HAETAE: Shorter Fiat-Shamir with Aborts Signature

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

1. Seoul National University
2. CryptoLab Inc.
3. ANSSI
4. Ruhr University Bochum
5. German Research Centre for Artifical Intelligence

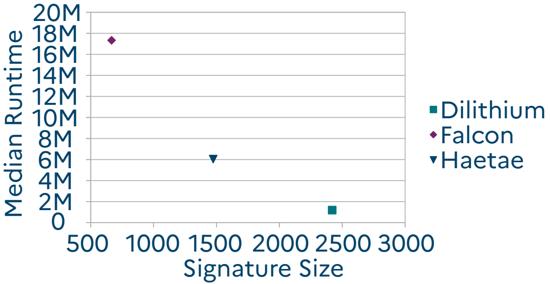
What's Haetae?

- Website: `https://kpqc.cryptolab.co.kr/haetae`
- Submission to NIST's additional PQC round
- Submission to South Korea PQC
- Same framework as DILITHIUM : Fiat-Shamir with Aborts over lattices

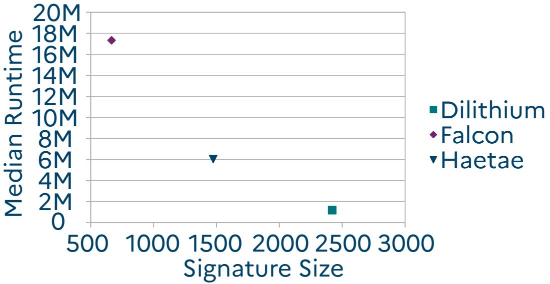
High-level comparison with Dilithium

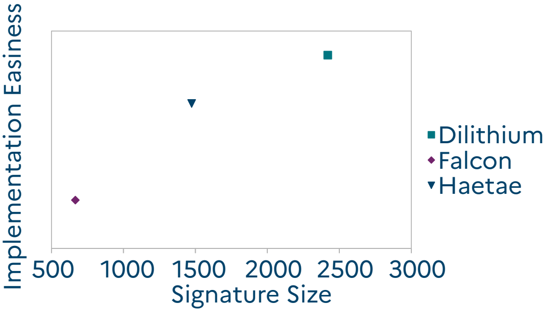
	Dilithium	Haetae
Goal	Implementer-friendly	Small signature size
Distribution		
Mode	Unimodal	Bimodal
Arithmetic operations	1.5	1
Bit-size of the modulus	23	16
Number of repetitions	4	6

Performances (Level II)



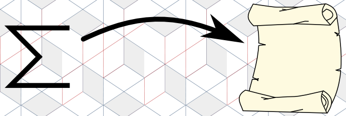
Performances (Level II)



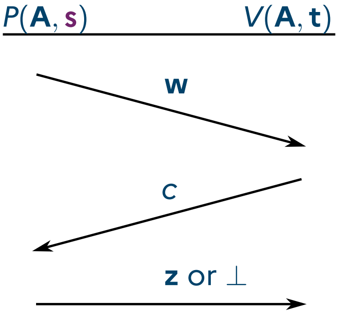


Outline

1. Fiat-Shamir with Aborts for Lattices

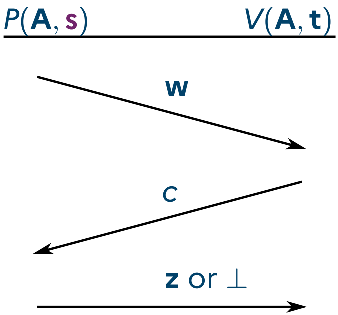


Σ-Protocols



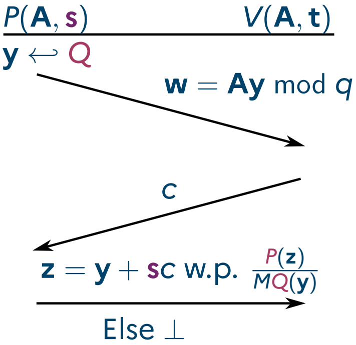
- $\mathbf{A}\mathbf{s} = (\mathbf{A}_0 | \mathbf{Id})\mathbf{s} = \mathbf{t} \bmod q$

