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

ARITH 2023

SIEMENS

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

[1] M. Safieh, **Algorithms and Architectures for Cryptography and Source Coding in Non-Volatile Flash Memories**, in *Springer 2021*, ISBN 978-3-658-34458-0, pp. 1-132.
[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian** Integers, in *Cryptography*. 2021; 5(1):6.
[3] M. Safieh and F. De Santis, **Efficient Reduction Algorithms for Special Gaussian Integer Moduli**, in *29th IEEE Symposium on Computer Arithmetic*, ARITH 2022, Lyon, France, Sept. 2022.

SIEMENS

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

[1] M. Safieh, **Algorithms and Architectures for Cryptography and Source Coding in Non-Volatile Flash Memories**, in *Springer 2021*, ISBN 978-3-658-34458-0, pp. 1-132.
[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian** Integers, in *Cryptography*. 2021; 5(1):6.
[3] M. Safieh and F. De Santis, **Efficient Reduction Algorithms for Special Gaussian Integer Moduli**, in *29th IEEE Symposium on Computer Arithmetic*, ARITH 2022, Lyon, France, Sept. 2022.

Introduction to Gaussian integers

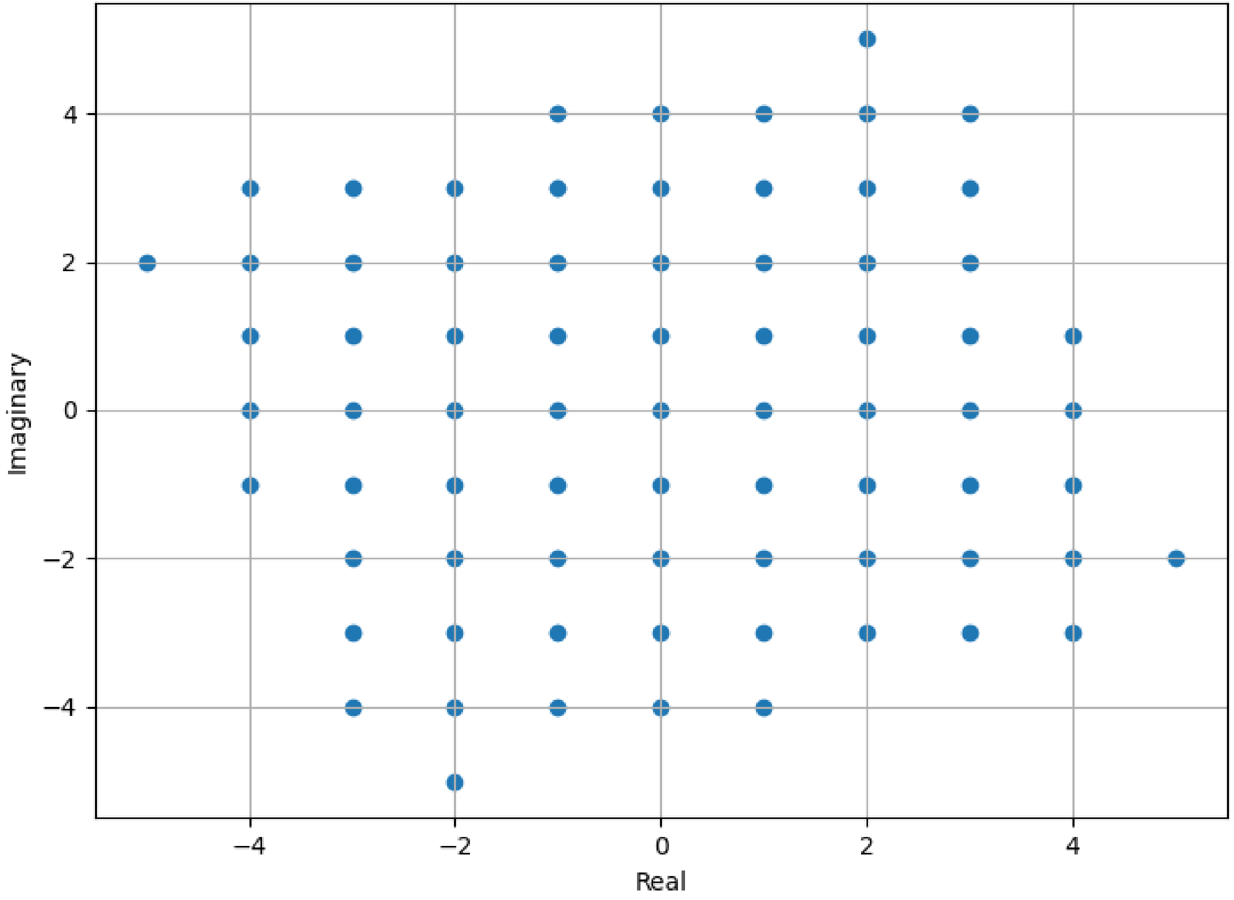
- Subset of complex numbers→ $x = a + b\mathrm{i}, \mathrm{i} = \sqrt{-1}$, a, and b are integer numbers

