HAETAE

Jung Hee Cheon[1,2], **Hyeongmin Choe**[1], Julien Devevey[3],
Tim Güneysu[4], Dongyeon Hong[2], Markus Krausz[4], Georg Land[4],
Marc Möller[4], Junbum Shin[2], Damien Stehlé[5], MinJune Yi[1,2]

[1]**Seoul National University**, [2]CryptoLab Inc.,
[3]ANSSI (FR), [4]Ruhr Universität Bochum (DE), [5]CryptoLab Inc. (FR)

February 27, 2024
2024 KpqC Winter Camp

Table of Contents

HAETAE

- Digital signature scheme
- Secure against quantum attacks!
  - based on **lattice hard problems** MLWE and MSIS
  - follows **Fiat-Shamir with aborts** framework, secure in QROM
- Simple but short!
  - simpler than Falcon[1] & shorter than Dilithium[1]
  - optimal rejection rate with simple rejection condition
- Design rationale
  - **Bimodal** rejection sampling
  - **Hyperball** distribution
- Candidate in *KpqC 2nd round & NIST PQC Additional Signatures*[2]

[1] NIST 2022 PQC signature standards
[2] NIST's on-ramp PQC signature competition, from 2023.

HAETAE

- Digital signature scheme
- Secure against quantum attacks!
  - based on **lattice hard problems** MLWE and MSIS
  - follows **Fiat-Shamir with aborts** framework, secure in QROM
- Simple but short!
  - simpler than Falcon[1] & shorter than Dilithium[1]
  - optimal rejection rate with simple rejection condition
- Design rationale
  - **Bimodal** rejection sampling
  - **Hyperball** distribution
- Candidate in *KpqC 2nd round & NIST PQC Additional Signatures*[2]

[1] NIST 2022 PQC signature standards

[2] NIST's on-ramp PQC signature competition, from 2023.

HAETAE

- Digital signature scheme
- Secure against quantum attacks!
  - based on **lattice hard problems** MLWE and MSIS
  - follows **Fiat-Shamir with aborts** framework, secure in QROM
- Simple but short!
  - simpler than Falcon[1] & shorter than Dilithium[1]
  - optimal rejection rate with simple rejection condition
- Design rationale
  - **Bimodal** rejection sampling
  - **Hyperball** distribution
- Candidate in *KpqC 2nd round & NIST PQC Additional Signatures*[2]

[1] NIST 2022 PQC signature standards

[2] NIST's on-ramp PQC signature competition, from 2023.

HAETAE

- Digital signature scheme
- Secure against quantum attacks!
  - based on **lattice hard problems** MLWE and MSIS
  - follows **Fiat-Shamir with aborts** framework, secure in QROM
- Simple but short!
  - simpler than Falcon[1] & shorter than Dilithium[1]
  - optimal rejection rate with simple rejection condition
- Design rationale
  - **Bimodal** rejection sampling
  - **Hyperball** distribution
- Candidate in *KpqC 2nd round & NIST PQC Additional Signatures*[2]

[1] NIST 2022 PQC signature standards

[2] NIST's on-ramp PQC signature competition, from 2023.

HAETAE

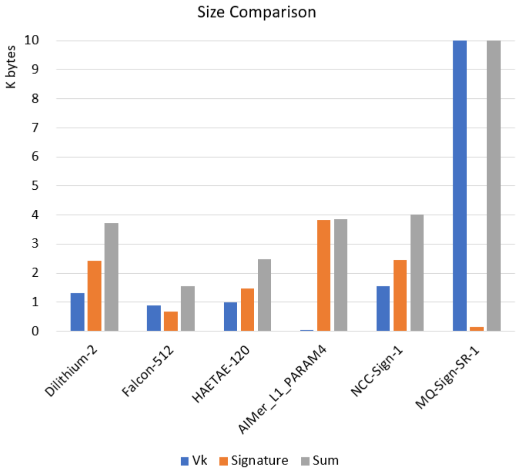
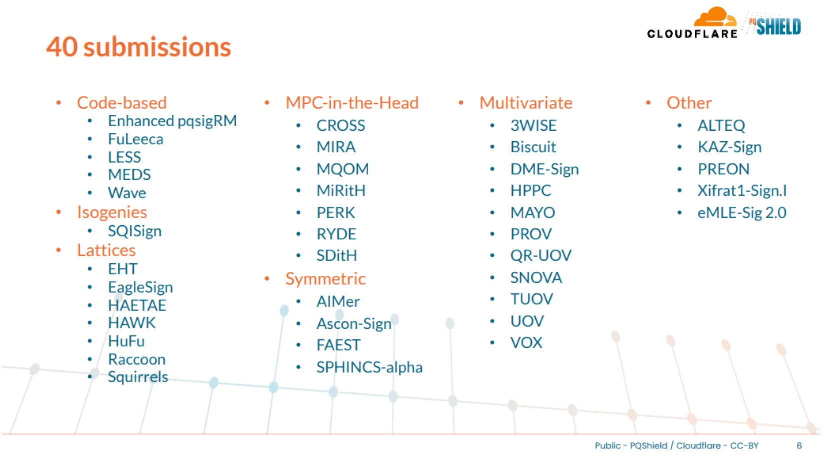


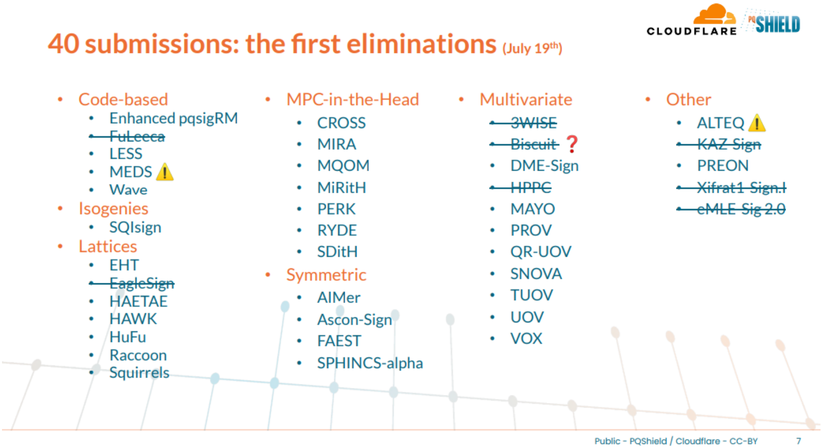
Figure: KpqC round 2, signature schemes

HAETAE



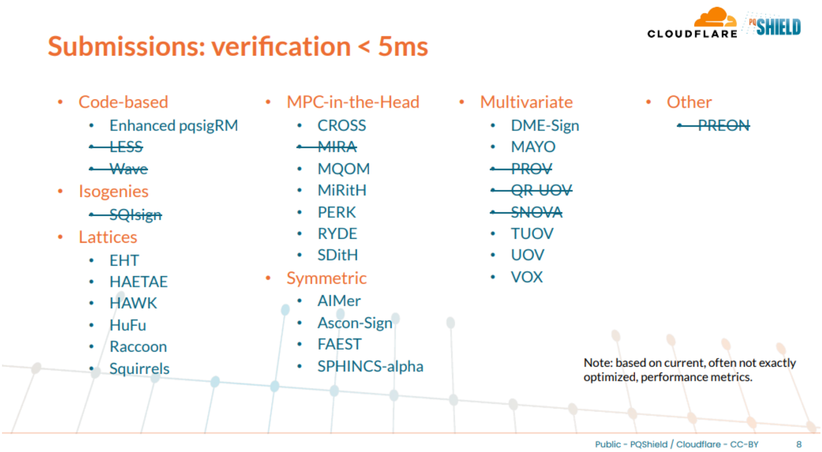
Slide from `https://datatracker.ietf.org/meeting/117/materials/slides-117-tls-new-post-quantum-signature-algorithms-on-the-horizon-00`