Σ-Protocols



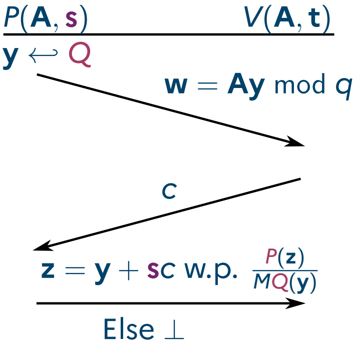
- $\mathbf{A}\mathbf{s} = (\mathbf{A}_0 | \mathbf{Id})\mathbf{s} = \mathbf{t} \bmod q$
- V accepts under some condition
- Nothing is revealed on $\mathbf{s}$
- Convincing V without $\mathbf{s}$ is hard

Lyubashevsky's Protocol [Lyu09, Lyu12]



- $\mathbf{As} = (\mathbf{A}_0|\mathbf{Id})\mathbf{s} = \mathbf{t} \bmod q$
- $\mathbf{y}, \mathbf{s}$ and c are small

Lyubashevsky's Protocol [Lyu09, Lyu12]

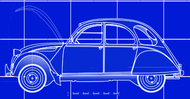


- $\mathbf{As} = (\mathbf{A}_0 | \mathbf{Id})\mathbf{s} = \mathbf{t} \bmod q$
- $\mathbf{y}, \mathbf{s}$ and c are small
- V accepts if $\mathbf{Az} - \mathbf{t}c = \mathbf{w} \bmod q$ and $\|\mathbf{z}\| \leq \gamma$
- $\mathbf{z} \hookleftarrow P$ independent of $\mathbf{s}$
- Convincing V without $\mathbf{s}$ is hard

Fiat-Shamir with Aborts

Sign($\mathbf{A}, \mathbf{s}, \mu$):
do
- $\mathbf{y} \hookleftarrow Q$
- $c = H(\mathbf{Ay} \bmod q, \mu)$
- $\mathbf{z} = \mathbf{y} + \mathbf{s}c$
- w.p. $\frac{P(\mathbf{z})}{M \cdot Q(\mathbf{y})}$
  - $\mathbf{z} = \perp$

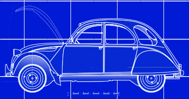
while $\mathbf{z} = \perp$
return $(\mathbf{z}, c)$



- Verification: recover $\mathbf{w}$, check if $c = H(\mathbf{w}, \mu)$ and $\|\mathbf{z}\| \leq \gamma$

Fiat-Shamir with Aborts

```
Sign(A, s, μ):
do
  | y ← Q
  | c = H(Ay mod q, μ)
  | z = y + sc
  | w.p. P(z) / (M·Q(y))
  | | z = ⊥
while z = ⊥
return (z, c)
```



- Verification: recover $\mathbf{w}$, check if $c = H(\mathbf{w}, \mu)$ and $\|\mathbf{z}\| \leq \gamma$
- Unforgeability if [BBD+23]:
  - Large min-entropy for $\mathbf{w}$
  - aHVZK: simulate accepting transcripts without $\mathbf{s}$
  - Soundness: $\mathcal{A}(\mathbf{A}, \mathbf{t})$ cannot convince $V(\mathbf{A}, \mathbf{t})$

Security Reduction

Soundness

$\Downarrow$ *(Only in the ROM)*

UF-NMA (Find a forgery given the verification key)

$\Downarrow$ *(Use the HVZK simulator)*

UF-CMA (Find a forgery given *vk* and access to a signing oracle)

Optimal Choice of Distribution

Haetae instantiates $Q \propto P = U(\bullet)$

Optimal Choice of Distribution

Haetae instantiates $Q \propto P = U(\bullet)$

- Smallest γ as with Gaussians [**D**FPS22]
- Easier rejection step than Gaussians

Optimal Choice of Distribution

Haetae instantiates $Q \propto P = U(\bullet)$

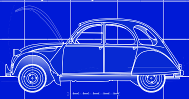
- Smallest γ as with Gaussians [DFPS22]
- Easier rejection step than Gaussians

But we can do better!