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

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

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

G_{73} with $\pi = 8 + 3\mathrm{i}, p = \pi\pi^* = 73$

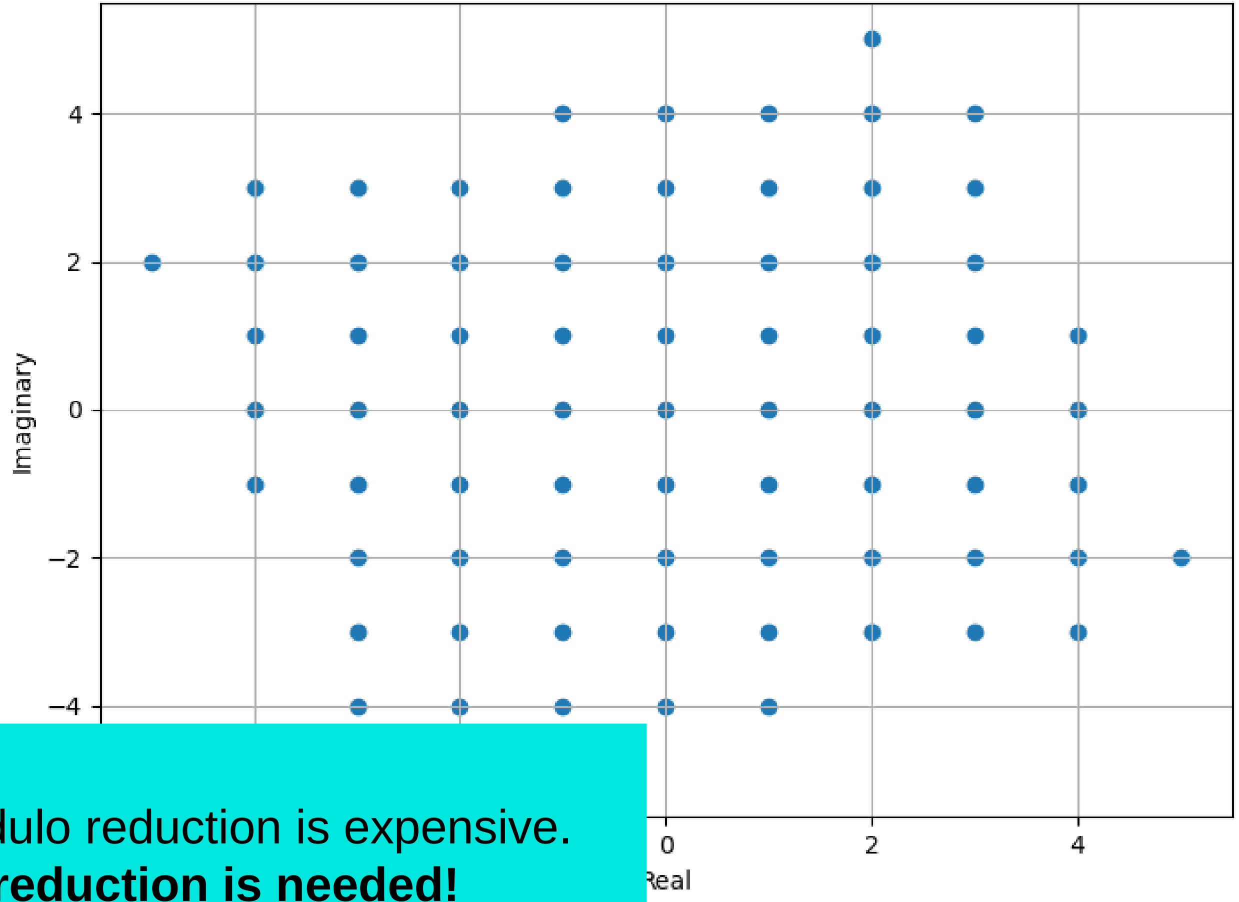


[1] M. Safieh, **Algorithms and Architectures for Cryptography and Source Coding in Non-Volatile Flash Memories**, in *Springer 2021*, ISBN 978-3-658-34458-0, pp. 1-132.
[6] K. Huber, **Codes over Gaussian integers**, in *IEEE Transactions on Information Theory*, pp. 207–216, 1994.

Introduction to Gaussian integers

- Subset of complex numbers→ $x = a + b\mathrm{i}, \mathrm{i} = \sqrt{-1}$, 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 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]

G_{73} with $\pi = 8 + 3\mathrm{i}, p = \pi\pi^* = 73$



The division for the naïve modulo reduction is expensive.
More efficient modulo reduction is needed!

[1] M. Safieh, **Algorithms and Architectures for Cryptography and Source Coding in Non-Volatile Flash Memories**, in *Springer 2021*, ISBN 978-3-658-34458-0, pp. 1-132.
[6] K. Huber, **Codes over Gaussian integers**, in *IEEE Transactions on Information Theory*, pp. 207–216, 1994.

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 τ can reduce the computational complexity of the PM. Let κ be the integer k converted to the base τ, the PM can be calculated as

$$\kappa \cdot P = \sum_{j=0}^{l-1} \kappa_j \tau^j \cdot P = \tau(\cdots \tau(\tau\kappa_{r-1} + \kappa_{r-2}P) + \cdots) + \kappa_0 P$$

[4] M. Safieh, J. Thiers, and J. Freudenberger, **Side channel attack resistance of the elliptic curve point multiplication using Gaussian integers**, *in 2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, May 2020, pp. 231–236.
[5] M. Hedabou, P. Pinel, and L. Bénéteau , **Countermeasures for preventing comb method against SCA attacks**, in *Information Security Practice and Experience*, R. H. Deng, F. Bao, H. Pang, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 85–96.

