BAT: Small and Fast KEM over NTRU Lattices

Pierre-Alain Fouque, Paul Kirchner, Thomas Pornin, Yang Yu

CHES 2022

September 2022
Our initial Goal

BAT: a New KEM companion for Falcon signature, based on NTRU
- Latency and efficiency of TLS are impacted by lattice schemes
- Maximal IP packet size: 1536 bytes
- Falcon: smallest $|ct| + |pk|$ among all known lattice signature schemes
- Falcon has some drawbacks: FPA and hard to protect against SCA
Our initial Goal

BAT: a New KEM companion for Falcon signature, based on NTRU

- Latency and efficiency of TLS are impacted by lattice schemes
- Maximal IP packet size: 1536 bytes
- Falcon: smallest $|ct| + |pk|$ among all known lattice signature schemes
- Falcon has some drawbacks: FPA and hard to protect against SCA

- BAT avoids these issues: no need to FPA and easier to protect
- BAT ciphertext size:
  1. 473 bytes for BAT-512 (NIST-I)
  2. 1006 bytes for BAT-1024 (NIST-V)
  3. LW-BAT: 203 bytes for 80-bit security
- BAT enjoys fast encap/decap $\simeq$ Kyber, but relatively slow keygen
## Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>Security</th>
<th>Ciphertext (bytes)</th>
<th>PK (bytes)</th>
<th>Keygen (kcycles)</th>
<th>Encaps (kcycles)</th>
<th>Decaps (kcycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BAT</strong></td>
<td>128 bits</td>
<td>473</td>
<td>521</td>
<td>30.6 \cdot 10^3</td>
<td>8.4</td>
<td>54.3</td>
</tr>
<tr>
<td></td>
<td>256 bits</td>
<td>1006</td>
<td>1230</td>
<td>185.7 \cdot 10^3</td>
<td>18.5</td>
<td>118.6</td>
</tr>
<tr>
<td>LW-BAT</td>
<td>80 bits (c)</td>
<td>203</td>
<td>225</td>
<td>23.6 \cdot 10^3</td>
<td>55.7</td>
<td>248.0</td>
</tr>
<tr>
<td><strong>NTRU-HRSS</strong></td>
<td>128 bits</td>
<td>1140</td>
<td>1140</td>
<td>220.3</td>
<td>34.6</td>
<td>65.0</td>
</tr>
<tr>
<td><strong>NTTRU</strong></td>
<td>128 bits</td>
<td>1248</td>
<td>1248</td>
<td>6.4</td>
<td>6.1</td>
<td>7.8</td>
</tr>
<tr>
<td><strong>Kyber</strong></td>
<td>128 bits</td>
<td>768</td>
<td>800</td>
<td>33.9</td>
<td>45.2</td>
<td>34.6</td>
</tr>
<tr>
<td><strong>Kyber</strong></td>
<td>256 bits</td>
<td>1568</td>
<td>1568</td>
<td>73.5</td>
<td>97.3</td>
<td>79.1</td>
</tr>
<tr>
<td><strong>Saber</strong></td>
<td>128 bits</td>
<td>736</td>
<td>672</td>
<td>45.2</td>
<td>62.2</td>
<td>62.6</td>
</tr>
<tr>
<td><strong>Saber</strong></td>
<td>256 bits</td>
<td>1472</td>
<td>1312</td>
<td>126.2</td>
<td>153.8</td>
<td>155.7</td>
</tr>
<tr>
<td><strong>LAC</strong></td>
<td>128 bits</td>
<td>712</td>
<td>544</td>
<td>59.6</td>
<td>89.1</td>
<td>140.2</td>
</tr>
<tr>
<td><strong>LAC</strong></td>
<td>256 bits</td>
<td>1424</td>
<td>1056</td>
<td>135.8</td>
<td>208.0</td>
<td>359.2</td>
</tr>
<tr>
<td><strong>Round5</strong></td>
<td>128 bits</td>
<td>620</td>
<td>461</td>
<td>46</td>
<td>68</td>
<td>95</td>
</tr>
<tr>
<td><strong>Round5</strong></td>
<td>256 bits</td>
<td>1285</td>
<td>978</td>
<td>105</td>
<td>166</td>
<td>247</td>
</tr>
<tr>
<td><strong>Round5-iot</strong></td>
<td>96 bits (c)</td>
<td>394</td>
<td>342</td>
<td>41</td>
<td>52</td>
<td>28</td>
</tr>
<tr>
<td><strong>RSA 4096 ECC</strong></td>
<td>128 bits (c)</td>
<td>512</td>
<td>512</td>
<td>2.19 \times 10^6</td>
<td>212.1</td>
<td>13690</td>
</tr>
<tr>
<td><strong>SIKE p434</strong></td>
<td>128 bits (c)</td>
<td>32</td>
<td>32</td>
<td>46</td>
<td>176</td>
<td>130</td>
</tr>
<tr>
<td><strong>Compressed SIKE p434</strong></td>
<td>128 bits</td>
<td>346</td>
<td>330</td>
<td>5.9 \times 10^3</td>
<td>9.7 \times 10^3</td>
<td>10.3 \times 10^3</td>
</tr>
<tr>
<td></td>
<td>128 bits</td>
<td>236</td>
<td>197</td>
<td>10.2 \times 10^3</td>
<td>15.1 \times 10^3</td>
<td>11.1 \times 10^3</td>
</tr>
</tbody>
</table>
BAT: a New decryption process for NTRU (more natural ?)

