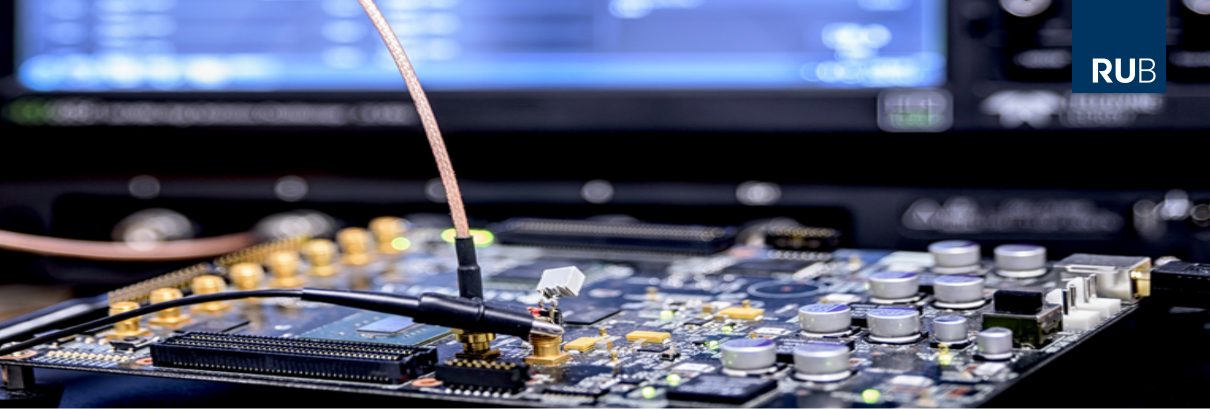


RUHR-UNIVERSITÄT BOCHUM

Efficiently Masking Polynomial Inversion at Arbitrary Order

Markus Krausz[1] Georg Land[1,2] Jan Richter-Brochmann[1] Tim Güneysu[1,2]

[1]Chair for Security Engineering, Ruhr-Universität Bochum
[2]DFKI GmbH, Cyber-Physical Systems, Bremen, Germany

September 29, 2022

Motivation

BIKE Key Generation

Require: BIKE parameters n, w, ℓ.
Ensure: Private key (h_0, h_1, σ) and public key h.

Generate $(h_0, h_1) \xleftarrow{\$} \mathcal{R}^2$ both of odd weight $|h_0| = |h_1| = w/2$.
Generate $\sigma \xleftarrow{\$} \{0,1\}^\ell$ uniformly at random.
Compute $h \leftarrow h_1 h_0^{-1}$.
Return (h_0, h_1, σ) and h.

NTRU Key Generation

Require: NTRU parameters n, p, q.
Ensure: Priv. key (f, f_p, f_q), pub. key h.

Generate $f \xleftarrow{\$} \mathcal{L}_f$
Generate $g \xleftarrow{\$} \mathcal{L}_g$
Compute $f_p \leftarrow 1/f$ in S_3
Compute $f_q \leftarrow 1/f$ in S_q
Compute $g \leftarrow 3 \cdot g \cdot f_g$ in R_q
Compute $h_q \leftarrow 1/h$ in S_q
Return (f, f_p, f_q) and h.

sNTRUp Key Generation

Require: sNTRUp parameter q.
Ensure: Priv. key (f, g_{inv}), pub. key h.

repeat
    Generate $g \xleftarrow{\$} R$, g small
until g is invertible in R_3
Generate $f \xleftarrow{\$} Short$
Compute $g_{inv} \leftarrow 1/g$ in R_3
Compute $h \leftarrow g/(3 \cdot f)$ in R_q
Return (f, g_{inv}) and h.

Observation: secret polynomials are inverted.

Motivation

Processing secrets requires **protection** against side-channel adversaries.

Motivation

Processing secrets requires **protection** against side-channel adversaries.

Constant time implementations counter timing attacks.
Embedded devices: Attacker can measure **power consumption**.

