Efficient and secure modular operations using the Polynomial Modular Number System Part 2

Jérémy Marrez

Team ALMASTY
Laboratory of Computer Sciences of Paris 6, LIP6
Sorbonne University

April 19th 2018

Context : Efficient and secure modular operations using the Polynomial Modular Number System

Problematic

- Side channel attacks (SCA) use the leakage of information during the execution of a cryptographic protocol (execution time, power consumption or electromagnetic emission) in order to totally or partially recover the secret.
- SCA have proven to be efficient in ECC : countermeasures should be included in the implementation of the scalar multiplication in ECC which, from a point P over a public elliptic curve, and a private integer k, computes kP.
- The classical double and add method is not resistant to SCA. The Montgomery ladder and its variant are more resistant but can still be attacked.

Idea

- Introduce randomization at the arithmetical level.

We want to use a random representation of point P each time this point is used during the scalar multiplication algorithm.

- to ensure the resistance to SCA and specific point attacks.

Approach

- ➤ The coordinates of P are represented in Polynomial Modular Number System, known to speed up modular arithmetic
  - ✓ We use the redundancy of PMNS to randomize the coordinates.
- ➤ We propose to randomise the scalar multiplication by the two following ways :
  - Randomisation of each initial coordinate of P using a suitable conversion procedure, ensuring the resistance to SCA and specific point attacks.
  - Randomisation of each multiplication between two elements in PMNS representation to be more resistant to SCA.

Properties about the representations in PMNS

The set of all representations of the integer a in the PMNS $\mathfrak{B} = (p, n, \gamma, \rho)$, noted $a_{\mathfrak{B}}$ is defined as

$$A \in a_{\mathfrak{B}} \iff \begin{cases} A(\gamma) \equiv a \pmod{p}, \\ \deg A < n, \\ \|A\|_\infty < \rho \end{cases}$$

with $\|.\|_\infty$ the infinity norm.

- ➤ How to ensure that operations are stable within the system ?

Ensuring stability

➤ Operations must be stable on the chosen system $\mathfrak{B} = (p, n, \gamma, \rho)$

Let $A \equiv a_{\mathfrak{B}}, B \equiv b_{\mathfrak{B}}$, then $A.B(\gamma) = ab \mod p$.

To compute a representation $(ab)_{\mathfrak{B}}$, two reductions are required :

- *External reduction* : $A \cdot B$ might not be in $\mathcal{B}$ since $\deg(A \cdot B) \geq n$. The *external reduction polynomial* E is used to reduce the degree.
  $\Rightarrow V = A.B \mod E \in \mathbb{Z}[X]$

  ✓ $\deg(V) < n$

- *Internal reduction* : V with $\deg(V) < n$ might not be a representation of ab (mod p) in $\mathcal{B}$, if $\|V\|_\infty \geq \rho$.
  $\Rightarrow S = \mathrm{RedCoeff}(V) \in \mathbb{Z}[X]$

  ✓ $\|S\|_\infty \leq \rho$, then $S = ab_{\mathfrak{B}}$

Standard algorithm

Algorithm 1 OpMod(A,B,$\mathfrak{B}$), Modular operation on PMNS $\mathfrak{B}$

```
Require: $A, B \equiv a_{\mathfrak{B}}, b_{\mathfrak{B}}$ with $\mathfrak{B} = (p, n, \gamma, \rho, E)$
Ensure: $S \equiv (a \cdot b)_{\mathfrak{B}}$, i.e. $S(\gamma) \equiv A(\gamma).B(\gamma) \pmod{p}$
    $V \leftarrow A.B \mod E$
    $S \leftarrow$ RedCoeff(V) via a Babaï-like method
    return $S$
```

Summary

- A Babaï-like method for the internal reduction
- Randomisation of the input data
- How many representations ? Bound on digits ? Collisions ?
- Randomisation of the multiplication
- Complexity and perspectives

Summary

- A Babaï-like method for the internal reduction
- Randomisation of the input data
- How many representations ? Bound on digits ? Collisions ?
- Randomisation of the multiplication
- Complexity and perspectives

A method without condition

Unlike the Montgomery-like algorithm, the Babaï-like algorithm

- does not require to maintain the elements of $\mathbb{Z}/p\mathbb{Z}$ in another representation domain,
- does not need a polynomial M with particular properties.

The representations in the PMNS system are considered as vectors.

