SMAUG: Pushing Lattice-based Key Encapsulation Mechanisms to the Limits

Jung Hee Cheon[1,2], **Hyeongmin Choe[1]**, Dongyeon Hong[3], MinJune Yi[1]

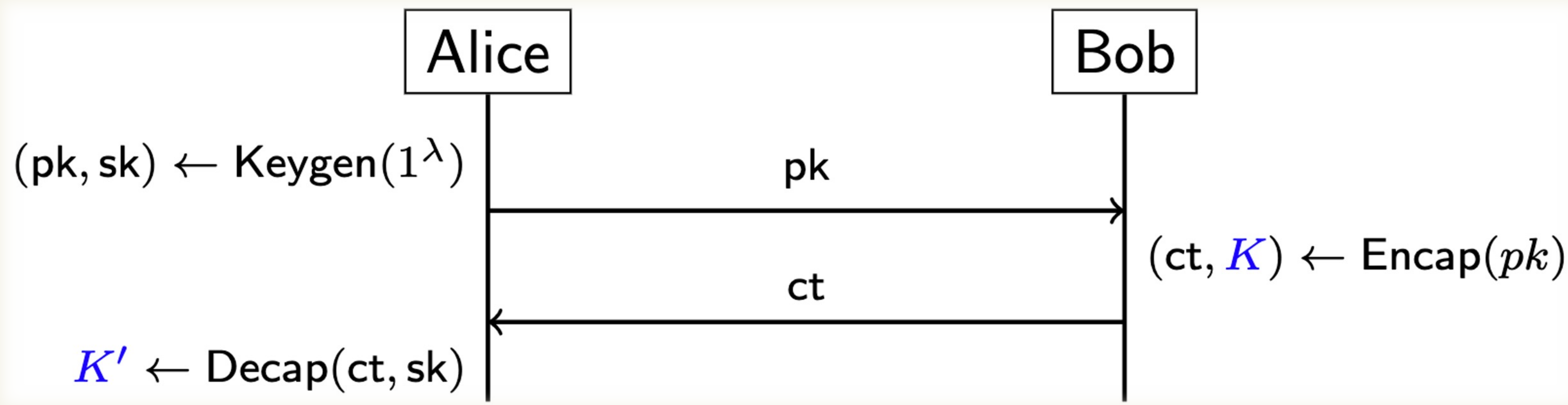
[1] Seoul National University, [2] CryptoLab Inc.,
[3] National Security Research Institute

August 16, 2023
SAC 2023

Lattice-based KEMs

KEMs in Post-Quantum World

- Key Encapsulation Mechanism (KEM)



KEMs in Post-Quantum World

- Key Encapsulation Mechanism (KEM)

Internet

TLS protocols

IoT devices

KEMs in Post-Quantum World

- Key Encapsulation Mechanism (KEM)

Internet

TLS protocols

IoT devices

- Current KEMs: vulnerable to quantum attacks

KEMs in Post-Quantum World

- Key Encapsulation Mechanism (KEM)

Internet

TLS protocols

IoT devices

- Current KEMs: vulnerable to quantum attacks

⇛ Since 2017, NIST PQC standardization is ongoing!

KEMs in Post-Quantum World

- Key Encapsulation Mechanism (KEM)

Internet　　TLS protocols

IoT devices

- Current KEMs: vulnerable to quantum attacks

$\Rightarrow$ Since 2017, NIST PQC standardization is ongoing!

Various lattice-based KEMs:

Kyber, Saber, NTRU, Round5, FrodoKEM, Rlizard,…

Requirements for KEMs

- Efficiency

- Security

Requirements for KEMs

- Efficiency
  - Small sizes

- Security

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance

- Security

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance

- Secure against…
  - IND-CCA2 attacks

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance

- Secure against…
  - Core-SVP hardness

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance
- Secure against…
  - Core-SVP hardness
  - Decryption failure attacks

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance

- Secure against…
  - Core-SVP hardness
  - Decryption failure attacks
  - Side-channel attacks

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance
- How to?
  - Module lattices
  - LWR problem
  - Centered Binomial Distribution (CBD)

