HAETAE: Shorter Fiat-Shamir with Aborts Signature

Jung Hee Cheon\textsuperscript{1,2}, Hyeongmin Choe\textsuperscript{1}, Julien Devevey\textsuperscript{3}, Tim Güneysu\textsuperscript{4,5}, Dongyeon Hong\textsuperscript{2}, Markus Krausz\textsuperscript{4}, Georg Land\textsuperscript{4}, Marc Möller\textsuperscript{4}, Damien Stehlé\textsuperscript{2}, MinJune Yi\textsuperscript{1}

\textsuperscript{1.} Seoul National University \hspace{1cm} \textsuperscript{2.} CryptoLab Inc. \hspace{1cm} \textsuperscript{3.} ENS de Lyon
\textsuperscript{4.} Ruhr University Bochum \hspace{1cm} \textsuperscript{5.} German Research Centre for Artificial Intelligence
**High-level Overview**

Same framework as Fiat-Shamir with Aborts over lattices

Our goals:

<table>
<thead>
<tr>
<th>Minimize signature size</th>
<th>Fixed-point arithmetic everywhere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace  with</td>
<td>Careful analysis of the sampler</td>
</tr>
<tr>
<td>Bimodal version of the scheme</td>
<td></td>
</tr>
</tbody>
</table>

J. Devevey - HAETAE
<table>
<thead>
<tr>
<th>Level</th>
<th>FALCON</th>
<th>DILITHIUM</th>
<th>HAETAE</th>
<th>H/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>vk</td>
<td>$</td>
<td>897</td>
<td>1312</td>
</tr>
<tr>
<td>$</td>
<td>\sigma</td>
<td>$</td>
<td>666</td>
<td>2420</td>
</tr>
<tr>
<td>KG cycles (average)</td>
<td>60M</td>
<td>339K</td>
<td>1.832M</td>
<td>540%</td>
</tr>
<tr>
<td>Sign cycles (average)</td>
<td>17M</td>
<td>1.446M</td>
<td>8.903M</td>
<td>616%</td>
</tr>
</tbody>
</table>

Haetae works over $R = Z[x]/(x^{256}+1)$ and uses a modulus $q = 64513$ (16 bits).

Between 1.5 and 2× less arithmetic operations than Dilithium.
<table>
<thead>
<tr>
<th></th>
<th><strong>FALCON</strong></th>
<th><strong>DILITHIUM</strong></th>
<th><strong>HAETAE</strong></th>
<th><strong>H/D</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level</strong></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vk</td>
<td></td>
<td>897</td>
<td>1312</td>
</tr>
<tr>
<td></td>
<td>σ</td>
<td></td>
<td>666</td>
<td>2420</td>
</tr>
<tr>
<td>KG cycles (average)</td>
<td>60M</td>
<td>339K</td>
<td>1.832M</td>
<td>540%</td>
</tr>
<tr>
<td>Sign cycles (average)</td>
<td>17M</td>
<td>1.446M</td>
<td>8.903M</td>
<td>616%</td>
</tr>
</tbody>
</table>

- Haetae works over $\mathcal{R} = \mathbb{Z}[x]/(x^{256} + 1)$ and uses a modulus $q = 64513$ (16 bits)
- Between 1.5 and $2\times$ less arithmetic operations than Dilithium
Fiat-Shamir with Aborts

KeyGen($1^\lambda$):

1: return $A, s$
   with $As = qj \mod 2q$

Sign($A, s, \mu$):

do

1: $y \leftarrow U(\bullet)$
2: $w = Ay \mod 2q$
3: $c = H(HB(w), LSB(w), \mu)$
4: $z = y + (-1)^b sc$
5: w.p. $p(z)$, set $z = \perp$

while $z = \perp$

6: $x = \text{compress}(z)$
7: return $(x, c)$
Fiat-Shamir with Aborts

\[
\text{do } \quad y \leftarrow Q
\]
\[
c = H(Ay \mod 2q, \mu)
\]
\[
z = y + (-1)^{U(\{0,1\})} \cdot sc
\]
\[
w.p. \quad \frac{2P(z)}{M(Q(z-sc)+Q(z+sc))}
\]
\[
z = \perp
\]
while \(z = \perp\)

return \((z, c)\)

- \(z \leftarrow P\)

- Verification relies on \(Az - qcj = Ay \mod 2q\) as \(As = -As = qj \mod 2q\)

- Bimodal [DDLL13] is more compact [DFPS22]

- Compactness depends on \(||sc||\)
Optimal Choice of Distribution

Our choice: continuous $U(\bullet)$

- Most compact [DFPS22]
- Easier rejection probability than Gaussians
- Rejection probability well-understood
- Rounding step before hashing $A[y]$ and compressing $\text{compress}([z])$
Rejection Step

KeyGen($1^\lambda$):
1. return $A, s$
   with $As = qj \mod 2q$

Sign($A, s, \mu$):
1. do
   1. $y \leftarrow U(\bullet)$
   2. $w = Ay \mod 2q$
   3. $c = H(HB(w), LSB(w), \mu)$
   4. $z = y + (-1)^b sc$
   5. w.p. $p(z)$, set $z = \bot$
while $z = \bot$
6. $x = \text{compress}(z)$
7. return $(x, c)$
Rejection Probability

\[ s_c - s_{c1}^{1/2} \]
Rejection Probability

Check $\|z\|$ and $\|2z - y\|$
Hyperball Sampler

**KeyGen($1^\lambda$):**

1. return $A, s$
   with $As = qj \mod 2q$

**Sign($A, s, \mu$):**

do

1. $y \leftarrow U(\mathbb{Z})$
2. $w = Ay \mod 2q$
3. $c = H(HB(w), \text{LSB}(w), \mu)$
4. $z = y + (-1)^b sc$
5. w.p. $p(z)$, set $z = \perp$

while $z = \perp$

6. $x = \text{compress}(z)$
7. return $(x, c)$

J. Devevey - HAETAE
Back to Gaussian sampling [VG17]

\[ n \pm 2 = D \cdot U(\cdot) \]

