KU LEUVEN

The MPC-in-the-Head Framework and the Limbo Protocol

C. Delpech de Saint Guilhem
imec-COSIC, KU Leuven

ISC Winter School, March 2, 2023

0 Outline

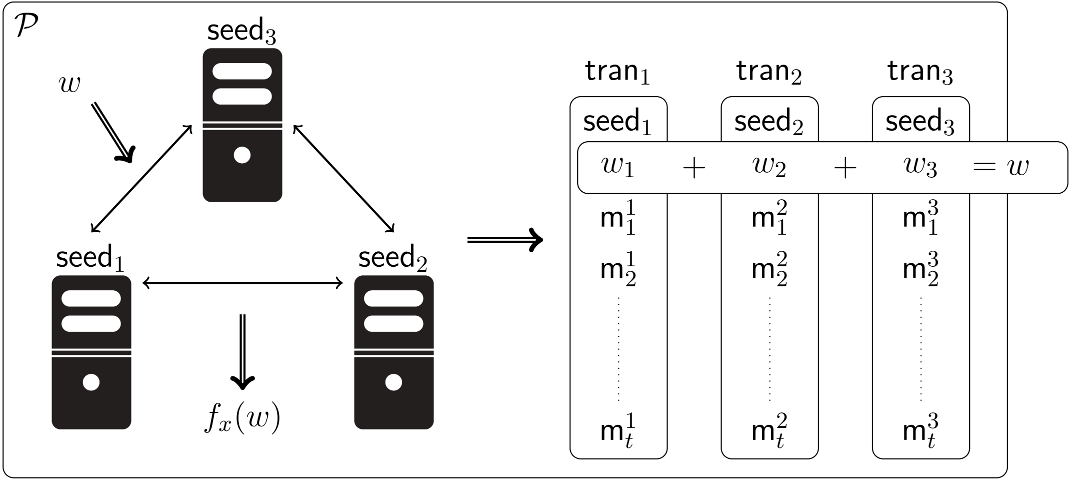
1 Outline

1 ZKPoK from MPC in the head

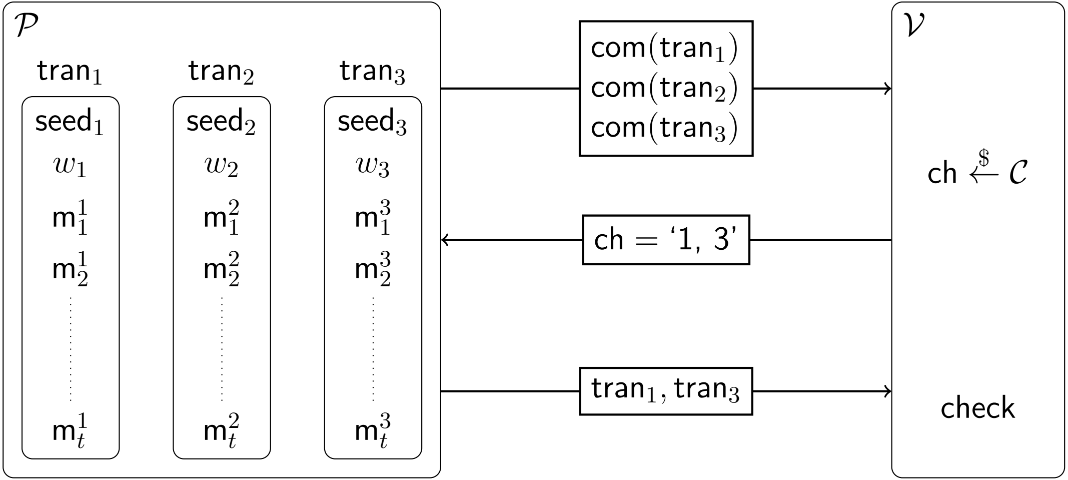
- Want efficient ZKPoK for arbitrary NP relation $\mathcal{R}$.
- Given
  1. x: public statement
  2. w: $\mathcal{P}$'s private witness,

  want to convince $\mathcal{V}$ that $\mathcal{R}(x, w) = 1$ without revealing w.
- [IKOS07]: multiparty computation (MPC) of $f_x(w) = \mathcal{R}(x, w)$.
  Simulated by $\mathcal{P}$ in the head and checked by $\mathcal{V}$.

