# The Picnic Digital Signature Algorithm

**NIST First PQC Standardization Conference**
**April 2018**

Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig and **Greg Zaverucha**

Microsoft · AIT Austrian Institute of Technology · Princeton University · DTU · Aarhus University · TU Graz

# Overview

## Security depends only on problems related to symmetric key primitives

- Unique design, conservative assumptions
- 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)

## 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 or Unruh transform

## Performance characteristics

- Keys are small, signing and verification times are fast
- Signatures are relatively large

# Picnic Signatures

## Key Generation:

Generate a random plaintext block $p$

Generate a random secret key $sk$

Compute $C$ = LowMC($sk$, $p$)

Picnic public key is $pk$ = ($C$, $p$), secret key is $sk$

## Sign($sk$, $pk$, $m$):

Prove knowledge of sk such that $C$ = 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 $C$ = LowMC($sk$, $p$) ← 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 $C$ = LowMC($sk$, $p$)

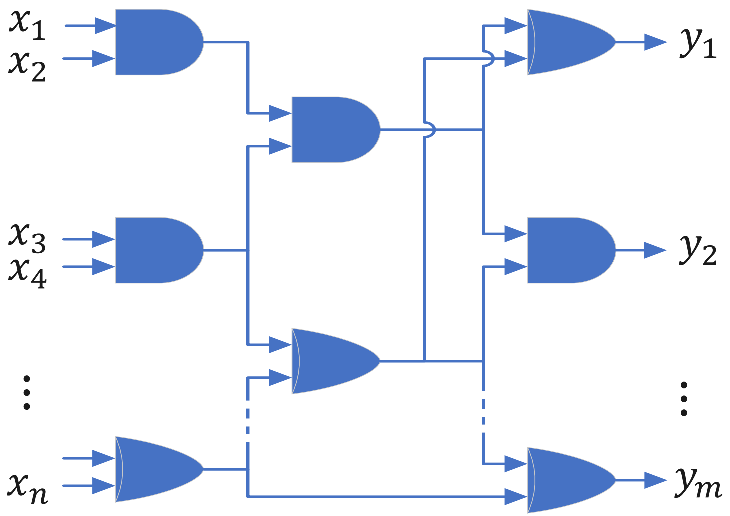
Message $m$ and public key $pk$ are bound to the proof when computing the challenge

Picnic signature is the proof ← Must be zero-knowledge

# Proof System

ZKBoo: zero knowledge proofs for statements about circuits.

Public circuit with AND and XOR gates




Prover knows $x_1 \dots x_n$ such that the circuit evaluates to $y_1 \dots y_m$

Signer

sk

Hard to invert

pk

I. Giacomelli, J. Madsen and C. Orlandi, [ZKBoo: Faster Zero-Knowledge for Boolean Circuits.] USENIX Security 2016.

# Proof system

- Picnic uses ZKB++, a variant optimized for short proofs
- Built with a hash function and KDF (both SHAKE)
- Non-interactive proofs
  - Fiat-Shamir transform gives ROM security
  - Unruh's transform gives QROM security, 1.6x larger signatures
- Proof size depends on
  - The security level; we use parallel repetition to achieve soundness
  - The number of AND gates in the circuit

# The LowMC Block Cipher

LowMC is a block cipher introduced by Albrecht et al. at Eurocrypt

Designed for nontraditional block cipher applications, like MPC and FHE

Compared to more common primitives:

- About 7x fewer than AES, and 30x fewer than SHA-256
- Newer design, but we only need key recovery to be difficult

Highly parameterizable, some of our choices

- Tradeoff between AND and XOR gates: balance signature size and signing time
- Only one plaintext-ciphertext pair is revealed per key
- Keysize = blocksize (128, 192 and 256 bits)

M. Albrecht, C. Rechberger, T. Schneider, T. Tiessen and M. Zohner. Ciphers for MPC and FHE. EUROCRYPT 2015.

# Parameter Sets

- Parameter sets for the three AES security levels, L1, L3, L5
  - Each with different LowMC, SHAKE and # of parallel repetitions
  - Fiat-Shamir (FS) and Unruh (UR) variants

# Performance: Key and Signature Size

## Signature and key sizes (bytes)

| Parameter Set | Public Key | Private Key | Signature |
|---|---|---|---|
| Picnic-L1-FS | 32 | 16 | 34,000 |
| Picnic-L1-UR | 32 | 16 | 53,929 |
| Picnic-L3-FS | 48 | 24 | 76,740 |
| Picnic-L3-UR | 48 | 24 | 121,813 |
| Picnic-L5-FS | 64 | 32 | 132,824 |
| Picnic-L5-UR | 64 | 32 | 209,474 |

# Performance: Timings

Optimized Implementation (ms), Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz

| Parameter Set | Keygen | Sign | Verify |
|---|---|---|---|
| Picnic-L1-FS | 0.00 | 1.95 | 1.36 |
| Picnic-L1-UR | 0.00 | 2.64 | 1.91 |
| Picnic-L3-FS | 0.01 | 6.61 | 4.63 |
| Picnic-L3-UR | 0.01 | 8.84 | 6.29 |
| Picnic-L5-FS | 0.02 | 14.71 | 10.64 |
| Picnic-L5-UR | 0.02 | 18.67 | 13.60 |

# TLS Experiments

Are there challenges to using Picnic in TLS?

- We added Picnic to the *Open Quantum Safe library* (OQS), the OQS fork of OpenSSL and Apache web server

Experiment:

- Use Picnic-signed X.509 certificates certifying Picnic keys
- L1-FS parameter set
- Use Picnic certificates to authenticate TLS 1.2 connections
- Fetch HTML files

Performance, client-side latency:

- For 45B files: increase of 1.4x to 1.7x
- For 100KB files: increase of 1.1x to 1.3x

Challenges:

- TLS 1.2 has limit of $2^{16} - 1$ bytes/signature: too short for our higher security parameter sets

| Ciphersuite KEX, SIG | Page Size | Mean fetch time (seconds) Slow network | Mean fetch time (seconds) Fast network |
|---|---|---|---|
| ECDHE-ECDSA | 45B | 0.470 | 0.299 |
| *(not PQ)* | 100K | 1.226 | 0.452 |
| LWEFRODO-RSA | 45B | 0.578 | 0.366 |
| | 100K | 1.335 | 0.518 |
| LWEFRODO-PICNIC | 45B | 0.984 | 0.513 |
| | 100K | 1.733 | 0.594 |
| SIDH-RSA | 45B | 0.655 | 0.385 |
| | 100K | 1.370 | 0.541 |
| SIDH-PICNIC | 45B | 1.084 | 0.523 |
| | 100K | 1.738 | 0.600 |

# HSM Experiments

- What if a CA wants to protect Picnic signing keys in a hardware security module?
- We experimented with the Utimaco SecurityServer Se50 LAN V4
- Experiment:
  - Implement Picnic key generation and signing in an HSM. Ported our  reference implementation.
  - Generate self-signed root cert using new Picnic key pair
  - Receive certificate signing request for RSA key pair and issue X.509 certificate
- Performance was acceptable and porting reference implementation was straightforward

# Highlights of Picnic

Unique design

Conservative assumptions

Efficient and tested in real world protocols

## More information: microsoft.github.io/Picnic/

Spec and design documents, research paper from CCS 2017

Talks and related work (RWC 2018 talk)

Link to OQS/OpenSSL, code from our HSM demo

Implementations