HAETAE



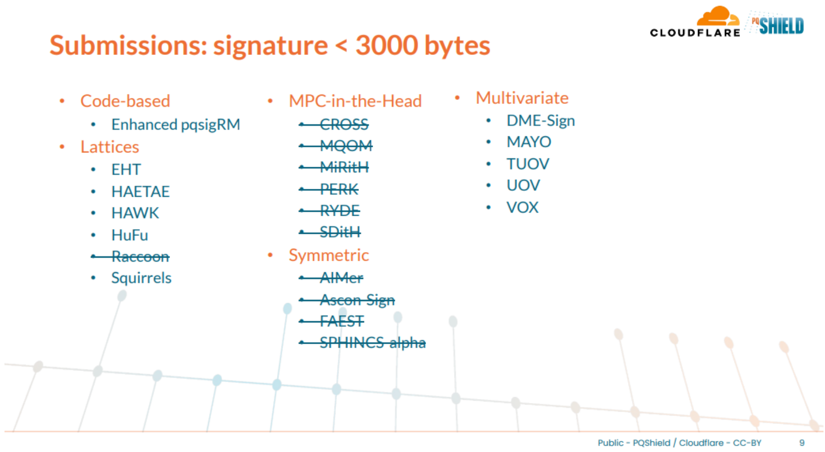
Slide from `https://datatracker.ietf.org/meeting/117/materials/slides-117-tls-new-post-quantum-signature-algorithms-on-the-horizon-00`

HAETAE



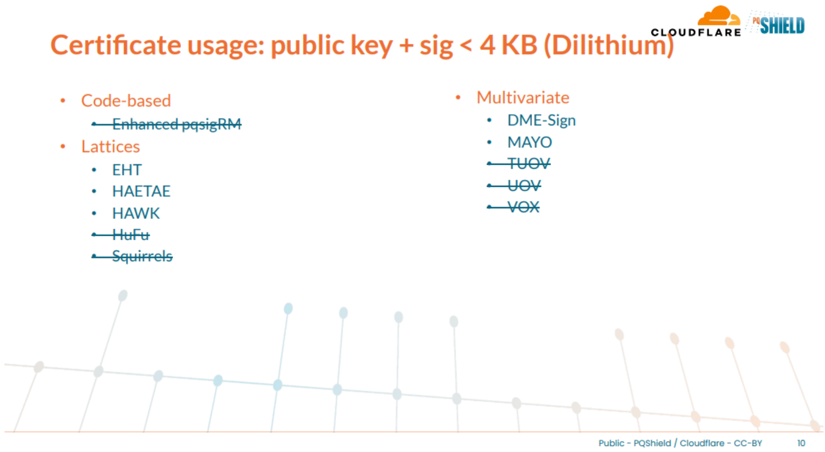
Slide from https://datatracker.ietf.org/meeting/117/materials/slides-117-tls-new-post-quantum-signature-algorithms-on-the-horizon-00

HAETAE



Slide from `https://datatracker.ietf.org/meeting/117/materials/slides-117-tls-new-post-quantum-signature-algorithms-on-the-horizon-00`

HAETAE



Slide from `https://datatracker.ietf.org/meeting/117/materials/slides-117-tls-new-post-quantum-signature-algorithms-on-the-horizon-00`

1. Brief Introduction to HAETAE

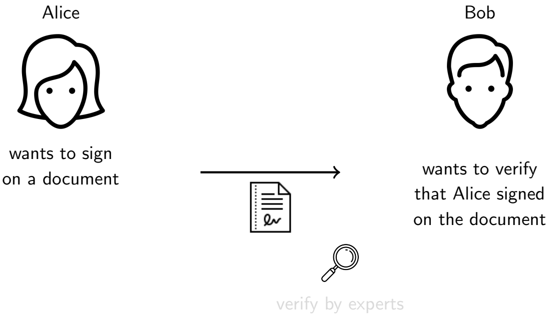
2. Preliminaries:
   - Digital signatures
   - Lattice hard problems
   - Lattice-based signatures
     - Bimodal rejection sampling

3. HAETAE:
   - Hyperball bimodal rejection sampling
   - Comparison to SotA lattice signatures

4. Changes after Round 1

Digital signatures

Conventional signatures:



Some images are from `https://kr.freepik.com/search?format=search&last_filter=type&last_value=icon&query=magnifier&selection=1&type=icon`

Digital signatures

Conventional signatures:

Some images are from https://kr.freepik.com/search?format=search&last_filter=type&last_value=icon&query=magnifier&selection=1&type=icon

Digital signatures

Conventional signatures:

Alice

wants to sign
on a document

Bob

wants to verify
that Alice signed
on the document

verify by experts

Digital signatures:

(sk, vk) $\leftarrow$ KeyGen and broadcast vk

Alice (knows sk)

signature $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$

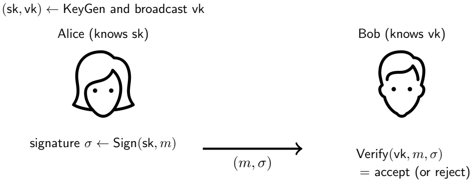
$$\xrightarrow[(m,\sigma)]{}$$

Bob (knows vk)

$\mathsf{Verify}(\mathsf{vk}, m, \sigma)$
$=$ accept (or reject)

Digital signatures

Digital signatures:



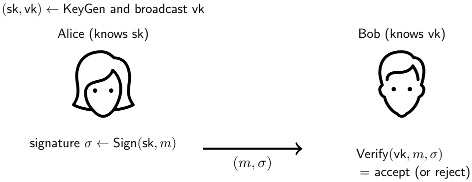
Anyone (who can access vk) can verify that (m, σ) is from Alice or not!

Correctness: Verify(vk, m, Sign(sk, m)) = accept

Unforgeability: No one but Alice can make a new signature.

Digital signatures

Digital signatures:



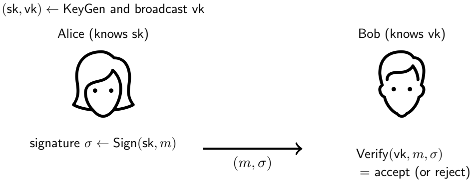
Anyone (who can access vk) can verify that (m, σ) is from Alice or not!

Correctness: $\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = \mathsf{accept}$

Unforgeability: No one but Alice can make a new signature.

Digital signatures

Digital signatures:



Anyone (who can access vk) can verify that (m, σ) is from Alice or not!

Correctness: Verify(vk, m, Sign(sk, m)) = accept

Unforgeability: No one but Alice can make a new signature.

1. Brief Introduction to HAETAE

2. Preliminaries:
   - Digital signatures
   - Lattice hard problems
   - Lattice-based signatures
     - Bimodal rejection sampling

3. HAETAE:
   - Hyperball bimodal rejection sampling
   - Comparison to SotA lattice signatures

4. Changes after Round 1

Lattice hard problems

Lattice-based cryptography is ... are currently important candidates for post-quantum cryptography.

