Verifying Post-Quantum Signatures in 8 kB of RAM

Ruben Gonzalez¹, Andreas Hülsing², Matthias J. Kannwischer³, Juliane Krämer⁴, Tanja Lange², Marc Stöttinger⁵, Elisabeth Waitz⁶, Thom Wiggers⁷, and Bo-Yin Yang⁸

Third PQC Standardization Conference, 07. - 09. June 2021
Presenter: Ruben Gonzalez
Background

- We’re at Round 3
  - Let’s look at the real world
- PQC for Embedded Systems Workshop
  - Bringing together industry and academia
Use Case

● Feature Activation in Cars
  ○ Short signed messages
### Use Case

- **Feature Activation in Cars**
  - Short signed messages
- **Protocol already exists**
  - Uses ECC

<table>
<thead>
<tr>
<th>User</th>
<th>Authorization Entity: $AE$</th>
<th>Device: $D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req. feature activation</td>
<td>Validate feature activation request</td>
<td></td>
</tr>
<tr>
<td>Req. authentication Authenticate</td>
<td>Verify authentication</td>
<td></td>
</tr>
<tr>
<td>$T_1 : \text{Sign}<em>{pr</em>{AE}}(A_{msg})$</td>
<td>Send ${A_{msg}</td>
<td>T_1}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update feature policies</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Activate feature</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Secure hash of the feature policies</td>
</tr>
<tr>
<td>Update feature polices of $D$</td>
<td>$\text{Send } A_{rec}$</td>
<td>Generate $A_{rec}$</td>
</tr>
</tbody>
</table>
Use Case

- **Feature Activation in Cars**
  - Short signed messages

- **Protocol already exists**
  - Uses ECC

- **HSM has to verify signatures and Pubkey**
  - Is resource constrained
  - Holds hash of public key
  - Stores activated features in secure memory
<table>
<thead>
<tr>
<th>Investigated Schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS+</td>
</tr>
<tr>
<td>GeMSS</td>
</tr>
<tr>
<td>Rainbow</td>
</tr>
<tr>
<td>Dilithium</td>
</tr>
<tr>
<td>Falcon</td>
</tr>
</tbody>
</table>

**SPHINCS+**
- Hash Based

**GeMSS**
- Multivariate

**Rainbow**
- Multivariate

**Dilithium**
- Lattice Based

**Falcon**
- Lattice Based
SPHINCS+ (SHA256-128)
SPHINCS+ (SHA256-128)

- Signature processed in order
- Pubkey fits in memory
- Chunk size of 16B possible
Rainbow (I-classic)

Rainbow (I-classic) Public Key

16,600B

Polynomial Coefficients

10100 × 

32B

Coefficients

64 × 

Coefficient in $F_{16}$

Rainbow (I-classic) Signature

66B

Variables

Salt

50B

4b

16B

100 × Variables in $F_{16}$
Rainbow (I-classic)

Rainbow (I-classic) Public Key
- 16,600B
  - Polynomial Coefficients
    - 32B
      - Coefficients
        - 10100 x
          - 4b
            - Coefficient in $F_{16}$

Rainbow (I-classic) Signature
- 66B
  - Variables
    - 50B
  - Salt
    - 16B

Mathematical expressions:

$p^{(1)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} p^{(1)}_{ij} \cdot x_i x_j + \sum_{i=1}^{n} (p^{(1)}_i \cdot x_i + p^{(1)}_0)$

$p^{(2)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} p^{(2)}_{ij} \cdot x_i x_j + \sum_{i=1}^{n} (p^{(2)}_i \cdot x_i + p^{(2)}_0)$

$p^{(m)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} p^{(m)}_{ij} \cdot x_i x_j + \sum_{i=1}^{n} (p^{(m)}_i \cdot x_i + p^{(m)}_0)$
Rainbow (I-classic)

- Public key processed in order
- Signature fits in memory
- Chunk size of 32B possible
GeMSS (128)
GeMSS (128)

- Verification has 4 iterations
  - Pubkey has to be streamed 4 times
- Signature fits in memory
- Chunk size of 2174B possible
  - Due to row wise storage
Dilithium (2)

Dilithium (2) Public Key

- **Seed**: 32B
- **Polynomials**: 1,280B
- Total: 1,312B

Dilithium (2) Signature

- **Challenge**: 32B
- **Polynomials ($z$)**: 2,304B
- **Polynomials ($h$)**: 84B
- Total: 2,420B

**Polynomials in $\mathbb{R}$**

$$w' = Az - c \cdot t_1 \cdot 2^d$$
Dilithium (2)

- Signature fits in memory
- Public key is streamed one polynomial at a time
- Chunk size of 2420B and then 320B possible
Falcon

- No streaming required
- Everything fits in memory
# Results

## Data Volume

<table>
<thead>
<tr>
<th></th>
<th>streaming data</th>
<th>streaming time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>500 kbit/s</td>
</tr>
<tr>
<td>sphincs-s(^a)</td>
<td>32</td>
<td>7856</td>
</tr>
<tr>
<td>sphincs-f(^b)</td>
<td>32</td>
<td>17088</td>
</tr>
<tr>
<td>rainbowI-classic</td>
<td>161 600</td>
<td>66</td>
</tr>
<tr>
<td>gemss-128</td>
<td>352 188</td>
<td>33</td>
</tr>
<tr>
<td>dilithium2</td>
<td>1 312</td>
<td>2420</td>
</tr>
<tr>
<td>falcon-512</td>
<td>897</td>
<td>690</td>
</tr>
</tbody>
</table>

\(^a\)-sha256-128s-simple  \(^b\)-sha256-128f-simple  \(^c\)4 \(\times |pk| + |sig|\)

## Cycle Counts

<table>
<thead>
<tr>
<th></th>
<th>w/o pk vrf.</th>
<th>w/ pk verification</th>
<th>w/ streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pk vrf.</td>
<td>total</td>
<td>time(^a)</td>
</tr>
<tr>
<td>sphincs-s(^a)</td>
<td>8 741 k</td>
<td>0</td>
<td>8 741 k</td>
</tr>
<tr>
<td>sphincs-f(^b)</td>
<td>26 186 k</td>
<td>0</td>
<td>26 186 k</td>
</tr>
<tr>
<td>rainbowI-classic</td>
<td>333 k</td>
<td>6 850 k(^d)</td>
<td>7 182 k</td>
</tr>
<tr>
<td>gemss-128</td>
<td>1 619 k</td>
<td>109 938 k(^e)</td>
<td>111 557 k</td>
</tr>
<tr>
<td>dilithium2</td>
<td>1 990 k</td>
<td>133 k(^f)</td>
<td>2 123 k</td>
</tr>
<tr>
<td>falcon-512</td>
<td>581 k</td>
<td>91 k(^f)</td>
<td>672 k</td>
</tr>
</tbody>
</table>

\(^a\)-sha256-128s-simple  \(^b\)-sha256-128f-simple  \(^c\)SHA-3/SHAKE

\(^d\)SHA-256  \(^e\)At 100 MHz (no wait states)

## Memory Usage

<table>
<thead>
<tr>
<th></th>
<th>memory</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>buffer</td>
</tr>
<tr>
<td>sphincs-s(^a)</td>
<td>6 904</td>
<td>4 928</td>
</tr>
<tr>
<td>sphincs-f(^b)</td>
<td>7 536</td>
<td>4 864</td>
</tr>
<tr>
<td>rainbowI-classic</td>
<td>8 168</td>
<td>6 848</td>
</tr>
<tr>
<td>gemss-128</td>
<td>8 176</td>
<td>4 560</td>
</tr>
<tr>
<td>dilithium2</td>
<td>8 048</td>
<td>40</td>
</tr>
<tr>
<td>falcon-512</td>
<td>6 552</td>
<td>897</td>
</tr>
</tbody>
</table>

\(^a\)-sha256-128s-simple  \(^b\)-sha256-128f-simple
Resources


Code: https://git.fslab.de/pqc/streaming-pq-sigs/

Images used in this presentation were licensed from colourbox.de or freesvg.org. Hyptertree visualization by Abid Khan.