An efficient Barrett reduction algorithm for Gaussian integer moduli

Presenter:
Dr. Malek Safieh, Security for Embedded Systems

Authors:
Malek Safieh, Fabrizio De Santis, and Andreas Furch
Introduction

- Gaussian integers are used in many applications, like Rivest-Shamir-Adleman (RSA), elliptic curve cryptography (ECC), post-quantum cryptography, error-correcting coding, and many other systems
  → All these applications can benefit from efficient modular arithmetic for Gaussian integers

- In my dissertation [1]: increased efficiency for ECC point multiplications using Montgomery arithmetic over Gaussian integers
  → Low complexity for the reduction with arbitrary Gaussian integer moduli [2]

- In [3]: more efficient reduction algorithms for Gaussian integer moduli of restricted form

Introduction

- Gaussian integers are used in many applications, like Rivest-Shamir-Adleman (RSA), elliptic curve cryptography (ECC), post-quantum cryptography, error-correcting coding, and many other systems
  ➔ All these applications can benefit from efficient modular arithmetic for Gaussian integers

- In my dissertation [1]: increased efficiency for ECC point multiplications using Montgomery arithmetic over Gaussian integers
  ➔ Low complexity for the reduction with arbitrary Gaussian integer moduli [2]

- In [3]: more efficient reduction algorithms for Gaussian integer moduli of restricted form

- In this work, a novel reduction algorithm for Gaussian integers based on Barrett’s concepts is derived:
  - Suitable for arbitrary Gaussian integer moduli, unlike algorithms from [3]
  - Provides equivalent computational complexity to the Montgomery reduction from [1, 2]
  - No need for Montgomery domain transformations

Introduction to Gaussian integers

- Subset of complex numbers \( x = a + bi, i = \sqrt{-1}, a, \) and \( b \) are integer numbers

- Naïve modulo function \( x \mod \pi = x - \left[ \frac{x \pi^*}{\pi \pi^*} \right] \cdot \pi \) [6]

- For \( p = \pi \pi^* \equiv 1 \mod 4 \), we have Gaussian integer fields \( \mathbb{G}_p \) isomorphic to prime fields \( \mathbb{F}_p \) [6]

- For \( n = cd, c \equiv d \equiv 1 \mod 4 \), \( \mathbb{G}_n \) is a Gaussian integer ring isomorphic to the ring over integer numbers \( \mathbb{Z}_n \) [1]

\[ G_{73} \text{ with } \pi = 8 + 3i, p = \pi \pi^* = 73 \]
Introduction to Gaussian integers

- Subset of complex numbers $x = a + bi, i = \sqrt{-1}$, $a$, and $b$ are integer numbers

- Naïve modulo function $\equiv x \mod \pi = x - \left[\frac{x\pi^*}{\pi\pi^*}\right] \cdot \pi$ [6]

- For $p = \pi\pi^* \equiv 1 \mod 4$, we have Gaussian integer fields $G_p$ isomorphic to prime fields $\mathbb{F}_p$ [6]

- For $n = cd, c \equiv d \equiv 1 \mod 4$, $G_n$ is a Gaussian integer ring isomorphic to the ring over integer numbers $\mathbb{Z}_n$ [1]

The division for the naïve modulo reduction is expensive. More efficient modulo reduction is needed!
Motivation for efficient Gaussian integer modular arithmetic with ECC system

• Elliptic curve cryptography (ECC) is suitable for resource-constrained devices (shorter keys than RSA)

• The ECC trapdoor function is the elliptic curve scalar point multiplication (PM)

• Consider the key $k$, the length of the key in bits $r$, and a point on the curve $P$, then the PM can be calculated using the Horner scheme as

$$k \cdot P = \sum_{j=0}^{r-1} k_j 2^j \cdot P = 2(\cdots 2(2k_{r-1} + k_{r-2}P) + \cdots) + k_0 P$$

• It was shown in [4,5] that representing the key with non-binary base $\tau$ can reduce the computational complexity of the PM. Let $\kappa$ be the integer $k$ converted to the base $\tau$, the PM can be calculated as

$$\kappa \cdot P = \sum_{j=0}^{l-1} \kappa_j \tau^j \cdot P = \tau(\cdots \tau(\tau_k \cdot P + \cdots) + \kappa_0 P$$