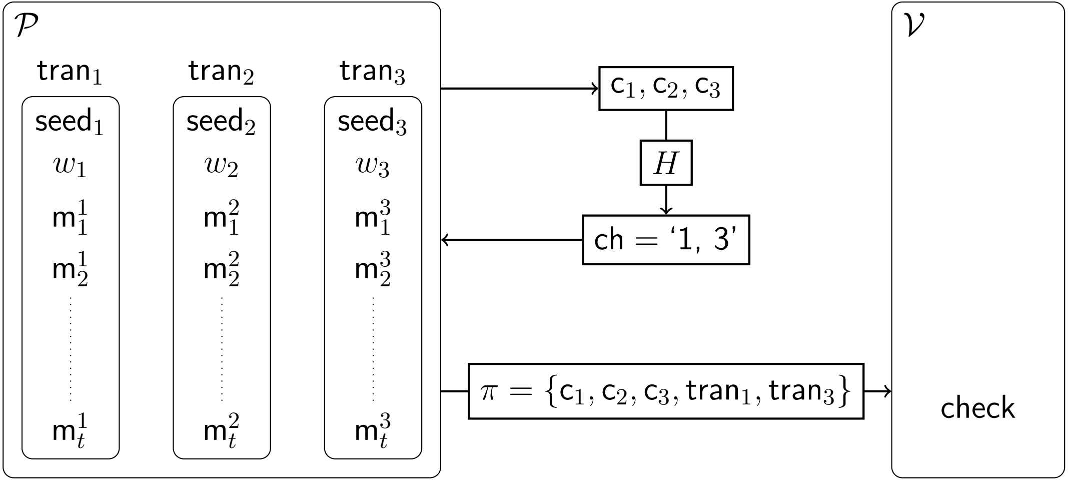
1 MPC in the head?



1 ZKPoK from MPC in the head



1 NIZKPoK from MPC in the head



1 Properties of the MPCitH Proof System

- *Correctness:* If MPC protocol for $f_x(w)$ is correct, then so is MPCitH.

- *Soundness:* If $f_x(w) \neq 1$ and opened parties show $f_x(w) = 1$ then $\mathcal{P}$ cheated during MPC protocol.
  - Assuming com is *binding*, $\mathcal{P}$ can't cheat for opened parties.
  - Assuming com is *binding*, $\mathcal{P}$ must cheat on hidden party *before* it sees ch.

  Soundness error is exactly $\frac{1}{N}$. In practice $N = 2, 4, 8, 16, 32, 64$.

- *Zero-knowledge:* First, com must be *hiding*.
  Second, $\mathcal{V}$ sees $N - 1$ transcripts, so MPC protocol must be $(N-1)$-*private*.

1 Optimizations in Practice

- Commitments $\mathsf{com}_1, \ldots, \mathsf{com}_N$ can be compressed with a Merkle tree.
  This means sending only 1 hash value in the first round, instead of N.

- Because $\frac{1}{N}$ is not cryptographically secure, the proof can be repeated in parallel with different seeds to increase soundness exponentially.

- Opening $N - 1$ transcripts can use a lot of data (and make a large proof). Using MPC protocol in the *broadcast model* means

  $$\text{"revealing } \mathsf{tran}_1, \mathsf{tran}_3\text{"} \approx \text{"sending } \mathsf{server}_2\text{'s output;"}$$

  everything else can be computed by $\mathcal{V}$ from the seeds.

1 MPCitH from MPC with Preprocessing

- Some MPC protocols use preprocessing / online paradigm.
  - Preprocessing generates *input-independent correlated randomness*.
  - Online phase uses it for *low communication* and *input-dependent* computation.
- This can be used for MPC in the head.
  - **aux**: *correlated randomness* without preprocessing communication included in π.
    So more correlated randomness = bigger proof π.