Our Work

1. First procedure for masking polynomial inversion.

2. It is efficient.

3. Possible at arbitrary masking order.

Conceptual Considerations

Attacker Model

- All inversions happen in key generation, which is executed **once**.
- The only valid attack is a (profiled) Simple Power Analysis with **one attack measurement**.

Shuffling and Masking

- Standard countermeasure: shuffling (randomizing the execution order of steps within the inversion procedure)
- Shuffling: highly non-trivial, probably infeasible for optimized inversion implementations.
- Masking: very efficient against SPA, possible on arithmetic shares using Fermat inversion (expensive)
- Usually, first-order masking is sufficient against realistic attackers.

Masking?

- Based on Shamir's secret sharing
- Hiding secret values from the CPU that processes it by splitting them up in multiple random shares
- Usual in PQC: boolean and additive masking

Basic Idea

$$a = \boxed{a_1} \oplus \boxed{a_2}$$

$$b = \boxed{b_1} \oplus \boxed{b_2}$$

Basic Idea

RUB

$$a = \boxed{a_1} \oplus \boxed{a_2}$$

$$\oplus$$

$$b = \boxed{b_1} \oplus \boxed{b_2}$$

$$a \oplus b$$

Basic Idea

$$
\begin{array}{ccccc}
a & = & \boxed{a_1} & \oplus & \boxed{a_2} \\
\oplus & & \oplus & & \oplus \\
b & = & \boxed{b_1} & \oplus & \boxed{b_2}
\end{array}
$$

$$
a \oplus b = \boxed{a_1 \oplus b_1} \oplus \boxed{a_2 \oplus b_2}
$$

Basic Idea

$$a = \boxed{a_1} + \boxed{a_2}$$

$$b = \boxed{b_1} + \boxed{b_2}$$

Basic Idea

$$a = \boxed{a_1} + \boxed{a_2}$$

$$+$$

$$b = \boxed{b_1} + \boxed{b_2}$$

$$a + b$$

Basic Idea

$$
\begin{array}{ccccc}
a & = & \boxed{a_1} & + & \boxed{a_2} \\
+ & & + & & + \\
b & = & \boxed{b_1} & + & \boxed{b_2}
\end{array}
$$

$$
a + b = \boxed{a_1 + b_1} + \boxed{a_2 + b_2}
$$

Basic Idea

Observation: Applying functions that are **linear in the masking domain** is cheap.

Basic Idea

Observation: Applying functions that are **linear in the masking domain** is cheap.

In which masking domain is **polynomial inversion linear**?

Polynomial-Multiplicative Masking

Old Idea: Multiplicative Masking

New: Shares are polynomials, whose polynomial product yields the secret polynomial.

Polynomial-Multiplicative Masking

Old Idea: Multiplicative Masking

New: Shares are polynomials, whose polynomial product yields the secret polynomial.

Old problem: Masking zero.

No new problem: The zero polynomial is not invertible and will never be masked.

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \boxed{m_1} \times \boxed{m_2}$$

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$\boxed{r}$$

$$a = \boxed{m_1} \times \boxed{m_2}$$

- $1\times$ sampling a random polynomial

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$\boxed{r a_1} + \boxed{r a_2} \qquad \boxed{r}$$

$$a = \boxed{m_1} \times \boxed{m_2}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \left(\boxed{r a_1} + \boxed{r a_2} \right) \times \boxed{r^{-1}}$$

$$a = \boxed{m_1} \times \boxed{m_2}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication
- $1\times$ polynomial inversion

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \boxed{\left(\boxed{r a_1} + \boxed{r a_2} \right)} \times \boxed{r^{-1}}$$

$$a = \boxed{m_1} \times \boxed{m_2}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication
- $1\times$ polynomial inversion
- $1\times$ polynomial addition

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \boxed{\left(\boxed{r a_1} + \boxed{r a_2}\right)} \times \boxed{r^{-1}}$$

$$a = \boxed{m_1} \times \boxed{m_2}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication
- $1\times$ polynomial inversion
- $1\times$ polynomial addition

