SPECK

Signature from Permutation Equivalence of Codes and Kernels

Rahmi El Mechri
r.elmechri@staff.univpm.it

Dipartimento di Ingegneria dell'Informazione
Università Politecnica delle Marche

Workshop on the mathematics of post-quantum cryptography @ UZH
June 3rd, 2025

SPECK

Linear codes over finite fields

- A **linear code** $\mathscr{C} \in \mathbb{F}_q^n$ of length n and dimension k is a k-dimensional linear subspace of $\mathbb{F}_q^n$.

Linear codes over finite fields

- A **linear code** $\mathscr{C} \in \mathbb{F}_q^n$ of length n and dimension k is a k-dimensional linear subspace of $\mathbb{F}_q^n$.

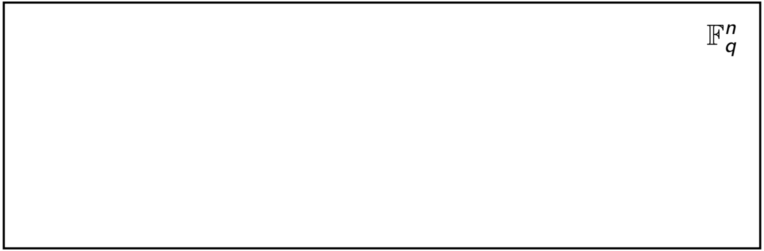
- A **generator matrix** for $\mathscr{C}$ is a matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ whose rows form a basis for $\mathscr{C}$.
  - $\mathscr{C} = \{\mathbf{uG} \mid \mathbf{u} \in \mathbb{F}_q^k\}$
  - A generator matrix is in **systematic form** when it is in the form $(\mathbf{I}_k \mid \mathbf{A})$.

- A **parity-check matrix** for $\mathscr{C}$ is a matrix $\mathbf{H} \in \mathbb{F}_q^{n-k \times n}$ whose rows form a basis for $\mathscr{C}^\perp$.
  - $\mathscr{C} = \{\mathbf{c} \in \mathbb{F}_q^n \mid \mathbf{cH}^\top = \mathbf{0}\}$
  - A parity-check matrix is in **systematic form** when it is in the form $(-\mathbf{A}^\top \mid \mathbf{I}_{n-k})$.

Dual codes and hulls

The **hull** of a code $\mathscr{C}$ is the subspace given by the intersection of $\mathscr{C}$ and its dual $\mathscr{C}^\perp$:

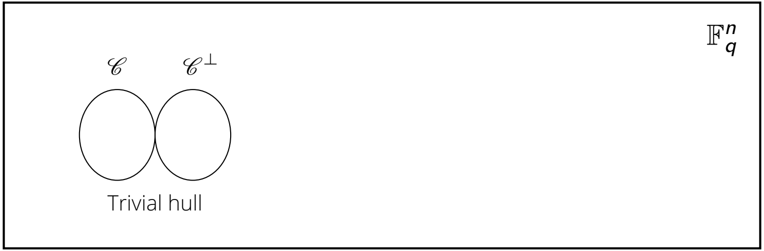
$$\mathcal{H}(\mathscr{C}) = \mathscr{C} \cap \mathscr{C}^\perp$$



Dual codes and hulls

The **hull** of a code $\mathscr{C}$ is the subspace given by the intersection of $\mathscr{C}$ and its dual $\mathscr{C}^\perp$:

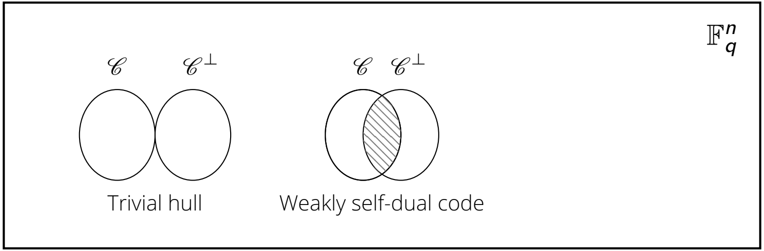
$$\mathcal{H}(\mathscr{C}) = \mathscr{C} \cap \mathscr{C}^\perp$$



Dual codes and hulls

The **hull** of a code $\mathscr{C}$ is the subspace given by the intersection of $\mathscr{C}$ and its dual $\mathscr{C}^\perp$:

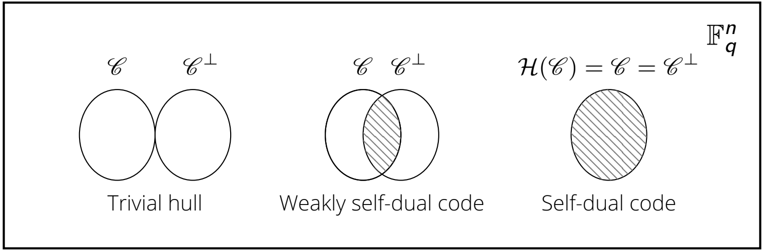
$$\mathcal{H}(\mathscr{C}) = \mathscr{C} \cap \mathscr{C}^\perp$$



Dual codes and hulls

The **hull** of a code $\mathscr{C}$ is the subspace given by the intersection of $\mathscr{C}$ and its dual $\mathscr{C}^\perp$:

$$\mathcal{H}(\mathscr{C}) = \mathscr{C} \cap \mathscr{C}^\perp$$



Maps Maps Maps

- We consider codes endowed with the **Hamming metric**:

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) = |\{i \ \text{ s.t. } \ a_i \neq b_i\}|$$

Maps Maps Maps

- We consider codes endowed with the **Hamming metric**:

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) = |\{i \text{ s.t. } a_i \neq b_i\}|$$

- We are interested in **isometries**, maps preserving the hamming distance:

Maps Maps Maps

- We consider codes endowed with the **Hamming metric**:

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) = |\{i \text{ s.t. } a_i \neq b_i\}|$$

- We are interested in **isometries**, maps preserving the hamming distance:
  - **Permutations group $\mathcal{S}_n$ of length-n**:

$$\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi^{-1}(1)}, a_{\pi^{-1}(2)}, \ldots, a_{\pi^{-1}(n)}\big)$$

Maps Maps Maps

- We consider codes endowed with the **Hamming metric**:

$$\mathrm{dist}(\mathbf{a}, \mathbf{b}) = |\{i \ \text{ s.t. } \ a_i \neq b_i\}|$$