- *Wikipedia* -

Lattice-based cryptography bases its security on lattice hard problems, which have strong theoretical backgrounds:

- SVP and GapSVP$_\lambda$: NP-hard! [Ajt96, HR07]
- Worst-case to average-case **reductions** [Ajt96]
- Useful hard problems: NTRU, LWE, SIS, MLWE, MSIS, etc

Lattice hard problems

Lattice-based cryptography is ... are currently important candidates for post-quantum cryptography.

- *Wikipedia* -

Lattice-based cryptography bases its security on lattice hard problems, which have strong theoretical backgrounds:

- SVP and GapSVP$_\lambda$: NP-hard! [Ajt96, HR07]
- Worst-case to average-case **reductions** [Ajt96]
- Useful hard problems: NTRU, LWE, SIS, MLWE, MSIS, etc

Lattice hard problems

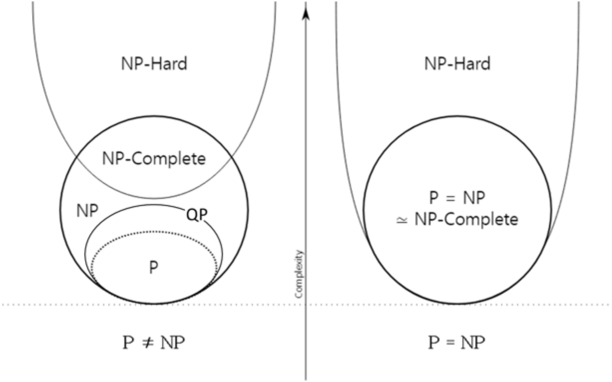
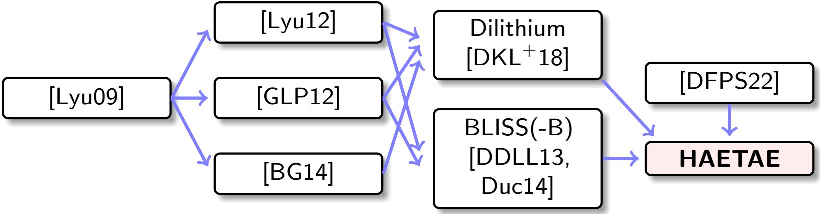


Figure: Category of hard problems when P≠NP and P=NP.

No proofs for Quantum Poly (QP), but is believed to be separated to NP-Hard problems.

Lattice-based signatures

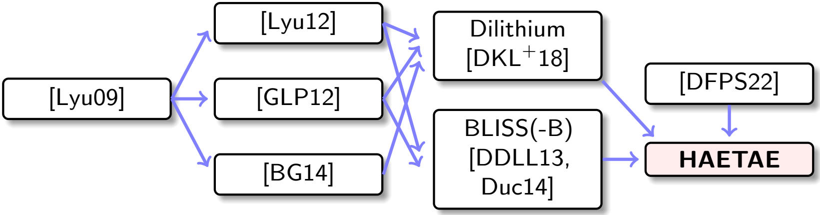
Fiat-Shamir with abort



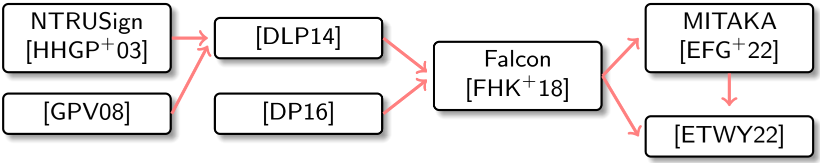
Hash-and-Sign

Lattice-based signatures

Fiat-Shamir with abort



Hash-and-Sign



Lattice-based signatures

Fiat-Shamir with abort:

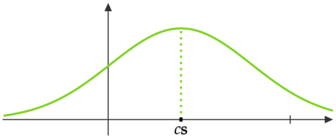
For secret $\mathbf{s}$, random $\mathbf{y}$, c, signature $\sigma = (c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$

$$\left(\boxed{8}, \begin{bmatrix} 9 \\ -9 \\ 6 \\ 2 \\ 1 \\ 11 \\ 8 \\ -9 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -2 \\ 2 \\ 1 \\ 3 \\ 0 \\ -1 \end{bmatrix} + \boxed{8} \begin{bmatrix} 1 \\ -1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ -1 \end{bmatrix} \right)$$

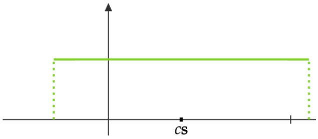
Lattice-based signatures

Leakage from $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$?

(High-level) With ∞ pairs of $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$, we may collect $\mathbf{z}$ for same c:



($y \leftarrow \mathcal{N}(0, \sigma^2)$)



($y \leftarrow U[-a, a]$)

$\Rightarrow$ Recover $\mathbf{s}$ from $c\mathbf{s}$.

How to make it safe?

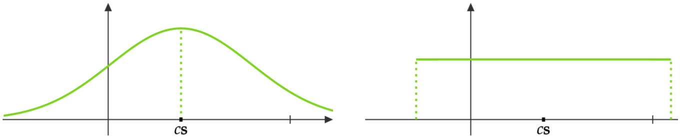
$$(c, \mathbf{z} = \mathbf{y} + c\mathbf{z}) \xrightarrow[\text{Rejection Sampling}]{\text{several trials}} \sigma = (c, \mathbf{z} = \mathbf{y} + c\mathbf{z})$$

not safe safe

Lattice-based signatures

Leakage from $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$?

(High-level) With ∞ pairs of $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$, we may collect $\mathbf{z}$ for same c:



$(\ y \leftarrow \mathcal{N}(0, \sigma^2)\ )$ $(\ y \leftarrow U[-a, a]\ )$

$\Rightarrow$ Recover s from cs.

How to make it safe?

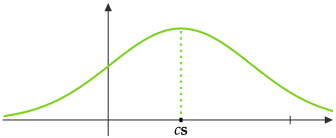
$$(c, \mathbf{z} = \mathbf{y} + c\mathbf{z}) \xrightarrow[\text{Rejection Sampling}]{\text{several trials}} \sigma = (c, \mathbf{z} = \mathbf{y} + c\mathbf{z})$$

not safe safe

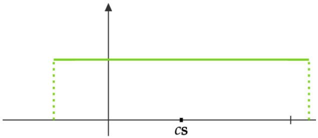
Lattice-based signatures

Leakage from $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$?

(High-level) With ∞ pairs of $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$, we may collect $\mathbf{z}$ for same c:



($y \leftarrow \mathcal{N}(0, \sigma^2)$)



($y \leftarrow U[-a, a]$)

$\Rightarrow$ Recover $\mathbf{s}$ from $c\mathbf{s}$.

How to make it safe?

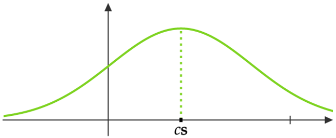
$$(c, \mathbf{z} = \mathbf{y} + c\mathbf{z}) \xrightarrow[\text{Rejection Sampling}]{\text{several trials}} \sigma = (c, \mathbf{z} = \mathbf{y} + c\mathbf{z})$$

not safe safe

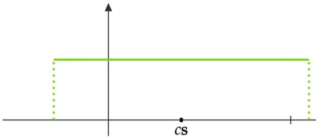
Lattice-based signatures

Leakage from $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$?

(High-level) With ∞ pairs of $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$, we may collect $\mathbf{z}$ for same c:



$(\ y \leftarrow \mathcal{N}(0, \sigma^2)\ )$



$(\ y \leftarrow U[-a, a]\ )$

$\Rightarrow$ Recover $\mathbf{s}$ from $c\mathbf{s}$.