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \boxed{(\ \boxed{r a_1} + \boxed{r a_2}\ )} \times \boxed{r^{-1}}$$

$$a^{-1} = \boxed{m_1^{-1}} \times \boxed{m_2^{-1}}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication
- $3\times$ polynomial inversion
- $1\times$ polynomial addition

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \boxed{(\boxed{r a_1} + \boxed{r a_2})} \times \boxed{r^{-1}}$$

$$a^{-1} = \boxed{m_1^{-1}} \times \boxed{m_2^{-1}}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication
- $3\times$ polynomial inversion
- $1\times$ polynomial addition
- $+$ conversion back to additive

Additive to Polynomial Multiplicative Conversion

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \left(\boxed{r\,a_1} + \boxed{r\,a_2} \right) \times \boxed{r^{-1}}$$

$$a^{-1} = \boxed{m_1^{-1}} \times \boxed{m_2^{-1}}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication
- $3\times$ polynomial inversion
- $1\times$ polynomial addition
- $+$ conversion back to additive

Additive to Polynomial Multiplicative Conversion

RUB

$$a = \boxed{a_1} + \boxed{a_2}$$

$$a = \boxed{(\; r\,a_1 + r\,a_2 \;)} \times \boxed{r}$$

$$a^{-1} = \boxed{m_1^{-1}} \times \boxed{m_2}$$

- $1\times$ sampling a random polynomial
- $2\times$ polynomial multiplication
- ~~$3\times$~~ $1\times$ polynomial inversion
- $1\times$ polynomial addition
- $+$ conversion back to additive

Invertibility: Two Cases

BIKE and NTRU

- There is an easy way of sampling invertible polynomials.
- BIKE: The polynomial must have an odd weight (sample uniform, check & correct).
- NTRU: All non-zero polynomials are invertible (sample uniform).

Invertibility: Two Cases

BIKE and NTRU

- There is an easy way of sampling invertible polynomials.
- BIKE: The polynomial must have an odd weight (sample uniform, check & correct).
- NTRU: All non-zero polynomials are invertible (sample uniform).

Streamlined NTRU Prime

- No easy way to sample invertible polynomials
- Even the (shared) input polynomial might not be invertible
- Invertibility check is done by inverting
- Uniform random polynomials are invertible with high probability
- Solution:
  1. Sample uniform random r
  2. If inversion of $ra_0 + ra_1$ fails, a or r were not invertible, start over

Multiplicative to Additive Conversion

$$p \quad = \quad a \quad \times \quad m$$

- Idea: usually the inverted polynomial is multiplied by another polynomial

Multiplicative to Additive Conversion

$$p \quad = \quad a \quad \times \quad m$$

$$p \quad = \left(\boxed{a_1} + \boxed{a_2} \right) \times \boxed{m_1} \times \boxed{m_2}$$

- Idea: usually the inverted polynomial is multiplied by another polynomial

Multiplicative to Additive Conversion

$$p \quad = \quad a \quad \times \quad m$$

$$p \quad = \left(\boxed{a_1} + \boxed{a_2} \right) \times \boxed{m_1} \times \boxed{m_2}$$

$$\boxed{a_1} + \boxed{a_2}$$

- Idea: usually the inverted polynomial is multiplied by another polynomial

Multiplicative to Additive Conversion

$$p \quad = \quad a \quad \times \quad m$$

$$p \quad = \left(\boxed{a_1} + \boxed{a_2} \right) \times \boxed{m_1} \times \boxed{m_2}$$

$$\boxed{a_1 m_1} + \boxed{a_2 m_1}$$

- Idea: usually the inverted polynomial is multiplied by another polynomial
- Cost: 2 polynomial multiplications

Multiplicative to Additive Conversion

$$p = a \times m$$

$$p = \left(\boxed{a_1} + \boxed{a_2} \right) \times \boxed{m_1} \times \boxed{m_2}$$

$$p = \boxed{a_1 m_1 m_2} + \boxed{a_2 m_1 m_2}$$

- Idea: usually the inverted polynomial is multiplied by another polynomial
- Cost: 4 polynomial multiplications