- We are interested in **isometries**, maps preserving the hamming distance:
  - **Permutations group $\mathcal{S}_n$ of length-n**:

$$\pi\big((a_1, a_2, \dots, a_n)\big) = \big(a_{\pi^{-1}(1)}, a_{\pi^{-1}(2)}, \dots, a_{\pi^{-1}(n)}\big)$$

  - **Monomials group $\mathcal{M}_n$ of length-n**:

$$\mu = (v; \pi) \in \mathbb{F}_q^{*n} \times \mathcal{S}_n \implies \mu\big((a_1, a_2, \dots, a_n)\big) = \big(v_1 \cdot a_{\pi^{-1}(1)}, \dots, v_n \cdot a_{\pi^{-1}(n)}\big)$$

Linear Equivalence

LESS (Linear Equivalence Signature Scheme) [2] signature scheme is based on:

Linear Equivalence Problem (LEP)

Given two linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$, with respective generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$, find (if it exists) a monomial matrix $\mathbf{Q} \in \mathcal{M}_n$ and a non singular $\mathbf{S} \in GL_k(\mathbb{F}_q)$ such that $\mathbf{G}' = \mathbf{SGQ}$.

Linear Equivalence

LESS (Linear Equivalence Signature Scheme) [2] signature scheme is based on:

Linear Equivalence Problem (LEP)

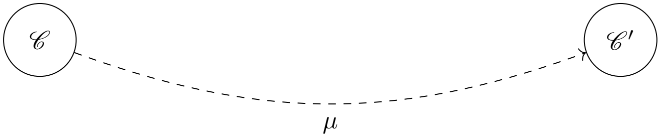
Given two linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$, with respective generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$, find (if it exists) a monomial matrix $\mathbf{Q} \in \mathcal{M}_n$ and a non singular $\mathbf{S} \in GL_k(\mathbb{F}_q)$ such that $\mathbf{G}' = \mathbf{SGQ}$.

Characteristics:

- The problem cannot be NP-complete (unless the polynomial hierarchy collapses).
- All known solvers take exponential time for average LEP if $q \geq 5$.

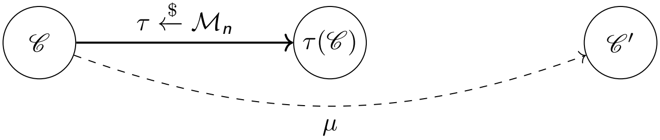
The LESS ZK protocol

sk: $\mu \xleftarrow{\$} \mathcal{M}_n$, pk: $(\mathscr{C}\ ,\ \mathscr{C}')$ such that $\mathscr{C}' = \mu(\mathscr{C})$



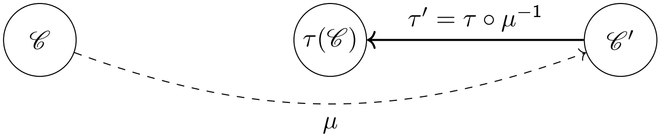
The LESS ZK protocol

sk: $\mu \xleftarrow{\$} \mathcal{M}_n$, pk: $(\mathscr{C}\ ,\ \mathscr{C}')$ such that $\mathscr{C}' = \mu(\mathscr{C})$



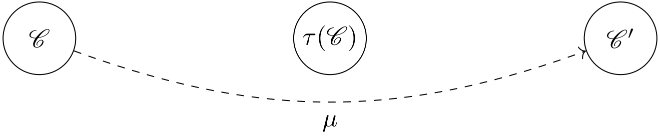
The LESS ZK protocol

sk: $\mu \xleftarrow{\$} \mathcal{M}_n$, pk: $(\mathscr{C}\ ,\ \mathscr{C}')$ such that $\mathscr{C}' = \mu(\mathscr{C})$



The LESS ZK protocol

sk: $\mu \xleftarrow{\$} \mathcal{M}_n$, pk: $(\mathscr{C}\ ,\ \mathscr{C}')$ such that $\mathscr{C}' = \mu(\mathscr{C})$

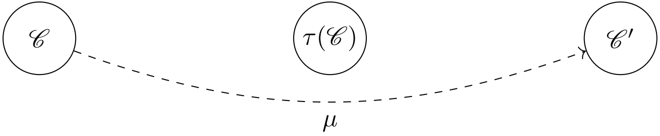


- LESS achieves very compact signatures ($\sim$ 2 KB) when **canonical forms** [3] are used:

$$\mathsf{CF}(\mathbf{A}) = \mathsf{CF}(\mathbf{M}_\mathrm{r} \cdot \mathbf{A} \cdot \mathbf{M}_\mathrm{c}), \ \ \forall \mathbf{M}_\mathrm{r} \in M_k,\ \forall \mathbf{M}_\mathrm{c} \in M_{n-k}$$

The LESS ZK protocol

$$\text{sk: } \mu \xleftarrow{\$} \mathcal{M}_n, \qquad \text{pk: } (\mathscr{C}, \mathscr{C}') \text{ such that } \mathscr{C}' = \mu(\mathscr{C})$$



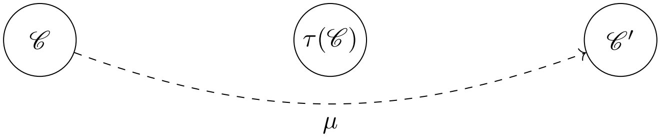
- LESS achieves very compact signatures ($\sim$ 2 KB) when **canonical forms** [3] are used:

$$\mathsf{CF}(\mathbf{A}) = \mathsf{CF}(\mathbf{M}_{\mathrm{r}} \cdot \mathbf{A} \cdot \mathbf{M}_{\mathrm{c}}), \quad \forall \mathbf{M}_{\mathrm{r}} \in M_k, \ \forall \mathbf{M}_{\mathrm{c}} \in M_{n-k}$$

- Verification requires $O(n^3)$ operations (Gaussian elimination).