Using the Euclidean lattice $\mathfrak{L}_{\mathcal{B}}$ associated with the PMNS $\mathcal{B}$

The algorithm runs on the lattice $\mathfrak{L}_{\mathcal{B}}$ composed of polynomials of degree at most $n-1$ with γ as root.

$$\mathfrak{L}_{\mathcal{B}} = \{A(X) \in \mathbb{Z}[X], \text{ such that : } deg(A) < n \text{ and } A(\gamma) \equiv 0 \mod p\}.$$

If $V(\gamma) \equiv U(\gamma) \mod p$, then $V \equiv U \mod \mathcal{L}_{\mathcal{B}}$

➤ $V - U \in \mathcal{L}_{\mathcal{B}}$

From $V \in \mathbb{Z}^n$, the Babai-like algorithm finds a vector U such that $V \equiv U \mod \mathcal{L}_{\mathcal{B}}$.

➤ $U \in V - \mathfrak{L}_{\mathcal{B}} := \{V - W \mid W \in \mathfrak{L}_{\mathcal{B}}\}$, with $\|U\|_\infty \leq \|V\|_\infty$

RedCoeff - Babaï like

$$\mathbf{A} = \begin{pmatrix} p & 0 & \dots & \dots & 0 & 0 \\ -\gamma & 1 & \dots & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \dots & -\gamma & 1 & \dots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & -\gamma & 1 \end{pmatrix}$$

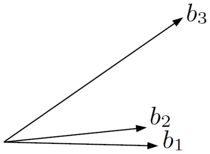


FIGURE – Reduction via Babaï in dimension 3. The selected hyperplanes are green.

Data : $B = \{b_i, 1 \leq i \leq n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$

RedCoeff - Babaï like

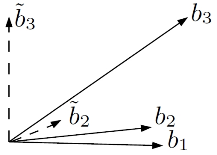


FIGURE – Reduction via Babaï in dimension 3. The selected hyperplanes are green.

Data : $B = \{b_i, 1 \leq i \leq n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$
$\widetilde{B} = \{\widetilde{b_i}, 1 \leq i \leq n\}$ the Gram-Schmidt base obtained from B

RedCoeff - Babaï like

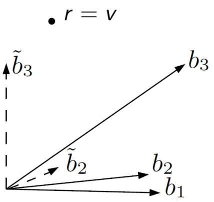


FIGURE – Reduction via Babaï in dimension 3. The selected hyperplanes are green.

Require: $V \in \mathbb{Z}[X]$ with $\deg(V) < n$, $\mathcal{B} = (p, n, \gamma, \rho, E)$

Data : $B = \{b_i, 1 \le i \le n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$

$\widetilde{B} = \{\widetilde{b_i}, 1 \le i \le n\}$ the Gram-Schmidt base obtained from B

```
1: $r \leftarrow v$
```

RedCoeff - Babaï like

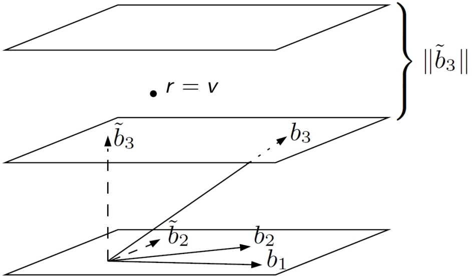


FIGURE – Reduction via Babaï in dimension 3. The selected hyperplanes are green.

Require: $V \in \mathbb{Z}[X]$ with $\deg(V) < n$, $\mathcal{B} = (p, n, \gamma, \rho, E)$

Data : $B = \{b_i, 1 \le i \le n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$

$\widetilde{B} = \{\widetilde{b_i}, 1 \le i \le n\}$ the Gram-Schmidt base obtained from B

```
1: $r \leftarrow v$
2: for $i = 1$ to $n$ do
```

RedCoeff - Babaï like

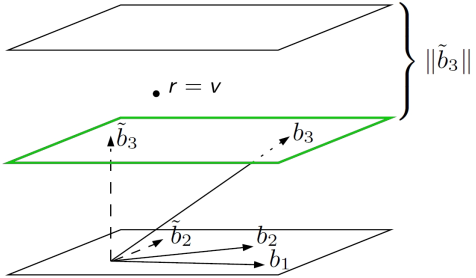


FIGURE – Reduction via Babaï in dimension 3. The selected hyperplanes are green.

