Improving the Performance of the Picnic Signature Scheme

Daniel Kales   Greg Zaverucha
CHES 2020, September, 2020
Outline

- Background
- Exploring Parameter Choices
- Protocol Optimizations
- Picnic3
Background
Post-Quantum Cryptography

- Shor’s Algorithm
  - Integer Factorization: RSA broken
  - Discrete Logarithm: (EC-)DSA, (EC-)DH, … broken
- Quantum computers
  - Theoretically viable, engineering effort to scale sizes
  - NIST has started a “Post Quantum Standardization Project” which has recently entered the third round.
    - key encapsulation, public key encryption, digital signatures
Background - Picnic: From MPC to PQ Signatures

Multi-Party Computation

- Allows \( n \) players to compute a function \( f \) on secret inputs \( x \) to arrive at a common output \( y \)
- Individual players learn nothing about the inputs of other players
- Individual players learn nothing any intermediate values in \( f \)
- Even \( n - 1 \) colluding players do not learn anything
Background - Picnic: From MPC to PQ Signatures

Zero-Knowledge Proofs from MPC:

- MPC-in-the-Head (Ishai et al., 2007)
  - Semi-Honest MPC $\rightarrow$ ZK-Proofs (of Knowledge)

- General Idea:
  - MPC protocol with $n$ players and $(n-1)$-privacy
  - Prover simulates execution of function in MPC
  - Prover commits to states of all $n$ parties
  - Verifier asks Prover to open all but 1 party (privacy)
  - Verifier gains assurance that overall execution was correct
  - Repeat until assurance is high enough (soundness)
Background - Picnic: From MPC to PQ Signatures

Non-interactive ZK-Proofs from ZK-Proofs:

- Fiat-Shamir transformation
  - Decades old standard technique
  - Prover calculates challenge himself
  - Set challenge \( c \leftarrow \mathcal{H}(\text{com}) \)
  - Proof in the ROM
    - Recent results for QROM

\[
\begin{array}{c|c}
\text{Prover} & \text{Verifier} \\
\hline
\text{com} \leftarrow \mathcal{P}_0(x) & \text{com} \\
ch & \text{ch} \leftarrow \text{ChS}(1^k) \\
\text{resp} \leftarrow \mathcal{P}_1(x, \text{com}, \text{ch}) & \text{resp} \\
b \leftarrow \mathcal{V}(y, \text{com}, \text{ch}, \text{resp})
\end{array}
\]
Non-interactive ZK-Proofs from ZK-Proofs:

- Fiat-Shamir transformation
- Decades old standard technique
- Prover calculates challenge himself
- Set challenge $c \leftarrow \mathcal{H}(\text{com})$
- Proof in the ROM
  - Recent results for QROM

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{com} \leftarrow \mathcal{P}_0(x)$</td>
<td>$\text{com, resp}$</td>
</tr>
<tr>
<td>$\text{ch} \leftarrow \mathcal{H}(\text{com})$</td>
<td>$\text{ch} \leftarrow \mathcal{H}(\text{com})$</td>
</tr>
<tr>
<td>$\text{resp} \leftarrow \mathcal{P}_1(x, \text{com}, \text{ch})$</td>
<td>$b \leftarrow \mathcal{V}(y, \text{com}, \text{ch}, \text{resp})$</td>
</tr>
</tbody>
</table>
Background - Picnic: From MPC to PQ Signatures

Signature Schemes from NIZK-Proofs:

- Identification scheme
  - Pick block cipher $E$
  - Publish $pk \leftarrow (C, P)$, where $C = E_{sk}(P)$
  - To identify, execute ZKPoK of $sk$ for $pk$

- Signature scheme
  - Non-Interactive via Fiat-Shamir
  - Include $m$ in challenge generation of NIZKPoK
    - $c \leftarrow \mathcal{H}(m||com)$
Background - Picnic Instances & Performance

**Picnic:**
- OWF $\leftarrow$ LowMC
- MPC $\leftarrow$ (2,3)-circuit decomposition
- $\mathcal{H} \leftarrow$ SHAKE

**Picnic2:**
- OWF $\leftarrow$ LowMC
- MPC $\leftarrow$ KKW
  - MPC protocol with preprocessing phase
- $\mathcal{H} \leftarrow$ SHAKE

