LESS-FM: Fine-tuning Signatures from the Code Equivalence Problem

A. Barenghi, J.-F. Biasse, E. Persichetti and P. Santini

20 July 2021

Charles E. Schmidt College of Science
Florida Atlantic University

In This Talk

- Motivation
- The Code Equivalence Problem
- LESS and Variants
- Performance and Conclusions

Part I

MOTIVATION

Code-based cryptography is one of the main players in PQC.

Code-based Signatures

Code-based cryptography is one of the main players in PQC.

Great KEM/PKE constructions, signatures still not yet up to speed.

Code-based cryptography is one of the main players in PQC.

Great KEM/PKE constructions, signatures still not yet up to speed.

Traditional protocols:

Code-based Signatures

Code-based cryptography is one of the main players in PQC.

Great KEM/PKE constructions, signatures still not yet up to speed.

Traditional protocols:

- Hash-and-sign $\rightarrow$ large keys (small signatures)
- ZK proof $\rightarrow$ large signatures (small keys)

Code-based Signatures

Code-based cryptography is one of the main players in PQC.

Great KEM/PKE constructions, signatures still not yet up to speed.

Traditional protocols:

- Hash-and-sign $\rightarrow$ large keys (small signatures)
- ZK proof $\rightarrow$ large signatures (small keys)

...all based on hardness of decoding.

CODE-BASED SIGNATURES

Code-based cryptography is one of the main players in PQC.

Great KEM/PKE constructions, signatures still not yet up to speed.

Traditional protocols:

- Hash-and-sign $\rightarrow$ large keys (small signatures)
- ZK proof $\rightarrow$ large signatures (small keys)

...all based on hardness of decoding.

LESS is a new proposal based on Code Equivalence.

(Biasse, Micheli, P., Santini, 2020)

Code-based Signatures

Code-based cryptography is one of the main players in PQC.

Great KEM/PKE constructions, signatures still not yet up to speed.

Traditional protocols:

- Hash-and-sign $\rightarrow$ large keys (small signatures)
- ZK proof $\rightarrow$ large signatures (small keys)

...all based on hardness of decoding.

LESS is a new proposal based on Code Equivalence.

(Biasse, Micheli, P., Santini, 2020)

In this work, we propose new variants and updated parameters, with optimized performance.

Part II

The Code Equivalence Problem

Traditional Code-based Approach

McEliece: first cryptosystem using error-correcting codes (1978).

Traditional Code-based Approach

McEliece: first cryptosystem using error-correcting codes (1978).

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error-correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Traditional Code-based Approach

McEliece: first cryptosystem using error-correcting codes (1978).

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error-correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Based on the hardness of decoding random linear codes.

Traditional Code-based Approach

McEliece: first cryptosystem using error-correcting codes (1978).

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error-correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Based on the hardness of decoding random linear codes.

Problem 1 (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ and $w \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq w$ such that $He^T = y$.

Traditional Code-based Approach

McEliece: first cryptosystem using error-correcting codes (1978).

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error-correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Based on the hardness of decoding random linear codes.

Problem 1 (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ and $w \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq w$ such that $He^T = y$.

Important that the chosen code is unrecognizable.

Traditional Code-based Approach

McEliece: first cryptosystem using error-correcting codes (1978).

$[n, k]$ linear code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$.

w-error-correcting: exists decoding algorithm that corrects up to w errors occurred on a codeword.

Based on the hardness of decoding random linear codes.

Problem 1 (Computational Syndrome Decoding)

Given: $H \in \mathbb{F}_q^{(n-k)\times n}$, $y \in \mathbb{F}_q^{(n-k)}$ and $w \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq w$ such that $He^T = y$.

Important that the chosen code is unrecognizable.

$\rightarrow$ the Code Equivalence Problem.