Require: $V \in \mathbb{Z}[X]$ with $\deg(V) < n$, $\mathcal{B} = (p, n, \gamma, \rho, E)$

Data : $B = \{b_i, 1 \leq i \leq n\}$ the LLL-reduced base of $\mathfrak{L}_\mathcal{B}$

$\widetilde{B} = \{\widetilde{b_i}, 1 \leq i \leq n\}$ the Gram-Schmidt base obtained from B

```
1: $r \leftarrow v$
2: for $i = 1$ to $n$ do
3:     $c \leftarrow \lfloor < r, \widetilde{b}_{n-i+1} > / \|\widetilde{b}_{n-i+1}\|^2 \rceil$
```

RedCoeff - Babaï like

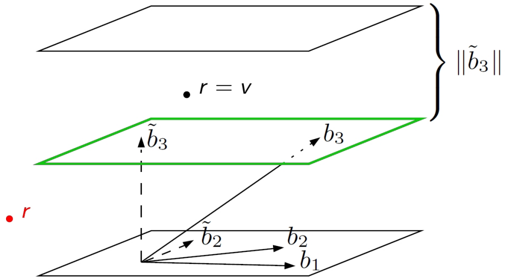


FIGURE – Reduction via Babaï in dimension 3. The selected hyperplanes are green.

Require: $V \in \mathbb{Z}[X]$ with $\deg(V) < n$, $\mathcal{B} = (p, n, \gamma, \rho, E)$

Data : $B = \{b_i, 1 \leq i \leq n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$

$\widetilde{B} = \{\widetilde{b_i}, 1 \leq i \leq n\}$ the Gram-Schmidt base obtained from B

```
1: $r \leftarrow v$
2: for $i = 1$ to $n$ do
3:     $c \leftarrow \lfloor < r, \widetilde{b}_{n-i+1} > / \|\widetilde{b}_{n-i+1}\|^2 \rceil$
4:     $r \leftarrow r - c \times b_{n-i+1}$
```

RedCoeff - Babaï like

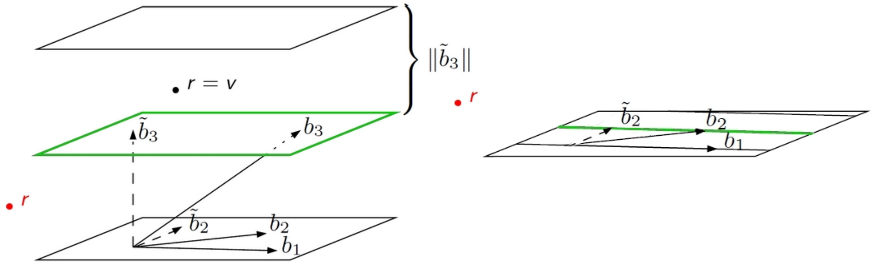


FIGURE – Reduction via Babaï in dimension 3. The selected hyperplanes are green.

Require: $V \in \mathbb{Z}[X]$ with $\deg(V) < n$, $\mathcal{B} = (p, n, \gamma, \rho, E)$

Data : $B = \{b_i, 1 \leq i \leq n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$

$\widetilde{B} = \{\widetilde{b_i}, 1 \leq i \leq n\}$ the Gram-Schmidt base obtained from B

Ensure: $r \in \mathcal{B}$ **such that** $\|R\|_\infty \leq \|V\|_\infty$ **and** $r(\gamma) = V(\gamma) \mod p$

```
1: $r \leftarrow v$
2: for $i = 1$ to $n$ do
3:     $c \leftarrow \lfloor < r, \widetilde{b}_{n-i+1} > / \|\widetilde{b}_{n-i+1}\|^2 \rceil$
4:     $r \leftarrow r - c \times b_{n-i+1}$
5: end for
6: return $r$
```

The output of the algorithm represents the same element modulo p as the input and lies in the rectangle

$$\left\{ \sum a_i \widetilde{D}_i \mid |a_i| \leq \frac{1}{2} \right\}.$$

Theorem

If ρ is such that :

$$\rho \geqslant \tfrac{1}{2} \, 2^{\frac{3n-1}{2}} \, p^{1/n},$$

then the algorithm computes R such that $\|R\|_\infty < \rho$ (i.e $R \in \mathcal{B}$).

This lower bound does not depend on the polynomial E unlike the Montgomery-like internal reduction.

Summary

- A Babaï-like method for the internal reduction
- Randomisation of the input data
- How many representations ? Bound on digits ? Collisions ?
- Randomisation of the multiplication
- Complexity and perspectives

Random polynomial generation

$$\begin{array}{c} z \in \mathbb{Z} \\ \text{chosen during the} \\ \text{PMNS generation process} \end{array} \rightarrow \text{randomPoly}(z) = \begin{array}{c} Z \in \mathbb{Z}[X] \\ \|Z\|_\infty \leq z,\ z \in \mathbb{N} \\ \deg Z < n \end{array}$$

We consider this function as safe.

z define the number of representations for the randomization

The integer z define the minimum number of distinct representations of any element of $\mathbb{Z}/p\mathbb{Z}$ in $\mathcal{B}$.