- Secure against…
  - Core-SVP hardness
  - Decryption failure attacks
  - Side-channel attacks

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance
- How to?
  - Module lattices
  - LWR problem
  - Centered Binomial Distribution (CBD)

- Secure against…
  - Core-SVP hardness
  - Decryption failure attacks
  - Side-channel attacks
- How to?
  - Ring lattices
  - LWE problem
  - Error Correction Codes (ECC)

Requirements for KEMs

- Efficiency
  - Small sizes
  - Fast performance
- How to?
  - Module lattices
  - LWR problem
  - Centered Binomial Distribution (CBD)

- Secure against…
  - Core-SVP hardness
  - Decryption failure attacks
  - Side-channel attacks
- How to?
  - Ring lattices
  - LWE problem
  - ~~Error Correction Codes (ECC)~~

Efficiency of Lattice-KEMs

- Recent lattice-based KEMs with low enough Decryption Failure Probability (DFP)
  - KEMs using ECCs or having higher DFPs are omitted

- FrodoKEM
- RLizard
- Kyber
  - LWE
  - Module
  - CBD
- Saber
  - LWR
  - Module
  - CBD
- NTRU
  - NTRU
  - Sparse secret

Efficiency of Lattice-KEMs

- Recent lattice-based KEMs with low enough Decryption Failure Probability (DFP)
  - KEMs using ECCs or having higher DFPs are omitted

- FrodoKEM
- RLizard
- Kyber
  - LWE
  - Module
  - CBD
- Saber
  - LWR
  - Module
  - CBD
- NTRU
  - NTRU
  - Sparse secret
- Sable (CHES'21)
  - Saber variant
  - Optimized sizes & performance

Efficiency of Lattice-KEMs

- Recent lattice-based KEMs with low enough Decryption Failure Probability (DFP)
  - KEMs using ECCs or having higher DFPs are omitted

- Sable (CHES'21)
  - Saber variant
  - Optimized sizes & performance
- NTRU
  - NTRU
  - Sparse secret
- Saber
  - LWR
- FrodoKEM

Scheme	sk	pk	ct ↑	DFP	Sec.	$\|K\|$	Assumption
Sable	800	608	672	-139	114	256	MLWR
NTRU	699	935	699	$-\infty$	106	256	NTRU
Saber	832	672	736	-120	118	256	MLWR
Kyber	1632	800	768	-139	118	256	MLWE
RLizard	385	4096	2080	-188	147	256	RLWE+RLWR
FrodoKEM	19888	9616	9752	-139	150	128	LWE

Can we further push efficiency of lattice-KEMs towards the limit?

Can we further push efficiency of lattice-KEMs towards the limit?

$\Rrightarrow$ **SMAUG**

Can we further push **efficiency** of lattice-KEMs towards the limit?

$\Rrightarrow$ **SMAUG**

- Module LWE & LWR problem
- Sparse secret
- Approximate discrete Gaussian

Can we further push efficiency of lattice-KEMs towards the limit?

Scheme	sk	pk	ct ↑	DFP	Sec.	$\lvert K \rvert$	Assumption
SMAUG	176	672	672	-120	120	256	MLWE+MLWR
Sable	800	608	672	-139	114	256	MLWR
NTRU	699	935	699	$-\infty$	106	256	NTRU
Saber	832	672	736	-120	118	256	MLWR
Kyber	1632	800	768	-139	118	256	MLWE
RLizard	385	4096	2080	-188	147	256	RLWE+RLWR
FrodoKEM	19888	9616	9752	-139	150	128	LWE

SMAUG

SMAUG

- IND-CPA secure PKE
  - MLWE: key generation

SMAUG

- IND-CPA secure PKE
  - MLWE: key generation
  - MLWR: encryption

SMAUG

- IND-CPA secure PKE
  - MLWE: key generation
  - MLWR: encryption

- + Sparse secret
  - Lower DFP
  - Sparsity-based faster operations

SMAUG

- IND-CPA secure PKE
  - MLWE: key generation
  - MLWR: encryption
- + Sparse secret
  - Lower DFP
  - Sparsity-based faster operations
- + Approximate discrete Gaussian
  - Fast and parallelizable

SMAUG

- IND-CPA secure PKE
  - MLWE: key generation
  - MLWR: encryption
- + Sparse secret
  - Lower DFP
  - Sparsity-based faster operations
- + Approximate discrete Gaussian
  - Fast and parallelizable

FO transform

$\Rrightarrow$ IND-CCA2 secure KEM

SMAUG

- IND-CPA secure PKE
  - MLWE: key generation
  - MLWR: encryption
- + Sparse secret
  - Lower DFP
  - Sparsity-based faster operations
- + Approximate discrete Gaussian
  - Fast and parallelizable

FO transform

$\Rightarrow$ IND-CCA2 secure KEM

Why MLWE + MLWR?

Why MLWE + MLWR?

- (M)LWE

$$b = (As + e + \Delta\mu \;\; mod\; q),\;\; e \leftarrow D_{\sigma}\text{: small}$$

Why MLWE + MLWR?

- (M)LWE

$$b = (As + e + \Delta\mu \ \ mod\ q),\ \ e \leftarrow D_\sigma\text{: small}$$

  - (+) Small noise $\Rrightarrow$ Decryption error ⇩

Why MLWE + MLWR?

- (M)LWE

$$b = (As + e + \Delta\mu \ \ mod\ q),\ e \leftarrow D_\sigma\text{: small}$$

  - (+) Small noise $\Rrightarrow$ Decryption error ⇩
  - (−) Noise sampling $\Rrightarrow$ Performance ⇩

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩
- (M)LWR

$$b = \left\lfloor \frac{p}{q} \cdot (As + \Delta\mu \ \ mod\ q) \right\rceil$$

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise $\Rightarrow$ Decryption error ⇩
  - (−) Noise sampling $\Rightarrow$ Performance ⇩
- (M)LWR

$$b = \left\lfloor \frac{p}{q} \cdot (As + \Delta\mu \ \ mod\ q) \right\rceil$$

$\approx$ (M)LWE with $e \leftarrow \mathrm{unif}\left(-\frac{p}{2q}, \cdots, \frac{p}{2q}\right]$

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩

- (M)LWR

$$b = \left\lfloor \frac{p}{q} \cdot (As + \Delta\mu \ \ mod \ q) \right\rceil$$

$$\approx \text{(M)LWE with } e \leftarrow \text{unif}\left(-\frac{p}{2q}, \cdots, \frac{p}{2q}\right]$$

  - (+) Scaling & rounding ⇛ Performance ⇧

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩

- (M)LWR

$$b = \left\lfloor \frac{p}{q} \cdot (As + \Delta\mu \ \ mod\ q) \right\rceil$$

$$\approx \text{(M)LWE with } e \leftarrow \mathrm{unif}\left(-\frac{p}{2q}, \cdots, \frac{p}{2q}\right]$$

  - (+) Scaling & rounding ⇛ Performance ⇧
  - (−) Rounding error ⇛ Decryption error ⇧

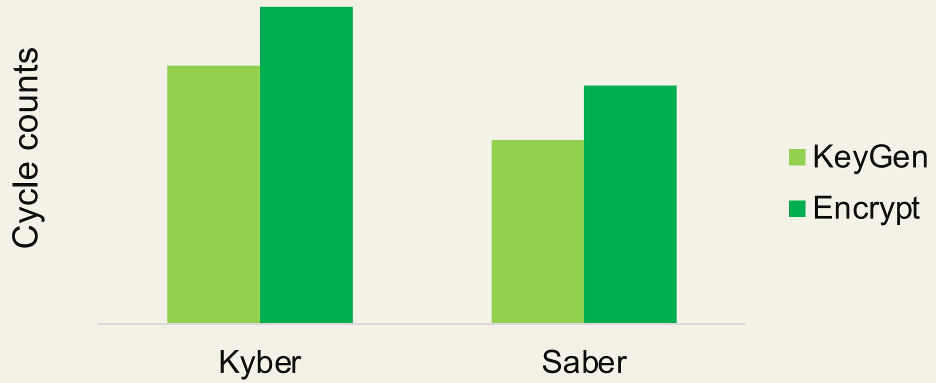
Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩

- (M)LWR
  - (+) Scaling & rounding ⇛ Performance ⇧
  - (−) Rounding error ⇛ Decryption error ⇧

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩
- (M)LWR
  - (+) Scaling & rounding ⇛ Performance ⇧
  - (−) Rounding error ⇛ Decryption error ⇧

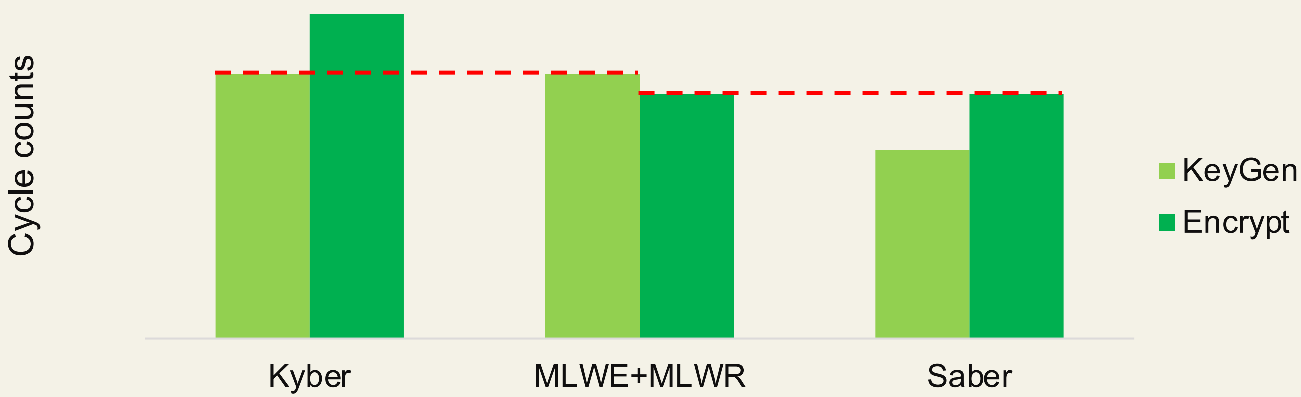


DFP: 2^{-139} 2^{-120}

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩

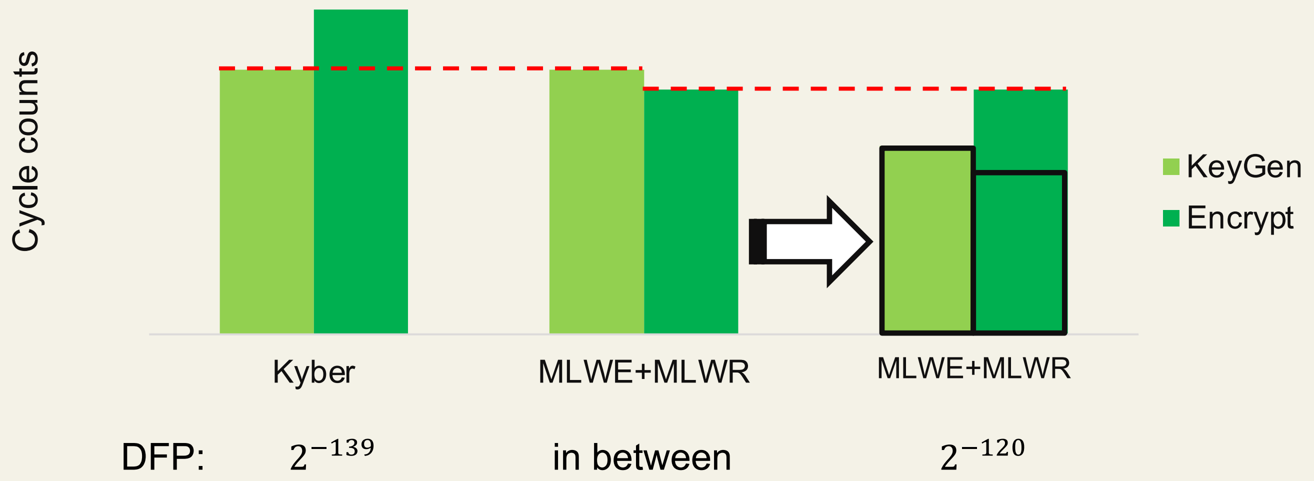
- (M)LWR
  - (+) Scaling & rounding ⇛ Performance ⇧
  - (−) Rounding error ⇛ Decryption error ⇧



