tr>
<tr>
<td><strong>Neutral element</strong></td>
<td>$1$</td>
<td>$0$ or $0$</td>
</tr>
<tr>
<td></td>
<td>$a^e$</td>
<td>$[k] \cdot P$</td>
</tr>
<tr>
<td><strong>Exponentiation</strong></td>
<td></td>
<td>scalar multiplication</td>
</tr>
<tr>
<td><strong>Discrete logarithm problem</strong></td>
<td>knowing $X$ and $a$, find $e$ such as $X = a^e$</td>
<td>knowing $X$ and $P$, find $k$ such as $X = [k] \cdot P$</td>
</tr>
</tbody>
</table>
## Multiplicative groups vs Abelian (additive) groups

<table>
<thead>
<tr>
<th></th>
<th>Multiplicative group $(G, \times, 1)$</th>
<th>Abelian group $(G, +, 0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group operation</td>
<td>$a \times b$</td>
<td>$P + Q$</td>
</tr>
<tr>
<td></td>
<td>$a^2$</td>
<td>$[2] \cdot P$</td>
</tr>
<tr>
<td>Neutral element</td>
<td>1</td>
<td>0 or $0$</td>
</tr>
<tr>
<td></td>
<td>$a^e$</td>
<td>$[k] \cdot P$</td>
</tr>
<tr>
<td></td>
<td>exponentiation</td>
<td>scalar multiplication</td>
</tr>
<tr>
<td>Discrete logarithm problem</td>
<td>knowing $X$ and $a$, find $e$ such as $X = a^e$</td>
<td>knowing $X$ and $P$, find $k$ such as $X = [k] \cdot P$</td>
</tr>
</tbody>
</table>

### Example: ElGamal encryption

<table>
<thead>
<tr>
<th>Alice’s private key</th>
<th>$x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice’s public key</td>
<td>$h \leftarrow g^x$</td>
</tr>
<tr>
<td></td>
<td>$(G, g, h)$</td>
</tr>
<tr>
<td></td>
<td>$H \leftarrow [x] \cdot P$</td>
</tr>
<tr>
<td></td>
<td>$(G, P, H)$</td>
</tr>
<tr>
<td>Bob’s encryption $(c_1, c_2)$</td>
<td>$y = $ Bob’s secret parameter</td>
</tr>
<tr>
<td></td>
<td>$c_1 \leftarrow g^y$, $s \leftarrow h^y$</td>
</tr>
<tr>
<td></td>
<td>$c_2 \leftarrow m \cdot s$</td>
</tr>
<tr>
<td>Alice’s decryption</td>
<td>$s \leftarrow c_1^x$</td>
</tr>
<tr>
<td></td>
<td>$m' \leftarrow c_2 \cdot s^{-1}$</td>
</tr>
<tr>
<td></td>
<td>$(m' \leftarrow m \cdot s \cdot s^{-1})$</td>
</tr>
<tr>
<td></td>
<td>$S \leftarrow [x] \cdot c_1$</td>
</tr>
<tr>
<td></td>
<td>$m' \leftarrow c_2 - S$</td>
</tr>
<tr>
<td></td>
<td>$(m' \leftarrow m + S - S)$</td>
</tr>
</tbody>
</table>
## ECC vs Exponentiation over \( \mathbb{F}_p \)

<table>
<thead>
<tr>
<th>Date</th>
<th>Minimum of Strength</th>
<th>Symmetric Algorithms</th>
<th>Factoring Modulus</th>
<th>Discrete Logarithm Key</th>
<th>Elliptic Curve</th>
<th>Hash (A)</th>
<th>Hash (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010 (Legacy)</td>
<td>80</td>
<td>2TDEA*</td>
<td>1024</td>
<td>160</td>
<td>1024</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>2011 - 2030</td>
<td>112</td>
<td>3TDEA</td>
<td>2048</td>
<td>224</td>
<td>2048</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>&gt; 2030</td>
<td>128</td>
<td>AES-128</td>
<td>3072</td>
<td>256</td>
<td>3072</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt; 2030</td>
<td>192</td>
<td>AES-192</td>
<td>7680</td>
<td>384</td>
<td>7680</td>
<td>384</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;&gt; 2030</td>
<td>256</td>
<td>AES-256</td>
<td>15360</td>
<td>512</td>
<td>15360</td>
<td>512</td>
<td></td>
</tr>
</tbody>
</table>


Jean-Marc ROBERT  Team DALI/LIRMM, Université de Perpignan, France
Our group: the set of Elliptic Curve points
Our group: the set of Elliptic Curve points

Example over $\mathbb{R}$:

- Point addition
- Point doubling
Our group: the set of Elliptic Curve points

Our curve is over a finite field $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$ (instead of $\mathbb{R}$):

$E : Y^2 = X^3 + aX + b$, $a, b \in \mathbb{F}_p$.

$E : Y^2 + XY = X^3 + aX^2 + b$, $a, b \in \mathbb{F}_{2^m}$.

- Finite field operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>$A, B \in \mathbb{F}_p$</th>
<th>$A, B \in \mathbb{F}_{2^m}$ ($= \mathbb{F}_2[x]/(f(x) \cdot \mathbb{F}_2[x])$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field element</td>
<td>$0 \leq A &lt; p$</td>
<td>$A = \sum_{i=0}^{m-1} a_i \cdot x^i$</td>
</tr>
<tr>
<td></td>
<td>$0 \leq B &lt; p$</td>
<td>$B = \sum_{i=0}^{m-1} b_i \cdot x^i$</td>
</tr>
<tr>
<td></td>
<td>$p$ large prime</td>
<td>$a_i, b_i \in {0, 1}$</td>
</tr>
<tr>
<td>Addition $A + B =$</td>
<td>$A + B \mod p$</td>
<td>$\sum_{i=0}^{m-1} (a_i + b_i) \cdot x^i$</td>
</tr>
<tr>
<td>Multiplication $A \times B =$</td>
<td>$A \times B \mod p$</td>
<td>$A \cdot B \mod f$</td>
</tr>
</tbody>
</table>
Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis

3. Conclusion
Elliptic Curve Point Operations

Doubling and addition formulas: \( P_1 + P_2 = P_3 \)
(with \( P_1 = (x_1, y_1) \), \( P_2 = (x_2, y_2) \), \( P_3 = (x_3, y_3) \).)