The LESS ZK protocol

sk: $\mu \xleftarrow{\$} \mathcal{M}_n$, pk: $(\mathscr{C}, \mathscr{C}')$ such that $\mathscr{C}' = \mu(\mathscr{C})$



- LESS achieves very compact signatures ($\sim$ 2 KB) when **canonical forms** [3] are used:

$$\mathsf{CF}(\mathbf{A}) = \mathsf{CF}(\mathbf{M}_{\mathrm{r}} \cdot \mathbf{A} \cdot \mathbf{M}_{\mathrm{c}}), \ \ \forall \mathbf{M}_{\mathrm{r}} \in M_k, \ \forall \mathbf{M}_{\mathrm{c}} \in M_{n-k}$$

- Verification requires $O(n^3)$ operations (Gaussian elimination). $\leftarrow$ **Computational bottleneck!**

Rationale

Can we tweak LESS in order to achieve a more efficient verification procedure?

Rationale

Can we tweak **LESS** in order to achieve a more efficient verification procedure?

- Prove the knowledge of the map μ on a codeword $\mathbf{c} \in \mathbf{G}$, rather than the whole code:

$$\underbrace{\mathsf{SF}(\mu(\mathbf{G})) = \mathsf{SF}(\mathbf{G}')}_{O(n^3)} \longrightarrow \underbrace{\mu(\mathbf{c})\mathbf{H}'^{\top} = \mathbf{0}}_{O(n^2)}$$

Rationale

Can we tweak **LESS** in order to achieve a more efficient verification procedure?

- Prove the knowledge of the map μ on a codeword $\mathbf{c} \in \mathbf{G}$, rather than the whole code:

$$\underbrace{\mathrm{SF}(\mu(\mathbf{G})) = \mathrm{SF}(\mathbf{G}')}_{O(n^3)} \longrightarrow \underbrace{\mu(\mathbf{c})\mathbf{H}'^{\top} = \mathbf{0}}_{O(n^2)}$$

- Forgery would mean finding a trasformation that sends $\mathbf{c}$ into $\mathscr{C}'$:

Rationale

Can we tweak **LESS** in order to achieve a more efficient verification procedure?

- Prove the knowledge of the map μ on a codeword $\mathbf{c} \in \mathbf{G}$, rather than the whole code:

$$\underbrace{\mathsf{SF}(\mu(\mathbf{G})) = \mathsf{SF}(\mathbf{G}')}_{O(n^3)} \longrightarrow \underbrace{\mu(\mathbf{c})\mathbf{H}'^{\top} = \mathbf{0}}_{O(n^2)}$$

- Forgery would mean finding a trasformation that sends $\mathbf{c}$ into $\mathscr{C}'$:
  - Trivial when monomial maps are considered, just find:

$$\mathbf{c}' \in \mathscr{C}' : \mathsf{wt}(\mathbf{c}') = \mathsf{wt}(\mathbf{c})$$

Rationale

Can we tweak **LESS** in order to achieve a more efficient verification procedure?

- Prove the knowledge of the map μ on a codeword $\mathbf{c} \in \mathbf{G}$, rather than the whole code:

$$\underbrace{\mathsf{SF}(\mu(\mathbf{G})) = \mathsf{SF}(\mathbf{G}')}_{O(n^3)} \longrightarrow \underbrace{\mu(\mathbf{c})\mathbf{H}'^{\top} = \mathbf{0}}_{O(n^2)}$$

- Forgery would mean finding a trasformation that sends $\mathbf{c}$ into $\mathscr{C}'$:
  - Trivial when monomial maps are considered, just find:

$$\mathbf{c}' \in \mathscr{C}' : \mathsf{wt}(\mathbf{c}') = \mathsf{wt}(\mathbf{c})$$

  - We need to rely on permutation equivalence.

The Permutation Equivalence Problem (PEP)

Permutation Equivalence Problem (PEP)

Given two linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$, with respective generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$, find (if it exists) a permutation $\mathbf{P} \in S_n$ and a non singular $\mathbf{S} \in GL_k(\mathbb{F}_q)$ such that $\mathbf{G}' = \mathbf{SGP}$.

The Permutation Equivalence Problem (PEP)

Permutation Equivalence Problem (PEP)

Given two linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$, with respective generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$, find (if it exists) a permutation $\mathbf{P} \in S_n$ and a non singular $\mathbf{S} \in GL_k(\mathbb{F}_q)$ such that $\mathbf{G}' = \mathbf{SGP}$.

Characteristics:

- Known solvers for PEP take polynomial time when random codes are considered [5] [1].
- Known solvers for PEP take exponential time when **(weakly) self-dual** codes are considered.

The Permuted Kernel Problem (PKP)

Permuted Kernel Problem (PKP)

Given a linear code $\mathscr{C} \subseteq \mathbb{F}_q^n$ with parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and a vector $\mathbf{c} \in \mathbb{F}_q^n$, find (if it exists) a permutation $\mathbf{P} \in S_n$ such that $\mathbf{cPH}^\top = \mathbf{0}$.

The Permuted Kernel Problem (PKP)

Permuted Kernel Problem (PKP)

Given a linear code $\mathscr{C} \subseteq \mathbb{F}_q^n$ with parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ and a vector $\mathbf{c} \in \mathbb{F}_q^n$, find (if it exists) a permutation $\mathbf{P} \in S_n$ such that $\mathbf{cPH}^\top = \mathbf{0}$.

Characteristics:

- It's a well known **NP**-Hard Problem [4].
- It allows fast verifcation of a given solution.

The Permutation Equivalence of Codes and Kernels (PECK) Problem

SPECK signature scheme is based on:

Permutation Equivalence of Codes and Kernels (PECK) Problem

Given two permutation equivalent codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ of dimension k, having respectively generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}'$ and a random $\mathbf{u} \in \mathbb{F}_q^k$, find a permutation $\mathbf{P} \in S_n$ such that $\mathbf{uGPH}'^\top = \mathbf{0}$.

The Permutation Equivalence of Codes and Kernels (PECK) Problem

SPECK signature scheme is based on:

Permutation Equivalence of Codes and Kernels (PECK) Problem

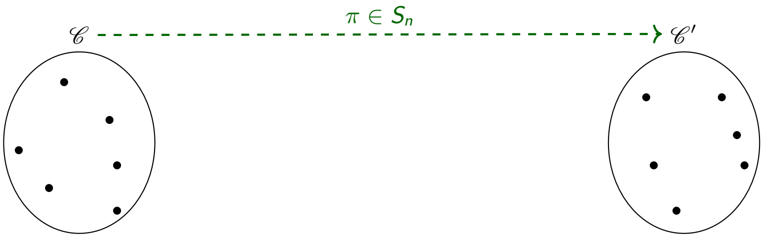
Given two permutation equivalent codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ of dimension k, having respectively generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}'$ and a random $\mathbf{u} \in \mathbb{F}_q^k$, find a permutation $\mathbf{P} \in S_n$ such that $\mathbf{uGPH}'^\top = \mathbf{0}$.