1. $\mathcal{P}$ simulates and commits to many preprocessing executions.
2. $\mathcal{V}$ challenges all-but-some of them to open and verify.
   Indep. of w, so $\mathcal{P}$ reveals both master seed and correlated rand **aux**.
3. With the rest, $\mathcal{P}$ simulates and commits to online executions.
4. $\mathcal{V}$ challenges and verifies as before; uses less data as online phase is cheap.

2 Outline

2 The Picnic Signature Scheme [CDG+17, ZCD+20]

Given block cipher $F_k(\boldsymbol{x}) : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$,

- Gen: $\boldsymbol{x} \xleftarrow{\$} \mathcal{X} = \{0,1\}^{\kappa}$, $k \xleftarrow{\$} \mathcal{K} = \{0,1\}^{\kappa}$, $\boldsymbol{y} \leftarrow F_k(\boldsymbol{x})$.

$$\mathsf{sk} = k \text{ and } \mathsf{pk} = (\boldsymbol{y}, \boldsymbol{x}).$$

  Security: function $G_{\boldsymbol{x}} : k \mapsto F_k(\boldsymbol{x})$ is OWF with respect to k.

- Sign: Given message m, compute $\sigma = \pi \leftarrow \mathsf{Prove}_m(f_{\boldsymbol{y},\boldsymbol{x}}(k))$.

$$f_{\boldsymbol{y},\boldsymbol{x}}(k) = \{F_k(\boldsymbol{x}) \stackrel{?}{=} \boldsymbol{y}\}$$

- Verify: verify the proof, including m in the challenge computation.

Choice of F_k: LowMC [ARS+15] as a binary circuit;

2 Picnic: MPC for binary circuits I [CDG+17, ZCD+20]

$f_x(w)$ as binary circuit C with wires and gates (XOR and AND).

- Wire α, real value is z_α, masked as $\hat{z}_\alpha = z_\alpha \oplus \lambda_\alpha$ for random $\lambda_\alpha \in \{0, 1\}$.
- $[\lambda_\alpha]$ is n-out-of-n XOR secret sharing

$$\lambda_\alpha = \bigoplus_{i=1}^{n} \lambda_\alpha^{(i)} \in \{0, 1\}.$$

- Each party P_i holds $\hat{z}_\alpha$ and a share $\lambda_\alpha^{(i)}$.
- Preprocessing prepares masks, online phase computes masked values $\hat{z}_\alpha$.

2 Picnic: MPC for binary circuits II

Preprocessing:

- For each *input* or *AND gate output* wire α, random mask $[\lambda_\alpha]$ from seed.
- For XOR gate $\gamma = \alpha \oplus \beta$, set $[\lambda_\gamma] = [\lambda_\alpha] \oplus [\lambda_\beta]$; local $=$ free.
- For AND gate $\gamma = \alpha \cdot \beta$, need $[\lambda_{\alpha,\beta}]$, where $\lambda_{\alpha,\beta} = \lambda_\alpha \cdot \lambda_\beta$; not free.
- P_i can set $\lambda^{(i)}_{\alpha,\beta}$ at random, for $i \in \{1, \ldots, N-1\}$, but P_N needs

$$\lambda^{(N)}_{\alpha,\beta} = \lambda_{\alpha,\beta} - \bigoplus_{i=1}^{N-1} \lambda^{(i)}_{\alpha,\beta}.$$

- This $\lambda^{(N)}_{\alpha,\beta}$ has to be computed by $\mathcal{P}$ and added to aux;
  1 bit/AND gate added to π.