Bimodal Lattice-based Fiat-Shamir with Aborts

Sign($\mathbf{A}, \mathbf{s}, \mu$):
do
- $\mathbf{y} \hookleftarrow Q$
- $c = H(\mathbf{Ay} \bmod 2q, \mu)$
- $\mathbf{z} = \mathbf{y} + (-1)^{U(\{0,1\})}\mathbf{s}c$
- w.p. $\frac{2P(\mathbf{z})}{M(Q(\mathbf{z}-\mathbf{s}c)+Q(\mathbf{z}+\mathbf{s}c))}$
  - $\mathbf{z} = \perp$

while $\mathbf{z} = \perp$
return $(\mathbf{z}, c)$

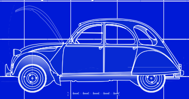


- New key equation: $\mathbf{As} = -\mathbf{As} = qc\mathbf{j} \bmod 2q$

Bimodal Lattice-based Fiat-Shamir with Aborts

Sign($\mathbf{A}, \mathbf{s}, \mu$):
do
- $\mathbf{y} \hookleftarrow Q$
- $c = H(\mathbf{Ay} \bmod 2q, \mu)$
- $\mathbf{z} = \mathbf{y} + (-1)^{U(\{0,1\})}\mathbf{s}c$
- w.p. $\frac{2P(\mathbf{z})}{M(Q(\mathbf{z}-\mathbf{s}c)+Q(\mathbf{z}+\mathbf{s}c))}$
  - $\mathbf{z} = \perp$

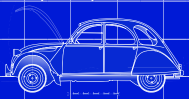
while $\mathbf{z} = \perp$
return $(\mathbf{z}, c)$



- New key equation: $\mathbf{As} = -\mathbf{As} = qc\mathbf{j} \bmod 2q$
- Verification:
  - Compute $\mathbf{w} = \mathbf{Az} - qc\mathbf{j} \bmod 2q$
  - Accept if $\|\mathbf{z}\| \leq \gamma$ and $c = H(\mathbf{w}, \mu)$

Bimodal Lattice-based Fiat-Shamir with Aborts

Sign($\mathbf{A}, \mathbf{s}, \mu$):
do
  $\mathbf{y} \hookleftarrow Q$
  $c = H(\mathbf{Ay} \bmod 2q, \mu)$
  $\mathbf{z} = \mathbf{y} + (-1)^{U(\{0,1\})}\mathbf{s}c$
  w.p. $\frac{2P(\mathbf{z})}{M(Q(\mathbf{z}-\mathbf{s}c)+Q(\mathbf{z}+\mathbf{s}c))}$
  | $\mathbf{z} = \perp$
while $\mathbf{z} = \perp$
return $(\mathbf{z}, c)$



- New key equation: $\mathbf{As} = -\mathbf{As} = qc\mathbf{j} \bmod 2q$
- Verification:
  - Compute $\mathbf{w} = \mathbf{Az} - qc\mathbf{j} \bmod 2q$
  - Accept if $\|\mathbf{z}\| \leq \gamma$ and $c = H(\mathbf{w}, \mu)$
- Allows for smaller γ at constant M [DFPS22]
- $\gamma \approx \frac{\sqrt{\dim(\mathbf{y})}\|\mathbf{s}c\|}{\sqrt{\log M}}$

Design Rationale

Smaller γ (i.e. smaller size)

$\Downarrow$

Forging becomes harder

$\Downarrow$

More security overall

$\Downarrow$

Smaller parameters

$\Downarrow$

Smaller size (i.e. smaller γ)

$\Downarrow$

...

2. Hyperballs

2.1. Rejection Step

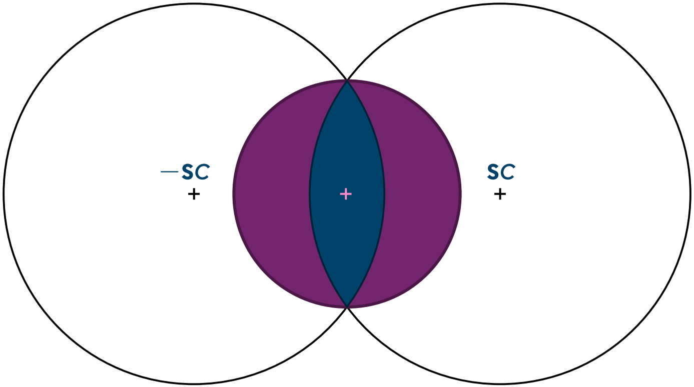
$\mathsf{KeyGen}(1^\lambda)$:

return $\mathbf{A}, \mathbf{s}$
with $\mathbf{As} = q\mathbf{j} \bmod 2q$

$\mathsf{Sign}(\mathbf{A}, \mathbf{s}, \mu)$:

do
$\mathbf{y} \hookleftarrow U($  $)$
$\mathbf{w} = \mathbf{Ay} \bmod 2q$
$c = H(\mathrm{HB}(\mathbf{w}), \mathrm{LSB}(\mathbf{w}), \mu)$
$\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}c$
w.p. $p(\mathbf{z})$, set $\mathbf{z} = \perp$
while $\mathbf{z} = \perp$
$x = \mathrm{compress}(\mathbf{z})$
return (x, c)

Rejection Probability

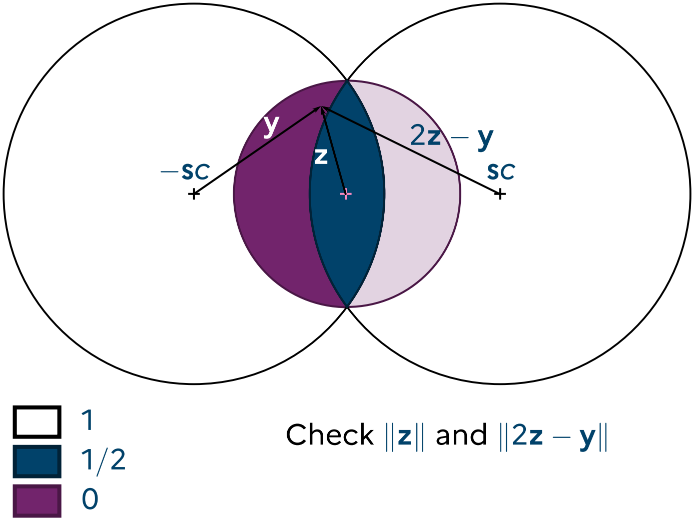


1

1/2

0

Rejection Probability



Check $\|\mathbf{z}\|$ and $\|2\mathbf{z} - \mathbf{y}\|$

2.2. Hyperball Sampler

KeyGen(1^λ):

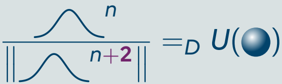
return $\mathbf{A}, \mathbf{s}$
with $\mathbf{As} = q\mathbf{j} \bmod 2q$

Sign$(\mathbf{A}, \mathbf{s}, \mu)$:

do
$\mathbf{y} \hookleftarrow U($$)$
$\mathbf{w} = \mathbf{Ay} \bmod 2q$
$c = H(\mathrm{HB}(\mathbf{w}), \mathrm{LSB}(\mathbf{w}), \mu)$
$\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}c$
w.p. $p(\mathbf{z})$, set $\mathbf{z} = \perp$
while $\mathbf{z} = \perp$
$x = \mathrm{compress}(\mathbf{z})$
return (x, c)

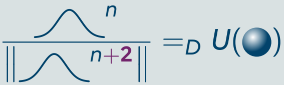
Main Theorem

Back to normal distributions [VG17]



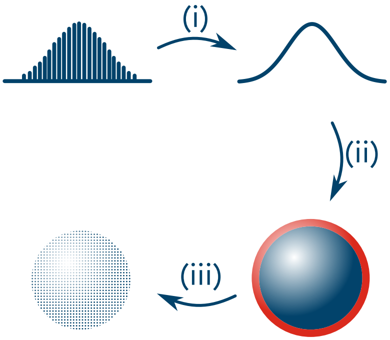
- Works for continuous distributions

Back to normal distributions [VG17]



- Works for continuous distributions
- Implemented using fixed-point arithmetic
- Requires $\approx$ 90 bits of precision

Implementation with Fixed-point Arithmetic



- (i) Discrete Gaussian to normal distribution "for free"
- (iii) Discretization step to balance rejection probability and efficiency

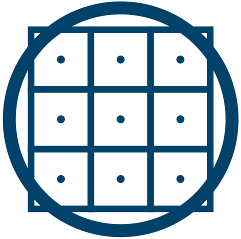
Setting the Discretization Step

Card()?

Setting the Discretization Step

Card()?