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

Representing the point on the curve P, the key κ, the digits of the key κ_j, and the base τ 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 τ can reduce the computational complexity of the PM. Let κ be the integer k converted to the base τ, the PM can be calculated as

$$\kappa \cdot P = \sum_{j=0}^{l-1} \kappa_j \tau^j \cdot P = \tau(\cdots \tau(\tau \kappa_{r-1} + \kappa_{r-2} P) + \cdots) + \kappa_0 P$$

[4] M. Safieh, J. Thiers, and J. Freudenberger, **Side channel attack resistance of the elliptic curve point multiplication using Gaussian integers**, *in 2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, May 2020, pp. 231–236.
[5] M. Hedabou, P. Pinel, and L. Bénéteau , **Countermeasures for preventing comb method against SCA attacks**, in *Information Security Practice and Experience*, R. H. Deng, F. Bao, H. Pang, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 85–96.

SIEMENS

Motivation for efficient Gaussian integer modular arithmetic with ECC system

- Precomputations to prevent side channel attacks for a non-binary base τ 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	$\lvert\tau\rvert^2$ or 2^w	Stored points	l	*M* for PM & precomp.
Gaussian integer key expansion [4]	17	5	$0.245r$	1678
Gaussian integer key expansion [4]	29	8	$0.206r$	1953
Proposed ordinary key expansion [5]	16	8	$0.2515r$	2726
Fixed-base ordinary key expansion [5]	16	15	$0.2515r$	2710
Proposed ordinary key expansion [5]	32	16	$0.203r$	2796
Fixed-base ordinary key expansion [5]	32	31	$0.203r$	2780

[4] M. Safieh, J. Thiers, and J. Freudenberger, **Side channel attack resistance of the elliptic curve point multiplication using Gaussian integers**, *in 2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, May 2020, pp. 231–236.
[5] M. Hedabou, P. Pinel, and L. Bénéteau , **Countermeasures for preventing comb method against SCA attacks**, in *Information Security Practice and Experience*, R. H. Deng, F. Bao, H. Pang, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 85–96.