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>$M$</th>
<th>$\tau$</th>
<th>$N$</th>
<th>Sign</th>
<th>Verify</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picnic-L1</td>
<td>219</td>
<td>219</td>
<td>3</td>
<td>1.37</td>
<td>1.10</td>
<td>32 862</td>
</tr>
<tr>
<td>Picnic2-L1</td>
<td>343</td>
<td>27</td>
<td>64</td>
<td>40.95</td>
<td>18.20</td>
<td>12 341</td>
</tr>
</tbody>
</table>

Times in ms, Size in Bytes.
Exploring Parameter Choices
Parameters in the KKW Proof System

The KKW proof system has a few important parameters:

- Number of MPC parties $N$
- Number of parallel repetitions $M$
- Number of opened online (preprocessing) phases $\tau (M - \tau)$

Exact formulas for security and size, but speed is harder to predict.

- We can count # of hashes etc. to get numbers to compare
- Real-world performance of optimized implementation most interesting
Size-Speed Tradeoffs

Changing these parameters allows for different tradeoffs in terms of signing/verification speed and signature size.

<table>
<thead>
<tr>
<th>Sec. level</th>
<th>N</th>
<th>M</th>
<th>$\tau$</th>
<th>Sign</th>
<th>Verify</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>64</td>
<td>343</td>
<td>27</td>
<td>41.16</td>
<td>18.21</td>
<td>12,347</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>258</td>
<td>32</td>
<td>18.69</td>
<td>8.37</td>
<td>13,162</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>252</td>
<td>36</td>
<td><strong>10.42</strong></td>
<td><strong>5.00</strong></td>
<td><strong>13,831</strong></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>200</td>
<td>48</td>
<td>5.24</td>
<td>2.95</td>
<td>15,995</td>
</tr>
</tbody>
</table>

Small increase in signature size gives large increase in performance.
LowMC Parameters

LowMC is a family of block ciphers, parametrizable on:

- Block size $n$
- Key size $k$
- Number of Sboxes per round $s$
- Allowed data complexity for attacker $d$

Original exploration of LowMC parameters naturally focused on $n = k = \{128, 192, 256\}$. We expanded search to include instances with full Sbox layer.
LowMC with a full Sbox layer

- Recent observation:
  - Instances with **full Sbox layer** are more efficient
    - less rounds needed for security in Picnic scenario
  - Problem: 3-bit Sbox, does not fit nicely in 128 & 256

- Move to new state sizes:
  - L1: 129-bit
  - L3: 192-bit
  - L5: 255-bit
Protocol Optimizations
The KKW MPC Protocol

Invariant for each wire \( a \) in the circuit:

\[
\hat{z}_a = z_a + \sum_i \lambda_a^i
\]

masked value \( \hat{z}_a \)
real value \( z_a \)
mask of party \( i \)

Linearity means XOR gates are "free", AND gates need multiplication triples:

\[
\sum_i \lambda_{ab}^i = \left( \sum_i \lambda_a^i \right) \cdot \left( \sum_i \lambda_b^i \right)
\]

In contrast to other MPC protocols, these multiplication triples are circuit-dependant (specific to the two input wires \( a \) and \( b \)).
Optimization 1: Improved Preprocessing

Generation of multiplication triples during preprocessing:

- For each AND gate with input wires $a$, $b$ and output wire $c$
  - Masks $\lambda^i_a$, $\lambda^i_b$, $\lambda^i_{ab}$ are read from random tapes
    - Not guaranteed to be valid multiplication triples
  - Calculate the error $e = \sum_i \lambda^i_{ab} - (\sum_i \lambda^i_a) \cdot (\sum_i \lambda^i_b)$
  - Correct the last party’s random tape to fix error

Observe: error only depends on sum of masks, not individual values

- Apply any linear gates only to sum instead of individual masks
- For LowMC’s heavy linear layer this results in 1.5x faster signing
Optimization 2: Improved Online Phase

Previous optimization allows for faster preprocessing

- In online phase, parties use output masks $\lambda_c$ to calculate input masks of next AND gate...
Optimization 2: Improved Online Phase

Previous optimization allows for faster preprocessing

- In online phase, parties use output masks $\lambda_c$ to calculate input masks of next AND gate...