The Permutation Equivalence of Codes and Kernels (PECK) Problem

SPECK signature scheme is based on:

Permutation Equivalence of Codes and Kernels (PECK) Problem

Given two permutation equivalent codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ of dimension k, having respectively generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}'$ and a random $\mathbf{u} \in \mathbb{F}_q^k$, find a permutation $\mathbf{P} \in S_n$ such that $\mathbf{uGPH'}^\top = \mathbf{0}$.

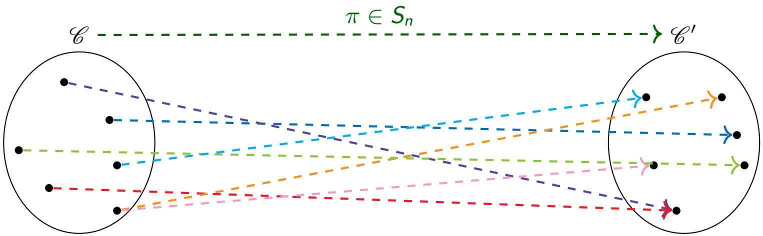


The Permutation Equivalence of Codes and Kernels (PECK) Problem

SPECK signature scheme is based on:

Permutation Equivalence of Codes and Kernels (PECK) Problem

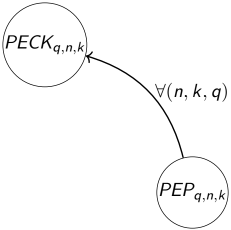
Given two permutation equivalent codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ of dimension k, having respectively generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}'$ and a random $\mathbf{u} \in \mathbb{F}_q^k$, find a permutation $\mathbf{P} \in S_n$ such that $\mathbf{uGPH}'^{\top} = \mathbf{0}$.



Hardness of PECK

$PECK_{q,n,k}$

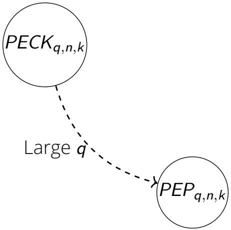
Hardness of PECK



PECK **is always easier than** PEP

- PECK instance $\{\mathbf{c}, \mathbf{G}, \mathbf{H}'\} \longrightarrow$ PEP instance $\{\mathbf{G}, \mathbf{H}'\}$
- Solver for PEP with input $\{\mathbf{G}, \mathbf{H}'\}$
- π also sends $\mathbf{c}$ to $\mathscr{C}'$ $\overset{\pi}{\longleftarrow}$ The solver returns $\pi \in \mathcal{S}_n$ which sends $\mathscr{C}$, ($\mathbf{G}$) to $\mathscr{C}'$ ($\mathbf{H}'$)

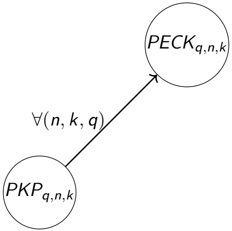
Hardness of PECK



PECK **is as hard as** PEP **when q is large**

- When $q >> n$ with high probability random codewords have no repeated values
- The unique solution for PECK sends the whole code to $\mathscr{C}'$
- PEP instance $\{\mathbf{G}, \mathbf{H}'\}$ $\xrightarrow{c \xleftarrow{\$} C}$ PECK instance $\{\mathbf{c}, \mathbf{G}, \mathbf{H}'\}$
- Solver for PECK with input $\{\mathbf{c}, \mathbf{G}, \mathbf{H}'\}$
- π sends sends $\mathscr{C}$ ($\mathbf{G}$) to $\mathscr{C}'$ ($\mathbf{H}'$) $\xleftarrow{\pi \in \mathcal{S}_n}$ The solver returns $\pi \in \mathcal{S}_n$ which sends $\mathbf{c}$ to $\mathscr{C}'$ ($\mathbf{H}'$)

Hardness of PECK

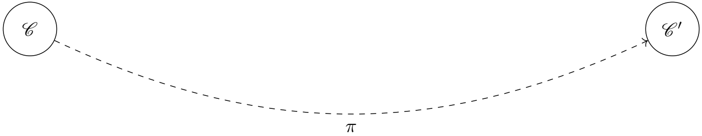


Relations with PKP

- Problem resembles PKP, which is hard to solve.
- The best ISD-solver for PKP can be adapted to PECK.

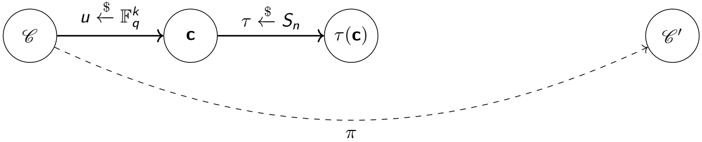
SPECK interactive protocol

sk: $\pi \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}\ ,\ \mathscr{C}')$ such that $\mathscr{C}' = \pi(\mathscr{C})$



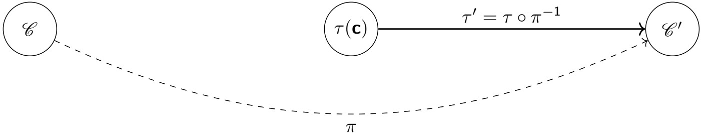
SPECK interactive protocol

sk: $\pi \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}\ ,\ \mathscr{C}')$ such that $\mathscr{C}' = \pi(\mathscr{C})$



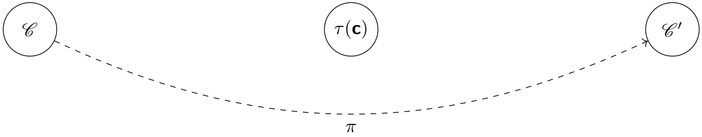
SPECK interactive protocol

sk: $\pi \xleftarrow{\$} \mathcal{S}_n$,   pk: $(\mathscr{C}, \mathscr{C}')$ such that $\mathscr{C}' = \pi(\mathscr{C})$



SPECK interactive protocol

$$\text{sk: } \pi \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathscr{C}\ ,\ \mathscr{C}') \text{ such that } \mathscr{C}' = \pi(\mathscr{C})$$



- SPECK signature scheme obtained by applying **Fiat-Shamir transform**.
- Two regimes:

SPECK – Low	SPECK – High
$q = 127$	$q = 8861$
Smaller keys and signatures Multiple solutions	Larger keys and signatures Unique solution

Optimizations

Protocol specific:

Optimizations

Protocol specific:

- Generator matrices in systematic form: $\mathsf{pk} = (\mathbf{A}, \mathbf{A}')$.

