BIKE
5th NIST PQC Standardization Workshop
April 10th, 2024

Nicolas Aragon, University of Limoges, France
Paulo L. Barreto, University of Washington Tacoma, USA
Slim Bettaieb, Worldline, France
Loïc Bidoux, Worldline, France
Olivier Blazy, University of Limoges, France
Jean-Christophe Deneuville, ENAC, Federal University of Toulouse, France
Philippe Gaborit, University of Limoges, France
Santosh Ghosh, Intel, USA
Shay Gueron, University of Haifa, and Meta, Israel & USA

Tim Güneysu, Ruhr-Universität Bochum & DFKI, Germany
Carlos Aguilar Melchor, University of Toulouse, France
Rafael Misoczki, Meta, USA
Edoardo Persichetti, Florida Atlantic University, USA
Jan Richter-Brockmann, Ruhr-Universität Bochum, Germany
Nicolas Sendrier, INRIA, France
Jean-Pierre Tillich, INRIA, France
Valentin Vasseur, Thales, France
Gilles Zémor, IMB, University of Bordeaux, France

https://bikesuite.org
Agenda

- BIKE Recap
- Proposed tweaks
  - Enhancing multi-target security
  - Constant-weight sampler
  - Decoder
BIKE Recap

### KeyGen

\[ \text{Input: } (h_0, h_1, \sigma) \in \mathcal{H}_w \times \mathcal{M}, \quad h \in \mathcal{R} \]

\[ \text{Output: } (h_0, h_1, \sigma) \in \mathcal{H}_w \times \mathcal{M}, \quad h \in \mathcal{R} \]

1. \( (h_0, h_1) \overset{\mathcal{D}}{\longleftarrow} \mathcal{H}_w \) \hspace{1cm} (1)
2. \( h \leftarrow h_1h_0^{-1} \)
3. \( \sigma \leftarrow \mathcal{M} \)

### Encaps

\[ \text{Input: } h \mapsto K, c \]

\[ \text{Output: } K \in \mathcal{K}, \quad c \in \mathcal{R} \times \mathcal{M} \]

1. \( m \leftarrow \mathcal{M} \)
2. \( (e_0, e_1) \leftarrow \mathbf{H}(m) \)
3. \( c \leftarrow (e_0 + e_1h, m \oplus \mathbf{L}(e_0, e_1)) \)
4. \( K \leftarrow K(m, c) \)

### Decaps

\[ \text{Input: } (h_0, h_1, \sigma), c \mapsto K \]

\[ \text{Output: } K \in \mathcal{K} \]

1. \( e' \leftarrow \text{decoder}(c_0h_0, h_0, h_1) \) \hspace{1cm} \( e' \in \mathcal{R}^2 \cup \{\bot\} \)
2. \( m' \leftarrow c_1 \oplus \mathbf{L}(e') \) \hspace{1cm} \( \text{with the convention } \bot = (0, 0) \)
3. \( \text{if } e' = \mathbf{H}(m') \text{ then } K \leftarrow K(m', c) \text{ else } K \leftarrow K(\sigma, c) \)

### Design

- Niederreiter-based KEM instantiated with QC-MDPC codes (faster polynomial inversion by [DGK’20]).
- Leverage Fujisaki-Okamoto Transform [DGKP’21].
- State-of-the-art QC-MDPC Decoding Failure Rate analysis.
- Black-Gray-Flip Decoder implemented in constant time.

### Notation

- \( \mathbb{F}_q \): Binary finite field.
- \( \mathcal{R} \): Cyclic polynomial ring \( \mathbb{F}_q[X]/(X^r - 1) \).
- \( \mathcal{H}_w \): Private key space \( \{(h_0, h_1) \in \mathbb{R}^2 \mid |h_0| = |h_1| = w/2\} \).
- \( \mathcal{E}_t \): Error space \( \{(e_0, e_1) \in \mathbb{R}^2 \mid |e_0| + |e_1| = t\} \).
- \( |\cdot| \): Hamming weight of a binary polynomial \( g \in \mathcal{R} \).
- \( \oplus \): Variable \( u \) is sampled uniformly at random from the set \( U \).
- \( \oplus \): exclusive or of two bits, componentwise with \( \oplus \).
- \( \mathcal{M} \): message space in \( \{0, 1\}^\ell \).
- \( \mathcal{K} \): key space in \( \{0, 1\}^\ell \).
- \( r \): block length.
- \( w \): row weight.
- \( t \): error weight.
- \( \ell \): shared secret size.