\[
\begin{align*}
\mathbb{F}_p & \quad | \quad \mathbb{F}_{2^m} \\
\begin{cases}
  x_3 = \lambda^2 - x_1 - x_2 \\
y_3 = (x_1 - x_3)\lambda - y_1
\end{cases} & \quad | \quad \begin{cases}
x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \\
y_3 = (x_1 + x_3)\lambda + x_3 + y_1
\end{cases}
\end{align*}
\]

with

\[
\begin{align*}
\lambda &= \frac{y_2 - y_1}{x_2 - x_1} \quad \text{if } P_1 \neq P_2 \\
\lambda &= \frac{3x_1^2 + a}{2y_1} \quad \text{if } P_1 = P_2
\end{align*}
\]

\[
\begin{align*}
\lambda &= \frac{y_1 + y_2}{x_1 + x_2} \quad \text{if } P_1 \neq P_2 \\
\lambda &= \frac{y_1}{x_1} + x_1 \quad \text{if } P_1 = P_2
\end{align*}
\]
Elliptic Curve Point Operations

Doubling and addition formulas: \( P_1 + P_2 = P_3 \)
(with \( P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_3 = (x_3, y_3) \).)

\[
\begin{align*}
\mathbb{F}_p & \quad \mathbb{F}_{2^m} \\
\begin{align*}
x_3 &= \lambda^2 - x_1 - x_2 \\
y_3 &= (x_1 - x_3)\lambda - y_1
\end{align*}
& \quad \begin{align*}
x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\
y_3 &= (x_1 + x_3)\lambda + x_3 + y_1
\end{align*}
\end{align*}
\]

with

\[
\lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{if} \quad P_1 \neq P_2 \]
\[
\lambda = \frac{y_1 + y_2}{x_1 + x_2} \quad \text{if} \quad P_1 \neq P_2
\]
\[
\lambda = \frac{3x_1^2 + a}{2y_1} \quad \text{if} \quad P_1 = P_2
\]
\[
\lambda = \frac{y_1}{x_1} + x_1 \quad \text{if} \quad P_1 = P_2
\]

- One **doubling** requires 1 inversion, 2 multiplications, 1 or 2 squaring(s), and some additions;
- One **addition** requires 1 inversion, 2 multiplications, 1 squaring, and some additions;
Elliptic curve point representation: Projective Coordinate Systems

In order to eliminate the field inversion (the costliest operation), we use projective coordinate systems.

Example over $\mathbb{F}_{2^m}$ (Lopez-Dahab system):

$P = (x, y)$ is transformed into $(X : Y : Z)$ with

$$
\begin{align*}
    x & = \frac{X}{Z} \\
    y & = \frac{Y}{Z^2}
\end{align*}
$$

2M + 1S + 1I \quad \text{(Affine)}

4M + 4S \quad \text{(Lopez-Dahab System, Projective + Affine)}

13M + 4S \quad \text{(Addition, Mixed Coordinates, Projective + Affine)}

9M + 5S \quad \text{(Addition, Projective + Affine)}
Elliptic curve point representation: Projective Coordinate Systems

- In order to eliminate the field inversion (the costliest operation), we use projective coordinate systems.
  Example over $\mathbb{F}_{2^m}$ (Lopez-Dahab system):

  $P = (x, y)$ is transformed into $(X : Y : Z)$ with

  $\begin{cases} 
  x = \frac{X}{Z} \\
  y = \frac{Y}{Z^2}
  \end{cases}$

- Now, the doubling is computed as follows:

  $2.(X : Y : Z) = (X_1 : Y_1 : Z_1)$ with

  $\begin{cases} 
  X_1 = X^4 + b \cdot Z^4 \\
  Y_1 = bZ^4 \cdot Z_1 + X_1 \cdot (aZ_1 + Y^2 + bZ^4) \\
  Z_1 = X^2 \cdot Z^2
  \end{cases}$
Elliptic curve point representation: Projective Coordinate Systems

- In order to eliminate the field inversion (the costliest operation), we use projective coordinate systems.

Example over $\mathbb{F}_{2^m}$ (Lopez-Dahab system):

$$P = (x, y) \text{ is transformed into } (X : Y : Z) \text{ with } \begin{cases} \frac{x}{Z} = X \\ \frac{y}{Z^2} = Y \end{cases}$$

- Now, the doubling is computed as follows:

$$2.(X : Y : Z) = (X_1 : Y_1 : Z_1) \text{ with } \begin{cases} X_1 = X^4 + b \cdot Z^4 \\ Y_1 = bZ^4 \cdot Z_1 + X_1 \cdot (aZ_1 + Y^2 + bZ^4) \\ Z_1 = X^2 \cdot Z^2 \end{cases}$$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Affine</th>
<th>Lopez-Dahab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doubling</td>
<td>2M + 1S + 1I</td>
<td>4M + 4S</td>
</tr>
<tr>
<td>Addition</td>
<td>2M + 1S + 1I</td>
<td>13M + 4S</td>
</tr>
<tr>
<td>Addition (mixed coordinates)</td>
<td>-</td>
<td>9M + 5S (Projective + Affine)</td>
</tr>
</tbody>
</table>
Outline

1 Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2 Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis

3 Conclusion
ECSM algorithm: Double-and-add

Let \( k = (k_{t-1}, \ldots, k_1, k_0)_2 \in \mathbb{N}, P \in E(\mathbb{F}_{2^m}) \)

\[
k \cdot P = (\sum_{i=0}^{t-1} 2^i k_i) \cdot P
= ((\ldots((k_{t-1} \cdot 2 + k_{t-2}) \cdot 2 + k_{t-3}) \cdot 2 \ldots + k_1) \cdot 2 + k_0) \cdot P
= ((\ldots((k_{t-1} \cdot P \cdot 2 + k_{t-2} \cdot P) \cdot 2 + k_{t-3} \cdot P) \cdot 2 \ldots + k_1 \cdot P) \cdot 2 + k_0 \cdot P)
\]
ECSM algorithm: *Double-and-add*