Motivation for efficient Gaussian integer modular arithmetic with ECC system

- Elliptic curve cryptography (ECC) is suitable for resource-constrained devices (shorter keys than RSA)
- The ECC trapdoor function is the elliptic curve scalar point multiplication (PM)
- Consider the key $k$, the length of the key in bits $r$, and a point on the curve $P$, then the PM can be calculated using the Horner scheme as

$$k \cdot P = \sum_{j=0}^{r-1} k_j \tau^j \cdot P = \tau(\cdots \tau(k_{r-1} + k_{r-2}P) + \cdots) + k_0P$$

Representing the point on the curve $P$, the key $k$, the digits of the key $k_j$, and the base $\tau$ as Gaussian integers reduces the computational complexity of the PM. This can also reduce the memory requirements for robust applications against side channel attacks (SCA)!

- It was shown in [4,5] that representing the key with non-binary base $\tau$ can reduce the computational complexity of the PM. Let $k$ be the integer $k$ converted to the base $\tau$, the PM can be calculated as


Motivation for efficient Gaussian integer modular arithmetic with ECC system

- Precomputations to prevent side channel attacks for a non-binary base \( \tau \) or \( w \)

- \( M \) describes multiplication-equivalent operations

- Binary key with \( r = 163 \) bits

- \( l \) is the number of iterations to calculate the point multiplication (PM)

- [5] introduces a memory reduction using ordinary integers for the key expansions

- [4] enables further memory reduction and lower computational complexity using Gaussian integer key expansions

| Reference | \(|\tau|^2 \text{ or } 2^w\) | Stored points | \( l \) | \( M \text{ for PM \\ & precomp.} \) |
|-----------|-----------------|--------------|------|------------------|
| Gaussian integer key expansion [4] | 17 | 5 | 0.245\(r\) | 1678 |
| Gaussian integer key expansion [4] | 29 | 8 | 0.206\(r\) | 1953 |
| Proposed ordinary key expansion [5] | 16 | 8 | 0.2515\(r\) | 2726 |
| Fixed-base ordinary key expansion [5] | 16 | 15 | 0.2515\(r\) | 2710 |
| Proposed ordinary key expansion [5] | 32 | 16 | 0.203\(r\) | 2796 |
| Fixed-base ordinary key expansion [5] | 32 | 31 | 0.203\(r\) | 2780 |


Motivation for efficient Gaussian integer modular arithmetic with ECC system

- Precomputations to prevent side channel attacks for a non-binary base $\tau$ or $w$

- $M$ describes multiplication-equivalent operations

- Binary key with $r = 163$ bits

- $l$ is the number of iterations to calculate the point multiplication (PM)

- [5] introduces a memory reduction using ordinary integers for the key expansion

- [4] enables further memory reduction and lower computational complexity using Gaussian integer key expansions

| Reference | $|\tau|^2$ or $2^w$ | Stored points | $l$ | $M$ for PM & precomp. |
|-----------|-----------------|--------------|-----|----------------------|
| Gaussian integer key expansion [4] | 17 | 5 | 0.245$r$ | 1678 |
| Gaussian integer key expansion [4] | 29 | 8 | 0.206$r$ | 1953 |
| Proposed ordinary key expansion [5] | 16 | 8 | 0.2515$r$ | 2726 |
| Fixed-base ordinary key expansion [5] | 16 | 15 | 0.2515$r$ | 2710 |

This example motivates the requirement of efficient modular arithmetic for Gaussian integers!


Computes \( r = z \mod m \) using \( \mu \) (precomputed), for any integer numbers \( r, z, m, \mu \) [7]

- Only additions, subtractions, multiplications, and digit operations are used
- No divisions are needed since \( \beta \) is a power of two (typically the word-size of the underlying processor)
- \( q_1 \) and \( q_3 \) can be calculated using digit shifts
- Lines 10 to 12 are denoted as final reduction to obtain the final result \( r \) from the approximated congruent \( r' \)