- Counting the number of points would help setting parameters
- Well-known for continuous hyperballs
- Choose a step making the comparison meaningful
- Other solution: empirical approach



Performances

Sign

Up to 80% of signing runtime!

3. Minimizing ||sc||

3.1. Key Generation

$\mathsf{KeyGen}(1^\lambda)$:

```
return A, s
    with As = qj mod 2q
```

$\mathsf{Sign}(\mathbf{A}, \mathbf{s}, \mu)$:

```
do
    y ↩ U(●)
    w = Ay mod 2q
    c = H(HB(w), LSB(w), μ)
    z = y + (−1)^b sc
    w.p. p(z), set z = ⊥
while z = ⊥
    x = compress(z)
    return (x, c)
```

Key Generation

1: $\mathbf{A}_0 \hookleftarrow U(\mathcal{R}_q^{k\times\ell-1})$
2: $\mathbf{s}_0, \mathbf{e}_0 \hookleftarrow U([-\eta \ldots \eta])^{\ell-1+k}$
3: $\mathbf{b} \leftarrow \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}_0 \bmod q$

Key Generation

1: $\mathbf{A}_0 \hookleftarrow U(\mathcal{R}_q^{k\times\ell-1})$
2: $\mathbf{s}_0, \mathbf{e}_0 \hookleftarrow U([-\eta\ldots\eta])^{\ell-1+k}$
3: $\mathbf{b} \leftarrow \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}_0 \bmod q$
4: $\mathbf{A} \leftarrow (-2\mathbf{b} + q\mathbf{j}|2\mathbf{A}_0|2\mathbf{I}_k) \bmod 2q$
5: $\mathbf{s} \leftarrow (1|{\mathbf{s}_0}^\top|{\mathbf{e}_0}^\top)^\top$

- $\mathbf{j} = (1, 0\ldots0)^\top$
- Add a trapdoor in the public matrix

Key Generation

1: $\mathbf{A}_0 \hookleftarrow U(\mathcal{R}_q^{k\times\ell-1})$
2: $\mathbf{s}_0, \mathbf{e}_0 \hookleftarrow U([-\eta\ldots\eta])^{\ell-1+k}$
3: $\mathbf{b} \leftarrow \mathbf{A}_0\mathbf{s}_0 + \mathbf{e}_0 \bmod q$
4: $\mathbf{A} \leftarrow (-2\mathbf{b} + q\mathbf{j}|2\mathbf{A}_0|2\mathbf{I}_k) \bmod 2q$
5: $\mathbf{s} \leftarrow (1|\mathbf{s}_0{}^\top|\mathbf{e}_0{}^\top)^\top$
6: restart if $f_\tau(\mathbf{s}) > n\beta^2/\tau$
7: return $\mathsf{vk} = \mathbf{A}, \mathsf{sk} = \mathbf{s}$

- $\mathbf{j} = (1, 0 \ldots 0)^\top$
- Add a trapdoor in the public matrix
- f_τ ensures that $\|\mathbf{s}c\| \leq \beta$ for any c with Hamming weight $\leq \tau$
- Acceptance rate from 10 to 25%

The f_τ Function

- Challenge c: binary polynomial with τ 1s
- f_τ uses canonical embedding to bound $\max_c \|\mathbf{s}c\|$
- Finer-grained than other upper bounds

	DILITHIUM	HAETAE
Challenge	Ternary	Binary
Entropy	$\binom{256}{\tau}+\tau$	$\binom{256}{\tau}$
Level II τ	39	58

Key Compression

Idea: transmit only $\mathbf{b} - \mathit{LeastSignificantBit}(\mathbf{b}) \Rightarrow$ saves 1 bit per coordinate

Key Compression

Idea: transmit only $\mathbf{b} - \mathit{LeastSignificantBit}(\mathbf{b}) \Rightarrow$ saves 1 bit per coordinate
Downside:

- Set $\mathbf{s}^\top = (1|\mathbf{s_0}^\top|\mathbf{e_0}^\top + \mathit{LSB}(\mathbf{b}))$
- Adapt KeyGen to keep $\mathbf{A}$ pseudo-uniform
- LSB is modified to keep $\mathbf{s}$ balanced

4. Signature Compression

KeyGen(1^λ):

return $\mathbf{A}, \mathbf{s}$
with $\mathbf{As} = q\mathbf{j} \bmod 2q$

Sign($\mathbf{A}, \mathbf{s}, \mu$):

do
$\mathbf{y} \hookleftarrow U(\bullet)$
$\mathbf{w} = \mathbf{Ay} \bmod 2q$
$c = H(\mathrm{HB}(\mathbf{w}), \mathrm{LSB}(\mathbf{w}), \mu)$
$\mathbf{z} = \mathbf{y} + (-1)^b \mathbf{s}c$
w.p. $p(\mathbf{z})$, set $\mathbf{z} = \perp$
while $\mathbf{z} = \perp$
$x = \mathrm{compress}(\mathbf{z})$
return (x, c)

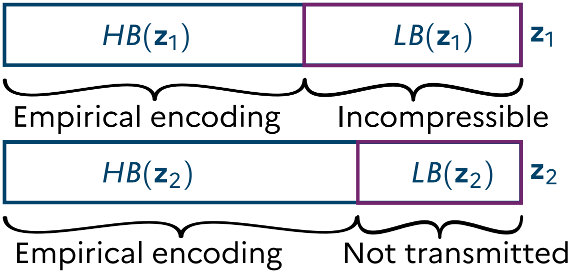
Low Bits Truncation

- Truncation technique from Bai and Galbraith
- $\mathbf{Ay} = \mathbf{A}_1\mathbf{z}_1 + 2\mathbf{z}_2 - q c \mathbf{j} \bmod 2q$ for some $\mathbf{A}_1$

- Exclude $LB(\mathbf{z}_2)$ from the signature
- Hash $HB(\mathbf{w})$ and $LSB(\mathbf{w})$

Transmitting the Signature

- Signature encoded using tANS (similar to [ETWY22])
- Low bits are sent as they are for reduced memory usage
- Allows for a signature size close to its entropy



5. Security Estimation

Security Assumptions

MSIS *(find a kernel element of* $\mathbf{A}$ *with norm* $\leq \gamma$*)*

$\Downarrow$ *(in the ROM)*

"BimodalSelfTargetMSIS" *(MSIS with hash collision)*

$\Downarrow$ *(with MLWE)*

Unforgeability of Haetae

Security Assumptions

MSIS *(find a kernel element of* $\mathbf{A}$ *with norm* $\leq \gamma$*)*

$\Downarrow$ *(in the ROM)*

"BimodalSelfTargetMSIS" *(MSIS with hash collision)*

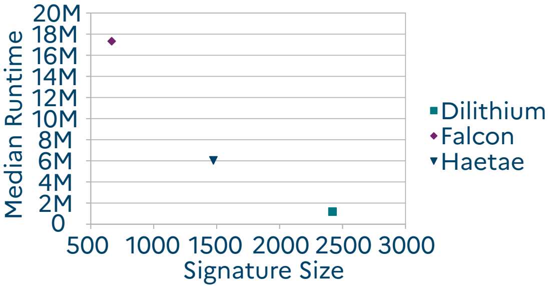
$\Downarrow$ *(with MLWE)*

Unforgeability of Haetae

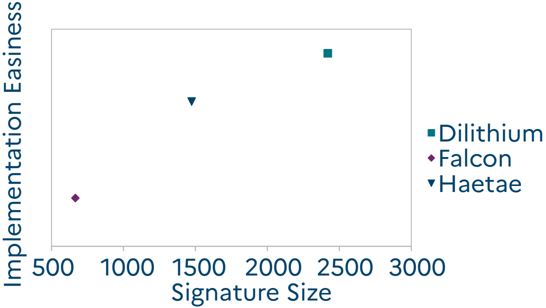
Estimating the Security

- Best approach for BimodalSelfTargetMSIS: solve MSIS
- MSIS and MLWE security estimated via the CoreSVP approach
- MLWE has refined estimates using [DSDGR20]
- This follows Dilithium's approach for easy comparison

Wrapping up



Median Runtime
20M
18M
16M
14M
12M
10M
8M
6M
4M
2M
0
500 1000 1500 2000 2500 3000
Signature Size
Dilithium
Falcon
Haetae



Implementation Easiness
500 1000 1500 2000 2500 3000
Signature Size
Dilithium
Falcon
Haetae

Any question?