SIEMENS

Motivation for efficient Gaussian integer modular arithmetic with ECC system

- Precomputations to prevent side channel attacks for a non-binary base τ 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
ordinary integers for the
- [4] enables further mem
lower computational complexity using Gaussian integer key expansions

Reference	$\lvert\tau\rvert^2$ or 2^w	Stored points	l	*M* for PM & precomp.
Gaussian integer key expansion [4]	17	5	$0.245r$	1678
Gaussian integer key expansion [4]	29	8	$0.206r$	1953
Proposed ordinary key expansion [5]	16	8	$0.2515r$	2726
Fixed-base ordinary key	16	15	$0.2515r$	2710
			$0.203r$	2796
expansion [5]			$0.203r$	2780

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

[4] M. Safieh, J. Thiers, and J. Freudenberger, **Side channel attack resistance of the elliptic curve point multiplication using Gaussian integers**, *in 2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, May 2020, pp. 231–236.
[5] M. Hedabou, P. Pinel, and L. Bénéteau , **Countermeasures for preventing comb method against SCA attacks**, in *Information Security Practice and Experience*, R. H. Deng, F. Bao, H. Pang, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 85–96.

SIEMENS

Concepts of Barrett reduction for integer numbers [7, Alg. 14.42]

- Computes $r = z \bmod m$ using μ (precomputed), for any integer numbers r, z, m, μ [7]
- Only additions, subtractions, multiplications, and digit operations are used
- No divisions are needed since β 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, $\mu = \lfloor \beta^{2k}/m \rfloor$, $\beta > 3$
output: Integer number $r = z \bmod m$

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

[7] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, **Handbook of Applied Cryptography**, in *CRC Press*, 2001. ISBN: 0-8493-8523-7.
[8] J.-F. Dhem, **Modified Version of the Barrett Algorithm**, in *technical report*, 1994.

Concepts of Barrett reduction for integer numbers [7, Alg. 14.42]

- Computes $r = z \bmod m$ using μ (precomputed), for any integer numbers r, z, m, μ [7]
- Only additions, subtractions, multiplications, and digit operations are used
- No divisions are needed since β 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 $q_3 = \left\lfloor \frac{\left\lfloor \frac{z}{\beta^{k-1}} \right\rfloor \left\lfloor \frac{\beta^{2k}}{m} \right\rfloor}{\beta^{k+1}} \right\rfloor$
- Improved version computes $q_3 = \left\lfloor \frac{\left\lfloor \frac{z}{\beta^{k+\delta}} \right\rfloor \left\lfloor \frac{\beta^{k+\gamma}}{m} \right\rfloor}{\beta^{\gamma-\delta}} \right\rfloor$ to reduce the complexity of the final reduction (γ, δ examples [8])

input: Two positive integer numbers z and m, $\mu = \lfloor \beta^{2k}/m \rfloor$, $\beta > 3$
output: Integer number $r = z \bmod m$

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

[7] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, **Handbook of Applied Cryptography**, in *CRC Press*, 2001. ISBN: 0-8493-8523-7.
[8] J.-F. Dhem, **Modified Version of the Barrett Algorithm**, in *technical report*, 1994.

Concepts of Barrett reduction for integer numbers [7, Alg. 14.42]

- Computes $r = z \bmod m$ using μ (precomputed), for any integer numbers r, z, m, μ [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 γ,δ

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

input: Two positive integer numbers z and m, $\mu = \lfloor \beta^{2k}/m \rfloor$, $\beta > 3$
output: Integer number $r = z \bmod m$

1: $q_1 \leftarrow \lfloor z/\beta^{k-1} \rfloor$
2: $q_2 \leftarrow q_1 \mu$
3: $q_3 \leftarrow \lfloor q_2/\beta^{k+1} \rfloor$
4: $r_1 \leftarrow z \bmod \beta^{k+1}$
5: $r_2 \leftarrow q_3 m \bmod \beta^{k+1}$
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

[7] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, **Handbook of Applied Cryptography**, in *CRC Press*, 2001. ISBN: 0-8493-8523-7.
[8] J.-F. Dhem, **Modified Version of the Barrett Algorithm**, in *technical report*, 1994.