**input:** Two positive integer numbers \( z \) and \( m \), \( \mu = \lceil \beta^{2k}/m \rceil \), \( \beta > 3 \)

**output:** Integer number \( r = z \mod m \)

```python
1: \( q_1 \leftarrow \lceil z/\beta^{k-1} \rceil \)
2: \( q_2 \leftarrow q_1 \mu \)
3: \( q_3 \leftarrow \lceil q_2/\beta^{k+1} \rceil \)
4: \( r_1 \leftarrow z \mod \beta^{k+1} \)
5: \( r_2 \leftarrow q_3 \mu \mod \beta^{k+1} \)
6: \( r' \leftarrow r_1 - r_2 \)
7: if \( (r' < 0) \) then
8:   \( r' \leftarrow r' + \beta^{k+1} \)
9: end if
10: while \( (r' \geq m) \) do
11:   \( r' \leftarrow r' - m \)
12: end while
13: \( r \leftarrow r' \)
14: return \( r \)
```
Concepts of Barrett reduction for integer numbers [7, Alg. 14.42]

- Computes \( r = z \mod m \) using \( \mu \) (precomputed), for any integer numbers \( r, z, m, \mu \) [7]

- Only additions, subtractions, multiplications, and digit operations are used

- No divisions are needed since \( \beta \) is a power of two (typically the word-size of the underlying processor)

- \( q_1 \) and \( q_3 \) can be calculated using digit shifts

- Lines 10 to 12 are denoted as final reduction to obtain the final result \( r \) from the approximated congruent \( r' \)

- This algorithm determines \( q_3 = \left\lfloor \frac{z\beta^{-1}}{\beta^{2k}} \left\lfloor \frac{\beta^{2k}}{m} \right\rfloor \right\rfloor \)

- Improved version computes \( q_3 = \left\lfloor \frac{z\beta^{-1} + \gamma}{\beta^k + \delta} \right\rfloor \) to reduce the complexity of the final reduction (\( \gamma, \delta \) examples [8])

\[ \text{input: Two positive integer numbers } z \text{ and } m, \mu = \left\lfloor \frac{\beta^{2k}}{m} \right\rfloor, \beta > 3 \]
\[ \text{output: Integer number } r = z \mod m \]

1: \( q_1 \leftarrow \left\lfloor \frac{z}{\beta^{k-1}} \right\rfloor \)
2: \( q_2 \leftarrow q_1 \mu \)
3: \( q_3 \leftarrow \left\lfloor \frac{q_2}{\beta^{k+1}} \right\rfloor \)
4: \( r_1 \leftarrow z \mod \beta^{k+1} \)
5: \( r_2 \leftarrow q_3 m \mod \beta^{k+1} \)
6: \( r' \leftarrow r_1 - r_2 \)
7: \( \text{if } (r' < 0) \text{ then } \)
8: \( r' \leftarrow r' + \beta^{k+1} \)
9: \( \text{end if } \)
10: \( \text{while } (r' \geq m) \text{ do } \)
11: \( r' \leftarrow r' - m \)
12: \( \text{end while } \)
13: \( r \leftarrow r' \)
14: \( \text{return } r \)


Computes $r = z \mod m$ using $\mu$ (precomputed), for any integer numbers $r, z, m, \mu$ [7]

- Only additions, subtractions, multiplications, and digit operations are used

- Replace the floor divisions with suitable low-cost rounding functions

- No need for steps 7 to 9, since Gaussian integers include negative integer numbers

- The final reduction for Gaussian integers is more complex

→ Use the improved Barrett and determine the corresponding values for $\gamma, \delta$

- Improved version computes $q_3 = \left\lfloor \frac{z}{\beta^{k+\gamma}} \right\rfloor \mod \beta^{k+\delta}$ to reduce the complexity of the final reduction ($\gamma, \delta$ examples [8])

**input:** Two positive integer numbers $z$ and $m$, $\mu = \left\lfloor \beta^{2k}/m \right\rfloor$, $\beta > 3$

**output:** Integer number $r = z \mod m$

1: $q_1 \leftarrow \left\lfloor \frac{z}{\beta^{k-1}} \right\rfloor$
2: $q_2 \leftarrow q_1 \mu$
3: $q_3 \leftarrow \left\lfloor \frac{q_2}{\beta^{k+1}} \right\rfloor$
4: $r_1 \leftarrow z \mod \beta^{k+1}$
5: $r_2 \leftarrow q_3m \mod \beta^{k+1}$
6: $r' \leftarrow r_1 - r_2$
7: if ($r' < 0$) then
8: $r' \leftarrow r' + \beta^{k+1}$
9: end if
10: while ($r' \geq m$) do
11: $r' \leftarrow r' - m$
12: end while
13: $r \leftarrow r'$
14: return $r$


Proposed novel reduction for Gaussian integers based on Barrett’s concepts

• Computes $r = z \mod \pi$ using $\mu = \beta^k + \delta \ cdiv \pi$ (precomputed), for any Gaussian integers $r, z, \pi, \mu$

• Uses only subtractions, multiplications, and digit operations (lines 1 to 6)

• No divisions are needed since $\beta$ is a power of two (typically the word-size of the underlying processor)

• $\text{fdiv}$ rounding towards zero (digit shifts)

• $\text{cdiv}$ rounding away from zero (digit shifts and conditional additions of const. 1)

**input:** Gaussian integers $z, \mu, \pi$, integer numbers $\beta, \gamma, \delta$

**output:** Gaussian integer $r = z \mod \pi$

```
1: q_1 \leftarrow z \ cdiv \beta^{k + \delta}
2: q_2 \leftarrow q_1 \mu
3: q_3 \leftarrow q_2 \ fdiv \beta^{\gamma - \delta}
4: r_1 \leftarrow z \ mod \beta^{\gamma - \delta}
5: r_2 \leftarrow q_3 \pi \ mod \beta^{\gamma - \delta}
6: r' \leftarrow r_1 - r_2
7: \text{if } (|r'| < |\pi|/(\sqrt{2} - 1) / \sqrt{2}) \text{ then}
8: \alpha \leftarrow 0
9: \text{else if } (|r'| < |\pi| / \sqrt{2}) \text{ then}
10: \alpha \leftarrow \arg \min_{\alpha \in \{0, \pm 1, \pm i\}} |r' - \hat{\alpha}\pi|
11: \text{else}
12: \alpha \leftarrow \arg \min_{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}} |r' - \hat{\alpha}\pi|
13: \text{end if}
14: r \leftarrow r' - \alpha \pi
15: \text{return } r
```
Proposed novel reduction for Gaussian integers based on Barrett’s concepts

- Computes \( r = z \mod \pi \) using \( \mu = \beta^{k+\delta} \text{ cdiv } \pi \) (precomputed), for any Gaussian integers \( r, z, \pi, \mu \)

- Uses only subtractions, multiplications, and digit operations (lines 1 to 6)

- No divisions are needed since \( \beta \) is a power of two (typically the word-size of the underlying processor)

- \text{fdiv} rounding \textbf{towards} zero (digit shifts)

- \text{cdiv} rounding \textbf{away} from zero (digit shifts and conditional additions of const. 1)

- The difference between \( |q_3| \) and \( |Q| = \left\lfloor \frac{z \pi^*}{\pi \pi^*} \right\rfloor \) from the naïve reduction [6] is upper bounded by \( \sqrt{2} \) (derivation in the paper)

- Using this bound, the final reduction (lines 7 to 14) obtains \( r \) from the approximated \( r' \) based on offset comparisons

\begin{verbatim}
input: Gaussian integers \( z, \mu, \pi, \) integer numbers \( \beta, \gamma, \delta \)
output: Gaussian integer \( r = z \mod \pi \)
1: \( q_1 \leftarrow z \text{ cdiv } \beta^{k+\delta} \)
2: \( q_2 \leftarrow q_1 \mu \)
3: \( q_3 \leftarrow q_2 \text{ fdiv } \beta^{\gamma-\delta} \)
4: \( r_1 \leftarrow z \mod \beta^{\gamma-\delta} \)
5: \( r_2 \leftarrow q_3 \pi \mod \beta^{\gamma-\delta} \)
6: \( r' \leftarrow r_1 - r_2 \)
7: \textbf{if} \( (|r'| < |\pi| (\sqrt{2} - 1)/\sqrt{2}) \) \textbf{then} \( \alpha \leftarrow 0 \)
8: \textbf{else if} \( (|r'| < |\pi| / \sqrt{2}) \) \textbf{then} \( \alpha \leftarrow \arg\min_{\hat{\alpha}\in\{0,\pm1,\pm i\}} |r' - \hat{\alpha}\pi| \)
9: \textbf{else} \( \alpha \leftarrow \arg\min_{\hat{\alpha}\in\{\pm1,\pm i, \pm 1\pm i\}} |r' - \hat{\alpha}\pi| \)
10: \textbf{end if} \( r \leftarrow r' - \alpha\pi \)
11: \textbf{return} \( r \)
\end{verbatim}

The final reduction computes $r = r' - \alpha \pi$

The upper bound $\sqrt{2}$ is used to limit the possible offset candidates to $\alpha \in \{0, \pm 1, \pm i, \pm 1 \pm i\}$

Concept to reduce the offset comparisons based on the absolute value [2]

- If $|r'| < \frac{\sqrt{2}-1}{\sqrt{2}} |\pi|$ then $\alpha = 0$
- Else if $|r'| < \frac{|r|}{\sqrt{2}}$ then $\alpha = \arg\min_{\alpha \in \{0, \pm 1, \pm i\}} |q - \alpha \pi|$
- Else $\alpha = \arg\min_{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}} |q - \alpha \pi|$

Further complexity reduction based on the sign of the real and imaginary parts of $r'$ in the paper

### Input
Gaussian integers $z, \mu, \pi$, integer numbers $\beta, \gamma, \delta$

### Output
Gaussian integer $r = z \mod \pi$

1: $q_1 \leftarrow z \ cd\ div \ \beta^{k+\delta}$
2: $q_2 \leftarrow q_1 \mu$
3: $q_3 \leftarrow q_2 \ f\ div \ \beta^{\gamma-\delta}$
4: $r_1 \leftarrow z \ mod \ \beta^{\gamma-\delta}$
5: $r_2 \leftarrow q_3 \pi \ mod \ \beta^{\gamma-\delta}$
6: $r' \leftarrow r_1 - r_2$
7: if ($|r'| < |\pi| \ (\sqrt{2} - 1) / \sqrt{2}$) then
8: $\alpha \leftarrow 0$
9: else if ($|r'| < |\pi| / \sqrt{2}$) then
10: $\alpha \leftarrow \arg\min_{\alpha \in \{0, \pm 1, \pm i\}} |r' - \alpha \pi|$
11: else
12: $\alpha \leftarrow \arg\min_{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}} |r' - \alpha \pi|$
13: end if
14: $r \leftarrow r' - \alpha \pi$
15: return $r$
Concept of the final reduction

- The final reduction computes \( r = r' - \alpha \pi \)

- The upper bound \( \sqrt{2} \) is used to limit the possible offset candidates to \( \alpha \in \{0, \pm 1, \pm i, \pm 1 \pm i\} \)

- Concept to reduce the offset comparisons based on the absolute value [2]
  - If \( |r'| < \frac{\sqrt{2}-1}{\sqrt{2}} |\pi| \) then \( \alpha = 0 \)
  - Else if \( |r'| < \frac{|\pi|}{\sqrt{2}} \) then \( \alpha = \arg \min_{\alpha \in \{0, \pm 1, \pm i\}} |q - \alpha \pi| \)
  - Else \( \alpha = \arg \min_{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}} |q - \alpha \pi| \)

- Further complexity reduction based on the sign of the real and imaginary parts of \( r' \) in the paper

![Example for \( G_{73} \) with \( \pi = 8 + 3i \)]
Concept of the final reduction

- The final reduction computes \( r = r' - \alpha \pi \)

- The upper bound \( \sqrt{2} \) is used to limit the possible offset candidates to \( \alpha \in \{0, \pm 1, \pm i, \pm 1 \pm i\} \)

- Concept to reduce the offset comparisons based on the absolute value [2]
  - If \( |r'| < \frac{\sqrt{2}-1}{\sqrt{2}} |\pi| \) then \( \alpha = 0 \)
  - Else if \( |r'| < \frac{|\pi|}{\sqrt{2}} \) then \( \alpha = \arg \min_{\alpha \in \{0, \pm 1, \pm i\}} |q - \alpha \pi| \)
  - Else \( \alpha = \arg \min_{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}} |q - \alpha \pi| \)

- Further complexity reduction based on the sign of the real and imaginary parts of \( r' \) in the paper

Example for \( G_{73} \) with \( \pi = 8 + 3i \)
Montgomery reduction for Gaussian integers according to [2]

- Computes $M = Z \mod \pi$ for any Gaussian integers $X, Y, \pi, Z$ in the Montgomery domain

- Uses only additions, multiplications, and digit operations (lines 1 to 6)

- No divisions are needed since $R$ is a power of two (typically the word-size of the underlying processor)

- The function div is identical to our $fdiv$ rounding towards zero (digit shifts)

- Final reduction depends on $|q|$, where $|q| \leq \sqrt{2}$ [2]

- Identical to the proposed final reduction, since

$$ \alpha' = \arg \min_{\alpha \in \{0, \pm 1, \pm i\}} |q - \alpha \pi| $$

$$ \alpha'' = \arg \min_{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}} |q - \alpha \pi| $$

Input: $Z = XY$, $\pi' = -\pi^{-1} \mod R$, $R = 2^l > \frac{|\pi|}{\sqrt{2}}$

Output: $M = \mu(Z) = ZR^{-1} \mod \pi$

1: $t = Z\pi' \mod R$ // bitwise AND of Re, Im with $R - 1$

2: $q = (Z + t\pi) \div R$ // shift Re, Im right by $l$

3: if ($|q| < \frac{\sqrt{2} - 1}{\sqrt{2}} |\pi|$) then

4: $M = q$

5: else if ($|q| < \frac{|\pi|}{\sqrt{2}}$) then

6: determine $\alpha'$

7: $M = q - \alpha' \pi$

8: else

9: determine $\alpha''$

10: $M = q - \alpha'' \pi$

11: end if

Montgomery reduction for Gaussian integers according to [2]

- Computes $M = Z \mod \pi$ for any Gaussian integers $X, Y, \pi, Z$ in the Montgomery domain.
- Uses only additions, multiplications, and digit operations (lines 1 to 6).
- No divisions are needed since $R$ is a power of two (typically the word-size of the underlying processor).
- The function $\text{div}$ is identical to our $\text{fdiv}$ rounding towards zero (digit shifts).
- Final reduction depends on $|q|$, where $|q| \leq \sqrt{2}$ [2].
- Identical to the proposed final reduction, since

$$
\alpha' = \text{argmin}_{\alpha \in \{0, \pm 1, \pm i\}} |q - \alpha \pi|
$$

$$
\alpha'' = \text{argmin}_{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}} |q - \alpha 
$$

input: $Z = XY$, $\pi' = -\pi^{-1} \mod R$, $R = 2^l > \frac{|\pi|}{\sqrt{2}}$

output: $M = \mu(Z) = Z \pi^{-1} \mod \pi$

1: $t = Z \pi' \mod R$ // bitwise AND of Re, Im with $R - 1$
2: $q = (Z + t \pi) \text{div } R$ // shift Re, Im right by $l$
3: if $(|q| < \frac{\sqrt{2} - 1}{\sqrt{2}} |\pi|)$ then
4: $M = q$
5: else if $(|q| < \frac{|\pi|}{\sqrt{2}})$ then
6: determine $\alpha'$
7: $M = q - \alpha' \pi$
8: else
9: determine $\alpha''$
10: $M = q - \alpha'' \pi$
11: end if

Capital letters demonstrate the representation in the Montgomery domain. Montgomery domain transformations are required!
Comparing the proposed reduction with the Montgomery reduction for Gaussian integers from [2]

**Montgomery reduction**

**input:** $Z = XY$, $\pi' = -\pi^{-1} \mod R$, $R = 2^l > \frac{|\pi|}{\sqrt{2}}$

**output:** $M = \mu(Z) = ZR^{-1} \mod \pi$

1. $t = Z \pi' \mod R$  // bitwise AND of Re, Im with $R - 1$
2. $q = (Z + t\pi) \div R$  // shift Re, Im right by $l$

Final reduction on $q$

**Proposed reduction**

**input:** Gaussian integers $z, \mu, \pi$, integer numbers $\beta, \gamma, \delta$

**output:** Gaussian integer $r = z \mod \pi$

1. $q_1 \leftarrow z \text{ cdiv } \beta^{k+\delta}$
2. $q_2 \leftarrow q_1 \mu$
3. $q_3 \leftarrow q_2 \text{ fdiv } \beta^{\gamma-\delta}$
4. $r_1 \leftarrow z \mod \beta^{\gamma-\delta}$
5. $r_2 \leftarrow q_3 \pi \mod \beta^{\gamma-\delta}$
6. $r' \leftarrow r_1 - r_2$

Final reduction on $r'$

The final reduction is not illustrated since it is identical

Comparing the proposed reduction with the Montgomery reduction for Gaussian integers from [2]

Montgomery reduction

- **input:** \( Z = XY, \pi' = -\pi^{-1} \mod R, R = 2^l > \left| \pi \right| / \sqrt{2} \)
- **output:** \( M = \mu(Z) = ZR^{-1} \mod \pi \)

1: \( t = Z\pi' \mod R \) // bitwise AND of Re, Im with \( R - 1 \)
2: \( q = (Z + t\pi) \div R \) // shift Re, Im right by \( l \)

\[ \text{Final reduction on } q \]

Proposed reduction

- **input:** Gaussian integers \( z, \mu, \pi \), integer numbers \( \beta, \gamma, \delta \)
- **output:** Gaussian integer \( r = z \mod \pi \)

1: \( q_1 \leftarrow z \div \beta^{k+\delta} \)
2: \( q_2 \leftarrow q_1 \mu \)
3: \( q_3 \leftarrow q_2 \text{fdiv} \beta^{\gamma-\delta} \)
4: \( r_1 \leftarrow z \mod \beta^{\gamma-\delta} \)
5: \( r_2 \leftarrow q_3 \pi \mod \beta^{\gamma-\delta} \)
6: \( r' \leftarrow r_1 - r_2 \)

\[ \text{Final reduction on } r' \]

The final reduction is not illustrated since it is identical

Two complex multiplications by a constant

Comparing the proposed reduction with the Montgomery reduction for Gaussian integers from [2]

Montgomery reduction

input: $Z = XY$, $\pi' = -\pi^{-1} \mod R$, $R = 2^l > \frac{|\pi|}{\sqrt{2}}$
output: $M = \mu(Z) = ZR^{-1} \mod \pi$

1: $t = Z\pi' \mod R$  // bitwise AND of Re, Im with $R - 1$
2: $q = (Z + t\pi) \div R$  // shift Re, Im right by $l$

Final reduction on $q$

Proposed reduction

input: Gaussian integers $z, \mu, \pi$, integer numbers $\beta, \gamma, \delta$
output: Gaussian integer $r = z \mod \pi$

1: $q_1 \leftarrow z \div \beta^{k+\delta}$
2: $q_2 \leftarrow q_1 \mu$
3: $q_3 \leftarrow q_2 \fdiv \beta^{\gamma-\delta}$
4: $r_1 \leftarrow z \mod \beta^{\gamma-\delta}$
5: $r_2 \leftarrow q_3 \pi \mod \beta^{\gamma-\delta}$
6: $r' \leftarrow r_1 - r_2$

Final reduction on $r'$

The final reduction is not illustrated since it is identical

Two complex multiplications by a constant

One complex addition/subtraction

Complexity comparison

- Naïve modulo reduction \( x \mod \pi = x - \left\lfloor \frac{x\pi^*}{\pi} \right\rfloor \cdot \pi \) [6]
- The costs for digit operations are not considered
- The Montgomery domain transformations are defined in [2]

<table>
<thead>
<tr>
<th></th>
<th>Addition / subtraction</th>
<th>Multiplication by a constant</th>
<th>Complex number division</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve reduction [6]</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Montgomery reduction [2]</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Montgomery domain</td>
<td>2</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>transformations [2]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed reduction</td>
<td>1</td>
<td>2</td>
<td>-</td>
</tr>
</tbody>
</table>

Conclusion

A novel and efficient reduction algorithm for Gaussian integers based on Barrett’s concepts is presented

- Suitable for arbitrary Gaussian integer moduli
- Providing similar computational complexity as the Montgomery reduction for Gaussian integers
- Not requiring domain transformations as the Montgomery reduction
- Suitable for any application where modular arithmetic over Gaussian integers is needed (not only ECC !)
Thanks for your attention

Questions !?