- Let \( k = (k_{t-1}, \ldots, k_1, k_0)_{2} \in \mathbb{N}, P \in E(\mathbb{F}_{2^m}) \)
- \( k \cdot P = (\sum_{i=0}^{t-1} 2^i k_i) \cdot P \)
  \[ = (((...((k_{t-1} \cdot 2 + k_{t-2}) \cdot 2 + k_{t-3}) \cdot 2 \ldots + k_1) \cdot 2 + k_0) \cdot P \]
  \[ = (((...((k_{t-1} \cdot P \cdot 2 + k_{t-2} \cdot P) \cdot 2 + k_{t-3} \cdot P) \cdot 2 \ldots + k_1 \cdot P) \cdot 2 + k_0 \cdot P) \]

- This is the following algorithm:

```latex
\begin{algorithm}
\caption{Left-to-Right double-and-add Elliptic Curve Scalar Multiplication (ECSM)}
\begin{algorithmic}[1]
\Require \( k = (k_{t-1}, \ldots, k_1, k_0), P \in E(\mathbb{F}_{2^m}) \)
\Ensure \( Q = k \cdot P \)
\State \( Q \leftarrow \emptyset \)
\For {\( i \) from \( t-1 \) downto 0}
\State \( Q \leftarrow 2 \cdot Q \)
\If {\( k_i = 1 \)}
\State \( Q \leftarrow Q + P \)
\EndIf
\EndFor
\State \textbf{return} \((Q)\)
\end{algorithmic}
\end{algorithm}
```
ECSM algorithm: *Double-and-add*

- Let \( k = (k_{t-1}, ..., k_1, k_0) \in \mathbb{N}, P \in E(\mathbb{F}_{2^m}) \)

\[
k \cdot P = \left( \sum_{i=0}^{t-1} 2^i k_i \right) \cdot P
= ((\ldots ((k_{t-1} \cdot 2 + k_{t-2}) \cdot 2 + k_{t-3}) \cdot 2 \ldots + k_1) \cdot 2 + k_0) \cdot P
= ((\ldots ((k_{t-1} \cdot P \cdot 2 + k_{t-2} \cdot P) \cdot 2 + k_{t-3} \cdot P) \cdot 2 \ldots + k_1 \cdot P) \cdot 2 + k_0 \cdot P)
\]

This is the following algorithm:

```
Left-to-Right double-and-add
Elliptic Curve Scalar Multiplication (ECSM)

Require: \( k = (k_{t-1}, ..., k_1, k_0), P \in E(\mathbb{F}_{2^m}) \)
Ensure: \( Q = k \cdot P \)
1: \( Q \leftarrow O \)
2: for \( i \) from \( t - 1 \) downto 0 do
3: \( Q \leftarrow 2 \cdot Q \)
4: if \( k_i = 1 \) then
5: \( Q \leftarrow Q + P \)
6: end if
7: end for
8: return \( (Q) \)
```

"mixed coordinate addition" \( \Rightarrow \)
Double-and-add improvement: NAF and W-NAF.

NAF replaces the sequences of consecutive 1: let $k \in \mathbb{N}$ such as $k = 2^i - 1$, then

$$(k)_2 = \overbrace{111...1}^{i \text{ times}}$$

and one has: $$(k)_{NAF} = \overbrace{100...00 - 1}^{i+1 \text{ digits}}.$$ 

$\Rightarrow$ Average number of non zero digits: from $n/2$ down to $n/3$. 

Double-and-add improvement: NAF and W-NAF.

- **NAF** replaces the sequences of consecutive 1: let \( k \in \mathbb{N} \) such as \( k = 2^i - 1 \), then

\[
(k)_2 = \underbrace{111...1}_{i \text{ times}} \quad \text{and one has: } (k)_{\text{NAF}} = \underbrace{100...00}_{i+1 \text{ digits}} - 1.
\]

\( \Rightarrow \) Average number of non zero digits: from \( n/2 \) down to \( n/3 \).

- **W-NAF** decreases even more the number of non zero digits: \( n/(w + 1) \) now by using:

\[
\{-2^{w-1} + 1, ..., -5, -3, -1, 0, 1, 3, 5, ..., 2^{w-1} - 1\}.
\]

- Complexity balance for the ECSM over \( E(\mathbb{F}_p) \) or \( E(\mathbb{F}_q) \):

<table>
<thead>
<tr>
<th></th>
<th>doubling number</th>
<th>additions number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Double-and-add</strong></td>
<td>( n )</td>
<td>( n/2 )</td>
</tr>
<tr>
<td><strong>NAF Double-and-add</strong></td>
<td>( n )</td>
<td>( n/3 )</td>
</tr>
<tr>
<td><strong>W-NAF Double-and-add</strong></td>
<td>( n )</td>
<td>( n/(w + 1) + 2^{w-2} )</td>
</tr>
</tbody>
</table>
## State of the art

<table>
<thead>
<tr>
<th>Scalar multiplication</th>
<th>Curve</th>
<th>Security</th>
<th>processor</th>
<th>Method</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Langley [7]</td>
<td>128</td>
<td>i7 SB</td>
<td>Montg. ladder</td>
<td>229000</td>
</tr>
<tr>
<td></td>
<td>Bernstein [2, 1]</td>
<td>128</td>
<td>i7 SB</td>
<td>Montg. ladder</td>
<td>194000</td>
</tr>
<tr>
<td></td>
<td>Longa et al. [8]</td>
<td>128</td>
<td>2 Duo</td>
<td>WNAF D&amp;A</td>
<td>337000</td>
</tr>
<tr>
<td></td>
<td>Longa et al. [8]</td>
<td>128</td>
<td>2 Duo</td>
<td>WNAF D&amp;A</td>
<td>281000</td>
</tr>
<tr>
<td>$E(\mathbb{F}_{2^m})$</td>
<td>Nègre et al. [9]</td>
<td>112</td>
<td>i7 SB</td>
<td>WNAF D-H&amp;A</td>
<td>98000</td>
</tr>
<tr>
<td></td>
<td>Taverne et al. [11]</td>
<td>112</td>
<td>i7 SB</td>
<td>WNAF D-H&amp;A</td>
<td>102000</td>
</tr>
<tr>
<td></td>
<td>Nègre et al. [9]</td>
<td>192</td>
<td>i7 SB</td>
<td>WNAF D-H&amp;A</td>
<td>347000</td>
</tr>
<tr>
<td></td>
<td>Taverne et al. [11]</td>
<td>192</td>
<td>i7 SB</td>
<td>WNAF D-H&amp;A</td>
<td>358000</td>
</tr>
</tbody>
</table>
State of the art