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, π, μ
- Uses only subtractions, multiplications, and digit operations (lines 1 to 6)
- No divisions are needed since β is a power of two (typically the word-size of the underlying processor)
  - fdiv rounding **towards** zero (digit shifts)
  - cdiv rounding **away** from zero (digit shifts and conditional additions of const. 1)

input: Gaussian integers z, μ, π, integer numbers β, γ, δ
output: Gaussian integer $r = z \bmod \pi$

1: $q_1 \leftarrow z \, \mathrm{cdiv} \, \beta^{k+\delta}$
2: $q_2 \leftarrow q_1 \mu$
3: $q_3 \leftarrow q_2 \, \mathrm{fdiv} \, \beta^{\gamma-\delta}$
4: $r_1 \leftarrow z \bmod \beta^{\gamma-\delta}$
5: $r_2 \leftarrow q_3 \pi \bmod \beta^{\gamma-\delta}$
6: $r' \leftarrow r_1 - r_2$
7: **if** $(|r'| < |\pi| \, (\sqrt{2}-1)/\sqrt{2})$ **then**
8: $\quad \alpha \leftarrow 0$
9: **else if** $(|r'| < |\pi| \, / \sqrt{2})$ **then**
10: $\quad \alpha \leftarrow \mathrm{argmin}_{\hat{\alpha} \in \{0, \pm 1, \pm \mathrm{i}\}} |r' - \hat{\alpha}\pi|$
11: **else**
12: $\quad \alpha \leftarrow \mathrm{argmin}_{\hat{\alpha} \in \{\pm 1, \pm \mathrm{i}, \pm 1 \pm \mathrm{i}\}} |r' - \hat{\alpha}\pi|$
13: **end if**
14: $r \leftarrow r' - \alpha\pi$
15: **return** r

[6] K. Huber, **Codes over Gaussian integers**, in *IEEE Transactions on Information Theory*, pp. 207–216, 1994.

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, π, μ
- Uses only subtractions, multiplications, and digit operations (lines 1 to 6)
- No divisions are needed since β is a power of two (typically the word-size of the underlying processor)
  - fdiv rounding **towards** zero (digit shifts)
  - cdiv rounding **away** from zero (digit shifts and conditional additions of const. 1)
- The difference between $|q_3|$ and $|Q| = \left|\left[\frac{z\pi^*}{\pi\pi^*}\right]\right|$ 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

input: Gaussian integers z, μ, π, integer numbers β, γ, δ
output: Gaussian integer $r = z \bmod \pi$

1: $q_1 \leftarrow z \; \mathrm{cdiv} \; \beta^{k+\delta}$
2: $q_2 \leftarrow q_1 \mu$
3: $q_3 \leftarrow q_2 \; \mathrm{fdiv} \; \beta^{\gamma-\delta}$
4: $r_1 \leftarrow z \bmod \beta^{\gamma-\delta}$
5: $r_2 \leftarrow q_3 \pi \bmod \beta^{\gamma-\delta}$
6: $r' \leftarrow r_1 - r_2$
7: **if** $(|r'| < |\pi| (\sqrt{2} - 1)/\sqrt{2})$ **then**
8: $\quad \alpha \leftarrow 0$
9: **else if** $(|r'| < |\pi| / \sqrt{2})$ **then**
10: $\quad \alpha \leftarrow \mathrm{argmin}_{\hat{\alpha} \in \{0, \pm 1, \pm \mathrm{i}\}} |r' - \hat{\alpha}\pi|$
11: **else**
12: $\quad \alpha \leftarrow \mathrm{argmin}_{\hat{\alpha} \in \{\pm 1, \pm \mathrm{i}, \pm 1 \pm \mathrm{i}\}} |r' - \hat{\alpha}\pi|$
13: **end if**
14: $r \leftarrow r' - \alpha\pi$
15: **return** r

[6] K. Huber, **Codes over Gaussian integers**, in *IEEE Transactions on Information Theory*, pp. 207–216, 1994.