How to make it safe?

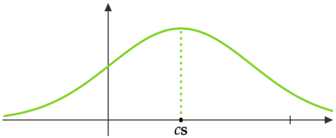
$$(c, \mathbf{z} = \mathbf{y} + c\mathbf{z}) \xrightarrow[\text{Rejection Sampling}]{\text{several trials}} \sigma = (c, \mathbf{z} = \mathbf{y} + c\mathbf{z})$$

not safe safe

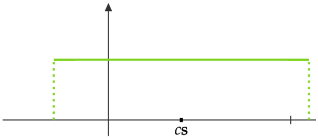
Lattice-based signatures

Leakage from $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$?

(High-level) With ∞ pairs of $(c, \mathbf{z} = \mathbf{y} + c\mathbf{s})$, we may collect $\mathbf{z}$ for same c:



($y \leftarrow \mathcal{N}(0, \sigma^2)$)



($y \leftarrow U[-a, a]$)

$\Rightarrow$ Recover $\mathbf{s}$ from $c\mathbf{s}$.

How to make it safe?

$$\underset{\text{not safe}}{(c, \mathbf{z} = \mathbf{y} + c\mathbf{z})} \xrightarrow[\text{Rejection Sampling}]{\text{several trials}} \underset{\text{safe}}{\sigma = (c, \mathbf{z} = \mathbf{y} + c\mathbf{z})}$$

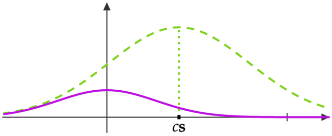
Rejection sampling

Rejection sampling

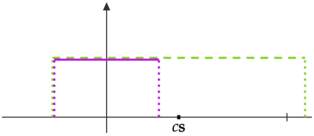
$$D_{\text{source}} = \{(c, \mathbf{z})\} \xrightarrow[\text{prob. } p(c,\mathbf{z})]{\text{reject with}} D_{\text{target}}$$

distribution of $(c, \mathbf{z})$, possibly leak $\mathbf{s}$

new distribution, independent of $\mathbf{s}$



$y \leftarrow \mathcal{N}(0, \sigma^2)$



$y \leftarrow U[-a, a]$

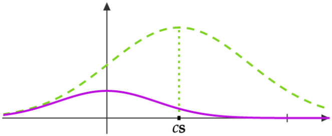
Rejection sampling

Rejection sampling

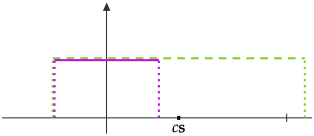
$$D_{\text{source}} = \{(c, \mathbf{z})\} \xrightarrow[\text{prob. } p(c,\mathbf{z})]{\text{reject with}} D_{\text{target}}$$

distribution of $(c, \mathbf{z})$, possibly leak $\mathbf{s}$

new distribution, independent of $\mathbf{s}$



$$y \leftarrow \mathcal{N}(0, \sigma^2)$$



$$y \leftarrow U[-a, a]$$

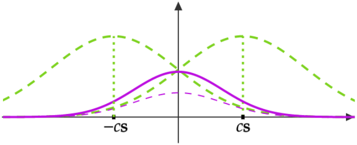
Bimodal rejection sampling

Run-time $\propto M$ ($\approx$ green area / purple area).

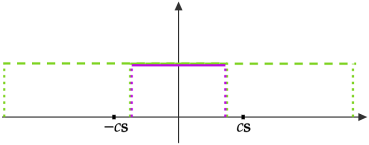
To decrease M, [DDLL13] uses

$$\mathbf{z} = \mathbf{y} + (-1)^b c\mathbf{s}$$

instead of $\mathbf{z} = \mathbf{y} + c\mathbf{s}$:



$$y \leftarrow \mathcal{N}(0, \sigma^2)$$



$$y \leftarrow U[-a, a]$$

Note, no change for the uniform case.

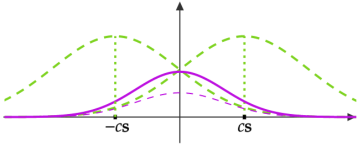
Bimodal rejection sampling

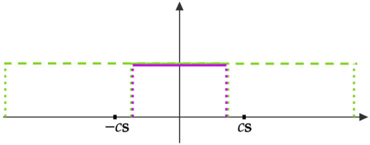
Run-time $\propto M$ ($\approx$ green area / purple area).

To decrease M, [DDLL13] uses

$$\mathbf{z} = \mathbf{y} + (-1)^b c\mathbf{s}$$

instead of $\mathbf{z} = \mathbf{y} + c\mathbf{s}$:





$y \leftarrow \mathcal{N}(0, \sigma^2)$ $y \leftarrow U[-a, a]$

Note, no change for the uniform case.

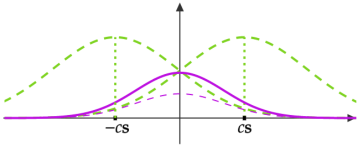
Bimodal rejection sampling

Run-time $\propto M$ ($\approx$ green area / purple area).

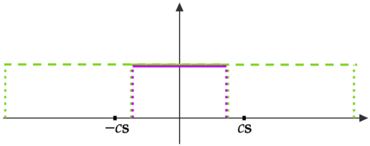
To decrease M, [DDLL13] uses

$$\mathbf{z} = \mathbf{y} + (-1)^b c\mathbf{s}$$

instead of $\mathbf{z} = \mathbf{y} + c\mathbf{s}$:



$y \leftarrow \mathcal{N}(0, \sigma^2)$



$y \leftarrow U[-a, a]$

Note, no change for the uniform case.

Bimodal rejection sampling

However, this makes "secure" implementation[3] much harder. It is basically due to "reject with probability a (transcendental) function of sk."

For e.g., for ≈120 bits security[4][5],

[3] an implementation secure against physical attacks (side-channel attacks)

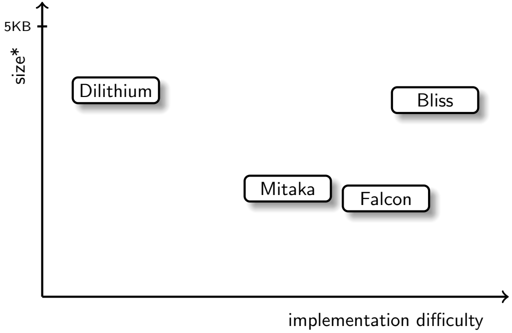
[4] core-SVP hardness

[5] size= |sig| + |vk|

Bimodal rejection sampling

However, this makes "secure" implementation[3] much harder. It is basically due to "reject with probability a (transcendental) function of sk."

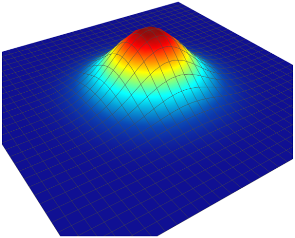
For e.g., for ≈120 bits security[4][5],

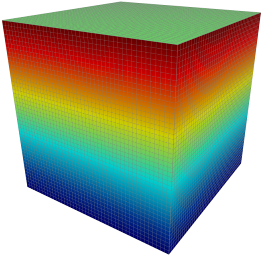


[3] an implementation secure against physical attacks (side-channel attacks)
[4] core-SVP hardness
[5] size= $|\text{sig}| + |\text{vk}|$

Hyperball bimodal rejection sampling

Previously, the randomness $\mathbf{y}$ was chosen from either discrete Gaussian or uniform hypercube[6].