Optimizations

Protocol specific:

- Generator matrices in systematic form: $\mathsf{pk} = (\mathbf{A}, \mathbf{A}')$.
- Encoding techniques for self-dual codes: $\mathsf{pk} = (\mathsf{Triang}(\mathbf{A}), \mathsf{Triang}(\mathbf{A}'))$

Optimizations

Protocol specific:

- Generator matrices in systematic form: $\mathsf{pk} = (\mathbf{A}, \mathbf{A}')$.
- Encoding techniques for self-dual codes: $\mathsf{pk} = (\mathsf{Triang}(\mathbf{A}), \mathsf{Triang}(\mathbf{A}'))$
- Lexicographical ordering as canonical representative: $\mathbf{x} = \mathsf{LexMin}(\mathbf{c})$.

Optimizations

Protocol specific:

- Generator matrices in systematic form: $\mathsf{pk} = (\mathbf{A}, \mathbf{A}')$.
- Encoding techniques for self-dual codes: $\mathsf{pk} = (\mathsf{Triang}(\mathbf{A}), \mathsf{Triang}(\mathbf{A}'))$
- Lexicographical ordering as canonical representative: $\mathbf{x} = \mathsf{LexMin}(\mathbf{c})$.
- Removing redundancy from codewords in response: $\mathbf{c}_2 = \mathbf{c}_1 \mathbf{A}^\top$.

Optimizations

Protocol specific:

- Generator matrices in systematic form: $\mathsf{pk} = (\mathbf{A}, \mathbf{A}')$.
- Encoding techniques for self-dual codes: $\mathsf{pk} = (\mathsf{Triang}(\mathbf{A}), \mathsf{Triang}(\mathbf{A}'))$
- Lexicographical ordering as canonical representative: $\mathbf{x} = \mathsf{LexMin}(\mathbf{c})$.
- Removing redundancy from codewords in response: $\mathbf{c}_2 = \mathbf{c}_1\mathbf{A}^\top$.
- Unique commitment: $\mathtt{cmt} = \mathsf{Hash}\big(\mathtt{cmt}^{(1)}||\cdots||\mathtt{cmt}^{(t)}\big)$.

Optimizations

Protocol specific:

- Generator matrices in systematic form: $\mathsf{pk} = (\mathbf{A}, \mathbf{A}')$.
- Encoding techniques for self-dual codes: $\mathsf{pk} = (\mathsf{Triang}(\mathbf{A}), \mathsf{Triang}(\mathbf{A}'))$
- Lexicographical ordering as canonical representative: $\mathbf{x} = \mathsf{LexMin}(\mathbf{c})$.
- Removing redundancy from codewords in response: $\mathbf{c}_2 = \mathbf{c}_1 \mathbf{A}^\top$.
- Unique commitment: $\mathtt{cmt} = \mathsf{Hash}\big(\mathtt{cmt}^{(1)} || \cdots || \mathtt{cmt}^{(t)}\big)$.

The usual:

- Fixed-weight string for challenge selection.
- Puncturable PRF w/ GGM Trees.

Optimizations

Protocol specific:

- Generator matrices in systematic form: $\mathsf{pk} = (\mathbf{A}, \mathbf{A}')$.
- Encoding techniques for self-dual codes: $\mathsf{pk} = (\mathsf{Triang}(\mathbf{A}), \mathsf{Triang}(\mathbf{A}'))$
- Lexicographical ordering as canonical representative: $\mathbf{x} = \mathsf{LexMin}(\mathbf{c})$.
- Removing redundancy from codewords in response: $\mathbf{c}_2 = \mathbf{c}_1\mathbf{A}^\top$.
- Unique commitment: $\mathtt{cmt} = \mathsf{Hash}\big(\mathtt{cmt}^{(1)}||\cdots||\mathtt{cmt}^{(t)}\big)$.

The usual:

- Fixed-weight string for challenge selection.
- Puncturable PRF w/ GGM Trees.
- **Multiple keys can't be used!**

Multiple keys: a cheating strategy

sk: $(\pi_1, \dots, \pi_{s-1})$, $\pi_i \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}_0, \mathscr{C}_1, \dots, \mathscr{C}_{s-1})$ such that $\mathscr{C}_i = \pi_i(\mathscr{C}_0)$

$\mathbf{y}_1$ $\mathscr{C}_1$

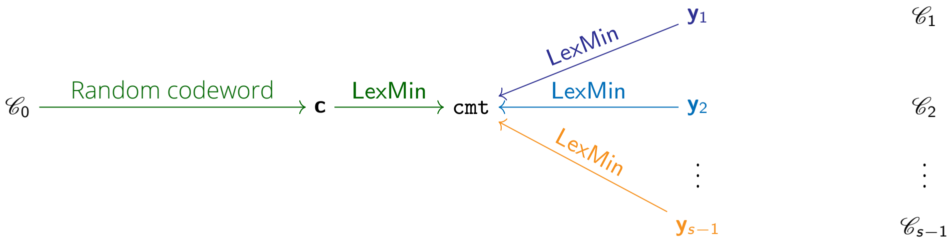
$\mathscr{C}_0 \xrightarrow{\text{Random codeword}} \mathbf{c} \xrightarrow{\text{LexMin}} \texttt{cmt}$ $\mathbf{y}_2$ $\mathscr{C}_2$

