XMSS
Practical Hash-Based Signatures

Andreas Hülsing
joint work with Johannes Buchmann and Erik Dahmen
Post-Quantum Signatures

Lattice, MQ, Coding

- Signature and/or key sizes
- Runtimes
- Secure parameters

\[ y_1 = x_1^2 + x_1 x_2 + x_1 x_4 + x_3 \]
\[ y_2 = x_3^2 + x_2 x_3 + x_2 x_4 + x_1 + 1 \]
\[ y_3 = \ldots \]
Hash-based Signature Schemes

[Mer89]

Post quantum
Only secure hash function
Security well understood
Fast
Inherently forward secure
Hash-based Signatures

\[
\text{SIG} = (i=2, \ H, \ SK, \ \bigcirc, \ \bigcirc, \ \bigcirc)
\]
XMSS
Results

Efficient

Minimal security assumptions

„Small signatures"

Forward secure

Full smartcard implementation
New Variants of the Winternitz One Time Signature Scheme
Winternitz OTS (WOTS)  
[Mer89; EGM96]

\[
\text{SIG} = \left( i, \pi, m, 0, 0, 0, 0 \right)
\]

\[
| \text{SIG} | = | \text{SIG} | = m \times | \text{SIG} |
\]

1. \( \text{SIG} = f(\text{SIG}) \)

2. Trade-off between runtime and signature size
\[
| \text{SIG} | \sim m / \log w \times | \text{SIG} |
\]
Theorem 3.9 (informally):

$W$-OTS$^+$ is strongly unforgeable under chosen message attacks if $F$ is a $2^{nd}$-preimage resistant, undetectable one-way function family.
XMSS
[BDH11]

Lamport-Diffie / WOTS → WOTS+

Tree construction
[DOTV08]

Pseudorandom key generation

23.09.2013 | TU Darmstadt | Andreas Hülsing | 10
XMSS* in Practice
## XMSS Implementations

### C Implementation [BDH11]

C Implementation, using OpenSSL

<table>
<thead>
<tr>
<th></th>
<th>Sign (ms)</th>
<th>Verify (ms)</th>
<th>Signature (bit)</th>
<th>Public Key (bit)</th>
<th>Secret Key (byte)</th>
<th>Bit Security</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMSS-SHA-2</td>
<td>35.60</td>
<td>1.98</td>
<td><strong>16,672</strong></td>
<td>13,600</td>
<td>3,364</td>
<td>157</td>
<td>h = 20, w = 64,</td>
</tr>
<tr>
<td>XMSS-AES-NI</td>
<td><strong>0.52</strong></td>
<td><strong>0.07</strong></td>
<td>19,616</td>
<td>7,328</td>
<td>1,684</td>
<td>84</td>
<td>h = 20, w = 4</td>
</tr>
<tr>
<td>XMSS-AES</td>
<td>1.06</td>
<td>0.11</td>
<td>19,616</td>
<td>7,328</td>
<td>1,684</td>
<td>84</td>
<td>h = 20, w = 4</td>
</tr>
<tr>
<td>RSA 2048</td>
<td><strong>3.08</strong></td>
<td><strong>0.09</strong></td>
<td>≤ 2,048</td>
<td>≤ 4,096</td>
<td>≤ 512</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>

Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz with Intel AES-NI
XMSS Implementations
Smartcard Implementation  [HBB12]

<table>
<thead>
<tr>
<th></th>
<th>Sign (ms)</th>
<th>Verify (ms)</th>
<th>Keygen (ms)</th>
<th>Signature (byte)</th>
<th>Public Key (byte)</th>
<th>Secret Key (byte)</th>
<th>Bit Sec.</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMSS</td>
<td>134</td>
<td>23</td>
<td>925,400</td>
<td>2,388</td>
<td>800</td>
<td>2,448</td>
<td>92</td>
<td>H = 16, w = 4</td>
</tr>
<tr>
<td>XMSS+</td>
<td>106</td>
<td>25</td>
<td>5,600</td>
<td>3,476</td>
<td>544</td>
<td>3,760</td>
<td>94</td>
<td>H = 16, w = 4</td>
</tr>
<tr>
<td>RSA 2048</td>
<td>190</td>
<td>7</td>
<td>11,000</td>
<td>≤ 256</td>
<td>≤ 512</td>
<td>≤ 512</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>

Infineon SLE78 16Bit-CPU@33MHz, 8KB RAM, TRNG, sym. & asym. co-processor
NVM: Card 16.5 million write cycles/ sector,
XMSS+ < 5 million write cycles (h=20)
Conclusion
Conclusion

Fast
Conservative Security
Compact
Forward secure
Future Work

Main Drawback: State

Easy Migration?
- Interfaces
- Key Management
Thank you! Questions?