The Picnic Digital Signature Algorithm

NIST Third PQC Standardization Conference
June 2021

Melissa Chase, David Derler, Steven Goldfeder, Daniel Kales, Jonathan Katz,
Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger,
Daniel Slamanig, Xiao Wang and Greg Zaverucha
Picnic Overview

Security depends only on problems related to symmetric key primitives
- Secure hash function (ROM/QROM analysis implies all the usual properties: CR, PR, etc.)
- Secure block cipher (key recovery given a single plaintext/ciphertext pair)
- Unique design, conservative assumptions

The core of Picnic is an efficient zero knowledge proof for binary circuits
- Create a signature scheme using a non-interactive proof
- Use the Fiat-Shamir transform

Performance characteristics
- Keys are small, signatures are relatively large, possible to tradeoff speed/size
Picnic Signatures

Key Generation:
- Generate a random plaintext block $p$
- Generate a random secret key $sk$
- Compute $C = \text{LowMC}(sk, p)$
- Picnic public key is $pk = (C, p)$, signing key is $sk$

Sign($sk, pk, m$):
- Prove knowledge of $sk$ such that $C = \text{LowMC}(sk, p)$
- Message $m$ and public key $pk$ are bound to the proof when computing the challenge
- Picnic signature is the proof
Picnic Signatures

Key Generation:
Generate a random plaintext block $p$
Generate a random secret key $sk$
Compute $C = \text{LowMC}(sk, p)$ \(\text{Must be hard to recover } sk\)
Picnic public key is $pk = (C, p)$, secret key is $sk$

Sign($sk$, $pk$, $m$):
Prove knowledge of $sk$ such that $C = \text{LowMC}(sk, p)$
Message $m$ and public key $pk$ are bound to the proof when computing the challenge
Picnic signature is the proof \(\text{Must be zero-knowledge}\)
Changes for Round 3
Parameter Sets

Round 2
- Picnic1-L1-FS
- Picnic1-L3-FS
- Picnic1-L5-FS
- Picnic1-L1-UR
- Picnic1-L3-UR
- Picnic1-L5-UR
- Picnic2-L1-FS
- Picnic2-L3-FS
- Picnic2-L5-FS

Round 3
- Picnic1-L1-FS
- Picnic1-L3-FS
- Picnic1-L5-FS
- Picnic1-L1-Full
- Picnic1-L3-Ful
- Picnic1-L5-Ful
- Picnic3-L1
- Picnic3-L3
- Picnic3-L5

New

Replaced

Picnic1
Faster
Larger signatures

Picnic3
Slower
Smaller signatures
Picnic1-Full

Same as Picnic1-FS but with alternative LowMC parameters

  The number of 3-bit S-boxes per round is flexible in LowMC
  Picnic previously used only LowMC instances with a partial S-box layer
  Partial: needs more rounds (e.g., 20) vs Full: needs far fewer (e.g., 4)
  LowMC Cryptanalysis Challenge [LCC] contains instances of both type
  Partial has (arguably) more margin due to Dinur’s cryptanalysis [Din21] of Full

Fewer rounds means faster sign and verify times and fewer round constants (lower code size)
Signatures are slightly shorter (fewer total AND gates)
Picnic3

Tweaks to Picnic2 aimed at making Sign and Verify faster

Picnic’s ZK proof is done using the MPC-in-the-head technique

- Prover simulates an N-party MPC protocol, commits to the execution, then reveals parts of it to the verifier

  Reduce the number of MPC parties \( N \) from 64 to 16. \( \sim 4x \) less computation

Change LowMC parameters to use a Full S-box layer

Tweaks to the MPC protocol

Security analysis still holds

Considering adding a more conservative option with a Partial S-box layer LowMC instance, due to [Din21] (as we have with Picnic1)
Performance

Security level L1, Intel Xeon W-1233 @ 3.6GHz

<table>
<thead>
<tr>
<th>Parameter set</th>
<th>Sign (ms)</th>
<th>Verify (ms)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picnic1-FS</td>
<td>1.37</td>
<td>1.10</td>
<td>32,862</td>
</tr>
<tr>
<td>Picnic1-Full</td>
<td>1.0</td>
<td>0.8</td>
<td>30,821</td>
</tr>
<tr>
<td>Picnic2</td>
<td>40.95</td>
<td>18.20</td>
<td>12,341</td>
</tr>
<tr>
<td>Picnic3</td>
<td>5.17</td>
<td>3.96</td>
<td>12,595</td>
</tr>
<tr>
<td>Picnic2, N=16</td>
<td>10.42</td>
<td>5.00</td>
<td>13,831</td>
</tr>
</tbody>
</table>