$\vdots$ $\vdots$

$\mathbf{y}_{s-1}$ $\mathscr{C}_{s-1}$

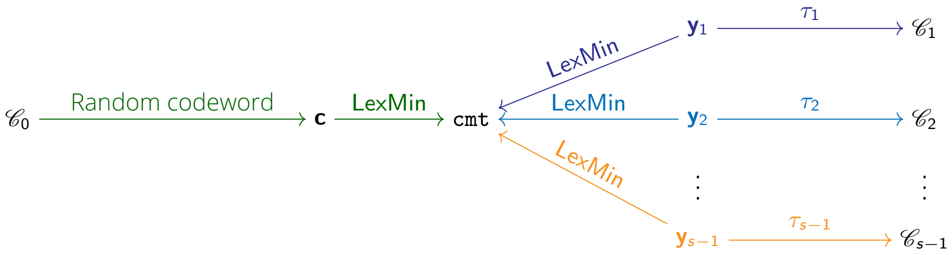
Multiple keys: a cheating strategy

$$\text{sk: } (\pi_1, \ldots, \pi_{s-1}), \quad \pi_i \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathscr{C}_0, \mathscr{C}_1, \ldots, \mathscr{C}_{s-1}) \text{ such that } \mathscr{C}_i = \pi_i(\mathscr{C}_0)$$



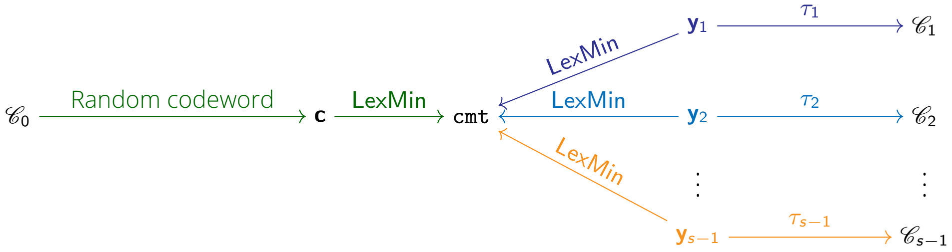
Multiple keys: a cheating strategy

sk: $(\pi_1, \dots, \pi_{s-1})$, $\pi_i \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}_0, \mathscr{C}_1, \dots, \mathscr{C}_{s-1})$ such that $\mathscr{C}_i = \pi_i(\mathscr{C}_0)$



Multiple keys: a cheating strategy

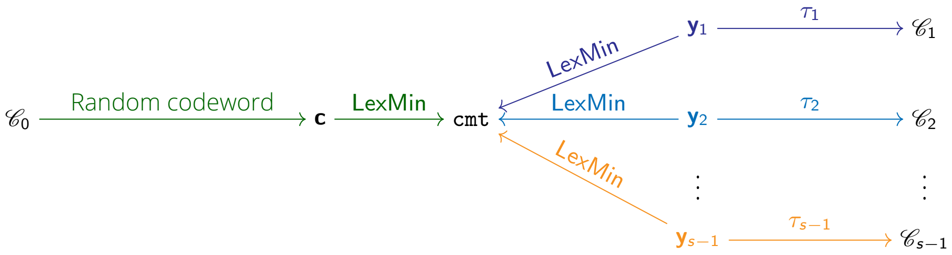
sk: $(\pi_1, \ldots, \pi_{s-1})$, $\pi_i \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}_0, \mathscr{C}_1, \ldots, \mathscr{C}_{s-1})$ such that $\mathscr{C}_i = \pi_i(\mathscr{C}_0)$



- Intersection between codes is not trivial with very high probability.

Multiple keys: a cheating strategy

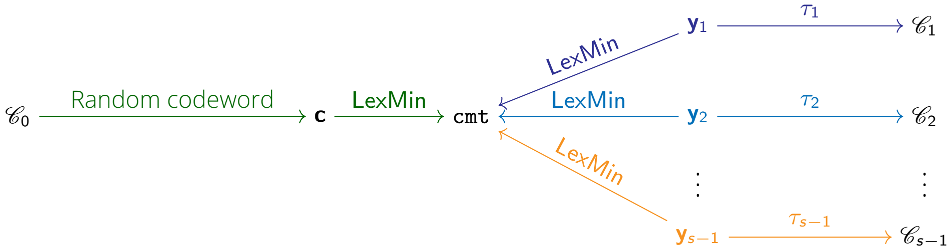
sk: $(\pi_1, \ldots, \pi_{s-1})$, $\pi_i \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}_0, \mathscr{C}_1, \ldots, \mathscr{C}_{s-1})$ such that $\mathscr{C}_i = \pi_i(\mathscr{C}_0)$



- Intersection between codes is not trivial with very high probability.
- Cheat by finding $\mathbf{c}_1, \ldots, \mathbf{c}_{s-1}$ such that $\text{LexMin}(\mathbf{c}_i) = \text{LexMin}(\mathbf{c}_j) \quad \forall i, j$

Multiple keys: a cheating strategy

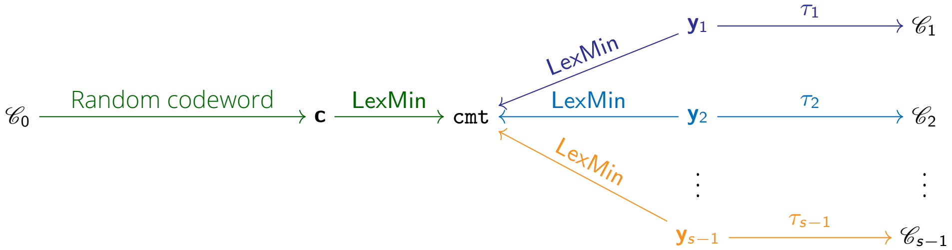
sk: $(\pi_1, \dots, \pi_{s-1})$, $\pi_i \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}_0, \mathscr{C}_1, \dots, \mathscr{C}_{s-1})$ such that $\mathscr{C}_i = \pi_i(\mathscr{C}_0)$



- Intersection between codes is not trivial with very high probability.
- Cheat by finding $\mathbf{c}_1, \dots, \mathbf{c}_{s-1}$ such that $\text{LexMin}(\mathbf{c}_i) = \text{LexMin}(\mathbf{c}_j) \quad \forall i, j$
- Derived soundness error is closer to $\frac{1}{2}$ than to $\frac{1}{s}$.

Multiple keys: a cheating strategy

sk: $(\pi_1, \ldots, \pi_{s-1})$, $\pi_i \xleftarrow{\$} \mathcal{S}_n$, pk: $(\mathscr{C}_0, \mathscr{C}_1, \ldots, \mathscr{C}_{s-1})$ such that $\mathscr{C}_i = \pi_i(\mathscr{C}_0)$



- Intersection between codes is not trivial with very high probability.
- Cheat by finding $\mathbf{c}_1, \ldots, \mathbf{c}_{s-1}$ such that $\text{LexMin}(\mathbf{c}_i) = \text{LexMin}(\mathbf{c}_j) \quad \forall i, j$
- Derived soundness error is closer to $\frac{1}{2}$ than to $\frac{1}{s}$.
- Not worth it considering the impact on keys' sizes.