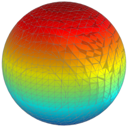


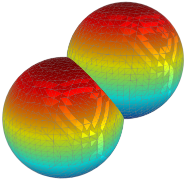
[6] The vectors $\mathbf{y}$ and $\mathbf{z}$ are high-dimensional vectors, so uniform in an interval is indeed a uniform hypercube.

Hyperball bimodal rejection sampling

We, instead, use **uniform hyperball** distribution for sampling $\mathbf{y}$ [DFPS22];

- to exploit optimal M,
- to reduce signature and verification key sizes,



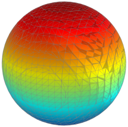


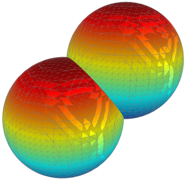
based on the **bimodal approach** [DDLL13].

Hyperball bimodal rejection sampling

We, instead, use **uniform hyperball** distribution for sampling $\mathbf{y}$ [DFPS22];

- to exploit optimal M,
- to reduce signature and verification key sizes,



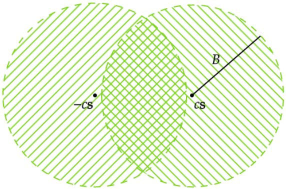


based on the **bimodal approach** [DDLL13].

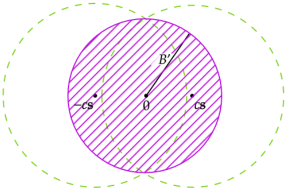
Hyperball bimodal rejection sampling

We reject $(c, \mathbf{z}) \sim D_\mathsf{s}$ (with p.d.f. p_s) to a target distribution D_t (with p.d.f. p_t), where

- p_s: uniform in hyperballs of radii B centered at $\pm c\mathbf{s}$
  - union of two large balls
- p_t: uniform in a smaller hyperball of radii B' centered at zero
  - a smaller ball in the middle



p_s

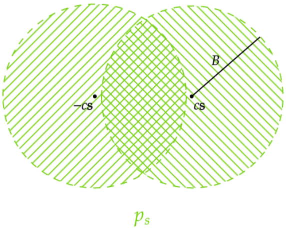


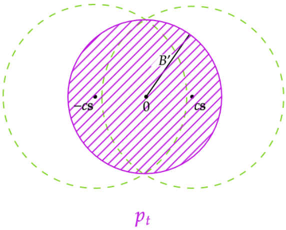
p_t

Hyperball bimodal rejection sampling

We reject $(c, \mathbf{z}) \sim D_{\mathsf{s}}$ (with p.d.f. p_{s}) to a target distribution D_{t} (with p.d.f. p_{t}), where

- p_{s}: uniform in hyperballs of radii B centered at $\pm c\mathbf{s}$
  - union of two large balls
- p_{t}: uniform in a smaller hyperball of radii B' centered at zero
  - a smaller ball in the middle





Hyperball bimodal rejection sampling

- $p_{\mathsf{s}}(\mathbf{x}) = \frac{1}{2\cdot\mathsf{vol}(\mathcal{B}(B))} \cdot \chi_{\|\mathbf{z}-c\mathbf{s}\|<B} + \frac{1}{2\cdot\mathsf{vol}(\mathcal{B}(B))} \cdot \chi_{\|\mathbf{z}+c\mathbf{s}\|<B}$,
- $p_{\mathsf{t}}(\mathbf{x}) = \frac{1}{\mathsf{vol}(\mathcal{B}(B))} \cdot \chi_{\|\mathbf{z}\|<B'}$.

$$\Rightarrow p(\mathbf{x}) = \frac{p_{\mathsf{t}}(\mathbf{x})}{M \cdot p_{\mathsf{s}}(\mathbf{x})} = \frac{\chi_{\|\mathbf{z}\|<B'}}{\chi_{\|\mathbf{z}-c\mathbf{s}\|<B} + \chi_{\|\mathbf{z}+c\mathbf{s}\|<B}}$$

$$= \begin{cases} 0 & \text{if } \mathbf{z} \notin \mathcal{B}(B'), \\ 1/2 & \text{if } \mathbf{z} \in \mathcal{B}(B') \cap \mathcal{B}(B, c\mathbf{s}) \cap \mathcal{B}(B, -c\mathbf{s}), \\ 1 & \text{if } \mathbf{z} \in \mathcal{B}(B') \setminus (\mathcal{B}(B, c\mathbf{s}) \cap \mathcal{B}(B, -c\mathbf{s})), \end{cases}$$

for some $M > 0$.

Hyperball bimodal rejection sampling

- $p_{\mathsf{s}}(\mathbf{x}) = \frac{1}{2\cdot\mathsf{vol}(\mathcal{B}(B))} \cdot \chi_{\|\mathbf{z}-c\mathbf{s}\|<B} + \frac{1}{2\cdot\mathsf{vol}(\mathcal{B}(B))} \cdot \chi_{\|\mathbf{z}+c\mathbf{s}\|<B}$,
- $p_{\mathsf{t}}(\mathbf{x}) = \frac{1}{\mathsf{vol}(\mathcal{B}(B))} \cdot \chi_{\|\mathbf{z}\|<B'}$.

$$\Rightarrow p(\mathbf{x}) = \frac{p_{\mathsf{t}}(\mathbf{x})}{M \cdot p_{\mathsf{s}}(\mathbf{x})} = \frac{\chi_{\|\mathbf{z}\|<B'}}{\chi_{\|\mathbf{z}-c\mathbf{s}\|<B} + \chi_{\|\mathbf{z}+c\mathbf{s}\|<B}}$$

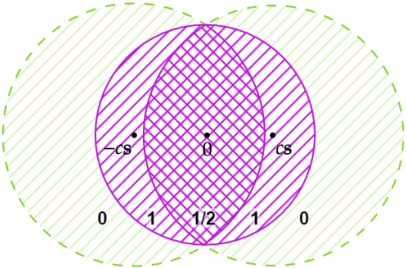
$$= \begin{array}{cl} 0 & \text{if } \mathbf{z} \notin \mathcal{B}(B'), \\ 1/2 & \text{if } \mathbf{z} \in \mathcal{B}(B') \cap \mathcal{B}(B, c\mathbf{s}) \cap \mathcal{B}(B, -c\mathbf{s}), \\ 1 & \text{if } \mathbf{z} \in \mathcal{B}(B') \setminus (\mathcal{B}(B, c\mathbf{s}) \cap \mathcal{B}(B, -c\mathbf{s})), \end{array}$$

for some $M > 0$.

Hyperball bimodal rejection sampling

That is, we return $\mathbf{x} = (c, \mathbf{z})$ with probability

- 0: if $\|\mathbf{z}\| \geq B'$,
- 1/2: else if $\|\mathbf{z} - c\mathbf{s}\| < B$ and $\|\mathbf{z} + c\mathbf{s}\| < B$,
- 1: otherwise.



1. Brief Introduction to HAETAE

2. Preliminaries:
   - Digital signatures
   - Lattice hard problems
   - Lattice-based signatures
     - Bimodal rejection sampling

3. HAETAE:
   - Hyperball bimodal rejection sampling
   - Comparison to SotA lattice signatures

4. Changes after Round 1

Comparison to SotA lattice signatures.

For 120-bit classical security. Sizes are in bytes.