- ➤ There are exactly $(2\,z+1)^n$ polynomials Z as defined above.

Questions :

- What bound ρ to guarantee there are at least $(2\,z+1)^n$ distinct representations of any element of $\mathbb{Z}/p\mathbb{Z}$ in the system ?
- How to use Z to reach all those representations ?

The randomisation of the Babaï-like method

Let the lattice $\mathfrak{L}_{\mathcal{B}}$ and its LLL-reduced base D. The randomization is based on two random vectors :

Let V be the ***mask* vector** : n random coefficients to generate a linear combination of the elements of D

- The resulting vector is added to the input at the beginning of the algorithm to randomize computations during execution. It does not affect the output.

Let Z be the ***shift* vector** : for each dimension of the Euclidean space, the element in D is multiplied by the corresponding random coefficient of Z

- Generate an additional translation during the algorithm and randomize the output.

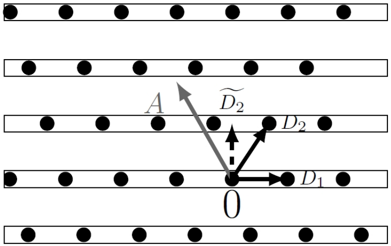
Algorithm 2 Conversion from classical representation to PMNS

```
Require: $a \in \mathbb{Z}/p\mathbb{Z}$   and   $\mathcal{B} = (p,\ n,\ \gamma,\ \rho, E)$
Ensure: $A \equiv a_{\mathcal{B}}$
Data : $P_i \equiv (\rho^i)_{\mathcal{B}}$, for $i = 0,\ \ldots, n-1$
  1: $b \leftarrow (a_{n-1}, ..., a_0)_\rho$ # radix-$\rho$ decomposition of $a$
  2: $U \leftarrow \sum_{i=0}^{n-1} b_i\, P_i$
  3: $A \leftarrow$ RedCoeff($U$)
  4: return $A$
```

Algorithm 3 Randomised conversion from classical representation to PMNS via Babaï

```
Require: $a \in \mathbb{Z}/p\mathbb{Z}$ and $\mathcal{B} = (p, n, \gamma, \rho, E)$
Ensure: $A(\gamma) \equiv a \pmod{p}$
Data : $P_i \equiv (\rho^i)_{\mathcal{B}}$, for $i = 0, \ldots, n-1$
       $D = \{D_i, 1 \le i \le n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$
       $\widetilde{D} = \{\widetilde{D_i}, 1 \le i \le n\}$ the Gram-Schmidt base
       $v \in \mathbb{N}$, $z \in \mathbb{N}$
 1: $V \leftarrow$ randomPoly($v$)
 2: $Z \leftarrow$ randomPoly($z$)
 3: $b \leftarrow (a_{n-1}, ..., a_0)_\rho$ # radix-$\rho$ decomposition of $a$
 4: $T \leftarrow \sum_{i=0}^{n-1} b_i P_i$
 5: $A \leftarrow T + \sum_{i=0}^{n-1} v_i D_{i+1}$
 6: for $i = 1$ to $n$ do
 7:     $c \leftarrow \lfloor < A, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rceil + z_{n-i}$
 8:     $A \leftarrow A - c \times D_{n-i+1}$
 9: end for
10: return $A$
```

Two executions of the Babai-like algorithm



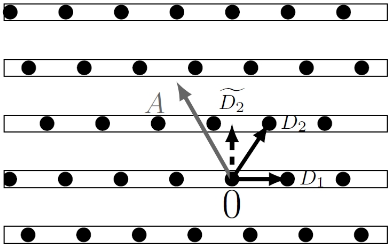


FIGURE – Reduction via Babaï in dimension 2. The selected hyperplanes in LB are shaded or surrounded.

Non randomized

Randomized

$z = 1 \rightarrow$ shift $Z = (-1, 1)$

Two executions of the Babai-like algorithm

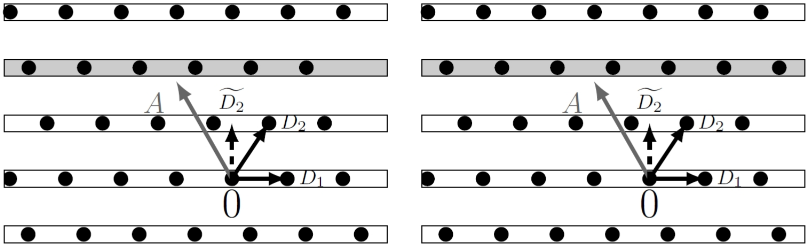


FIGURE – Reduction via Babaï in dimension 2. The selected hyperplanes in LB are shaded or surrounded.

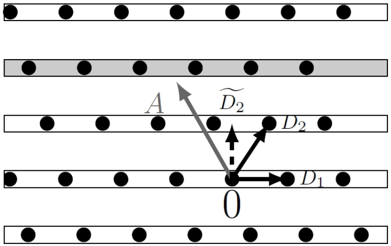
Non randomized

Randomized

$z = 1 \rightarrow$ shift $Z = (-1, 1)$

$$c \leftarrow \lfloor < A, \widetilde{D}_2 > / \|\widetilde{D}_2\|^2 \rceil$$

Two executions of the Babai-like algorithm



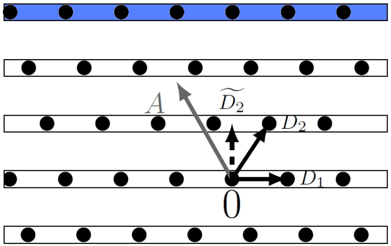


FIGURE – Reduction via Babaï in dimension 2. The selected hyperplanes in LB are shaded or surrounded.

Non randomized

Randomized

$z = 1 \rightarrow$ shift $Z = (-1, \underline{1})$

$$c \leftarrow \lfloor < A, \widetilde{D}_2 > / \|\widetilde{D}_2\|^2 \rceil + Z_2$$

Two executions of the Babai-like algorithm

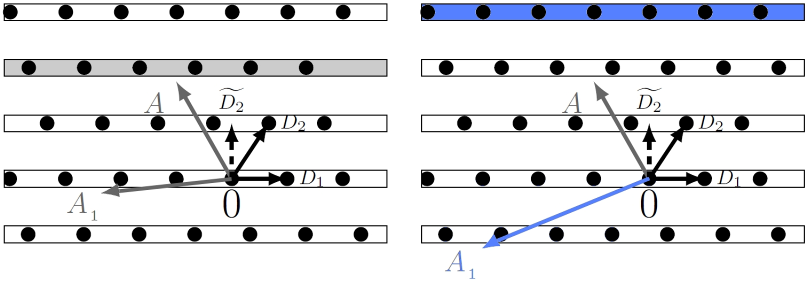


FIGURE – Reduction via Babaï in dimension 2. The selected hyperplanes in LB are shaded or surrounded.

Non randomized

Randomized

$z = 1 \rightarrow$ shift $Z = (-1, \underline{1})$

$$A_1 \leftarrow A - c \times D_2$$

Two executions of the Babai-like algorithm

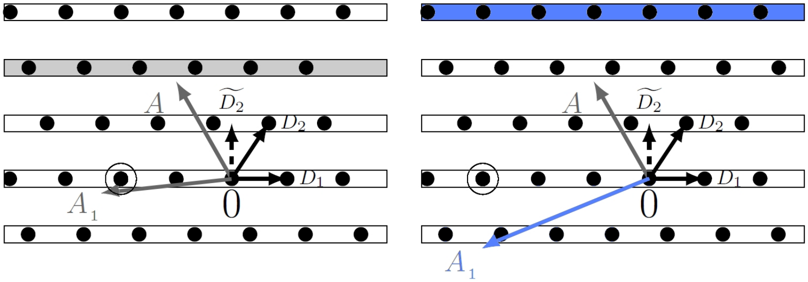


FIGURE – Reduction via Babaï in dimension 2. The selected hyperplanes in LB are shaded or surrounded.

Non randomized

Randomized

$z = 1 \rightarrow$ shift $Z = (-1, \underline{1})$

$$c \leftarrow \lfloor < A_1, \widetilde{D}_1 > / \|\widetilde{D}_1\|^2 \rceil$$

Two executions of the Babai-like algorithm

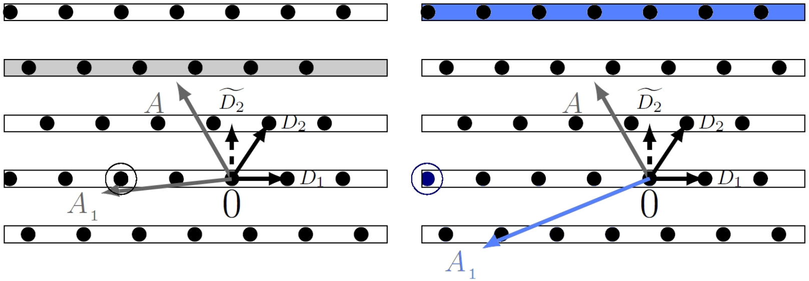


FIGURE – Reduction via Babaï in dimension 2. The selected hyperplanes in LB are shaded or surrounded.