One round of SPECK

$$\mathsf{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \mathsf{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

VERIFIER

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A} \ , \ \mathbf{A}')$$

PROVER VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$

One round of SPECK

$$\text{sk}: \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$

Get $\mathbf{u} \leftarrow \mathsf{PRF}(\texttt{Seed})$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \mathsf{PRF}(\texttt{Seed})$

Compute $\mathbf{c} := \mathbf{uG}$

One round of SPECK

$$\mathsf{sk}: \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \mathsf{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

VERIFIER

Sample $\mathtt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \mathsf{PRF}(\mathtt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$

Compute $\mathbf{x} := \mathsf{LexMin}(\mathbf{c})$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A} \ , \ \mathbf{A}')$$

PROVER

VERIFIER

Sample $\mathtt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \mathsf{PRF}(\mathtt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \mathsf{LexMin}(\mathbf{c})$

Set $\mathtt{cmt} := \mathsf{Hash}(\mathbf{x})$

One round of SPECK

$$\text{sk}: \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$$\xrightarrow{\texttt{cmt}}$$

Sample $b \xleftarrow{\$} \{0,1\}$

One round of SPECK

$$\mathsf{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \mathsf{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \mathsf{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \mathsf{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \mathsf{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^{\lambda}$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:

Set $\texttt{rsp} := \texttt{Seed}$

One round of SPECK

$$\mathsf{sk}: \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \mathsf{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | VERIFIER

Sample $\mathtt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \mathsf{PRF}(\mathtt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \mathsf{LexMin}(\mathbf{c})$
Set $\mathtt{cmt} := \mathsf{Hash}(\mathbf{x})$

$\xrightarrow{\mathtt{cmt}}$

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\mathtt{rsp} := \mathtt{Seed}$

Else:

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^{\lambda}$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:

Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^{\lambda}$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$

Set $\texttt{rsp} := (\mathbf{y}_1)$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | | VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
  Set $\texttt{rsp} := \texttt{Seed}$
Else:
  Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
  Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk: } (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

If $b = 0$:

One round of SPECK

$$\text{sk}: \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \quad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

If $b = 0$:
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER | VERIFIER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

If $b = 0$:
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c}_{\texttt{rsp}} := \mathbf{uG}$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^{\lambda}$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

If $b = 0$:
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c}_{\texttt{rsp}} := \mathbf{uG}$

Else:

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^{\lambda}$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

If $b = 0$:
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c}_{\texttt{rsp}} := \mathbf{uG}$
Else:

Compute $\mathbf{c}_{\texttt{rsp}} := (\mathbf{y}_1, -\mathbf{y}_1 * \mathbf{A}'^{\top})$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^{\lambda}$
Get $\mathbf{u} \leftarrow \mathsf{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \mathsf{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \mathsf{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

If $b = 0$:
Get $\mathbf{u} \leftarrow \mathsf{PRF}(\texttt{Seed})$
Compute $\mathbf{c}_{\texttt{rsp}} := \mathbf{uG}$
Else:
Compute $\mathbf{c}_{\texttt{rsp}} := (\mathbf{y}_1, -\mathbf{y}_1 * \mathbf{A}'^{\top})$

Compute $\mathbf{x} := \mathsf{LexMin}(\mathbf{c}_{\texttt{rsp}})$

One round of SPECK

$$\text{sk} : \mathbf{P} \xleftarrow{\$} \mathcal{S}_n, \qquad \text{pk}: (\mathbf{A}\ ,\ \mathbf{A}')$$

PROVER

Sample $\texttt{Seed} \xleftarrow{\$} \{0,1\}^\lambda$
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c} := \mathbf{uG}$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c})$
Set $\texttt{cmt} := \text{Hash}(\mathbf{x})$

$\xrightarrow{\texttt{cmt}}$

VERIFIER

Sample $b \xleftarrow{\$} \{0,1\}$

$\xleftarrow{b}$

If $b = 0$:
Set $\texttt{rsp} := \texttt{Seed}$
Else:
Compute $\mathbf{y} := (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{cP}$
Set $\texttt{rsp} := (\mathbf{y}_1)$

$\xrightarrow{\texttt{rsp}}$

If $b = 0$:
Get $\mathbf{u} \leftarrow \text{PRF}(\texttt{Seed})$
Compute $\mathbf{c}_{\texttt{rsp}} := \mathbf{uG}$
Else:
Compute $\mathbf{c}_{\texttt{rsp}} := (\mathbf{y}_1, -\mathbf{y}_1 * \mathbf{A}'^\top)$
Compute $\mathbf{x} := \text{LexMin}(\mathbf{c}_{\texttt{rsp}})$

Accept if $\texttt{cmt} = \text{Hash}(\mathbf{x})$

Signature size

Signature size

Rsp

Signature size

$$|\mathtt{Rsp}| \leq$$

Signature size

$$|\mathtt{Rsp}| \leq \underbrace{4\lambda}_{\text{Salt and commitment}} +$$

Signature size

$$|\mathtt{Rsp}| \leq \underbrace{4\lambda}_{\text{Salt and commitment}} + \underbrace{\lambda w \log_2(t/w) + \mathrm{wt}(t) - 1}_{\text{Intermediate seeds}} +$$

Signature size

$$|\mathtt{Rsp}| \leq \underbrace{4\lambda}_{\substack{\text{Salt and}\\\text{commitment}}} + \underbrace{\lambda w \log_2(t/w) + \mathrm{wt}(t) - 1}_{\text{Intermediate seeds}} + \underbrace{wk \log_2(q)}_{\substack{\text{Responses for}\\\text{rounds with } b^{(i)} = 1}}$$

Performances

Instance	KeyGen		Sign		Verify	
	ms	MCycles	ms	MCycles	ms	MCycles
$\mathrm{Speck-Low-133-60}$	1.20	3.11	1.48	3.89	1.45	3.79
$\mathrm{Speck-Low-256-30}$	1.23	3.22	2.14	5.60	2.12	5.52
$\mathrm{Speck-Low-512-23}$	1.16	3.03	3.53	9.21	3.50	9.14
$\mathrm{Speck-Low-768-20}$	1.15	2.99	4.88	12.73	4.94	12.89
$\mathrm{Speck-Low-4096-14}$	1.16	3.03	23.30	60.86	23.51	61.38

Table: Timings for the SPECK instances in the Low q regime. Timings have been benchmarked on a 13th Gen Intel(R) Core(TM) i7-1355U and are given both as ms and MCycles, as averages of 128 runs.