DFP: 2^{-139} in between 2^{-120}

Why MLWE + MLWR?

- (M)LWE
  - (+) Small noise ⇛ Decryption error ⇩
  - (−) Noise sampling ⇛ Performance ⇩
- (M)LWR
  - (+) Scaling & rounding ⇛ Performance ⇧
  - (−) Rounding error ⇛ Decryption error ⇧



Sparse Secret

Sparse Secret

- Homomorphic encryption
  - Noise propagation ⇩

Sparse Secret

- Homomorphic encryption
  - Noise propagation ⇩
  - Homomorphic operations speed ⇧

Sparse Secret

- Homomorphic encryption
  - Noise propagation ⇩
  - Homomorphic operations speed ⇧
- PKE
  - Decryption error ⇩

Sparse Secret

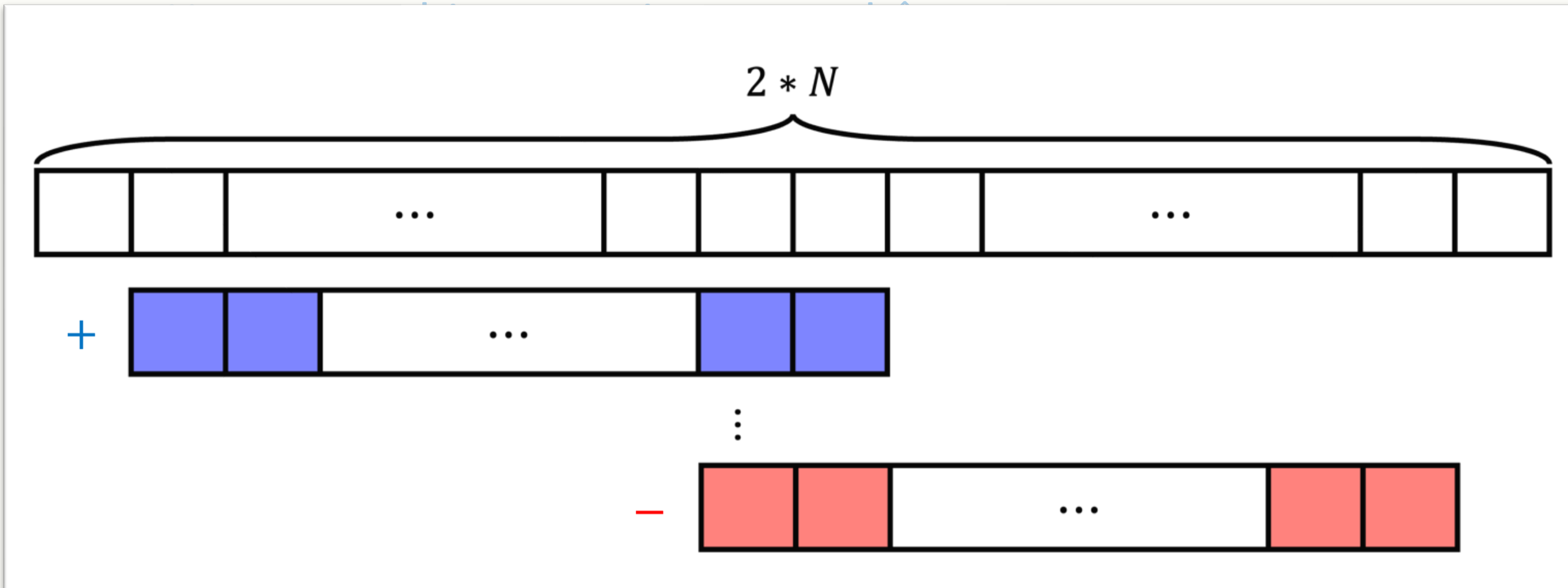
- Homomorphic encryption
  - Noise propagation ⇩
  - Homomorphic operations speed ⇧
- PKE
  - Decryption error ⇩
  - Performance ⇧