Non randomized

Randomized

$z = 1 \rightarrow$ shift $Z = (\underline{-1}, 1)$

$$c \leftarrow \lfloor < A_1, \widetilde{D}_1 > / \|\widetilde{D}_1\|^2 \rceil + Z_1$$

Two executions of the Babai-like algorithm

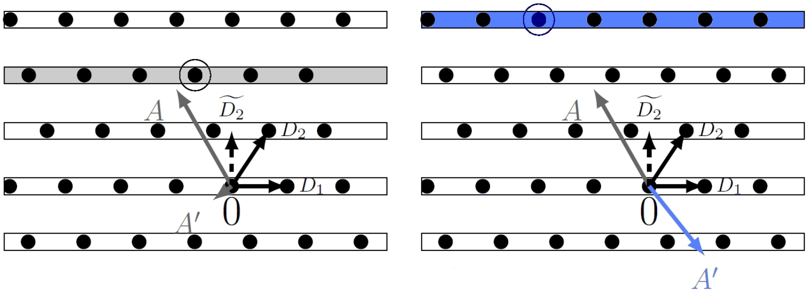


FIGURE – Reduction via Babaï in dimension 2. The selected hyperplanes in LB are shaded or surrounded.

Non randomized

Randomized

$z = 1 \rightarrow$ shift $Z = (\underline{-1}, 1)$

$$A' \leftarrow A_1 - c \times D_1$$

Summary

- A Babaï-like method for the internal reduction
- Randomisation of the input data
- How many representations ? Bound on digits ? Collisions ?
- Randomisation of the multiplication
- Complexity and perspectives

Theorem

Let :

- $\mathcal{B} = (p, n, \gamma, \rho, E)$ a PMNS.
- $a \in \mathbb{Z}/p\mathbb{Z}$
- v and $z \in \mathbb{Z}$, the inputs of the randomPoly procedure (for the mask and the shift polynomial).

We consider $\mathcal{B}$, a, v and z as the inputs and data of Algorithm 3.

If ρ satisfies

$$\rho \geq \left(\frac{1}{2} + z\right)\left(2^{\frac{3n-1}{2}} p^{1/n}\right),$$

then Algorithm 3 can generate $(2z + 1)^n$ distinct outputs, all representing a and belonging to the PMNS $\mathcal{B} = (p, n, \gamma, \rho, E)$.

The mask vector has no effect on the output

- Replacing T by $T + V$ for $V \in \mathfrak{L}$ has no effect on the output.

Input T : $(A_i)_{0 \leq i \leq n}, (c_i)_{0 \leq i \leq n}$ the values of A and c before and at each step of the loop.

Input $T + D_i$: $(A'_i)_{0 \leq i \leq n}$ and $(c'_i)_{0 \leq i \leq n}$.

Here $A_0 = T = T + D_i - D_i = {A'}_0 - D_i$.
By induction, for $j \leq n - i$,

$$< D_{n-j+1}, {A'}_{j-1} - D_i > = < D_{n-j+1}, {A'}_{j-1} >$$
$$\Rightarrow c_j = {c'}_j$$

The mask vector has no effect on the output

Then,

$$A_j = A_{j-1} - c_j D_{n-j+1} = A'_{j-1} - D_i - c'_j D_{n-j+1} = A'_j - D_i.$$

Then, at step $j = n - i + 1$, we obtain

$$c_j = \frac{< A_{j-1}, \widetilde{D}_i >}{\|\widetilde{D}_i\|^2} = \frac{< A'_{j-1} - D_i, \widetilde{D}_i >}{\|\widetilde{D}_i\|^2} = \frac{< A'_{j-1}, \widetilde{D}_i >}{\|\widetilde{D}_i\|^2} - 1 = c'_j - 1.$$

For this j, this implies

$$\begin{aligned} A_j &= A_{j-1} - c_j D_i \\ &= A'_{j-1} - D_i - (c'_j - 1) D_i \\ &= A'_{j-1} - c'_j D_i = A'_j \end{aligned}$$

✓ The output is identical.