SPECK vs. The World

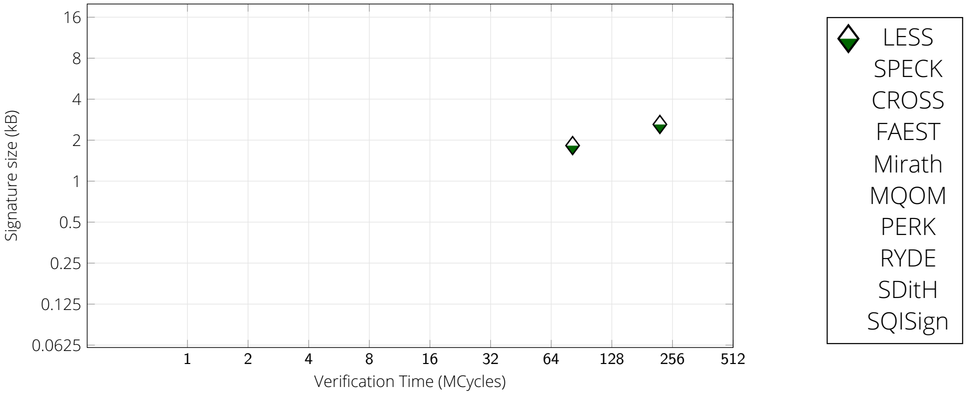


Figure: Overview of the signature size and the verification key size of round 2 NIST additional signatures based on ZK, MPCitH and VOLE-in-the-Head frameworks. Timings have been taken from `https://pqsort.tii.ae/`.

SPECK vs. The World

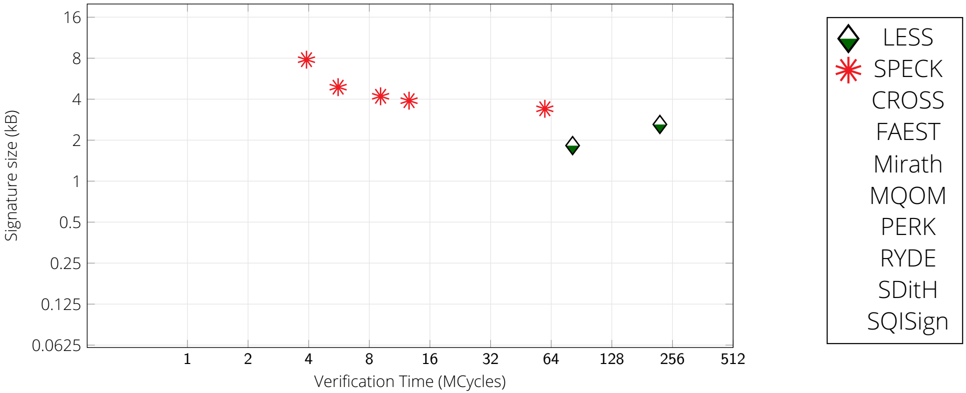


Figure: Overview of the signature size and the verification key size of round 2 NIST additional signatures based on ZK, MPCitH and VOLE-in-the-Head frameworks. Timings have been taken from `https://pqsort.tii.ae/`.

SPECK vs. The World

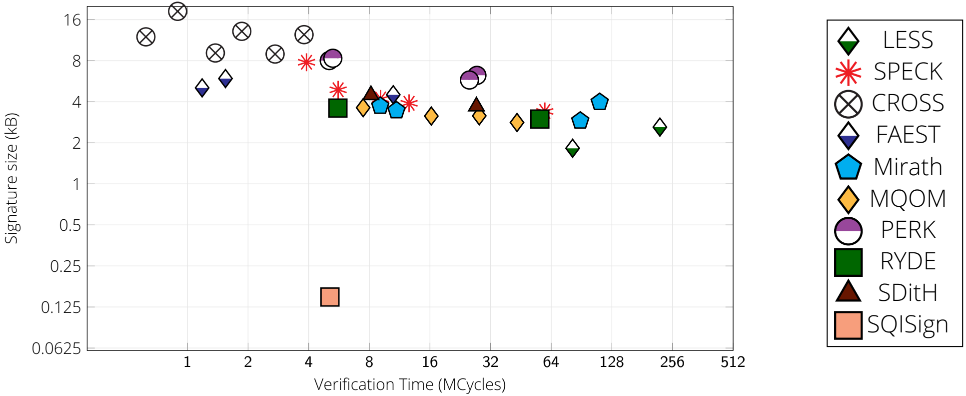


Figure: Overview of the signature size and the verification key size of round 2 NIST additional signatures based on ZK, MPCitH and VOLE-in-the-Head frameworks. Timings have been taken from `https://pqsort.tii.ae/`.

References

[1] M. Bardet, A. Otmani, and M. Saeed-Taha. "Permutation code equivalence is not harder than graph isomorphism when hulls are trivial". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2019, pp. 2464–2468.

[2] J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini. "LESS is more: code-based signatures without syndromes". In: *International Conference on Cryptology in Africa*. Springer. 2020, pp. 45–65.

[3] T. Chou, E. Persichetti, and P. Santini. "On linear equivalence, canonical forms, and digital signatures". In: *Designs, Codes and Cryptography* (2025), pp. 1–43.

[4] P. Santini, M. Baldi, and F. Chiaraluce. "Computational Hardness of the Permuted Kernel and Subcode Equivalence Problems". In: *IEEE Trans. Inf. Theor.* 70.3 (Mar. 2024), 2254–2270. DOI: `10.1109/TIT.2023.3323068`. URL: `https://doi.org/10.1109/TIT.2023.3323068`.

[5] N. Sendrier. "Finding the permutation between equivalent linear codes: The support splitting algorithm". In: *IEEE Transactions on Information Theory* 46.4 (2000), pp. 1193–1203.

Thank you for listening!

ia.cr/2025/923

Finanziato dall'Unione europea
NextGenerationEU

Ministero delle Infrastrutture e dei Trasporti

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA