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

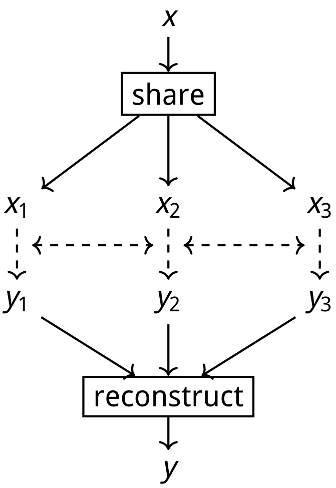
Post-Quantum Cryptography

- 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 $c \leftarrow \mathcal{H}(\mathsf{com})$
  - Proof in the ROM
    - Recent results for QROM

Prover		**Verifier**
$\mathsf{com} \leftarrow \mathcal{P}_0(x)$		
	$\xrightarrow{\mathsf{com}}$	
		$\mathsf{ch} \xleftarrow{\$} \mathsf{ChS}(1^k)$
	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow \mathcal{P}_1(x, \mathsf{com}, \mathsf{ch})$		
	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathcal{V}(y, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

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 $c \leftarrow \mathcal{H}(\mathrm{com})$
  - Proof in the ROM
    - Recent results for QROM

Prover		**Verifier**
$\mathrm{com} \leftarrow \mathcal{P}_0(x)$		
$\mathrm{ch} \leftarrow \mathcal{H}(\mathrm{com})$		
$\mathrm{resp} \leftarrow \mathcal{P}_1(x, \mathrm{com}, \mathrm{ch})$	$\xrightarrow{\mathrm{com}, \mathrm{resp}}$	
		$\mathrm{ch} \leftarrow \mathcal{H}(\mathrm{com})$
		$b \leftarrow \mathcal{V}(y, \mathrm{com}, \mathrm{ch}, \mathrm{resp})$

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
    - $c \leftarrow \mathcal{H}(m \| \mathrm{com})$

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

Parameter set	M	τ	N	Sign	Verify	Size
`Picnic-L1`	219	219	3	1.37	1.10	32 862
`Picnic2-L1`	343	27	64	40.95	18.20	12 341

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 τ $(M - \tau)$

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.

Sec. level	N	M	τ	Sign	Verify	Size
L1	64	343	27	41.16	18.21	12,347
	32	258	32	18.69	8.37	13,162
	16	**252**	**36**	**10.42**	**5.00**	**13,831**
	8	200	48	5.24	2.95	15,995

Small increase in signature size gives large increase in performance.

LowMC Parameters

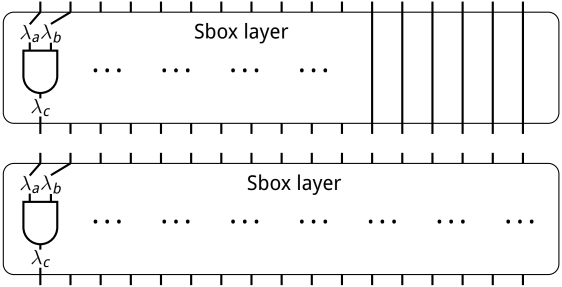
LowMC is a family of block ciphers, parametrizable on:

- 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:

$$\underbrace{\hat{z}_a}_{\text{masked value}} = \underbrace{z_a}_{\text{real value}} + \sum_i \underbrace{\lambda_a^i}_{\text{mask of party } i}$$

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

$$\sum_i \lambda_{ab}^i \stackrel{!}{=} \left(\sum_i \lambda_a^i\right) \cdot \left(\sum_i \lambda_b^i\right)$$

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 λ_a^i, λ_b^i, λ_{ab}^i are read from random tapes
    - Not guaranteed to be valid multiplication triples
  - Calculate the error $e = \sum_i \lambda_{ab}^i - (\sum_i \lambda_a^i) \cdot (\sum_i \lambda_b^i)$
  - Correct the last party's random tape to fix error

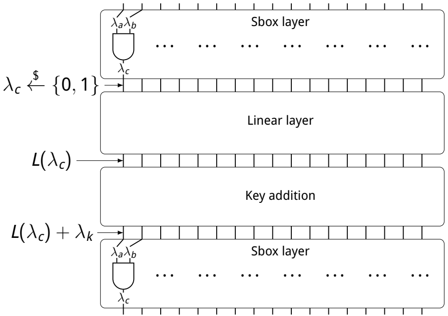
Observe: error only depends on sum of masks, not individual values

- 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

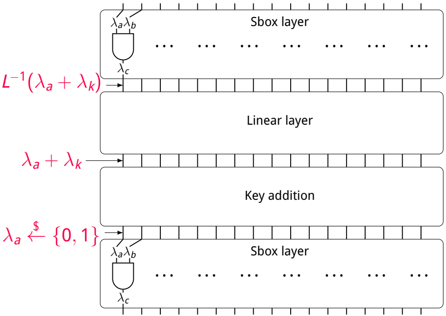
- In online phase, parties use output masks λ_c to calculate input masks of next AND gate...



Optimization 2: Improved Online Phase

Previous optimization allows for faster preprocessing

- In online phase, parties use output masks λ_c to calculate input masks of next AND gate...
- Change to preprocessing:
  - Choose random λ_a instead, and calculate backwards
- This enables improved online phase:
  - Parties can read input masks λ_a, λ_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

Parameter set	M	τ	N	n	s	r	Sign	Verify	Size
`Picnic-L1`	219	219	3	128	10	20	1.37	1.10	32 862
`Picnic2-L1`	343	27	64	128	10	20	40.95	18.20	12 341
`Picnic3-L1`	**252**	**36**	**16**	**129**	**43**	**4**	**5.17**	**3.96**	**12 595**
`Picnic3-5-L1`	252	36	16	129	43	5	5.59	4.63	13 703

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:

Parameter set	Sign	Verify	Size
`Picnic3-L1`	5.17	3.96	12 595
`SPHINCS`$^+$ `fast L1 (f128sha256simple)`	14.69	1.74	16 976
`SPHINCS`$^+$ `small L1 (s128sha256simple)`	238.33	0.72	8 080

Times in ms, Size in Bytes.

Conclusion

- Revisited parameter choices for Picnic
- Optimizations for MPC protocol
- Proposed Picnic3 instances
  - 8-14x faster sign, $\approx$ 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