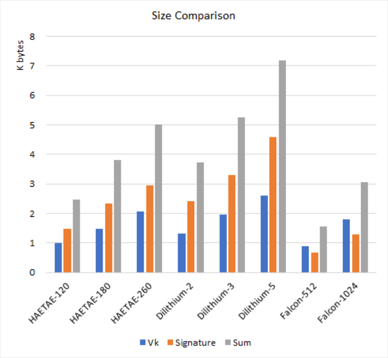
Scheme	sig	vk	KeyGen	Sign	
				sampling	rejection
Dilithium-2	2420	1312	fast	Hypercube	$\|\cdot\|_\infty < B$
Bliss-1024[7]	1700	1792	fast	dGaussian at 0	reject with prob. f(sk, Sig)
HAETAE120	1468	1056	fast	dHyperball at 0	$\|\cdot\|_2 < B$
Mitaka-512[8]	713	896	slow	dGaussian at 0 & intGaussian at $H(m)$	none
Falcon-512	666	897	slow	dGaussian at $H(m)$	none

Table: Comparison between different lattice-based signature schemes.

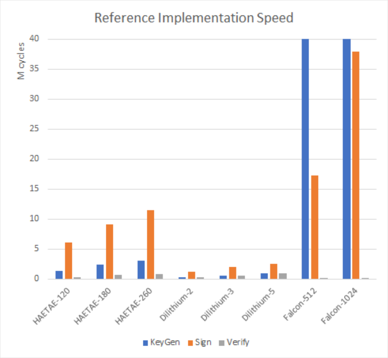
[7] modified Bliss (to ≥ 120 bit-security) in Dilithium paper.

[8] Mitaka-512 has 102 bits of security

Numbers - Updated Reference Implementation

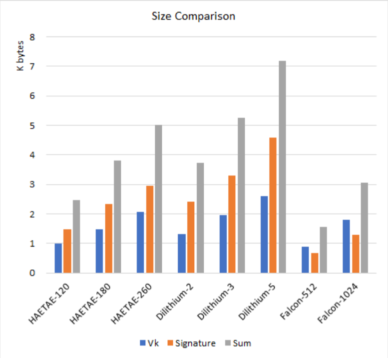


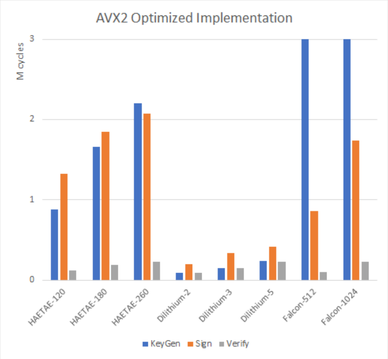
Size



Performance

Numbers - AVX2 optimized Implementation



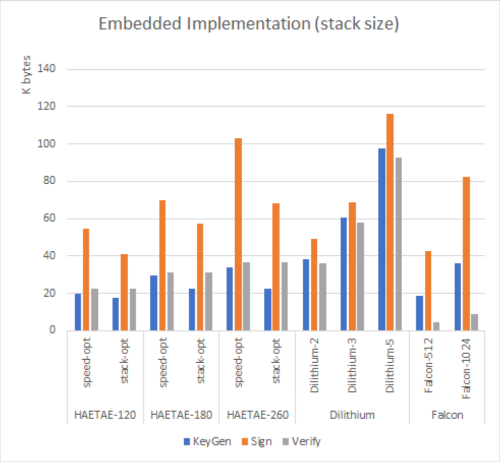


Size

Performance

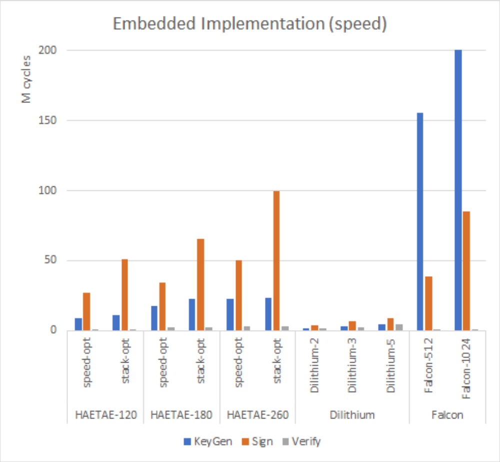
Numbers - Embedded Implementation on Cortex-M4

Stack-size of HAETAE and others on Cortex-M4.



Numbers - Embedded Implementation on Cortex-M4

Speed of HAETAE and others on Cortex-M4.



Update Logs after Round 1

Nov, 2022 (v0.9): KpqC round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: rANS, secret key rejection
- implementation: fixed-point, constant-time

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: Bug-fix, AVX2 optimized, embedded (Cortex-M4)

Feb, 2024 (v2.1): KpqC round 2

- spec: HVZK for compressed HAETAE, more precise security bound, "refined" security estimation

Update Logs after Round 1

Nov, 2022 (v0.9): KpqC round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: rANS, secret key rejection
- implementation: fixed-point, constant-time

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: Bug-fix, AVX2 optimized, embedded (Cortex-M4)

Feb, 2024 (v2.1): KpqC round 2

- spec: HVZK for compressed HAETAE, more precise security bound, "refined" security estimation

Update Logs after Round 1

Nov, 2022 (v0.9): KpqC round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: rANS, secret key rejection
- implementation: fixed-point, constant-time

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: Bug-fix, AVX2 optimized, embedded (Cortex-M4)

Feb, 2024 (v2.1): KpqC round 2

- spec: HVZK for compressed HAETAE, more precise security bound, "refined" security estimation

Update Logs after Round 1

Nov, 2022 (v0.9): KpqC round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: rANS, secret key rejection
- implementation: fixed-point, constant-time

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: Bug-fix, AVX2 optimized, embedded (Cortex-M4)

Feb, 2024 (v2.1): KpqC round 2

- spec: HVZK for compressed HAETAE, more precise security bound, "refined" security estimation

Update Logs after Round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: hint compression, secret key rejection
- implementation: fixed-point, constant-time

- **Hint vector compression:** The hint vector h, a part of the signature, is compressed via $\mathrm{LowBits}^h$, $\mathrm{HighBits}^h$, and rANS encoding.

- **Secret key rejection:** Bounding $\|c\mathbf{s}\|_2 \leq \gamma\sqrt{\tau}$ via bounding $\mathcal{N}(\mathbf{s}) \leq \gamma^2 n$:

$$\mathcal{N}(\mathbf{s}) := \tau \cdot \sum_{i=1}^{m} \max_{0 \leq j < 2n}^{i\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2 + r \cdot \max_{0 \leq j < 2n}^{(m+1)\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2,$$

which can be efficiently checked.

- **Fixed-Point everywhere!**

Update Logs after Round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: hint compression, secret key rejection
- implementation: fixed-point, constant-time

- **Hint vector compression:** The hint vector h, a part of the signature, is compressed via LowBits^h, HighBits^h, and rANS encoding.

- **Secret key rejection:** Bounding $\|c\mathbf{s}\|_2 \leq \gamma\sqrt{\tau}$ via bounding $\mathcal{N}(\mathbf{s}) \leq \gamma^2 n$:

$$\mathcal{N}(\mathbf{s}) := \tau \cdot \sum_{i=1}^{m} \max_{0 \leq j < 2n}^{i\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2 + r \cdot \max_{0 \leq j < 2n}^{(m+1)\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2,$$

which can be efficiently checked.

- **Fixed-Point everywhere!**

Update Logs after Round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: hint compression, secret key rejection
- implementation: fixed-point, constant-time

- **Hint vector compression:** The hint vector h, a part of the signature, is compressed via LowBits^h, HighBits^h, and rANS encoding.

- **Secret key rejection:** Bounding $\|c\mathbf{s}\|_2 \leq \gamma\sqrt{\tau}$ via bounding $\mathcal{N}(\mathbf{s}) \leq \gamma^2 n$:

$$\mathcal{N}(\mathbf{s}) := \tau \cdot \sum_{i=1}^{m} \max_{0 \leq j < 2n}^{i\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2 + r \cdot \max_{0 \leq j < 2n}^{(m+1)\text{-th}} \|\mathbf{s}(\omega_j)\|_2^2,$$

which can be efficiently checked.

- **Fixed-Point everywhere!**

Update Logs after Round 1

May, 2023 (v1.0)

- spec: missing parts inclusion, min-entropy analysis
- improved: hint compression, secret key rejection
- implementation: fixed-point, constant-time

- **Hint vector compression:** The hint vector h, a part of the signature, is compressed via LowBits^h, HighBits^h, and rANS encoding.

- **Secret key rejection:** Bounding $\|c\mathbf{s}\|_2 \leq \gamma\sqrt{\tau}$ via bounding $\mathcal{N}(\mathbf{s}) \leq \gamma^2 n$:

$$\mathcal{N}(\mathbf{s}) := \tau \cdot \sum_{i=1}^{m} \overset{i\text{-th}}{\max_{0 \leq j < 2n}} \|\mathbf{s}(\omega_j)\|_2^2 + r \cdot \overset{(m+1)\text{-th}}{\max_{0 \leq j < 2n}} \|\mathbf{s}(\omega_j)\|_2^2,$$

which can be efficiently checked.

- **Fixed-Point everywhere!**

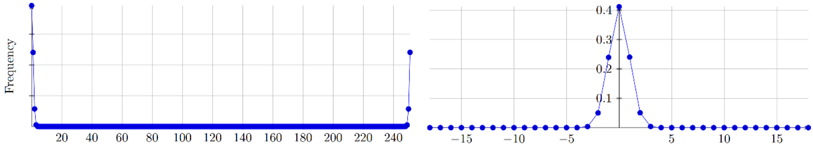
Update Logs after Round 1

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: Bug-fix, AVX2 optimized, embedded (Cortex-M4)

- **Implementation security:** Mainly on *Fix-Point Arithmetic* with no fix-point multiplication and *Protecting the Hyperball Sampler*.

- **Reduced precomputation table:** Cut off the extremely low-frequency symbols ($< 0.1\%$ in total):



(a) h Coefficient values **(b)** HB($\mathbf{z_1}$) Coefficient values

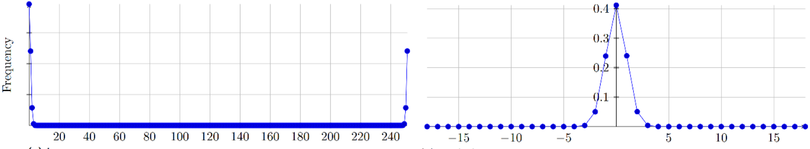
Update Logs after Round 1

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: Bug-fix, AVX2 optimized, embedded (Cortex-M4)

- **Implementation security:** Mainly on *Fix-Point Arithmetic* with no fix-point multiplication and *Protecting the Hyperball Sampler*.

- **Reduced precomputation table:** Cut off the extremely low-frequency symbols ($< 0.1\%$ in total):



(a) h **(b)** HB($\mathbf{z_1}$)

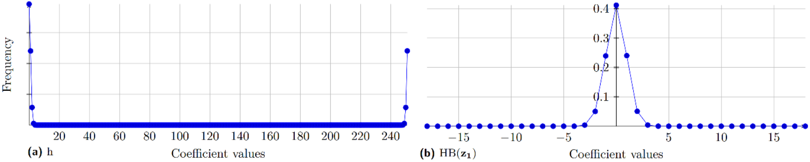
Update Logs after Round 1

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: Bug-fix, AVX2 optimized, embedded (Cortex-M4)

- **Implementation security:** Mainly on *Fix-Point Arithmetic* with no fix-point multiplication and *Protecting the Hyperball Sampler*.

- **Reduced precomputation table:** Cut off the extremely low-frequency symbols ($< 0.1\%$ in total):



(a) h **(b)** HB($\mathbf{z_1}$)

Update Logs after Round 1

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: AVX2 optimized, embedded (Cortex-M4)

- **Bug-fix:** Implementation-specific bugs (reported via KpqC workshops/ bulletin/PQC forum/ourselves) are fixed.

- **AVX2 optimization:** Mainly on *Vectorized Hyperball Sampling* (as Keccak and NTT use existing optimized code) via parallel polynomial samplings. For HAETAE-120, 4.6x speed-up.

- **Embedded Cortex-M4 implementation:** Stack/speed optimizations, resulting in 40 to 54 KiB maximum stack size for HAETAE-120.

Update Logs after Round 1

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: AVX2 optimized, embedded (Cortex-M4)

- **Bug-fix:** Implementation-specific bugs (reported via KpqC workshops/ bulletin/PQC forum/ourselves) are fixed.

- **AVX2 optimization:** Mainly on *Vectorized Hyperball Sampling* (as Keccak and NTT use existing optimized code) via parallel polynomial samplings. For HAETAE-120, 4.6x speed-up.

- **Embedded Cortex-M4 implementation:** Stack/speed optimizations, resulting in 40 to 54 KiB maximum stack size for HAETAE-120.

Update Logs after Round 1

Nov, 2023 (v2.0)

- spec: implementation security
- improved: reduced precomputation table for rANS
- implementation: AVX2 optimized, embedded (Cortex-M4)

- **Bug-fix:** Implementation-specific bugs (reported via KpqC workshops/ bulletin/PQC forum/ourselves) are fixed.

- **AVX2 optimization:** Mainly on *Vectorized Hyperball Sampling* (as Keccak and NTT use existing optimized code) via parallel polynomial samplings. For HAETAE-120, 4.6x speed-up.

- **Embedded Cortex-M4 implementation:** Stack/speed optimizations, resulting in 40 to 54 KiB maximum stack size for HAETAE-120.

Update Logs after Round 1

Feb, 2024 (v2.1)

- spec: HVZK for compressed HAETAE, more precise security bound, "refined" security estimation

- **HVZK for compressed HAETAE:** Proof for HVZK is extended to cover the compressed HAETAE.

- **Precise security bound and "refined" security estimation:** Along with the original security bounds with BKZ block size based on GSA and Core-SVP analysis, we also give a security estimation for MLWE based on the leaky LWE estimator [DDGR20].

Update Logs after Round 1

Feb, 2024 (v2.1)

- spec: HVZK for compressed HAETAE, more precise security bound, "refined" security estimation

- **HVZK for compressed HAETAE:** Proof for HVZK is extended to cover the compressed HAETAE.

- **Precise security bound and "refined" security estimation:** Along with the original security bounds with BKZ block size based on GSA and Core-SVP analysis, we also give a security estimation for MLWE based on the leaky LWE estimator [DDGR20].

Update Logs after Round 1

Feb, 2024 (v2.1)

- spec: HVZK for compressed HAETAE, more precise security bound, "refined" security estimation

- **HVZK for compressed HAETAE:** Proof for HVZK is extended to cover the compressed HAETAE.

- **Precise security bound and "refined" security estimation:** Along with the original security bounds with BKZ block size based on GSA and Core-SVP analysis, we also give a security estimation for MLWE based on the leaky LWE estimator [DDGR20].

Thanks!

Check http://kpqc.cryptolab.co.kr/haetae!

Check https://github.com/mupq/pqm4 for the *embedded code*!

Any question?

References I

[Ajt96] M. Ajtai.
Generating hard instances of lattice problems (extended abstract).
In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.

[BG14] Shi Bai and Steven D Galbraith.
An improved compression technique for signatures based on learning with errors.
In Cryptographers' Track at the RSA Conference, pages 28–47. Springer, 2014.

[DDGR20] Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi.
Lwe with side information: Attacks and concrete security estimation, 2020.

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky.
Lattice signatures and bimodal gaussians.
In Annual Cryptology Conference, pages 40–56. Springer, 2013.

[DFPS22] Julien Devevey, Omar Fawzi, Alain Passelègue, and Damien Stehlé.
On rejection sampling in lyubashevsky's signature scheme.
Cryptology ePrint Archive, Number 2022/1249, 2022.
To be appeared in Asiacrypt, 2022. https://eprint.iacr.org/2022/1249.

References II

[DKL$^+$18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé.
Crystals-dilithium: A lattice-based digital signature scheme.
IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 238–268, 2018.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest.
Efficient identity-based encryption over ntru lattices.
In International Conference on the Theory and Application of Cryptology and Information Security, pages 22–41. Springer, 2014.

[DP16] Léo Ducas and Thomas Prest.
Fast fourier orthogonalization.
In Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, pages 191–198, 2016.

[Duc14] Léo Ducas.
Accelerating bliss: the geometry of ternary polynomials.
Cryptology ePrint Archive, Paper 2014/874, 2014.
https://eprint.iacr.org/2014/874.

References III

[EFG+22] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu.
Mitaka: A simpler, parallelizable, maskable variant of.
In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 222–253. Springer, 2022.

[ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu.
Shorter hash-and-sign lattice-based signatures.
In Yevgeniy Dodis and Thomas Shrimpton, editors, Advances in Cryptology – CRYPTO, 2022.

[FHK+18] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang.
Falcon: Fast-fourier lattice-based compact signatures over ntru.
Submission to the NIST's post-quantum cryptography standardization process, 36(5), 2018.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann.
Practical lattice-based cryptography: A signature scheme for embedded systems.
In International Workshop on Cryptographic Hardware and Embedded Systems, pages 530–547. Springer, 2012.

References IV

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan.
Trapdoors for hard lattices and new cryptographic constructions.
In Proceedings of the fortieth annual ACM symposium on Theory of computing, pages 197–206, 2008.

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H Silverman, and William Whyte.
Ntrusign: Digital signatures using the ntru lattice.
In Cryptographers' track at the RSA conference, pages 122–140. Springer, 2003.

[HR07] Ishay Haviv and Oded Regev.
Tensor-based hardness of the shortest vector problem to within almost polynomial factors.
In Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, STOC '07, page 469–477, New York, NY, USA, 2007. Association for Computing Machinery.

[Lyu09] Vadim Lyubashevsky.
Fiat-shamir with aborts: Applications to lattice and factoring-based signatures.
In International Conference on the Theory and Application of Cryptology and Information Security, pages 598–616. Springer, 2009.

References V

[Lyu12] Vadim Lyubashevsky.
Lattice signatures without trapdoors.
In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 738–755. Springer, 2012.

HAETAE description (high-level)

KeyGen(1^λ)

1: $\mathbf{A}_{\text{gen}} \leftarrow \mathcal{R}_q^{k\times(\ell-1)}$ and $(\mathbf{s}_{\text{gen}}, \mathbf{e}_{\text{gen}}) \leftarrow S_\eta^{\ell-1} \times S_\eta^k$
2: $\mathbf{b} = \mathbf{A}_{\text{gen}} \cdot \mathbf{s}_{\text{gen}} + \mathbf{e}_{\text{gen}} \in \mathcal{R}_q^k$
3: $\mathbf{A} = (-2\mathbf{b} + q\mathbf{j} \,|\, 2\mathbf{A}_{\text{gen}} \,|\, 2\mathbf{Id}_k) \bmod 2q$ and write $\mathbf{A} = (\mathbf{A}_1 \,|\, 2\mathbf{Id}_k)$
4: $\mathbf{s} = (1, \mathbf{s}_{\text{gen}}, \mathbf{e}_{\text{gen}})$
5: if $\sigma_{\max}(\mathrm{rot}(\mathbf{s}_{\text{gen}})) > \gamma$, then restart
6: Return $\mathsf{sk} = \mathbf{s}$, $\mathsf{vk} = \mathbf{A}$

Sign(sk, M)

1: $\mathbf{y} \leftarrow U(\mathcal{B}_{(1/N)\mathcal{R},(k+\ell)}(B))$
2: $c = H(\mathrm{HighBits}_{2q}^{\mathrm{hint}}(\mathbf{A}\lfloor\mathbf{y}\rceil, \alpha), \mathrm{LSB}(\lfloor y_0 \rceil), M) \in \mathcal{R}_2$
3: $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) = \mathbf{y} + (-1)^b c \cdot \mathbf{s}$ for $b \leftarrow U(\{0,1\})$
4: $\mathbf{h} = \mathrm{HighBits}_{2q}^{\mathrm{hint}}(\mathbf{A}\lfloor\mathbf{z}\rceil - qc\mathbf{j}, \alpha) - \mathrm{HighBits}_{2q}^{\mathrm{hint}}(\mathbf{A}_1\lfloor\mathbf{z}_1\rceil - qc\mathbf{j}, \alpha) \bmod^{+} \frac{2(q-1)}{\alpha}$
5: if $\|\mathbf{z}\|_2 \geq B'$, then restart
6: if $\|2\mathbf{z} - \mathbf{y}\|_2 < B$, then restart with probability $1/2$
7: Return $\sigma = (\mathrm{Encode}(\mathrm{HighBits}(\lfloor\mathbf{z}_1\rceil, a)), \mathrm{LowBits}(\lfloor\mathbf{z}_1\rceil, a), \mathrm{Encode}(\mathbf{h}), c)$

Verify$(\mathsf{vk}, M, \sigma = (x, \mathbf{v}, h, c))$

1: $\tilde{\mathbf{z}}_1 = \mathrm{Decode}(x) \cdot a + \mathbf{v}$ and $\tilde{\mathbf{h}} = \mathrm{Decode}(h)$
2: $\mathbf{w} = \tilde{\mathbf{h}} + \mathrm{HighBits}_{2q}^{\mathrm{hint}}(\mathbf{A}_1\tilde{\mathbf{z}}_1 - qc\mathbf{j}, \alpha) \bmod^{+} \frac{2(q-1)}{\alpha}$
3: $w' = \mathrm{LSB}(\tilde{z}_0 - c)$
4: $\tilde{\mathbf{z}}_2 = \left[\mathbf{w} \cdot \alpha + w'\mathbf{j} - (\mathbf{A}_1\tilde{z}_1 - qc\mathbf{j})\right]/2 \bmod^{\pm} q$
5: $\tilde{\mathbf{z}} = (\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2)$
6: Return $(\, c = H(\mathbf{w}, w', M) \,) \wedge (\, \|\tilde{\mathbf{z}}\| < B'' \,)$