Giving a bound for ρ

The Gram-Schmidt vectors are orthogonal, then every $u \in \mathbb{R}^n$ can be written as

$$u = \sum \frac{< \widetilde{D}_i, u >}{\|\widetilde{D}_i\|^2} \cdot \widetilde{D}_i.$$

At the i-th step, the algorithm removes to A the fixed vector $(\lfloor < A, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rfloor + z_{n-i}) \cdot D_{n-i+1}$, the output lies in the rectangle

$$\left\{ \sum a_i \widetilde{D}_i \text{ with } |a_i| \leq \frac{1}{2} + z \right\},$$

with z the bound of the shift polynomial. Then

$$\|A\|^2 \leq \left(\frac{1}{2} + z \right)^2 \left(\sum_{i=1}^{n} \|\widetilde{D}_i\| \right)^2.$$

Giving a bound for ρ

Since the base D is an LLL-reduced basis, for $1 \leq i \leq n$, $\|\widetilde{D}_i\| \leq 2^{\frac{n-i}{2}} \|\widetilde{D}_n\|$, and

$$\begin{aligned} \|A\| &\leq \left(\frac{1}{2} + z\right) \left(\sum_{i=1}^{n} 2^{\frac{n-i}{2}} \|\widetilde{D}_n\|\right), \\ &\leq \left(\frac{1}{2} + z\right) \left(2^{\frac{n-1}{2}} \|\widetilde{D}_n\|\right). \end{aligned}$$

A bound on $\widetilde{D}_n$ is given by $\|\widetilde{D}_n\| \leq 2^n p^{1/n}$. Hence,

$$\|A\|_\infty \leq \|A\| \leq \left(\frac{1}{2} + z\right) \left(2^{\frac{3n-1}{2}} p^{1/n}\right).$$

✓ The bound on ρ is proved.

No collision

For the input A and shift vector Z, for i from $n-1$ to 0, the i-th coordinate z_i of Z represents the additional translation $-z_i \, D_{i+1}$ performed on A.

To compute the i-th coordinate of the result, the algorithm subtracts a point in the hyperplane of dimension i and at a distance of $z_i \, \|\widetilde{D}_{i+1}\|$ from the point $\lfloor < A, \widetilde{D}_{i+1} > / \|\widetilde{D}_{i+1}\|^2 \rceil D_{i+1}$ in the lattice.

- ➤ the choice of one hyperplane over another induces a different result.

✓ The algorithm returns distinct outputs from two distinct shift vectors.

Summary

- A Babaï-like method for the internal reduction
- Randomisation of the input data
- How many representations ? Bound on digits ? Collisions ?
- Randomisation of the multiplication
- Complexity and perspectives

Algorithm 4 PMNS Multiplication via Babaï - Randomised Version

```
Require: $\mathcal{B} = (p, n, \gamma, \rho, E)$ and $A, B \in \mathcal{B}$
Ensure: $R \in \mathcal{B}$ such that $R(\gamma) = A(\gamma)B(\gamma) \mod p$
Data : $P_i \equiv (\rho^i)_{\mathcal{B}}$, for $i = 0, \ldots, n-1$
       $D = \{D_i, 1 \leq i \leq n\}$ the LLL-reduced base of $\mathfrak{L}_{\mathcal{B}}$
       $\widetilde{D} = \{\widetilde{D_i}, 1 \leq i \leq n\}$ the Gram-Schmidt base
       $v \in \mathbb{N}$, $z \in \mathbb{N}$
 1: $V \leftarrow$ randomPoly($v$)
 2: $Z \leftarrow$ randomPoly($z$)
 3: $J \leftarrow \sum_{i=0}^{n-1} v_i D_{i+1}$
 4: $B' \leftarrow B + J$
 5: $R \leftarrow A \times B' \mod E$
 6: for $i := 1$ to $n$ do
 7:     $c \leftarrow \lfloor < R, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rceil + z_{n-i}$
 8:     $R \leftarrow R - c \times D_{n-i+1}$
 9: end for
10: return $R$
```

Theorem

Let :

- $\mathcal{B} = (p, n, \gamma, \rho, E)$ a PMNS.
- $A, B \in \mathcal{B}$
- v and $z \in \mathbb{Z}$, the inputs of the randomPoly procedure (for the mask and the shift polynomial).

We consider $\mathcal{B}$, A, B, v and z as the inputs and data of Algorithm 4.

If ρ satisfies

$$\rho \geq \left(\frac{1}{2} + z\right)\left(2^{\frac{3n-1}{2}} p^{1/n}\right),$$

then Algorithm 4 can generate $(2\,z + 1)^n$ distinct outputs representing $A(\gamma)B(\gamma)$ mod p in the PMNS $\mathcal{B} = (p, n, \gamma, \rho, E)$.