KU LEUVEN

2 Picnic: MPC for binary circuits III

Online:

- Public reconstruction of $[\lambda_\alpha]$ is done by broadcast of each $\lambda_\alpha^{(i)}$;
- Parties begin with masked $\hat{z}_\alpha$ given by $\mathcal{P}$ and $[\lambda_\alpha]$ from seed.
- Computation proceeds by computing $\hat{z}_\gamma$ for each gate $(\alpha, \beta) \to \gamma$ in C.
- XOR gate: $\hat{z}_\gamma = \hat{z}_\alpha \oplus \hat{z}_\beta$; local = free.
- AND gate: locally compute

$$[s] = \hat{z}_\alpha[\lambda_\beta] \oplus \hat{z}_\beta[\lambda_\alpha] \oplus [\lambda_{\alpha,\beta}] \oplus [\lambda_\gamma],$$

reconstruct s (1 bit of communication per party),
and compute $\hat{z}_\gamma = s \oplus \hat{z}_\alpha \hat{z}_\beta$.

2 The AES Algorithm

AES is a 128-bit state block-cipher with key length of 128, 192 or 256 bits. The round function is composed of four operations on the state:

1. AddRoundKey
2. SubBytes – only non-linear component, aka S-box
3. ShiftRows
4. Mix Columns

SubBytes applies the function $s \mapsto \begin{cases} s^{-1} & \text{if } s \neq 0 \\ 0 & \text{o/w} \end{cases}$ over $\mathbb{F}_{2^8}$, followed by a public affine transformation.

KU LEUVEN

2 BBQ: MPC for arithmetic circuit I

$f_x(w)$ represented as arithmetic circuit C with values and gates ($+$ and $\times$). [dDOS19, BN20]

- Wire value $x \in \mathbb{F}$ is randomly shared as $\langle x \rangle = (x^{(1)}, \ldots, x^{(n)})$ such that

$$x = \sum_{i=1}^{n} x^{(i)}.$$

  $\langle x \rangle$ is n-out-of-n additive sharing of x.
- Each party P_i holds only a share $x^{(i)}$.
- Preprocessing prepares shared randomness.
  Online phase computes $+$ and $\times$ gates.

2 MPC for arithmetic circuit II

Operations with public constants are free, and so are additions between shared values (because of additive sharing).

Only multiplication $\langle z \rangle = \langle x \cdot y \rangle$ costs preprocessing and communication.

Given *triple* $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ such that $c = a \cdot b$, multiplication is:

1. Compute $\langle \alpha \rangle = \langle x - a \rangle$ and $\langle \beta \rangle = \langle y - b \rangle$.
2. Open α and β.
3. Locally compute $\langle z \rangle = \langle c \rangle - \alpha \cdot \langle b \rangle - \beta \cdot \langle a \rangle + \alpha \cdot \beta$.

Computing $\langle c \rangle$ adds $\log_2 |\mathbb{F}|$ bits to aux in π as P_N needs

$$c^{(N)} = a \cdot b - \sum_{i=1}^{N-1} c^{(i)}.$$

2 Computing inversion in $\mathbb{F}$ for AES

Given S-box input $\langle s \rangle$ and random $\langle r \rangle$:

1. Compute $\langle s \cdot r \rangle$.
2. Open $s \cdot r$.
3. Locally compute $\langle s^{-1} \rangle = (s \cdot r)^{-1} \cdot \langle r \rangle$.

KU LEUVEN

3 Outline

1. NIZKPoK from MPC in the Head with Preprocessing
2. MPCitH from Circuit Computation: Picnic and BBQ
3. MPCitH from Circuit Verification: Banquet and Limbo
   - Witness Extension and Verification
   - Banquet: Verification of Multiplications (and Inverses)
   - Limbo: Improved Multiplication Verification

KU LEUVEN

3 Witness extension and verification

Idea from sacrificing techniques in MPC

- Prover "injects" the results of multiplications—no need to compute.
  - The witness is *extended* with the outputs of non-linear gates.
- MPC parties execute a *verification* protocol—batching possibilities.
  - e.g. Sacrifice one "suspicious" operation to verify another.

ZKPoK protocol sketch [BN20]

MPC parties receive "suspicious" multiplication results and verify them by sacrificing "suspicious" random triples ⇒ no cut & choose.

$$0 \stackrel{?}{=} \langle v \rangle = \epsilon \langle z \rangle - \langle c \rangle + \alpha \langle b \rangle + \beta \langle a \rangle - \alpha \cdot \beta$$

3 Increased Number of Rounds

- In order to *soundly* sacrifice triples, the parties need a *random challenge* ϵ.
- This challenge comes from $\mathcal{V}$, *after* suspicious data is committed.
- *After*, $\mathcal{P}$ still has to commit to MPC execution of sacrificing *verification*.
  After receving ϵ, the MPC parties continue to check $v \stackrel{?}{=} 0$.

This yields a protocol with 5 or more rounds:

1. $\mathcal{P}$ commits to (views of) suspicious data;
2. $\mathcal{V}$ sends sacrificing (i.e. verification) challenge;
3. $\mathcal{P}$ commits to (views of) data verification protocol;
4. $\mathcal{V}$ sends party-opening challenge (as usual);
5. $\mathcal{P}$ opens selected parties.

3 Banquet: Verification of Inverses

$\mathcal{P}$ injects m "suspicious" inverses $t = s^{-1}$, so MPCitH parties have pairs (s, t) such that $s \cdot t = 1$ *allegedly*.

Naïve verification protocol

For the ℓ-th inverse operation:

1: Set multiplication tuple $(s_\ell, t_\ell, 1)$.
2: Sacrifice with triple (a, b, c).

This is expensive: each multiplication requires 1 correlated triple, and 1 sacrifice. $4|C| + 1$ elts. in total.

3 Banquet: Polynomial-based Verification I

Define polynomials S, T and $P = S \cdot T$ as:

$$\begin{array}{ccc} S(1) = s_1 & T(1) = t_1 & P(1) = s_1 \cdot t_1 = 1 \\ \vdots & \vdots & \vdots \\ S(m) = s_m & T(m) = t_m & P(m) = s_m \cdot t_m = 1 \end{array}$$

Check $P \stackrel{?}{=} S \cdot T$:

1: Sample random $R \leftarrow \mathbb{F} \setminus \{1, \ldots, m\}$;
2: Open $P(R), S(R), T(R)$
3: Check

$$P(R) \stackrel{?}{=} S(R) \cdot T(R).$$

3 Banquet: Polynomial-based Verification II

Lemma (Schwartz–Zippel)

Let $Q \in \mathbb{F}[x]$ be non-zero of degree $d \geq 0$; for any $\mathbb{S} \subseteq \mathbb{F}$, $\Pr_{R \leftarrow \mathbb{S}}[Q(R) = 0] \leq \frac{d}{|\mathbb{S}|}$.

- Here, $Q = P - S \cdot T$; non-zero iff $t_\ell \neq s_\ell^{-1}$ for some ℓ.
- Opening $S(R)$, $T(R)$ leaks information $\Rightarrow$ add random points $S(0), T(0)$.
- P (and also Q) is of degree $d = 2m$ and $|\mathbb{S}| = |\mathbb{F} - m|$, so

$$\Pr_{R \leftarrow \mathbb{S}}[Q(R) = 0] \leq \frac{2m}{|\mathbb{F} - m|}.$$

3 Polynomial-based Verification III

Improved protocol

1. Prover commits to S (randomized) and T.
2. Prover commits to P.
3. MPC parties open $Q(R) = P(R) - S(R) \cdot T(R)$, for random R.

$2|C| + 4$ elts.; no cut & choose, no triple. Actually one triple, but hidden!

(Extra randomness $r_1, \ldots, r_m$ in S prevents further cheating.)

KU LEUVEN

3 Generalized polynomial-based checking I

Previous protocol verifies:

$$\begin{pmatrix} r_1 s_1 & \cdots & r_m s_m \end{pmatrix} \begin{pmatrix} t_1 \\ \vdots \\ t_m \end{pmatrix} \stackrel{?}{=} \sum_{\ell=1}^{m} r_\ell.$$

Now, let $m = m_1 \cdot m_2$, and instead verify:

$$\begin{pmatrix} r_1 s_{1,k} & \cdots & r_{m_1} s_{m_1,k} \end{pmatrix} \begin{pmatrix} t_{1,k} \\ \vdots \\ t_{m_1,k} \end{pmatrix} \stackrel{?}{=} \sum_{j=1}^{m_1} r_j, \qquad k \in \{0, \ldots, m_2 - 1\}.$$

($s_{j,k}$ and $t_{j,k}$ are rearranged from s_ℓ and t_ℓ.)

3 Generalized polynomial-based checking II

Define S_j and T_j as

$$\begin{aligned} S_j(k) &= r_j \cdot s_{j,k} & T_j(k) &= t_{j,k} & k \in \{0, \ldots, m_2 - 1\} \\ S_j(m_2) &= \bar{s}_j & T_j(m_2) &= \bar{t}_j; \end{aligned}$$

and let $P = \sum_{j=1}^{m_1} S_j \cdot T_j$.
Generalized verification protocol

1. Prover commits to S_j (randomized) and T_j;
2. Prover commits to P;
3. MPC parties open $Q(R) = P(R) - \sum_{j=1}^{m_1} S_j(R) \cdot T_j(R)$, for random R;

Total: $= |C| + O(\sqrt{|C|})$, instead of $2|C|$.

3 Inner-product compression

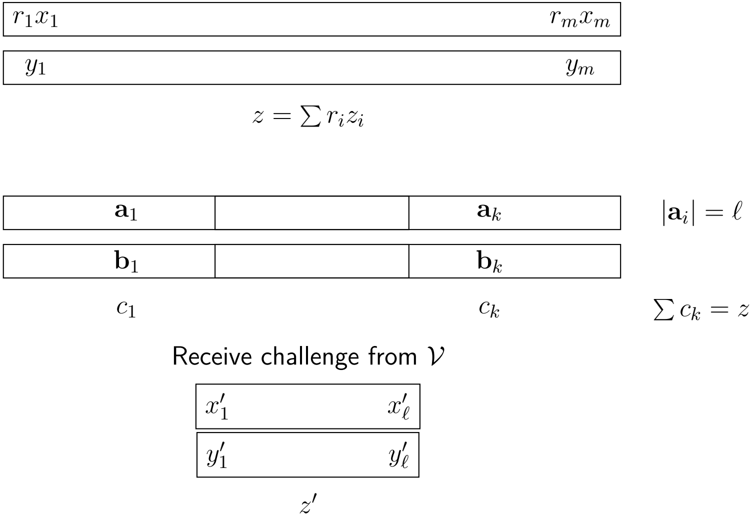
Limbo [dOT21]: several compression rounds.

Checking

$$\langle \tilde{\mathbf{x}}, \mathbf{y} \rangle \stackrel{?}{=} z$$

becomes

$$\langle \mathbf{x}', \mathbf{y}' \rangle \stackrel{?}{=} z'.$$



Questions?

Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner.
Ciphers for MPC and FHE.
In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 430–454. Springer, Heidelberg, April 2015.

Carsten Baum and Ariel Nof.
Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography.
In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 495–526. Springer, Heidelberg, May 2020.

Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha.
Post-quantum zero-knowledge and signatures from symmetric-key primitives.
In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017.

Cyprien de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart.
BBQ: Using AES in picnic signatures.
In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 669–692. Springer, Heidelberg, August 2019.

Cyprien de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy.
Limbo: Efficient zero-knowledge MPCitH-based arguments.
In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021.

Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens.
SPHINCS+.
Technical report, National Institute of Standards and Technology, 2022.
available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai.
Zero-knowledge from secure multiparty computation.
In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

Daniel Kales and Greg Zaverucha.
Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures.
Cryptology ePrint Archive, Report 2022/588, 2022.
https://eprint.iacr.org/2022/588.

Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai.
CRYSTALS-DILITHIUM.
Technical report, National Institute of Standards and Technology, 2022.
available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang.
FALCON.
Technical report, National Institute of Standards and Technology, 2022.
available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales.
Picnic.
Technical report, National Institute of Standards and Technology, 2020.
available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.