SIEMENS

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 $\alpha = \underset{\alpha\in\{0,\pm 1,\pm i\}}{argmin} |q - \alpha\pi|$
  - Else $\alpha = \underset{\alpha\in\{\pm 1,\pm i,\pm 1 \pm i\}}{argmin} |q - \alpha\pi|$

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

input: Gaussian integers z, μ, π, integer numbers β, γ, δ
output: Gaussian integer $r = z \bmod \pi$

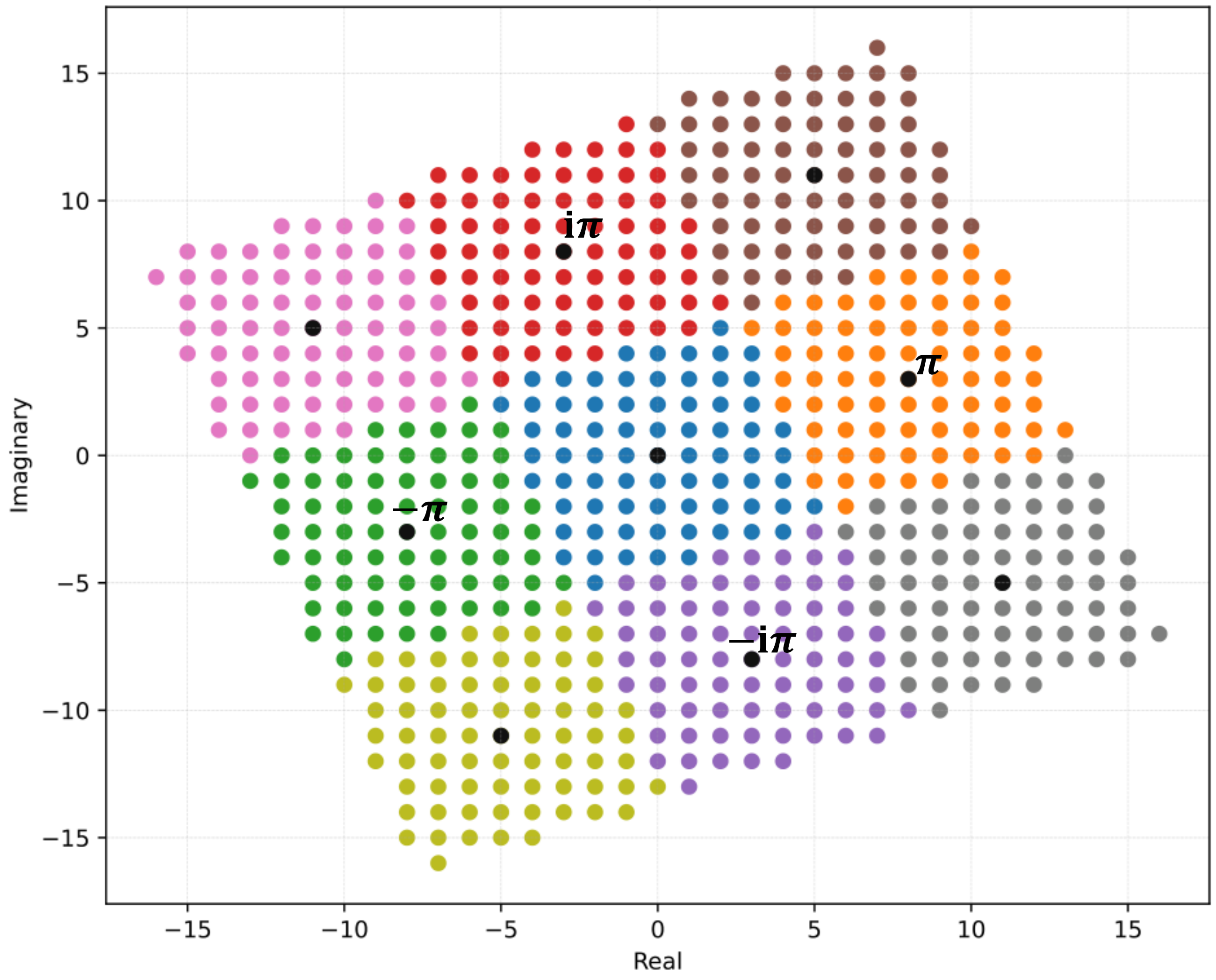
$q_1 \leftarrow z \operatorname{cdiv} \beta^{k+\delta}$
$q_2 \leftarrow q_1\mu$
$q_3 \leftarrow q_2 \operatorname{fdiv} \beta^{\gamma-\delta}$
$r_1 \leftarrow z \bmod \beta^{\gamma-\delta}$
$r_2 \leftarrow q_3\pi \bmod \beta^{\gamma-\delta}$
$r' \leftarrow r_1 - r_2$
if $(|r'| < |\pi|(\sqrt{2}-1)/\sqrt{2})$ **then**
    $\alpha \leftarrow 0$
else if $(|r'| < |\pi|/\sqrt{2})$ **then**
    $\alpha \leftarrow \operatorname{argmin}_{\hat{\alpha}\in\{0,\pm 1,\pm \mathrm{i}\}} |r' - \hat{\alpha}\pi|$
else
    $\alpha \leftarrow \operatorname{argmin}_{\hat{\alpha}\in\{\pm 1,\pm \mathrm{i},\pm 1 \pm \mathrm{i}\}} |r' - \hat{\alpha}\pi|$
end if
$r \leftarrow r' - \alpha\pi$
return r