Sparse Secret

- Homomorphic encryption
  - Noise propagation ⇩
  - Homomorphic operations speed ⇧
- PKE
  - Decryption error ⇩
  - Performance ⇧
- Polynomial multiplication
  - Schoolbook multiplication using $+/-$

Sparse Secret

- Homomorphic encryption
  - Noise propagation ⇩



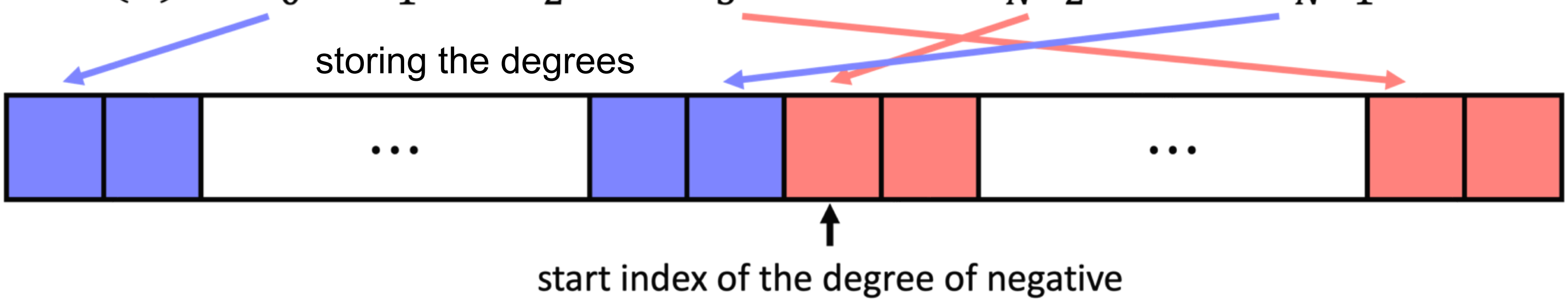
Sparse Secret

- Homomorphic encryption
  - Noise propagation ⇩
  - Homomorphic operations speed ⇧
- PKE
  - Decryption error ⇩
  - Performance ⇧
- Polynomial multiplication
  - Schoolbook multiplication using $+/-$
- Small secret key
  - Ready-to-use

Sparse Secret

- Homomorphic encryption
  - Noise propagation ⇩
  - Homomorphic operations speed ⇧

$$a(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_{N-2}x^{N-2} + a_{N-1}x^{N-1}$$



- Small secret key
  - Ready-to-use

Approximating Discrete Gaussian

Approximating Discrete Gaussian

- Scale dGaussian

Approximating Discrete Gaussian

- Scale dGaussian
  - Bound security loss using Réyni divergence

Parameter set	Scale factor	α	R_α	ΔSecurity
SMAUG-128	2^{10}	200	1.0016	1.8
SMAUG-192	2^{11}	75	1.0022	4.8
SMAUG-256	2^{10}	200	1.0016	5.7

Approximating Discrete Gaussian

- Scale dGaussian
  - Bound security loss using Réyni divergence

Parameter set	Scale factor	α	R_α	ΔSecurity
SMAUG-128	2^{10}	200	1.0016	1.8
SMAUG-192	2^{11}	75	1.0022	4.8
SMAUG-256	2^{10}	200	1.0016	5.7

  - Only for KeyGen $\Rrightarrow$ efficiently bounded!

Approximating Discrete Gaussian

- Scale dGaussian
  - Bound security loss using Réyni divergence

Parameter set	Scale factor	α	R_α	ΔSecurity
SMAUG-128	2^{10}	200	1.0016	1.8
SMAUG-192	2^{11}	75	1.0022	4.8
SMAUG-256	2^{10}	200	1.0016	5.7

  - Only for KeyGen $\Rightarrow$ efficiently bounded!

- Cumulative Distribution Table (CDT)

Approximating Discrete Gaussian

- Scale dGaussian
  - Bound security loss using Réyni divergence

Parameter set	Scale factor	α	R_α	ΔSecurity
SMAUG-128	2^{10}	200	1.0016	1.8
SMAUG-192	2^{11}	75	1.0022	4.8
SMAUG-256	2^{10}	200	1.0016	5.7

  - Only for KeyGen $\Rrightarrow$ efficiently bounded!
- Cumulative Distribution Table (CDT)
- Booleanize CDT
  - Quine-McCluskey's algorithm
  - Logic minimization

Approximating Discrete Gaussian

- Scale dGaussian
  - Bound security loss using Réyni divergence

Parameter set	Scale factor	α	R_α	ΔSecurity
SMAUG-128	2^{10}	200	1.0016	1.8
SMAUG-192	2^{11}	75	1.0022	4.8
SMAUG-256	2^{10}	200	1.0016	5.7

  - Only for KeyGen $\Rrightarrow$ efficiently bounded!
- Cumulative Distribution Table (CDT)
- Booleanize CDT
  - Quine-McCluskey's algorithm
  - Logic minimization

$\Rrightarrow$ Boolean algorithm for dGaussian

Approximating Discrete Gaussian

- Scale dGaussian
  - Bound security loss using Réyni divergence

<u>dGaussian$_\sigma(x)$:</u>

Require: $x = x_0x_1x_2x_3x_4x_5x_6x_7x_8x_9 \in \{0,1\}^{10}$

$s = s_1 s_0 = 00 \in \{0,1\}^2$

$s_0 = x_0x_1x_2x_3x_4x_5x_7\overline{x_8}$

$s_0 \mathrel{+}= (x_0x_3x_4x_5x_6x_8) + (x_1x_3x_4x_5x_6x_8) + (x_2x_3x_4x_5x_6x_8)$

$s_0 \mathrel{+}= (\overline{x_2x_3x_6}x_8) + (\overline{x_1x_3x_6}x_8)$

$s_0 \mathrel{+}= (x_6x_7\overline{x_8}) + (\overline{x_5x_6}x_8) + (\overline{x_4x_6}x_8) + (\overline{x_7}x_8)$

$s_1 = (x_1x_2x_4x_5x_7x_8) + (x_3x_4x_5x_7x_8) + (x_6x_7x_8)$

$s = (-1)^{x_9} \cdot s$ ▷ · is the arithmetic multiplication

return s

- Booleanize CDT
  - Quine-McCluskey's algorithm
  - Logic minimization

⇒ Boolean algorithm for dGaussian

Implementation

Parameter Sets

- Target: NIST’s security levels 1, 3, and 5

Parameter Sets

- Target: NIST's security levels 1, 3, and 5
- Security
  - Core-SVP hardness from Lattice-estimator
  - Algebraic/combinatorial attacks

Parameter Sets

- Target: NIST's security levels 1, 3, and 5

- Security
  - Core-SVP hardness from Lattice-estimator
  - Algebraic/combinatorial attacks
  - Especially for LWE problems with sparse secret

Parameter Sets

- Target: NIST's security levels 1, 3, and 5

- Security
  - Core-SVP hardness from Lattice-estimator
  - Algebraic/combinatorial attacks
  - Especially for LWE problems with sparse secret

- Decryption Failure Probability
  - At least as low as Saber

Parameter Sets

- Target: NIST's security levels 1, 3, and 5

- Security
  - Core-SVP hardness from Lattice-estimator
  - Algebraic/combinatorial attacks
  - Especially for LWE problems with sparse secret

- Decryption Failure Probability
  - At least as low as Saber

$\Rightarrow$ Smallest ciphertexts & public keys

Size Comparison

- NIST's security level 1

Schemes	Sizes (ratio)			Security	
	sk	pk	ct	Classic.	DFP
Kyber512	9.4	1.2	1.1	118	-139
LightSaber	4.8	1	1.1	118	-120
LightSable	4.6	0.9	1	114	-139
SMAUG-128	1	1	1	120	-120

- Sizes: proportion to SMAUG
- SMAUG wins, loses, tie

Full Size & Performance Comparison

- NIST's security levels 1, 3, and 5

Schemes	Sizes (ratio) sk	pk	ct	Cycles (ratio) KeyGen	Encap	Decap	Security Classic.	DFP
Kyber512	9.4	1.2	1.1	1.7	2.1	2.03	118	-139
LightSaber	4.8	1	1.1	1.21	1.58	1.44	118	-120
LightSable	4.6	0.9	1	1.1	1.48	1.39	114	-139
SMAUG-128	1	1	1	1	1	1	120	-120
Kyber768	10.4	1.1	1.1	1.38	1.84	1.75	183	-164
Saber	5.4	0.9	1.1	1.21	1.64	1.47	189	-136
Sable	5	0.8	1	1.1	1.55	1.45	185	-143
SMAUG-192	1	1	1	1	1	1	181	-136
Kyber1024	15.2	0.9	1.1	1.25	1.38	1.36	256	-174
FireSaber	8	0.7	1	1.08	1.29	1.25	260	-165
FireSable	7.8	0.7	0.9	1.03	1.25	1.22	223	-208
SMAUG-256	1	1	1	1	1	1	264	-167

- Constant-time, non-vectorized C reference codes
- Sizes & Cycles: proportion to SMAUG
- SMAUG wins, loses, tie

Conclusion

Conclusion

- Design of SMAUG:
  - MLWE key + MLWR ciphertext
  - Sparse secret and approximate dGaussian noise
  - Constant-time C reference code: www.kpqc.cryptolab.co.kr/smaug

Conclusion

- Design of SMAUG:
  - MLWE key + MLWR ciphertext
  - Sparse secret and approximate dGaussian noise
  - Constant-time C reference code: www.kpqc.cryptolab.co.kr/smaug
- Efficiency
  - Smallest[1] ciphertext sizes
  - Performance: 20-110% faster than Kyber, Saber, Sable

1. the smallest among lattice-KEMs with NIST's security level 1, 3, and 5, having low-enough DFP & maskable against SCAs

Conclusion

- Design of SMAUG:
  - MLWE key + MLWR ciphertext
  - Sparse secret and approximate dGaussian noise
  - Constant-time C reference code: www.kpqc.cryptolab.co.kr/smaug
- Efficiency
  - Smallest[1] ciphertext sizes
  - Performance: 20-110% faster than Kyber, Saber, Sable
- Answer to **the question**:

1. the smallest among lattice-KEMs with NIST's security level 1, 3, and 5, having low-enough DFP & maskable against SCAs

Conclusion

- Design of SMAUG:
  - MLWE key + MLWR ciphertext
  - Sparse secret and approximate dGaussian noise
  - Constant-time C reference code: www.kpqc.cryptolab.co.kr/smaug
- Efficiency
  - Smallest[1] ciphertext sizes
  - Performance: 20-110% faster than Kyber, Saber, Sable
- Answer to **the question**:

SMAUG achieves the smallest ciphertext sizes
with extra room for trade-off:

1. the smallest among lattice-KEMs with NIST's security level 1, 3, and 5, having low-enough DFP & maskable against SCAs

Conclusion

- Design of SMAUG:
  - MLWE key + MLWR ciphertext
  - Sparse secret and approximate dGaussian noise
  - Constant-time C reference code: www.kpqc.cryptolab.co.kr/smaug
- Efficiency
  - Smallest[1] ciphertext sizes
  - Performance: 20-110% faster than Kyber, Saber, Sable
- Answer to **the question**:

SMAUG achieves the smallest ciphertext sizes
with extra room for trade-off:
performance & small secret VS. small public key

1. the smallest among lattice-KEMs with NIST's security level 1, 3, and 5, having low-enough DFP & maskable against SCAs

*SMAUG, *The Hobbits*, J. R. R. Tolkien.