Summary

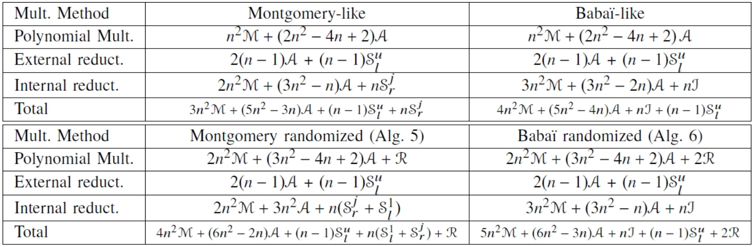
- A Babaï-like method for the internal reduction
- Randomisation of the input data
- How many representations ? Bound on digits ? Collisions ?
- Randomisation of the multiplication
- Complexity and perspectives

$\mathcal{M}$ and $\mathcal{A}$: multiplication and sum of two w-bits integers

$\mathcal{I}$: division by an integer of $2\,w\,\lfloor \log_2(n) \rfloor$ bits

$\mathcal{R}$: cost of one call to the `randPoly` function.

We also respectively denote $\mathcal{S}_l^i$ and $\mathcal{S}_r^i$ a left shift and a right shift of i bits.



Mult. Method	Montgomery-like	Babaï-like
Polynomial Mult.	$n^2\mathcal{M} + (2n^2 - 4n + 2)\mathcal{A}$	$n^2\mathcal{M} + (2n^2 - 4n + 2)\mathcal{A}$
External reduct.	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$
Internal reduct.	$2n^2\mathcal{M} + (3n^2 - n)\mathcal{A} + n\mathcal{S}_r^j$	$3n^2\mathcal{M} + (3n^2 - 2n)\mathcal{A} + n\mathcal{I}$
Total	$3n^2\mathcal{M} + (5n^2 - 3n)\mathcal{A} + (n-1)\mathcal{S}_l^u + n\mathcal{S}_r^j$	$4n^2\mathcal{M} + (5n^2 - 4n)\mathcal{A} + n\mathcal{I} + (n-1)\mathcal{S}_l^u$

Mult. Method	Montgomery randomized (Alg. 5)	Babaï randomized (Alg. 6)
Polynomial Mult.	$2n^2\mathcal{M} + (3n^2 - 4n + 2)\mathcal{A} + \mathcal{R}$	$2n^2\mathcal{M} + (3n^2 - 4n + 2)\mathcal{A} + 2\mathcal{R}$
External reduct.	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$
Internal reduct.	$2n^2\mathcal{M} + 3n^2\mathcal{A} + n(\mathcal{S}_r^j + \mathcal{S}_l^1)$	$3n^2\mathcal{M} + (3n^2 - n)\mathcal{A} + n\mathcal{I}$
Total	$4n^2\mathcal{M} + (6n^2 - 2n)\mathcal{A} + (n-1)\mathcal{S}_l^u + n(\mathcal{S}_l^1 + \mathcal{S}_r^j) + \mathcal{R}$	$5n^2\mathcal{M} + (6n^2 - 3n)\mathcal{A} + n\mathcal{I} + (n-1)\mathcal{S}_l^u + 2\mathcal{R}$

TABLE I
THEORETICAL COST OF OPERATIONS, WHERE $E(X) = X^n - \lambda$ WITH $\lambda = \pm 2^u$, $r = 2^j$.

Advantages

In classical binary representation, common countermeasures appear to be inefficient against Goubin's attack. It is possible to thwart this attack at the cost of an additional ECSM which must be done in addition to the common countermeasures.

Advantages of PMNS based randomized solutions :

- ➤ regardless the type of curve, this attack cannot be performed.
  - ✓ at least $(2z+1)^n$ distinct representatives of $0 \in \mathbb{Z}/p\mathbb{Z}$ which do not have special shapes.

  For z big enough, the attacker should not be able to exploit any information to perform this attack.
  - ✓ The only randomization of the conversion process using Montgomery or Babaï suffices to counter the Goubin's attack even if non-randomized multiplications are used later.
- ➤ it operates at arithmetic level.
- ✓ can be combined with other classical countermeasures (point blinding, scalar blinding) to randomize $\mathcal{P}$ and the intermediate points.

Conclusion

- ➤ We show for the first time how to use the redundancy of the PMNS to define arithmetical protections against DPA attacks
- ➤ We described how to randomize the inputs during the forward conversion to PMNS through two methods.
- ➤ We also gave two randomized modular multiplications in PMNS.
  - ✓ These methods can be used to apply classical countermeasures on the elliptic curve scalar multiplication.
- ➤ We showed that randomizing only the conversion process suffices to protect against Goubin's attack.

Perspectives

- These results are a first step in using randomization for arithmetic operations in PMNS. This work opens up new perspectives in the area of countermeasures for SCA attacks.
- A deeper study on practical efficiency and an exhaustive comparison with existing countermeasures will soon follow in order to establish the relevance of these methods.

Thank you !