- Works for continuous distributions
Main Theorem

Back to Gaussian sampling [VG17]

\[ \begin{array}{c}
\underset{n}{\underbrace{}} \quad \underset{n+2}{\underbrace{}} \\
\end{array} =_D U(\bullet) \]

- Works for continuous distributions
- Adapted to work from discrete gaussian over \( \frac{1}{N} \mathbb{Z}^{n+2} \) to \( U(\bullet \cap \frac{1}{N} \mathbb{Z}^n) \)
- Requires large enough standard deviation and \( N \)
Implementation with Fixed-point Arithmetic

- Reject from discrete to discrete
- New average rejection probability?
- Close enough to the previous one for large $N$
- Balanced out with the previous constraint
Performances

Hyperball Sampler

Sign

Up to 80% of signing runtime!

J. Devevey - HAETAE
Hashing to a Ball

KeyGen($1^\lambda$):

1: return $A, s$

with $As = qj \mod 2q$

Sign$(A, s, µ)$:

do

1: $y \leftarrow U(\cdot)$
2: $w = Ay \mod 2q$
3: $c = H(HB(w), LSB(w), µ)$
4: $z = y + (-1)^b sc$
5: w.p. $p(z)$, set $z = \perp$

while $z = \perp$

6: $x = compress(z)$
7: return $(x, c)$
## Hash-related Choices

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Ternary $\binom{256}{\tau} + \tau$</th>
<th>Binary $\binom{256}{\tau}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>39</td>
<td>58</td>
</tr>
<tr>
<td>Level II $\tau$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NB: for Level V, to get 255 bits of entropy, we take $\tau$ with Hamming weight $< 128$ and half of those with Hamming weight 128

### SampleInBall($\tau$):

1. $c_0 \ldots c_{255} = 0^{256}$
2. **For** $i = 256 - \tau$ **to** 255
3. $j \leftarrow U(\{0 \ldots i\})$
4. $c_i = c_j$
5. $c_j = 1$
6. return $c$
Key Generation

KeyGen($1^\lambda$):
1: return $A, s$
   with $As = qj \mod 2q$

Sign($A, s, \mu$):

do
1: $y \leftarrow U(\bullet)$
2: $w = Ay \mod 2q$
3: $c = H(HB(w), LSB(w), \mu)$
4: $z = y + (-1)^b sc$
5: w.p. $p(z)$, set $z = \perp$
while $z = \perp$
6: $x = compress(z)$
7: return $(x, c)$
Key Generation

1: $A_0 \leftarrow U(\mathcal{R}_q^{k \times \ell - 1})$
2: $s_0, e_0 \leftarrow U([-\eta \ldots \eta])^{\ell - 1 + k}$
3: $b \leftarrow A_0 s_0 + e_0 \mod q$
4: $A \leftarrow (-2b + qj|2A_0|2I_k) \mod 2q$
5: $s \leftarrow (1|s_o^T|e_o^T)^T$
6: restart if $f_\tau(s) > n\beta^2/\tau$
7: return $vk = A, sk = s$

• $j = (1, 0 \ldots 0)^T$
• Add a trapdoor in the public matrix
• $f_\tau$ ensures that $\|sc\| \leq \beta$ for any $c$ with Hamming weight $\tau$
• Acceptance rate from 10 to 25%
Signature Compression (Two Ways)

KeyGen($1^\lambda$):
1: return $A, s$
   with $As = qj \mod 2q$

Sign($A, s, \mu$):

do
1: $y \leftarrow U(\bullet)$
2: $w = Ay \mod 2q$
3: $c = H(HB(w), LSB(w), \mu)$
4: $z = y + (-1)^b sc$
5: w.p. $p(z)$, set $z = \perp$
while $z = \perp$
6: $x = \text{compress}(z)$
7: return $(x, c)$
Low Bits Truncation

- Truncation technique from Bai and Galbraith

- $A_y = A_1 z_1 + 2 z_2 - q c j \mod 2q$ for some $A_1$

- Exclude $LB(z_2)$ from the signature

- Hash $HB(w)$ and $LSB(w)$
Similar to [ETWY22]
- range Asymmetric Numeral System used to encode/decode
- Swapped with tANS to reduce RAM usage
- Negligible cost in sign runtime (< 1%)
Security Estimation
<table>
<thead>
<tr>
<th>Theoretical</th>
<th>Practical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similar to Dilithium</td>
<td>Similar to Dilithium</td>
</tr>
<tr>
<td>• Reduction in the QROM depends on SelfTargetMSIS</td>
<td>• Key Recovery attacks solve an LWE instance</td>
</tr>
<tr>
<td>• Lossy-soundness in specific parameters regime</td>
<td>• Forgery attacks solve a SIS instance</td>
</tr>
</tbody>
</table>
Thank you!

Any question?