<table>
<thead>
<tr>
<th>Scalar multiplication</th>
<th>Curve</th>
<th>Security</th>
<th>processor</th>
<th>Method</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E(\mathbb{F}_p) )</td>
<td>Hamburg [4]</td>
<td>Montgomery</td>
<td>126</td>
<td>i7 SB</td>
<td>Montg. ladder</td>
</tr>
<tr>
<td></td>
<td>Langley [7]</td>
<td>Curve25519</td>
<td>128</td>
<td>i7 SB</td>
<td>Montg. ladder</td>
</tr>
<tr>
<td></td>
<td>Bernstein [2, 1]</td>
<td>Curve25519</td>
<td>128</td>
<td>i7 SB</td>
<td>Montg. ladder</td>
</tr>
<tr>
<td></td>
<td>Longa et al. [8]</td>
<td>jac256189</td>
<td>128</td>
<td>2 Duo</td>
<td>WNAF D&amp;A</td>
</tr>
<tr>
<td></td>
<td>Longa et al. [8]</td>
<td>ted256189</td>
<td>128</td>
<td>2 Duo</td>
<td>WNAF D&amp;A</td>
</tr>
<tr>
<td>( E(\mathbb{F}_{2^m}) )</td>
<td>Nègre et al. [9]</td>
<td>B233</td>
<td>112</td>
<td>i7 SB</td>
<td>WNAF D-H&amp;A</td>
</tr>
<tr>
<td></td>
<td>Taverne et al. [11]</td>
<td>B233</td>
<td>112</td>
<td>i7 SB</td>
<td>WNAF D-H&amp;A</td>
</tr>
<tr>
<td></td>
<td>Nègre et al. [9]</td>
<td>B409</td>
<td>192</td>
<td>i7 SB</td>
<td>WNAF D-H&amp;A</td>
</tr>
</tbody>
</table>

This is several hundreds of time faster than the modular exponentiation over \( \mathbb{Z}/p\mathbb{Z} \)!
Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2. Side-channel Attacks
   - First Attack: Simple Power Analysis
     - How to thwart SPA?
   - Second Attack: Differential Power Analysis
     - How to thwart DPA?
   - Synthesis

3. Conclusion
Side-Channel attack: What is it?

Cryptographic device
(e.g., smart card and reader)

Control, Cyphertexts

Control, Waveform data

Oscilloscope

Computer

Side Channel test bench.
Jean-Marc ROBERT  Team DALI/LIRMM, Université de Perpignan, France
Vulnerable ECSM: Double-And-Add case

Left-to-Right double-and-add
Elliptic Curve Scalar Multiplication (ECSM)

Require: $k = (k_{t-1}, \ldots, k_1, k_0), P \in E(\mathbb{F}_2^m)$
Ensure: $Q = k \cdot P$

1: $Q \leftarrow O$
2: for $i$ from $t - 1$ downto 0 do
3: \hspace{1em} $Q \leftarrow 2 \cdot Q$
4: \hspace{1em} if $k_i = 1$ then
5: \hspace{2em} $Q \leftarrow Q + P$
6: \hspace{1em} end if
7: end for
8: return $(Q)$
Vulnerable ECSM: Double-And-Add case

Left-to-Right double-and-add
Elliptic Curve Scalar Multiplication (ECSM)

Require: \( k = (k_{t-1}, \ldots, k_1, k_0), P \in E(\mathbb{F}_{2^m}) \)
Ensure: \( Q = k \cdot P \)
1: \( Q \leftarrow O \)
2: for \( i \) from \( t - 1 \) downto 0 do
3: \( Q \leftarrow 2 \cdot Q \)
4: \( \text{if} \ k_i = 1 \ \text{then} \)
5: \( Q \leftarrow Q + P \)
6: \( \text{end if} \)
7: \( \text{end for} \)
8: return \( (Q) \)

→ Vulnerable: the sequence of operations leaks the secret scalar (no regularity)

→ Simple Power Analysis
Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis

3. Conclusion
How to thwart SPA: first idea, Double-and-add-always

As a counter-measure, Coron in [3] suggested the following algorithm:

Double-and-add-always

Require: $k = (k_{t-1}, \ldots, k_1, k_0)$ with $k_{t-1} = 1$, $P \in E(\mathbb{F}_q)$
Ensure: $Q = k \cdot P$
1: $Q_0 \leftarrow \mathcal{O}$, $Q_1 \leftarrow \mathcal{O}$
2: for $i$ from $t-1$ downto 0 do
3: $Q_0 \leftarrow 2 \cdot Q_0$
4: if ($k_i = 0$) then
5: $Q_1 \leftarrow Q_0 + P$
6: else
7: $Q_0 \leftarrow Q_0 + P$
8: end if
9: end for
10: return ($Q_0$)

Weak against Fault Injection Attacks (see Oviedo in [10]).
How to thwart SPA: first idea, Double-and-add-always

As a counter-measure, Coron in [3] suggested the following algorithm:

**Double-and-add-always**

Require: \( k = (k_{t-1}, \ldots, k_1, k_0) \) with \( k_{t-1} = 1, P \in E(\mathbb{F}_q) \)
Ensure: \( Q = k \cdot P \)

1: \( Q_0 \leftarrow O, Q_1 \leftarrow O \)
2: for \( i \) from \( t - 1 \) downto \( 0 \) do
3: \( Q_0 \leftarrow 2 \cdot Q_0 \)
4: if \( (k_i = 0) \) then
5: \( Q_1 \leftarrow Q_0 + P \) \( ⇐ \) "dummy addition"
6: else
7: \( Q_0 \leftarrow Q_0 + P \)
8: end if
9: end for
10: return \( (Q_0) \)

Double-and-add-always

Weak against Fault Injection Attacks (see Oviedo in [10]).
How to thwart SPA: Montgomery ladder

Montgomery

Require: \( k = (k_{t-1}, \ldots, k_1, k_0) \) with \( k_{t-1} = 1, P \in E(\mathbb{F}_q) \)

Ensure: \( Q = k \cdot P \)

1: \( Q_0 \leftarrow P, Q_1 \leftarrow 2P \)
2: for \( i \) from \( t-2 \) downto 0 do
3: \hspace{1em} if \( (k_i = 0) \) then
4: \hspace{2em} \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
5: \hspace{1em} else
6: \hspace{2em} \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
7: \hspace{1em} end if
8: end for
9: return \( (Q_0) \)

Basic Montgomery’s ladder ECSM
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0\times2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P. \)

1: if \( (k_i = 0) \) then
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: else
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: end if

<table>
<thead>
<tr>
<th>( i )</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_i )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( Q_0 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Q_1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: if ($k_i = 0$) then
2: $Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: else
4: $Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: end if

<table>
<thead>
<tr>
<th>$i$</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_i$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$Q_0$</td>
<td>$P$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: if ($k_i = 0$) then
2: $Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: else
4: $Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: end if

<table>
<thead>
<tr>
<th>$i$</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_i$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$Q_0$</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Q_1$</td>
<td>2P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P. \)

1: if \( k_i = 0 \) then
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: else
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: end if

<table>
<thead>
<tr>
<th>( i )</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_i )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( Q_0 )</td>
<td>( P )</td>
<td>2P</td>
<td>3P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Q_1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P \).