NTRU

Public key: $h = g/f \mod q$
$g, f$ small coeff. $\in \mathbb{Z}_q[X]$
Secret key: $(f, f^{-1} \mod p)$
Encrypt: $c = phr + m \mod q$
$p$ masking mod., $r$ random
Decrypt:
$c' = fc = pgr + fm \mod q$
$c' = pgr + fm$ over $\mathbb{Z}$
$(f^{-1} \mod p)c' = m \mod p$

BAT

Public key: $h = g/f \mod q$
Secret key: $B_{f,g} = \begin{pmatrix} g & G \\ f & F \end{pmatrix}$
s.t. $gF - fG = q$
Encrypt: $c = hm + r \mod q$
Decrypt: Dec. $\begin{pmatrix} c \\ 0 \end{pmatrix}$ w/ $B_{f,g}$
$L_{h,q} = \{(u, v)^t | u = hv \mod q\}$ has $B_{f,g}$ (short basis)

Solving 2 linear equations with 2 unknowns:
$\begin{pmatrix} Fc \\ -fc \end{pmatrix} = \begin{pmatrix} F & -G \\ -f & g \end{pmatrix} \begin{pmatrix} r \\ -m \end{pmatrix}$
New efficient NTRU Decoding

Decoding algorithms

- Given $c$, find short polynomials $(e, s)$ s.t. $c = hs + e$
- All operations simple and efficient: no high-precision arithmetic
- Large decoding distance means high security level: large errors $(s, e)$
- Babai Rounding: efficient and simple but cannot decode large errors
- Nearest Plane: decode larger errors but require high-prec. arithmetic
New efficient NTRU Decoding

Decoding algorithms

- Given $c$, find short polynomials $(e, s)$ s.t. $c = hs + e$
- All operations simple and efficient: no high-precision arithmetic
- Large decoding distance means high security level: large errors $(s, e)$
- Babai Rounding: efficient and simple but cannot decode large errors
- Nearest Plane: decode larger errors but require high-prec. arithmetic

Falcon and the BAT New NTRU decoding

- Falcon: Signature size $\propto$ maximal Gram-Schmidt norm of $B_{f,g}$ (NP)
- BAT: Decode larger errors than Babai Rounding
- BAT: All algorithms can be implemented using fixed-point arithmetic
- BAT: expensive computations are pre-computed in the Key Gen
- BAT: Optimization between the distribution of $e$ and $s$
Babai Rounding vs. Nearest Plane Decoding