### Functions

- \( \mathbf{H} : \mathcal{M} \mapsto \mathcal{E}_t \).
- \( \mathbf{K} : \mathcal{M} \times \mathcal{R} \times \mathcal{M} \mapsto \mathcal{K} \).
- \( \mathbf{L} : \mathcal{R}^2 \mapsto \mathcal{M} \).
## BIKE Recap

### Level 1

<table>
<thead>
<tr>
<th></th>
<th>Public key size (bytes)</th>
<th>Ciphertext size (bytes)</th>
<th>KeyGen (kilocycles)</th>
<th>Encaps (kilocycles)</th>
<th>Decaps (kilocycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIKE</td>
<td>1,540</td>
<td>1,572</td>
<td>589</td>
<td>97</td>
<td>1,135</td>
</tr>
<tr>
<td>HQC</td>
<td>2,249</td>
<td>4,497</td>
<td>187</td>
<td>419</td>
<td>833</td>
</tr>
<tr>
<td>mceliece348864</td>
<td>261,120</td>
<td>128</td>
<td>140,870</td>
<td>46</td>
<td>137</td>
</tr>
<tr>
<td>Kyber-512</td>
<td>800</td>
<td>768</td>
<td>123</td>
<td>155</td>
<td>289</td>
</tr>
</tbody>
</table>

### Level 3

<table>
<thead>
<tr>
<th></th>
<th>Public key size (bytes)</th>
<th>Ciphertext size (bytes)</th>
<th>KeyGen (kilocycles)</th>
<th>Encaps (kilocycles)</th>
<th>Decaps (kilocycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIKE</td>
<td>3,082</td>
<td>3,114</td>
<td>1,823</td>
<td>223</td>
<td>3,887</td>
</tr>
<tr>
<td>HQC</td>
<td>4,522</td>
<td>9,042</td>
<td>422</td>
<td>946</td>
<td>1,662</td>
</tr>
<tr>
<td>mceliece460896</td>
<td>524,160</td>
<td>188</td>
<td>441,517</td>
<td>83</td>
<td>273</td>
</tr>
<tr>
<td>Kyber-768</td>
<td>1,184</td>
<td>1,088</td>
<td>213</td>
<td>249</td>
<td>275</td>
</tr>
</tbody>
</table>

BIKE performance numbers from Drucker, Gueron, Kostic, "Additional implementation of BIKE (Bit Flipping Key Encapsulation)".

[https://github.com/awslabs/bike-kem](https://github.com/awslabs/bike-kem)
Security in the Multi-Target Setting

- In [WWW’23], a multi-target attack against the CCA variant of BIKE leveraging decryption failures was presented
  - The attack needs to first identify a key (out of many, e.g. $2^{87}$ keys) so that the gathering property is observed
  - Queries per target: $\sim 2^{29}$ steps
  - Total complexity: $\sim 2^{116}$ steps
- The attack is defeated by binding the public key to the ciphertext

Security in the Multi-Target Setting

- In [DGK’21], a fix has been proposed, performance-studied and implemented.
- BIKE will adopt this protection moving forward, thus defeating such multi-target attacks.

\[ f_1 : \mathcal{M} \times \mathcal{PK} \rightarrow \{0, 1\}^{256} \]
\[ (m, pk) \mapsto H(m \parallel pk) \]

\[ f_2 : \mathcal{M} \times \mathcal{PK} \rightarrow \{0, 1\}^{256} \]
\[ (m, pk) \mapsto H(m \parallel H'(pk)) \]

Constant Weight Sampling

- Variant of Fisher-Yates algorithm was vulnerable to timing attacks [GHJ’22]. Latest spec fixed this: a constant-time variant (slightly biased output).

- Using a biased sampler in Encaps/Decaps has no impact to security [Sen’23]. To avoid code duplication, we also use the biased sampler in KeyGen.

- However, in KeyGen there would be an impact to the security reduction [DHK’23] (assumption $h = h_0^{-1} h_1$ must be indistinguishable from random when $h_0$ and $h_1$ are produced by the biased sampler).

- Proposed tweak: Revert BIKE KeyGen to unbiased constant-weight sampler.


New Bit-Flipping Decoder

- Simpler algorithm (no black/gray iterations) with modified threshold schedule.

- State-of-the-art extrapolation techniques [SV’20] for the waterfall region predict a DFR at most $2^{-180}$ for level 1 with blocklength 12,323.

- Better resistance to weak key attacks [WWW’23]. In the region where simulation is possible, the attack requires $2^{168}$ decapsulation queries instead of $2^{116}$.

New Bit-Flipping Decoder

Algorithm 1 New BIKE Decoder

Input: $s \in \mathbb{F}_2^n$, $H \in \mathbb{F}_2^{r \times n}$

1: $\tilde{e} \leftarrow 0^n$ ; $\tilde{s} \leftarrow s$

2: for $i = 1, \ldots, \text{NbIter}$ do

3: $T \leftarrow \text{THRESHOLD}(i, s, \tilde{s})$

4: for $j = 0, \ldots, n - 1$ do

5: $\sigma_j \leftarrow \text{ctr}(H, \tilde{s}, j)$

6: for $j = 0, \ldots, n - 1$ do

7: if $\sigma_j \geq T$ then

8: $\tilde{e}_j \leftarrow \tilde{e}_j \oplus 1$

9: $\tilde{s} \leftarrow \tilde{s} - \text{col}(H, j)$

10: return $\tilde{e}$

$\text{ctr}(H, \tilde{s}, j)$ number of unsatisfied equations involving position $j$

1: function $\text{THRESHOLD}(i, s, \tilde{s})$

2: $T' \leftarrow f_t(|s|)$ \quad \triangleright \text{optimal}$

3: $M \leftarrow (d + 1)/2$ \quad \triangleright \text{majority}$

4: if $i = 1$ then $T \leftarrow T' + \delta$

5: if $i = 2$ then $T \leftarrow (2T' + M)/3 + \delta$

6: if $i = 3$ then $T \leftarrow (T' + 2M)/3 + \delta$

7: if $i \geq 4$ then $T \leftarrow M + \delta$

8: return $\max(f_t(|\tilde{s}|), T)$

$f_t(x) = 0.006258 \cdot x + 11.094$, $\delta = 3$ (level 1)
References

- [DGK]: Drucker, Gueron, Kostic, "Additional implementation of BIKE (Bit Flipping Key Encapsulation)". https://github.com/awslabs/bike-kem.
Questions?

https://bikesuite.org
Constant Weight Sampling

Algorithm 3 WSHAKE256-PRF(seed, len, wt)

Require: seed (32 bytes), len, wt
Ensure: A list (wlist) of wt distinct elements in \{0, \ldots, len − 1\}.

1: wlist ← () \quad \triangleright \text{empty list}
2: s_0, \ldots, s_{wt−1} ← SHAKE256-Stream(seed, 32 \cdot wt)
   \quad \triangleright \text{parse as a sequence of } wt \text{ non negative 32-bits integers}
3: for i = (wt − 1), \ldots, 1, 0 do \quad \triangleright i \text{ decreasing from } wt − 1 \text{ to } 0
4: \quad \text{pos} ← i + [(len − i)s_i/2^{32}]
5: \quad wlist ← wlist, (\text{pos } \in \text{ wlist}) ? i : \text{pos}
6: \text{return wlist}