1: if \( (k_i = 0) \) then
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: else
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: end if

<table>
<thead>
<tr>
<th>( i )</th>
<th>( k_i )</th>
<th>( Q_0 )</th>
<th>( Q_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>1</td>
<td>( P )</td>
<td>2P</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>( 2P )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3P</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Double:

\( Q_0 \leftarrow 2 \cdot Q_0 \)

\( (Q_1 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P \)
therefore \( Q_1 \leftarrow Q_0 = P \)
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P \).

\[
\begin{align*}
1: & \text{ if } (k_i = 0) \text{ then} \\
2: & Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \\
3: & \text{ else} \\
4: & Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \\
5: & \text{ end if}
\end{align*}
\]

Double-and-add:

\[
Q_0 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P \\
(Q_1 \leftarrow 2 \cdot (Q_0 + P) \therefore Q_1 - Q_0 = P)
\]

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
i & \text{init} & 4 & 3 & 2 & 1 & 0 \\
\hline
k_i & 1 & 0 & 1 & 1 & 0 & 1 \\
Q_0 & P & 2P & \text{3P} & \text{5P} & & \\
Q_1 & 2P & & & & & \\
\hline
\end{array}
\]
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P. \)

1: if \( (k_i = 0) \) then
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: else
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: end if

<table>
<thead>
<tr>
<th>( i )</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_i )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( Q_0 )</td>
<td>( P )</td>
<td>2P</td>
<td>2P</td>
<td>5P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Q_1 )</td>
<td>2P</td>
<td>3P</td>
<td>6P</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: if ($k_i = 0$) then
2: $Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: else
4: $Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: end if

Double-and-add:

$Q_0 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P$

$(Q_1 \leftarrow 2 \cdot (Q_0 + P)$

therefore $Q_1 - Q_0 = P$)

<table>
<thead>
<tr>
<th>$i$</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_i$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$Q_0$</td>
<td>P</td>
<td>2P</td>
<td>2P</td>
<td>5P</td>
<td>11P</td>
<td></td>
</tr>
<tr>
<td>$Q_1$</td>
<td>2P</td>
<td>3P</td>
<td>6P</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: $k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2$

At the beginning, we set: $Q_0 = P; Q_1 = 2P$.

1: if $(k_i = 0)$ then
2: $Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0$
3: else
4: $Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1$
5: end if

<table>
<thead>
<tr>
<th>$i$</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_i$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$Q_0$</td>
<td>$P$</td>
<td>$2P$</td>
<td>$3P$</td>
<td>$5P$</td>
<td>$11P$</td>
<td></td>
</tr>
<tr>
<td>$Q_1$</td>
<td>$2P$</td>
<td>$3P$</td>
<td>$6P$</td>
<td>$12P$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P. \)

1: if \( k_i = 0 \) then
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: else
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: end if

<table>
<thead>
<tr>
<th>( i )</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_i )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( Q_0 )</td>
<td>P</td>
<td>2P</td>
<td>3P</td>
<td>5P</td>
<td>11P</td>
<td>12P</td>
</tr>
<tr>
<td>( Q_1 )</td>
<td>2P</td>
<td>3P</td>
<td>6P</td>
<td>12P</td>
<td>23P</td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P \).

1: if \( (k_i = 0) \) then
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: else
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: end if

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
i & \text{init} & 4 & 3 & 2 & 1 & 0 \\
\hline
k_i & 1 & 0 & 1 & 1 & 0 & 1 \\
Q_0 & P & 2P & 2P & 5P & 11P & 22P \\
Q_1 & 2P & 3P & 6P & 12P & 23P \\
\hline
\end{array}
\]

Double:
\( Q_0 \leftarrow 2 \cdot Q_0 \)

\( (Q_1 \leftarrow Q_0 + Q_1 = 2 \cdot Q_0 + P \) therefore \( Q_1 - Q_0 = P \) \)
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0x2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P. \)

1: \( \text{if } (k_i = 0) \text{ then} \)
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: \( \text{else} \)
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: \( \text{end if} \)

<table>
<thead>
<tr>
<th>( i )</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_i )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( Q_0 )</td>
<td>( P )</td>
<td>2P</td>
<td>2P</td>
<td>5P</td>
<td>11P</td>
<td>22P</td>
</tr>
<tr>
<td>( Q_1 )</td>
<td>( 2P )</td>
<td>3P</td>
<td>6P</td>
<td>12P</td>
<td>23P</td>
<td></td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0 \times 2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P. \)

1: \textbf{if} \((k_i = 0)\) \textbf{then}
2: \hspace{1em} \(Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0\)
3: \textbf{else}
4: \hspace{1em} \(Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1\)
5: \textbf{end if}

<table>
<thead>
<tr>
<th>(i)</th>
<th>init</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k_i)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(Q_0)</td>
<td>(P)</td>
<td>(2P)</td>
<td>(3P)</td>
<td>(5P)</td>
<td>(11P)</td>
<td>(22P)</td>
</tr>
<tr>
<td>(Q_1)</td>
<td>(2P)</td>
<td>(3P)</td>
<td>(6P)</td>
<td>(12P)</td>
<td>(23P)</td>
<td>(46P)</td>
</tr>
</tbody>
</table>
How does the Montgomery ladder work?

Example: \( k = 45_{10} = 0 \times 2D = [1, 0, 1, 1, 0, 1]_2 \)

At the beginning, we set: \( Q_0 = P; Q_1 = 2P. \)

1: if \( (k_i = 0) \) then
2: \( Q_1 \leftarrow Q_0 + Q_1, Q_0 \leftarrow 2 \cdot Q_0 \)
3: else
4: \( Q_0 \leftarrow Q_0 + Q_1, Q_1 \leftarrow 2 \cdot Q_1 \)
5: end if

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
i & init & 4 & 3 & 2 & 1 & 0 \\
\hline
k_i & 1 & 0 & 1 & 1 & 0 & 1 \\
Q_0 & P & 2P & 5P & 11P & 22P & 45P \\
Q_1 & 2P & 3P & 6P & 12P & 23P & 46P \\
\hline
\end{array}
\]

The algorithm returns:

\[ \rightarrow Q_0 = 45P \]
Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis

3. Conclusion
Second Attack: *Differential Power Analysis*

These attacks are presented in Kocher *et al.* [5, 6]. The attacker uses a power consumption model of the device. Let $\mathcal{H}(k)$ be the Hamming weight of the processed data $\Delta$ at time $t$. Then, our model is:

$$P(t) = C \cdot \mathcal{H}(\Delta) + \epsilon + K$$

($C$ and $K$ are constants, $\epsilon$ represents the noise.)
Second Attack: *Differential Power Analysis*

A toy example: The attacker asks the device to compute a series of multiplications of chosen \( I[i] \) by a secret constant \( k \) over 12 bits (CPA).

Assume the attacker knows the seven first bits of \( k : [1, 0, 1, 1, 0, 0]_2 = 0x5A \).

The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ( I[i] )</th>
<th>intermediate value loop round 6 ( Q_0 )</th>
<th>Assumption ( k_i = 1 ) first 8 bits ( \rightarrow 0xB5 ) ( (= (0x5A &lt;&lt; 1) \land 0x1) )</th>
<th>selected bit: ( n^8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

When the device processes the selected bit \( n^8 \), if the assumption on \( k_i \) is correct, the “red ones” need more electric power than the “black ones.”

Jean-Marc ROBERT Team DALI/LIRMM, Université de Perpignan, France
Second Attack: Differential Power Analysis

A toy example: The attacker asks the device to compute a series of multiplications of chosen \( I[i] \) by a secret constant \( k \) over 12 bits (CPA).

Assume the attacker knows the seven first bits of \( k : [1, 0, 1, 1, 0, 0]_2 = 0x5A \).

The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ((I[i]))</th>
<th>intermediate value loop round 6 ((Q_0))</th>
<th>Assumption (k_i = 1) first 8 bits → 0xB5 ((= (0x5A &lt;&lt; 1) \land 0x1))</th>
<th>selected bit: (n^8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the device processes the selected bit \( (n^8) \), if the assumption on \( k_i \) is correct, the “red ones” need more electric power than the “black ones”.

Jean-Marc ROBERT  Team DALI/LIRMM, Université de Perpignan, France
Second Attack: **Differential Power Analysis**

A toy example: The attacker asks the device to compute a series of multiplications of chosen $I[i]$ by a secret constant $k$ over 12 bits (CPA).

Assume the attacker knows the seven first bits of $k$: $[1, 0, 1, 1, 1, 0, 0]_2 = 0x5A$. The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ($I[i]$)</th>
<th>intermediate value loop round 6 ($Q_0$)</th>
<th>Assumption $k_i = 1$ first 8 bits $\rightarrow 0xB5$ ($= (0x5A &lt;&lt; 1) \land 0x1$)</th>
<th>selected bit: $n^8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td>0xD6F</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the device processes the selected bit ($n^8$), if the assumption on $k_i$ is correct, the “red ones” need more electric power than the “black ones.”

Jean-Marc ROBERT  Team DALI/LIRMM, Université de Perpignan, France
Second Attack: *Differential Power Analysis*

A toy example: The attacker asks the device to compute a series of multiplications of chosen $I[i]$ by a secret constant $k$ over 12 bits (CPA).

Assume the attacker knows the seven first bits of $k$: $[1, 0, 1, 1, 0, 0]_2 = 0x5A$.

The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ($I[i]$)</th>
<th>intermediate value loop round 6 ($Q_0$)</th>
<th>Assumption $k_i = 1$ first 8 bits $\rightarrow$ 0xB5 ($= (0x5A &lt;&lt; 1) \land 0x1$)</th>
<th>selected bit: $n^\circ 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td>0xD6F</td>
<td>1</td>
</tr>
<tr>
<td>0xB2</td>
<td>0x3E94</td>
<td>0x7DDA</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...500–1000 computations over 16 bits, average Hamming weight of the “red ones”: 8.5; average Hamming weight of the “black ones”: 7.5.

When the device processes the selected bit ($n^\circ 8$), if the assumption on $k_i$ is correct, the “red ones” need more electric power than the “black ones.”
Second Attack: Differential Power Analysis

A toy example: The attacker ask the device to compute a serie of multiplications of chosen \( I[i] \) by a secret constant \( k \) over 12 bits (CPA).
Assume the attacker knows the seven first bits of \( k \): \([1, 0, 1, 1, 1, 0, 0]_2 = 0x5A\).
The attacker select one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ((I[i]))</th>
<th>intermediate value loop round 6 ((Q_0))</th>
<th>Assumption ( k_i = 1 ) first 8 bits (\rightarrow 0xB5) ((= (0x5A &lt;&lt; 1) \land 0x1))</th>
<th>selected bit: (n^\circ 8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td>0xD6F</td>
<td>1</td>
</tr>
<tr>
<td>0xB2</td>
<td>0x3E94</td>
<td>0x7DDA</td>
<td>1</td>
</tr>
<tr>
<td>0x56</td>
<td>0x1E3C</td>
<td>0x3CCE</td>
<td>0</td>
</tr>
</tbody>
</table>

... 500-1000 computations over 16 bits, average Hamming weight of the "red ones": 8.5; average Hamming weight of the "black ones": 7.5.
When the device processes the selected bit \((n^\circ 8)\), if the assumption on \( k_i \) is correct, the "red ones" need more electric power than the "black ones".
Second Attack: Differential Power Analysis

A toy example: The attacker asks the device to compute a series of multiplications of chosen $I[i]$ by a secret constant $k$ over 12 bits (CPA).
Assume the attacker knows the seven first bits of $k$: $[1, 0, 1, 1, 0, 0]_2 = 0x5A$.
The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ($I[i]$)</th>
<th>intermediate value loop round 6 ($Q_0$)</th>
<th>Assumption $k_i = 1$ first 8 bits $\rightarrow 0xB5$ ($= (0x5A &lt;&lt; 1) \land 0x1$)</th>
<th>selected bit: $n^\circ 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td>0xD6F</td>
<td>1</td>
</tr>
<tr>
<td>0xB2</td>
<td>0x3E94</td>
<td>0x7DDA</td>
<td>1</td>
</tr>
<tr>
<td>0x56</td>
<td>0x1E3C</td>
<td>0x3CCE</td>
<td>0</td>
</tr>
<tr>
<td>0xC2</td>
<td>0x4434</td>
<td>0x892A</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
A toy example: The attacker asks the device to compute a series of multiplications of chosen $l[i]$ by a secret constant $k$ over 12 bits (CPA).

Assume the attacker knows the seven first bits of $k$: $[1, 0, 1, 1, 1, 0, 0]_2 = 0x5A$.

The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ($l[i]$)</th>
<th>intermediate value loop round 6 ($Q_0$)</th>
<th>Assumption $k_i = 1$ first 8 bits → $0xB5$ ($= (0x5A &lt;&lt; 1) \land 0x1$)</th>
<th>selected bit: $n^8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td>0xD6F</td>
<td>1</td>
</tr>
<tr>
<td>0xB2</td>
<td>0x3E94</td>
<td>0x7DDA</td>
<td>1</td>
</tr>
<tr>
<td>0x56</td>
<td>0x1E3C</td>
<td>0x3CCE</td>
<td>0</td>
</tr>
<tr>
<td>0xC2</td>
<td>0x4434</td>
<td>0x892A</td>
<td>1</td>
</tr>
</tbody>
</table>

Second Attack: *Differential Power Analysis*

A toy example: The attacker asks the device to compute a series of multiplications of chosen $I[i]$ by a secret constant $k$ over 12 bits (CPA).

Assume the attacker knows the seven first bits of $k$: $[1, 0, 1, 1, 1, 0, 0]_2 = 0x5A$.

The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ($I[i]$)</th>
<th>intermediate value loop round 6 ($Q_0$)</th>
<th>Assumption $k_i = 1$ first 8 bits $\rightarrow 0xB5$ ($= (0x5A &lt;&lt; 1) \land 0x1$)</th>
<th>selected bit: $n^\circ 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td>0xD6F</td>
<td>1</td>
</tr>
<tr>
<td>0xB2</td>
<td>0x3E94</td>
<td>0x7DDA</td>
<td>1</td>
</tr>
<tr>
<td>0x56</td>
<td>0x1E3C</td>
<td>0x3CCE</td>
<td>0</td>
</tr>
<tr>
<td>0xC2</td>
<td>0x4434</td>
<td>0x892A</td>
<td>1</td>
</tr>
</tbody>
</table>

... 500-1000 computations

over 16 bits,

- average Hamming weight of the "red ones": 8.5;
- average Hamming weight of the "black ones": 7.5.
Second Attack: *Differential Power Analysis*

A toy example: The attacker asks the device to compute a series of multiplications of chosen $I[i]$ by a secret constant $k$ over 12 bits (CPA).

Assume the attacker knows the seven first bits of $k: [1, 0, 1, 1, 0, 0, 0]_2 = 0x5A$.

The attacker selects one bit of the intermediate value computed:

<table>
<thead>
<tr>
<th>Chosen value ($I[i]$)</th>
<th>intermediate value loop round 6 ($Q_0$)</th>
<th>Assumption $k_i = 1$ first 8 bits $\rightarrow 0xB5$ ($= (0x5A \ll 1) \land 0x1$)</th>
<th>selected bit: $n^8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x25</td>
<td>0xD02</td>
<td>0x1A29</td>
<td>0</td>
</tr>
<tr>
<td>0x13</td>
<td>0x6AE</td>
<td>0xD6F</td>
<td>1</td>
</tr>
<tr>
<td>0xB2</td>
<td>0x3E94</td>
<td>0x7DDA</td>
<td>1</td>
</tr>
<tr>
<td>0x56</td>
<td>0x1E3C</td>
<td>0x3CCE</td>
<td>0</td>
</tr>
<tr>
<td>0xC2</td>
<td>0x4434</td>
<td>0x892A</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

500-1000 computations

over 16 bits,

- average Hamming weight of the "red ones": 8.5;
- average Hamming weight of the "black ones": 7.5.

When the device processes the selected bit ($n^8$), if the assumption on $k_i$ is correct, the "red ones" need more electric power than the "black ones".
Second Attack: *Differential Power Analysis*

The attacker collects the $m$ power traces of the $m$ computations $T_1...m[j]$. Power trace of a "red one":

![Graph showing power trace over time](image_url)
Second Attack: *Differential Power Analysis*

The attacker collects the $m$ power traces of the $m$ computations $T_{1...m[j]}$. Power trace of a "black one":

![Graph showing power trace over time](image)
Second Attack: *Differential Power Analysis*

Both power traces:
Second Attack: \textit{Differential Power Analysis}

The attacker now computes the average trace for the red ones and the black ones, and the difference between both average traces:

\[
\Delta_D[j] = \frac{\sum_{i=1}^{m} D(P_i, s) \cdot T_i[j]}{\sum_{i=1}^{m} D(P_i, s)} - \frac{\sum_{i=1}^{m} (1 - D(P_i, s)) \cdot T_i[j]}{\sum_{i=1}^{m} (1 - D(P_i, s))}
\]

\[
\approx 2 \cdot \left(\frac{\sum_{i=1}^{m} D(P_i, s) \cdot T_i[j]}{\sum_{i=1}^{m} D(P_i, s)} - \frac{\sum_{i=1}^{m} T_i[j]}{m}\right).
\]

with:

- \(D(P_i, s) = \text{bit 8 of 0xB5 \cdot I[i]}\)
- \(T_{1...m}[j]\)
Second Attack: *Differential Power Analysis*

The attacker now computes the average trace for the red ones and the black ones, and the difference between both average traces:

→ At the time the selected bit is processed, one can see a peak: good guess!

(The eight\(^{th}\) bit of the secret scalar is 1!)
Second Attack: **Differential Power Analysis**

We now do the same with the assumption of a bit equal to 0 (same computations as previously, $0xB4$ instead of $0xB5$):

→ Only noise! Wrong guess!
(However, the attacker wins the game anyway: if the eight$^{th}$ bit of the secret scalar is not 0, it is 1 !)
In this exemple quoted form Coron, a smart card computes an elliptic curve scalar multiplication $d \cdot P$ with the Left-to-right-Double-and-add algorithm, and the attacker guesses the second bit to be processed.