- Change to preprocessing:
  - Choose random $\lambda_a$ instead, and calculate backwards

- This enables improved online phase:
  - Parties can read input masks $\lambda_a, \lambda_b$ directly from random tape, linear operations are “baked in”
Optimization 3: Improved Output Phase

At the end of MPC protocol, parties broadcast their share of output masks

- Check if unmasked output is equal to public key

Observation:

- Only linear operations between last Sbox layer and ciphertext
- Combine communication for Sbox AND gates and output broadcast
  - New communication can be computed from old, so no leak
- Saves $n$ bits of communication per MPC instance
Picnic3

Putting it all together
New Parameter Sets

We propose new Picnic instances (Picnic3) combining the following improvements:

- Using $N = 16$ parties in the KKW MPC protocol
- Using a LowMC instance with full Sbox layer
- Using the presented optimizations to the KKW MPC protocol

Existing security analysis for Picnic2 is still applicable due to similar structure!
Performance Comparison

Comparing Performance to existing Picnic instances

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>(M)</th>
<th>(\tau)</th>
<th>(N)</th>
<th>(n)</th>
<th>(s)</th>
<th>(r)</th>
<th>Sign (ms)</th>
<th>Verify (ms)</th>
<th>Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picnic-L1</td>
<td>219</td>
<td>219</td>
<td>3</td>
<td>128</td>
<td>10</td>
<td>20</td>
<td>1.37</td>
<td>1.10</td>
<td>32 862</td>
</tr>
<tr>
<td>Picnic2-L1</td>
<td>343</td>
<td>27</td>
<td>64</td>
<td>128</td>
<td>10</td>
<td>20</td>
<td>40.95</td>
<td>18.20</td>
<td>12 341</td>
</tr>
<tr>
<td>Picnic3-L1</td>
<td>252</td>
<td>36</td>
<td>16</td>
<td>129</td>
<td>43</td>
<td>4</td>
<td>5.17</td>
<td>3.96</td>
<td>12 595</td>
</tr>
<tr>
<td>Picnic3-5-L1</td>
<td>252</td>
<td>36</td>
<td>16</td>
<td>129</td>
<td>43</td>
<td>5</td>
<td>5.59</td>
<td>4.63</td>
<td>13 703</td>
</tr>
</tbody>
</table>

Times in ms, Size in Bytes.
Performance Comparison (cont.)

SPHINCS\(^+\) is another submission in the NIST post quantum standardization project. Its security is also only based on the security of symmetric primitives.

Comparison of parameters at the L1 (128-bit) security level:

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>Sign</th>
<th>Verify</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picnic3-L1</td>
<td>5.17</td>
<td>3.96</td>
<td>12595</td>
</tr>
<tr>
<td>SPHINCS(^+) fast L1 (f128sha256simple)</td>
<td>14.69</td>
<td>1.74</td>
<td>16976</td>
</tr>
<tr>
<td>SPHINCS(^+) small L1 (s128sha256simple)</td>
<td>238.33</td>
<td>0.72</td>
<td>8080</td>
</tr>
</tbody>
</table>

Times in ms, Size in Bytes.
Conclusion

- Revisited parameter choices for Picnic
- Optimizations for MPC protocol
- Proposed Picnic3 instances
  - 8-14x faster sign, ≈ 5x faster verify compared to Picnic2
  - Optimized Implementation (https://github.com/IAIK/Picnic)
  - Round 3 submission to NIST (alternate candidate)
- In paper: Optimized instances for interactive identification, benchmarks for alternative hash functions
Future Work

- Cryptanalysis of LowMC
  - Specific attacker scenario (only 1 PT/CT pair)
  - Participate in ongoing LowMC cryptanalysis challenge
    - Total prize fund of USD 100 000
    - https://lowmcchallenge.github.io/

- Optimized implementation for embedded platform
  - This work focused on x86 targets
  - Resource constraints have to be considered