Multiplicative to Additive Conversion

$$p \quad = \quad a \quad \times \quad m$$

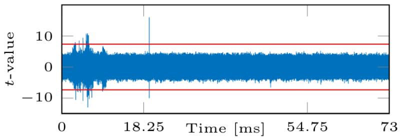
$$p \quad = \left(\boxed{a_1} + \boxed{a_2} \right) \times \boxed{m_1} \times \boxed{m_2}$$

$$p \quad = \boxed{a_1 m_1 m_2} + \boxed{a_2 m_1 m_2}$$

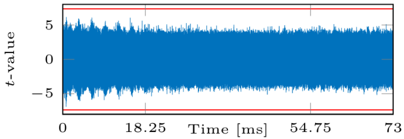
- Idea: usually the inverted polynomial is multiplied by another polynomial
- Cost: 4 polynomial multiplications
- Higher orders: re-sharing needed as intermediate steps
- Multiplication of two secret polynomials in additive domain would require re-sharing already for first order!

Side-Channel Evaluation

Additive to Polynomial-Multiplicative Conversion with implicit Inversion



t-test (2 000 traces) without randomness:
leakage



t-test (100 000 traces) with randomness:
no leakage

More evaluation: see the paper.

Performance Evaluation

NTRU-HPS-2048677

Ord. d	A2M Inversion			M2A Mul.			M2A Conversion		
	Cycles	MUL	INV	Cycles	MUL	ADD	Cycles	MUL	ADD
1	1 723 778	2	1	885 773	4	4	486 165	2	3
2	2 372 502	5	1	2 090 841	9	12	1 230 767	5	8
3	3 211 410	9	1	3 802 004	16	24	2 238 833	9	15
4	4 260 732	14	1	6 057 128	25	40	3 503 189	14	24
5	5 524 861	20	1	8 848 501	36	60	5 049 140	20	35
6	6 991 050	27	1	12 097 869	49	84	6 859 272	27	48

Unprotected operations:

- Addition: 18 340 clock cycles
- Multiplication: 201 383 clock cycles
- Inversion: 1 273 864 clock cycles

Performance Evaluation

NTRU-HPS-2048677

Ord. d	A2M Inversion			M2A Mul.			M2A Conversion		
	Cycles	MUL	INV	Cycles	MUL	ADD	Cycles	MUL	ADD
1	1 723 778	2	1	885 773	4	4	486 165	2	3
2	2 372 502	5	1	2 090 841	9	12	1 230 767	5	8
3	3 211 410	9	1	3 802 004	16	24	2 238 833	9	15
4	4 260 732	14	1	6 057 128	25	40	3 503 189	14	24
5	5 524 861	20	1	8 848 501	36	60	5 049 140	20	35
6	6 991 050	27	1	12 097 869	49	84	6 859 272	27	48

Unprotected operations:

- Addition: 18 340 clock cycles
- Multiplication: 201 383 clock cycles
- Inversion: 1 273 864 clock cycles

1st order A2M with inversion
vs unprotected inversion:
35 % overhead

Performance Evaluation

RUB

NTRU-HPS-2048677

Ord. d	A2M Inversion			M2A Mul.			M2A Conversion		
	Cycles	MUL	INV	Cycles	MUL	ADD	Cycles	MUL	ADD
1	1 723 778	2	1	885 773	4	4	486 165	2	3
2	2 372 502	5	1	2 090 841	9	12	1 230 767	5	8
3	3 211 410	9	1	3 802 004	16	24	2 238 833	9	15
4	4 260 732	14	1	6 057 128	25	40	3 503 189	14	24
5	5 524 861	20	1	8 848 501	36	60	5 049 140	20	35
6	6 991 050	27	1	12 097 869	49	84	6 859 272	27	48

Unprotected operations:

- Addition: 18 340 clock cycles
- Multiplication: 201 383 clock cycles
- Inversion: 1 273 864 clock cycles