**Fig. 2.** Simulated correlation function $g(t)$ between the points $4P_i$ and power consumption $C_i(t)$ when $d_{i-2} = 1$. No peak is observed since the points $4P_i$ are never computed by the card.
DPA in pictures, Coron, CHES 99

Fig. 1. Simulated correlation function $g(t)$ between the points $4P_i$ and power consumption $C_i(t)$ when $d_{\ell-2} = 0$. A peak is observed corresponding to the computation of $4P_i$ inside the card.

In this exemple quoted form Coron, a smart card computes an elliptic curve scalar multiplication $d \cdot P$ with the Left-to-right-Double-and-add algorithm, and the attacker guesses the second bit to be processed.
Analysis of the algorithms: **Double-and-Add-Always**

<table>
<thead>
<tr>
<th>round $i$:</th>
<th>$i = l - 2$</th>
<th>$i = l - 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{l-1} = 1$</td>
<td>$k_{l-2}$</td>
<td>$k_{l-3}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table:** First loop rounds of Algorithm 19, **Double-and-Add-Always**
## Analysis of the algorithms: Montgomery’s Binary Ladder

Montgomery’s binary ladder

<table>
<thead>
<tr>
<th>round $i$</th>
<th>$i = l - 2$</th>
<th>$i = l - 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{l-1} = 1$</td>
<td>$k_{l-2}$</td>
<td>$k_{l-3}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table**: First loop rounds of Algorithm 20, Montgomery’s binary ladder
Outline

1. Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2. Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis

3. Conclusion
Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- Randomization of the private exponent
Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- **Randomization of the private exponent**

An elliptic curve over $\mathbb{F}_{2^m}$ or $\mathbb{F}_p$ is a finite set of points, whose number of points is denoted $\#\mathcal{E}$. The order of a point divides $\#\mathcal{E}$.

$$Q = d \cdot P = (d + k \cdot \#\mathcal{E}) \cdot P, \forall k \in \mathbb{Z}.$$ 

(Indeed, one has: $\#\mathcal{E} \cdot P = \mathcal{O}$).
Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- **Randomization of the private exponent**
- **Blinding the point $P$**
Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- **Randomization of the private exponent**

- **Blinding the point $P$**

One adds a random point $R$ to the base point $P$, whose multiple $S = d \cdot R$ is known in advance.

$\rightarrow$ Point used in the operation $P' = P + R$

Variant:

$\rightarrow R \leftarrow (-1)^b 2R, \quad S \leftarrow (-1)^b 2S, \quad (b \text{ is a random bit}).$
Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- *Randomization of the private exponent*

- *Blinding the point P*

- *Randomized projective coordinates*
Randomization

The DPA attack works by averaging traces. A natural way to nail these average traces is to randomize the values processed.

- **Randomization of the private exponent**

- **Blinding the point $P$**

- **Randomized projective coordinates**

For one point in affine coordinates $(x, y)$, there is a huge quantity of projective points $(X, Y, Z)$ corresponding such as $(x = X/Z, y = Y/Z^2)$. 
Outline

1 Problematic
   - Cryptographic Protocols: what about the Group?
   - Elliptic Curve Point Operations
   - Elliptic Curve Scalar Multiplication

2 Side-channel Attacks
   - First Attack: Simple Power Analysis
   - How to thwart SPA?
   - Second Attack: Differential Power Analysis
   - How to thwart DPA?
   - Synthesis

3 Conclusion
The DPA attack:

- requires more power form the attacker (multiple Chosen Plaintext Attack and access to the device);
The DPA attack:

- requires more power from the attacker (multiple Chosen Plaintext Attack and access to the device);
- is also sensitive to the noise;
  - improvements by CPA, Template Attacks...
  - Hardware counter-measures;
The DPA attack:

- requires more power from the attacker (multiple Chosen Plaintext Attack and access to the device);
- is also sensitive to the noise;
  - improvements by CPA, Template Attacks...
  - Hardware counter-measures;
- Moreover, the counter-measures are costly.
Conclusion

Elliptic curve cryptographic main operation: the Scalar Multiplication; much faster than fast exponentiation, smaller data for the same level of security;

Side channel attacks are real threats!

SPA attack;

DPA attack;

Other attacks every day... New counter-measures...

Jean-Marc ROBERT Team DALI/LIRMM, Université de Perpignan, France
Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!
Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!
- SPA attack;
Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!
- SPA attack;
- DPA attack;
Conclusion

- Elliptic curve cryptographic main operation: the Scalar Multiplication;
- Much faster than fast exponentiation, smaller data for the same level of security;
- Side channel attacks are real threats!
- SPA attack;
- DPA attack;
- Other attacks every day... New counter-measures...
Thank you for your attention,

Any questions?

mail: jean-marc.robert@univ-perp.fr

home page: http://perso.univ-perp.fr/jeanmarc.robert
D. J. Bernstein.

D.J. Bernstein and Lange T. (eds).
eBACS: ECRYPT Benchmarking of Cryptographic Systems.
accessed May 25th, 14.

Jean-Sébastien Coron.
Resistance against differential power analysis for elliptic curve cryptosystems.

Mike Hamburg.
Fast and compact elliptic-curve cryptography.
http://eprint.iacr.org/.

Paul C. Kocher, Joshua Jaffe, and Benjamin Jun.
Differential power analysis.

Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis.

A. Langley.
C25519 code.
http://code.google.com/p/curve25519-donna/.

P. Longa and C. H. Gebotys.
Efficient Techniques for High-Speed Elliptic Curve Cryptography.

Christophe Nègre and Jean-Marc Robert.
Impact of optimized field operations $ab$, $ac$ and $ab + cd$ in scalar multiplication over binary elliptic curve.
Conclusion


Agustin Dominguez Oviedo.

*On Fault-based Attacks and Countermeasures for Elliptic Curve Cryptosystems.*

Jonathan Taverne, Armando Faz-Hernández, Diego F. Aranha, Francisco Rodríguez-Henríquez, Darrel Hankerson, and Julio López.

Speeding scalar multiplication over binary elliptic curves using the new carry-less multiplication instruction.