[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

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 $\alpha = \underset{\alpha \in \{0, \pm 1, \pm i\}}{argmin} |q - \alpha\pi|$
  - Else $\alpha = \underset{\alpha \in \{\pm 1, \pm i, \pm 1 \pm i\}}{argmin} |q - \alpha\pi|$
- Further complexity reduction based on the sign of the real and imaginary parts of r' in the paper

Example for G_{73} with $\pi = 8 + 3\mathrm{i}$



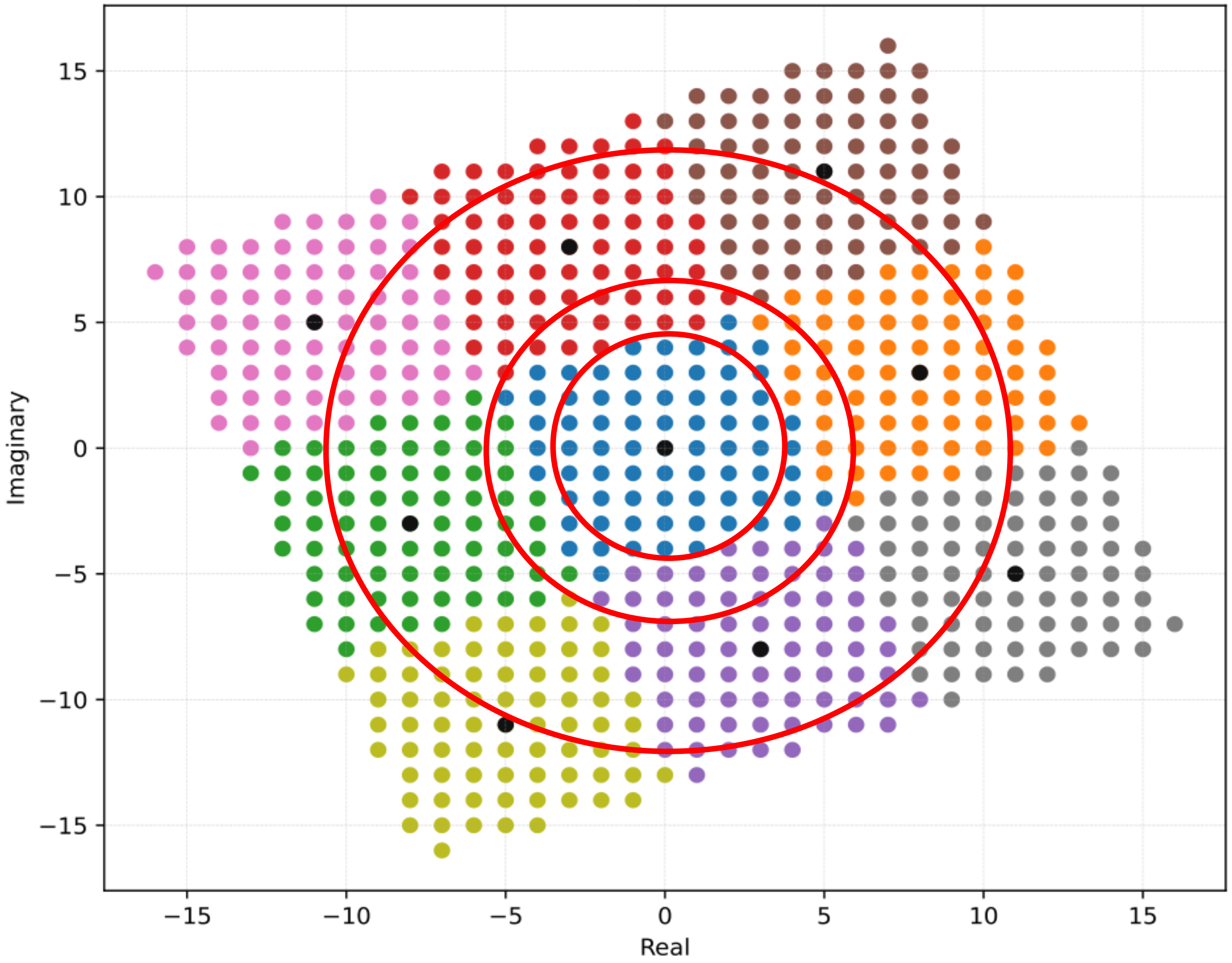
[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

SIEMENS

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 $\alpha = \underset{\alpha\in\{0,\pm 1,\pm i\}}{argmin}\ |q - \alpha\pi|$
  - Else $\alpha = \underset{\alpha\in\{\pm 1,\pm i,\pm 1\pm i\}}{argmin}\ |q - \alpha\pi|$
- Further complexity reduction based on the sign of the real and imaginary parts of r' in the paper

Example for G_{73} with $\pi = 8 + 3i$



[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

SIEMENS

Montgomery reduction for Gaussian integers according to [2]

- Computes $M = Z \; mod \; \pi$ for any Gaussian integers X, Y, π, 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

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

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

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

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

```
1:  t = Zπ' mod R          // bitwise AND of Re, Im with R − 1
2:  q = (Z + tπ) div R     // shift Re, Im right by l
3:  if (|q| < ((√2−1)/√2) |π|) then
4:      M = q
5:  else if (|q| < |π|/√2) then
6:      determine α'
7:      M = q − α'π
8:  else
9:      determine α''
10:     M = q − α''π
11: end if
```

[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

Montgomery reduction for Gaussian integers according to [2]

- Computes $M = Z \; mod \; \pi$ for any Gaussian integers X, Y, π, 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

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

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

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

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

Capital letters demonstrate the representation in the Montgomery domain.
Montgomery domain transformations are required!

[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

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

Montgomery reduction

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

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

⋮

Final reduction on q

Proposed reduction

input: Gaussian integers z, μ, π, integer numbers β, γ, δ
output: Gaussian integer $r = z \bmod \pi$

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

⋮

Final reduction on r'

The final reduction is not illustrated since it is identical

[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

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

Montgomery reduction

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

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

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

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

⋮

Final reduction on q

Proposed reduction

input: Gaussian integers z, μ, π, integer numbers β, γ, δ

output: Gaussian integer $r = z \bmod \pi$

1: $q_1 \leftarrow z \operatorname{fdiv} \beta^{k+\delta}$

2: $q_2 \leftarrow q_1\mu$

3: $q_3 \leftarrow q_2 \operatorname{fdiv} \beta^{\gamma-\delta}$

4: $r_1 \leftarrow z \bmod \beta^{\gamma-\delta}$

5: $r_2 \leftarrow q_3\pi \bmod \beta^{\gamma-\delta}$

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

[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

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

Montgomery reduction

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

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

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

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

⋮

Final reduction on q

Proposed reduction

input: Gaussian integers z, μ, π, integer numbers β, γ, δ

output: Gaussian integer $r = z \bmod \pi$

1: $q_1 \leftarrow z \operatorname{fdiv} \beta^{k+\delta}$

2: $q_2 \leftarrow q_1\mu$

3: $q_3 \leftarrow q_2 \operatorname{fdiv} \beta^{\gamma-\delta}$

4: $r_1 \leftarrow z \bmod \beta^{\gamma-\delta}$

5: $r_2 \leftarrow q_3\pi \bmod \beta^{\gamma-\delta}$

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

[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.

Complexity comparison

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

	Addition / subtraction	Multiplication by a constant	Complex number division
Naïve reduction [6]	1	2	1
Montgomery reduction [2]	1	2	-
Montgomery domain transformations [2]	2	5	-
Proposed reduction	1	2	-

[2] M. Safieh, J. Freudenberger, **Montgomery Reduction for Gaussian Integers**, in *Cryptography*. 2021; 5(1):6.
[6] K. Huber, **Codes over Gaussian integers**, in *IEEE Transactions on Information Theory*, pp. 207–216, 1994.

SIEMENS

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 !?

SIEMENS