BIKE Level 1

Ord. d	A2M Inversion			M2A Mul.			M2A Conversion		
	Cycles	MUL	INV	Cycles	MUL	ADD	Cycles	MUL	ADD
1	21 317 392	2	1	4 240 017	4	4	2 131 405	2	3
2	24 487 146	5	1	9 584 999	9	12	5 342 630	5	8
3	28 736 397	9	1	17 068 753	16	24	9 622 491	9	15
4	34 007 250	14	1	26 740 596	25	40	14 994 627	14	24
5	40 275 530	20	1	38 507 790	36	60	21 419 851	20	35
6	47 744 390	27	1	52 493 255	49	84	28 945 019	27	48

Unprotected operations:

- Addition: 3 534 clock cycles
- Multiplication: 1 052 253 clock cycles
- Inversion: 19 182 916 clock cycles

1$^{\text{st}}$ order A2M with inversion
vs unprotected inversion:
35 % overhead

Performance Evaluation

NTRU-HPS-2048677

Ord. d	A2M Inversion			M2A Mul.			M2A Conversion		
	Cycles	MUL	INV	Cycles	MUL	ADD	Cycles	MUL	ADD
1	1 723 778	2	1	885 773	4	4	486 165	2	3
2	2 372 502	5	1	2 090 841	9	12	1 230 767	5	8
3	3 211 410	9	1	3 802 004	16	24	2 238 833	9	15
4	4 260 732	14	1	6 057 128	25	40	3 503 189	14	24
5	5 524 861	20	1	8 848 501	36	60	5 049 140	20	35
6	6 991 050	27	1	12 097 869	49	84	6 859 272	27	48

Unprotected operations:

- Addition: 18 340 clock cycles
- Multiplication: 201 383 clock cycles
- Inversion: 1 273 864 clock cycles

1st order A2M with inversion
vs unprotected inversion:
35 % overhead

BIKE Level 1

Ord. d	A2M Inversion			M2A Mul.			M2A Conversion		
	Cycles	MUL	INV	Cycles	MUL	ADD	Cycles	MUL	ADD
1	21 317 392	2	1	4 240 017	4	4	2 131 405	2	3
2	24 487 146	5	1	9 584 999	9	12	5 342 630	5	8
3	28 736 397	9	1	17 068 753	16	24	9 622 491	9	15
4	34 007 250	14	1	26 740 596	25	40	14 994 627	14	24
5	40 275 530	20	1	38 507 790	36	60	21 419 851	20	35
6	47 744 390	27	1	52 493 255	49	84	28 945 019	27	48

Unprotected operations:

- Addition: 3 534 clock cycles
- Multiplication: 1 052 253 clock cycles
- Inversion: 19 182 916 clock cycles

1st order A2M with inversion
vs unprotected inversion:
11 % overhead

Conclusion

- **Efficient** method for masking polynomial inversion to counter Simple Power Analysis

Conclusion

- **Efficient** method for masking polynomial inversion to counter Simple Power Analysis
- In the paper:
  - M2A conversion without implicit multiplication
  - Generalization to higher masking orders

Conclusion

RUB

- **Efficient** method for masking polynomial inversion to counter Simple Power Analysis
- In the paper:
  - M2A conversion without implicit multiplication
  - Generalization to higher masking orders
- Recent work by Coron et al.[1] shows theoretical vulnerability of A2M algorithm **starting from third order**
  - Can be mitigated with low cost (more random sampling)
  - Higher-order single-trace SPA attackers are rather theoretical

[1] "High-order masking of NTRU", `https://eprint.iacr.org/2022/1188`

Conclusion

- **Efficient** method for masking polynomial inversion to counter Simple Power Analysis
- In the paper:
  - M2A conversion without implicit multiplication
  - Generalization to higher masking orders
- Recent work by Coron et al.[1] shows theoretical vulnerability of A2M algorithm **starting from third order**
  - Can be mitigated with low cost (more random sampling)
  - Higher-order single-trace SPA attackers are rather theoretical
- Future work: formal proofs of our algorithms

[1] "High-order masking of NTRU", `https://eprint.iacr.org/2022/1188`