See [KZ20] for additional details and benchmarks for L3 and L5
New Picnic implementations:
Resource-constrained and side-channel protected
Resource-Constrained Implementations

Our x64 implementations do not optimize for memory use or code size

Recent effort to reduce RAM usage, target is the ARM Cortex-M4
  Must carefully compute parts of the signature as-needed
  Recompute some values, rather than store
  E.g., in Picnic3-L1 there are 250 MPC instances, and the prover will need 36 of them to respond to the challenge; we recompute these 36 rather than storing all 250

Simple tweaks can further reduce RAM usage and improve times
  (e.g., order that values are hashed/derived)
Masked Picnic

[GSE20] and [SBWE20] demonstrated some probing side-channel attacks on Picnic1.

We recently applied these attacks to Picnic3, and found a new one [ABE+21]. Analyzed, implemented and benchmarked a masked version of Picnic signing.

Takeaway: Under mild assumptions, masking overheads can be as low as 1.86x (for first-order protection). Watch the talk on Tuesday!
## ARM Cortex-M4 Benchmarks

Using [pqm4] on the STM32F4 device, security level L1

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Implementation</th>
<th>Sign</th>
<th>Verify</th>
<th>Sig. Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picnic3-L1</td>
<td>opt</td>
<td>304M cycles</td>
<td>203M cycles</td>
<td>12.6KB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32KB RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>opt-mem</td>
<td>310M cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>24KB RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>masked</td>
<td>546M cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>32KB RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picnic1-L1-FS</td>
<td>opt</td>
<td>289M cycles</td>
<td>126M cycles</td>
<td>32.9KB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4KB RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lowmem-mod</td>
<td>152M cycles</td>
<td>55M cycles</td>
<td>30.8KB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.2KB RAM</td>
<td>3.5KB RAM</td>
<td></td>
</tr>
</tbody>
</table>

Ongoing effort: [https://github.com/dkales/picnic_m4](https://github.com/dkales/picnic_m4)
Research and Security Analysis
Security Analysis

[GHHM20]: found a mistake in the QROM proof of Picnic3 and helped correct it.

Also generalizes results on the fault-resistance of Picnic (and related FS schemes) from [AOTZ20] from the ROM to the QROM

Tight QROM security of Picnic ([DFMS21], [Cha19])

The recent results in [DFMS21] improve the existing, highly non-tight results for Picnic3. Alternate approach to tighter Picnic1 QROM analysis in [Cha19, 2021 revision].
Security Analysis

[CDF+20]: Picnic has stronger security properties than standard strong unforgeability. These properties are sometimes implicitly relied on by protocols. (Watch the talk on Wednesday!)

Efficient Implementation

[WBS20]: multiple ways to reduce Picnic1 RAM usage, and how to stream parts of the signature from a resource constrained device to a host device. Some ideas implemented in our M4 implementations of Picnic1.
AES-Based Signatures

FAQ: Can Picnic use AES instead of LowMC?
   Yes, but larger signatures

[BDK+21]: Banquet, a new AES-based MPCitH signature scheme

[DKR+21] and [DOT21]: Slightly faster and shorter signatures with AES
   Based on Banquet. Still slower or larger than Picnic3 (depending on parameters), but the gap is
   much smaller than previously
   E.g., 13.2KB signatures and ~20ms sign/verify

[DKR+21]: ciphers with large S-boxes (based on field inversion) improve performance much further
   E.g., 5.5 KB signatures and same sign/verify times as Picnic3, or
   8.5 KB signatures with sign/verify times <1 ms
Formal Verification

There is some recent work towards a formally verifying MPC-in-the-head (MPCitH) ZK proof protocols.

[SOS21]: formal verification of MPCitH protocols, EasyCrypt **machine checked security proof** of ZKBoo (a simpler version of the proof system used in Picnic1).

[ABEG+21]: Machine checked security proofs **and implementations** for MPCitH (more general than Picnic but simpler MPC protocol)
Summary

Compared to Round 2:
  Picnic3 is 4.5x to 13.9x faster than Picnic2
  New Picnic1 parameter sets are ~40% faster and ~10% shorter
  Implementations for the M4

Ongoing strong interest from the research community is amazing!

More information:  microsoft.github.io/Picnic/
References are at the end of the presentation.
References


References


References


References