- Babai Rounding decoder:
  \[
  \begin{pmatrix}
  e' \\
  -s'
  \end{pmatrix} = \begin{pmatrix}
  c \\
  0
  \end{pmatrix} - \mathbf{B}^{-1}_{f,g} \begin{pmatrix}
  \mathbf{B}^{-1}_{f,g} & c \\
  0 & 0
  \end{pmatrix}
  \]

- As \( \mathbf{B}^{-1}_{f,g} = \frac{1}{q} \begin{pmatrix}
  F & -G \\
  -f & g
  \end{pmatrix} \), \((e', s') = (e, s)\) if
  \[
  \begin{pmatrix}
  F & -G \\
  -f & g
  \end{pmatrix} \begin{pmatrix}
  e \\
  -s
  \end{pmatrix} = \begin{pmatrix}
  Fc \mod q \\
  -fc \mod q
  \end{pmatrix}
  \]

- It is correct iff \( \max\{ \| fe + gs \|_\infty, \| Fe + Gs \|_\infty \} \leq q/2 \)
Babai Rounding vs. Nearest Plane Decoding

- Babai Rounding decoder: \[
\begin{pmatrix}
e' \\
s'
\end{pmatrix} = \begin{pmatrix}c \\ 0\end{pmatrix} - B_{f,g} \begin{pmatrix}b_{f,g}^{-1} \\ 0\end{pmatrix}
\]

- As \(B_{f,g}^{-1} = \frac{1}{q} \begin{pmatrix}F & -G \\
-f & g\end{pmatrix}\), \((e', s') = (e, s)\) if

\[
\begin{pmatrix}F & -G \\
-f & g\end{pmatrix} \begin{pmatrix}e \\ -s\end{pmatrix} = \begin{pmatrix}Fc \mod q \\ -fc \mod q\end{pmatrix}
\]

- It is correct iff \(\max\{\|fe + gs\|_{\infty}, \|Fe + Gs\|_{\infty}\} \leq q/2\)

- NP decoder correct iff \(\max\{\|fe + gs\|_{\infty}, \|F^*e + G^*s\|_{\infty}\} \leq q/2\), where \(F^*, G^*\) are the Gram-Schmidt orthogonalized basis \(B_{f,g}^*\)

- \(\|(g, f)\| \approx \|(G^*, F^*)\|\), but \(\|(G, F)\| \approx \sqrt{\frac{n}{12}} \cdot \|(g, f)\|\)

- \(\|(e, s)\|\) is dominated by \(\|(G, F)\|\) in Babai Rounding, \(\|(g, f)\|\) in NP
A New Decoding Algorithm for NTRU

Goal: Replace the large \((G, F)\) by some small \((G', F')\) of size \(\approx \| (g, f) \|\)

\[ B_{f,g}^* = \begin{pmatrix} g \\ f \end{pmatrix} \quad G^* = G - vg \]

\[ F^* = F - vf \]

where \(v = \frac{F\bar{f} + G\bar{g}}{ff + gg} \)

\((G', F') = (G - g\lfloor v \rfloor_{q'}, F - f\lfloor v \rfloor_{q'})\)

If \(q'\) is large, \((G', F')\) converges to \((G^*, F^*)\) whose norm is \(\approx \| (g, f) \|\)
A New Decoding Algorithm for NTRU

Goal: Replace the large \((G, F)\) by some small \((G', F')\) of size \(\approx \|(g, f)\|\)

- \(B^*_{f, g} = \begin{pmatrix} g & G^* = G - vg \\ f & F^* = F - vf \end{pmatrix}\) where \(v = \frac{F\bar{f} + G\bar{g}}{f\bar{f} + g\bar{g}}\)
- \((G', F') = (G - g\lfloor v \rfloor q', F - f\lfloor v \rfloor q')\)
- If \(q'\) is large, \((G', F')\) converges to \((G^*, F^*)\) whose norm is \(\approx \|(g, f)\|\)

Refinement: Distributions of \(e\) and \(s\) are not the same \(\|s\| \ll \|e\|\)

- Better decoding if we tweak the \(\begin{pmatrix} g' & G' \\ f' & F' \end{pmatrix}\)
  with \(\|g'\| > \|g\| \approx \|f\| > \|f'\|\) and \(\|G'\| > \|G\| \approx \|F\| > \|F'\|\)
- \(\gamma = \sigma_e / \sigma_s\) tweaking parameter
**BAT Encryption**

\[
c_1 = \left\lfloor \frac{hs \mod q}{k} \right\rfloor \\
\]

\[
e = (hs \mod q) - kc_1 \\
\]

if \(\|(\gamma s, e)\| > \text{thres.}\), return ⊥

else return \((c_1, c_2 = m \oplus H(s))\)

---

**BAT Decryption**

\[
(e, s) \leftarrow \text{Decode} (c_1, k, sk) \\
\]

if \(\|(\gamma s, e)\| > \text{thres.}\), return ⊥

else if \(c_1 = \left\lfloor \frac{hs \mod q}{k} \right\rfloor\), return

\[
H(s) \oplus c_2 \\
\]

---

Key Encapsulation Method derived from the Encryption Scheme using Duman et al. [eprint 2021/1352]. Security proof in the QROM
(Decision) NTRU assumption: \( \mathcal{R} = \mathbb{Z}[x]/(x^n + 1) \), \( \chi \) distrib. of \( f, g \)

\[
\text{Adv}_{\mathcal{R}, q, \chi}^{\text{NTRU}}(A) = \Pr[b = 1 \mid f, g \leftarrow \chi; b \leftarrow A(f^{-1}g \mod q)] - \Pr[b = 1 \mid u \leftarrow U(\mathcal{R}_q^\times); b \leftarrow A(u)]
\]

(Search) Ring-LWR assumption: \( \chi = U(\mathcal{R} \mod 2) \)

\[
\text{Adv}_{\mathcal{R}, q, k, \chi}^{\text{RLWR}}(A) = \Pr_{a \leftarrow U(\mathcal{R}_q^\times), s \leftarrow \chi} \left[ A \left( a, \left\lfloor \frac{(as \mod q)}{k} \right\rfloor \right) = s \right]
\]
Security parameters and attack cost

- $n = 2^\ell$ and $q'$ is used to control the decryption failure rate.
- $q = bk + 1$: $b$ size of each ciphertext coefficient, $k$ decoding distance.

<table>
<thead>
<tr>
<th>Security</th>
<th>$n$</th>
<th>$(b, k, q)$</th>
<th>$\sigma_f$</th>
<th>$q'$</th>
<th>Decrypt. Fail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 bits</td>
<td>256</td>
<td>(64, 2, 128)</td>
<td>0.595</td>
<td>64513</td>
<td>$2^{-71.9}$</td>
</tr>
<tr>
<td>128 bits</td>
<td>512</td>
<td>(128, 2, 257)</td>
<td>0.596</td>
<td>64513</td>
<td>$2^{-146.7}$</td>
</tr>
<tr>
<td>256 bits</td>
<td>1024</td>
<td>(192, 4, 769)</td>
<td>0.659</td>
<td>64513</td>
<td>$2^{-166.7}$</td>
</tr>
</tbody>
</table>
Security parameters and attack cost

- $n = 2^\ell$ and $q'$ is used to control the decryption failure rate.
- $q = bk + 1$: $b$ size of each ciphertext coefficient, $k$ decoding distance.

<table>
<thead>
<tr>
<th>Security</th>
<th>$n$</th>
<th>$(b, k, q)$</th>
<th>$\sigma_f$</th>
<th>$q'$</th>
<th>Decrypt. Fail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 bits</td>
<td>256</td>
<td>(64, 2, 128)</td>
<td>0.595</td>
<td>64513</td>
<td>$2^{-71.9}$</td>
</tr>
<tr>
<td>128 bits</td>
<td>512</td>
<td>(128, 2, 257)</td>
<td>0.596</td>
<td>64513</td>
<td>$2^{-146.7}$</td>
</tr>
<tr>
<td>256 bits</td>
<td>1024</td>
<td>(192, 4, 769)</td>
<td>0.659</td>
<td>64513</td>
<td>$2^{-166.7}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security</th>
<th>Key Recovery</th>
<th>Message Recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>primal</td>
<td>hybrid</td>
</tr>
<tr>
<td>80 bits</td>
<td>87.8 / 236</td>
<td>90.3</td>
</tr>
<tr>
<td>128 bits</td>
<td>152.1 / 475</td>
<td>164.1</td>
</tr>
<tr>
<td>256 bits</td>
<td>274.4 / 933</td>
<td>314.4</td>
</tr>
</tbody>
</table>

**Table:** Concrete security estimate. “A/B”: attack cost A and BKZ blocksize B.
Table: The required storage (full format, including the header byte)

<table>
<thead>
<tr>
<th>Security</th>
<th>Public Key (bytes)</th>
<th>Ciphertext (with FO, bytes)</th>
<th>Private key (short, bytes)</th>
<th>Secret Key (long, bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 bits</td>
<td>225</td>
<td>203</td>
<td>225</td>
<td>1473</td>
</tr>
<tr>
<td>128 bits</td>
<td>521</td>
<td>473</td>
<td>417</td>
<td>2953</td>
</tr>
<tr>
<td>256 bits</td>
<td>1230</td>
<td>1006</td>
<td>801</td>
<td>6030</td>
</tr>
</tbody>
</table>

Measured on Intel i5-8259U CPU clocked at 2.3 GHz; TurboBoost is disabled.
### BAT: Storage and Speed Performances

**Table:** The required storage (full format, including the header byte)

<table>
<thead>
<tr>
<th>Security</th>
<th>Public Key (bytes)</th>
<th>Ciphertext (with FO, bytes)</th>
<th>Private key (short, bytes)</th>
<th>Secret Key (long, bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 bits</td>
<td>225</td>
<td>203</td>
<td>225</td>
<td>1473</td>
</tr>
<tr>
<td>128 bits</td>
<td>521</td>
<td>473</td>
<td>417</td>
<td>2953</td>
</tr>
<tr>
<td>256 bits</td>
<td>1230</td>
<td>1006</td>
<td>801</td>
<td>6030</td>
</tr>
</tbody>
</table>

**Table:** The performance of the plain C implementation

<table>
<thead>
<tr>
<th>Security</th>
<th>Key Generation (cycles)</th>
<th>Encapsulation (cycles)</th>
<th>Decapsulation (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 bits</td>
<td>$\approx 23.8 \times 10^6$</td>
<td>82131</td>
<td>392036</td>
</tr>
<tr>
<td>128 bits</td>
<td>$\approx 37.2 \times 10^6$</td>
<td>35785</td>
<td>279260</td>
</tr>
<tr>
<td>256 bits</td>
<td>$\approx 264.7 \times 10^6$</td>
<td>71007</td>
<td>537580</td>
</tr>
</tbody>
</table>

Measured on Intel i5-8259U CPU clocked at 2.3 GHz; TurboBoost is disabled
Conclusion

We present a new NTRU-based KEM, called BAT

- more compact than all known lattice-based KEMs
- encap/decap are fast comparable to Kyber
Conclusion

We present a new NTRU-based KEM, called BAT

- more compact than all known lattice-based KEMs
- encap/decap are fast comparable to Kyber

The complexity of the code as well as its running time is asymmetric

- cheap daily operations are favourable to small devices
- expensive keygen can be compensated by frequent key usage or regular key creation for forward secrecy
Conclusion

We present a new NTRU-based KEM, called BAT
- more compact than all known lattice-based KEMs
- encap/decap are fast comparable to Kyber

The complexity of the code as well as its running time is asymmetric
- cheap daily operations are favourable to small devices
- expensive keygen can be compensated by frequent key usage or regular key creation for forward secrecy

BAT and Falcon use similar key structure
- BAT has simpler and faster daily operations
- BAT is implemented fully over integers and smaller, thus more compatible with small devices