Code Equivalence Notions

Problem 2 (Permutation Code Equivalence)

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *permutationally equivalent*, or $\mathfrak{C} \overset{\mathsf{PE}}{\sim} \mathfrak{C}'$, if there is a permutation $\pi \in S_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}$, i.e.

$$\mathfrak{C}' = \{\pi(x), \ \ x \in \mathfrak{C}\}.$$

Code Equivalence Notions

Problem 2 (Permutation Code Equivalence)

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *permutationally equivalent*, or $\mathfrak{C} \overset{\text{PE}}{\sim} \mathfrak{C}'$, if there is a permutation $\pi \in S_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}$, i.e.

$$\mathfrak{C}' = \{\pi(x), \;\; x \in \mathfrak{C}\}.$$

This notion can be extended using linear isometries.

Code Equivalence Notions

Problem 2 (Permutation Code Equivalence)

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *permutationally equivalent*, or $\mathfrak{C} \overset{\text{PE}}{\sim} \mathfrak{C}'$, if there is a permutation $\pi \in S_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}$, i.e.

$$\mathfrak{C}' = \{\pi(x),\ \ x \in \mathfrak{C}\}.$$

This notion can be extended using linear isometries.

Problem 3 (Linear Code Equivalence)

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *linearly equivalent*, or $\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}'$, if there is a linear isometry $\mu = (v, \pi) \in \mathbb{F}_q^{*n} \rtimes S_n$ such that $\mathfrak{C}' = \mu(\mathfrak{C})$, i.e.

$$\mathfrak{C}' = \{\mu(x),\ \ x \in \mathfrak{C}\}.$$

Code Equivalence Notions

Problem 2 (Permutation Code Equivalence)

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *permutationally equivalent*, or $\mathfrak{C} \overset{\text{PE}}{\sim} \mathfrak{C}'$, if there is a permutation $\pi \in S_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}$, i.e.

$$\mathfrak{C}' = \{\pi(x), \;\; x \in \mathfrak{C}\}.$$

This notion can be extended using linear isometries.

Problem 3 (Linear Code Equivalence)

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *linearly equivalent*, or $\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}'$, if there is a linear isometry $\mu = (v, \pi) \in \mathbb{F}_q^{*n} \rtimes S_n$ such that $\mathfrak{C}' = \mu(\mathfrak{C})$, i.e.

$$\mathfrak{C}' = \{\mu(x), \;\; x \in \mathfrak{C}\}.$$

We do not consider here the case of *semilinear* isometries.

Code equivalence can be described using generator matrices.
Clearly:

Code equivalence can be described using generator matrices. Clearly:

$$\mathfrak{C} \overset{\mathsf{PE}}{\sim} \mathfrak{C}' \iff \exists (S,P) \in \mathsf{GL}_k(q) \times S_n \text{ s.t. } G' = SGP,$$
$$\mathfrak{C} \overset{\mathsf{LE}}{\sim} \mathfrak{C}' \iff \exists (S,Q) \in \mathsf{GL}_k(q) \times M_n(q) \text{ s.t. } G' = SGQ,$$

where P is a permutation matrix, and Q a monomial matrix.

Code equivalence can be described using generator matrices. Clearly:

$$\mathfrak{C} \overset{\mathsf{PE}}{\sim} \mathfrak{C}' \iff \exists (S, P) \in \mathrm{GL}_k(q) \times S_n \text{ s.t. } G' = SGP,$$
$$\mathfrak{C} \overset{\mathsf{LE}}{\sim} \mathfrak{C}' \iff \exists (S, Q) \in \mathrm{GL}_k(q) \times M_n(q) \text{ s.t. } G' = SGQ,$$

where P is a permutation matrix, and Q a monomial matrix.

Can be seen as a group action of $\mathrm{GL}_k(q) \times M_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k \times n}$.

The Code Equivalence Problem

Code equivalence can be described using generator matrices. Clearly:

$$\mathfrak{C} \overset{\text{PE}}{\sim} \mathfrak{C}' \iff \exists (S, P) \in \mathrm{GL}_k(q) \times S_n \ \text{ s.t. } \ G' = SGP,$$
$$\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}' \iff \exists (S, Q) \in \mathrm{GL}_k(q) \times M_n(q) \ \text{ s.t. } \ G' = SGQ,$$

where P is a permutation matrix, and Q a monomial matrix.

Can be seen as a group action of $\mathrm{GL}_k(q) \times M_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k\times n}$.

Permutation (Linear) Code Equivalence Problem

Let $\mathfrak{C}$ and $\mathfrak{C}'$ be two $[n, k]$ linear codes over $\mathbb{F}_q$, having generator matrices G and G', respectively. Determine whether the two codes are permutationally (linearly) equivalent, i.e. if there exist matrices $S \in \mathrm{GL}$ and $P \in S_n$ ($Q \in M_n(q)$) such that $G' = SGP$ ($G' = SGQ$).

Hardness at a Glance

Studied for a very long time.

Hardness at a Glance

Studied for a very long time.

Unlikely to be NP-complete (unless polynomial hierarchy collapses).
(Petrank and Roth, 1997)

Hardness at a Glance

Studied for a very long time.

Unlikely to be NP-complete (unless polynomial hierarchy collapses).
(Petrank and Roth, 1997)

Existing algorithms efficiently attack particular cases, however...

HARDNESS AT A GLANCE

Studied for a very long time.

Unlikely to be NP-complete (unless polynomial hierarchy collapses).
(Petrank and Roth, 1997)

Existing algorithms efficiently attack particular cases, however...

...underlying exponential complexity makes it easy to find intractable instances.

Part III

LESS and Variants

LESS ZK Identification Scheme

Key Generation

- Choose linear code $\mathfrak{C}$ with generator matrix G.
- SK: invertible matrix S and monomial matrix Q.
- PK: matrix $G' = SGQ$ (can be systematic form).

Prover's Computation

- Choose random monomial matrix $\tilde{Q}$.
- Set $\tilde{G} = SystForm(G\tilde{Q})$ and $h = Hash(\tilde{G})$.
  (After receiving challenge bit b).
- If $b = 0$ respond with $\mu = \tilde{Q}$.
- If $b = 1$ respond with $\mu = Q^{-1}\tilde{Q}$.

Verifier's Computation

- If $b = 0$ verify that $Hash(SystForm(G\mu)) = h$.
- If $b = 1$ verify that $Hash(SystForm(G'\mu)) = h$.

LESS ZK Identification Scheme

Key Generation

- Choose linear code $\mathfrak{C}$ with generator matrix G.
- SK: invertible matrix S and monomial matrix Q.
- PK: matrix $G' = SGQ$ (can be systematic form).

Prover's Computation

- Choose random monomial matrix $\tilde{Q}$.
- Set $\tilde{G} = SystForm(G\tilde{Q})$ and $h = Hash(\tilde{G})$.
  (After receiving challenge bit b).
- If $b = 0$ respond with $\mu = \tilde{Q}$.
- If $b = 1$ respond with $\mu = Q^{-1}\tilde{Q}$.

Verifier's Computation

- If $b = 0$ verify that $Hash(SystForm(G\mu)) = h$.
- If $b = 1$ verify that $Hash(SystForm(G'\mu)) = h$.

Repeat t rounds and convert to signature using Fiat-Shamir.
EUF-CMA proof using Forking Lemma.

LESS-M: Multi-bit Challenges

Idea: use multiple public keys and use challenge(s) to pick one.

LESS-M: Multi-bit Challenges

Idea: use multiple public keys and use challenge(s) to pick one.

Key Generation

- SK: monomial matrices $Q_0 \dots Q_{r-1}$.
- PK: generator matrices $G_0 \dots G_{r-1}$ where $G_i = SystForm(GQ_i)$.

LESS-M: Multi-bit Challenges

Idea: use multiple public keys and use challenge(s) to pick one.

Key Generation

- SK: monomial matrices $Q_0 \dots Q_{r-1}$.
- PK: generator matrices $G_0 \dots G_{r-1}$ where $G_i = \mathit{SystForm}(GQ_i)$.

Signer

- Choose random monomial matrices $\tilde{Q}_j$.
- Set $\tilde{G}_j = \mathit{SystForm}(G\tilde{Q}_j)$ and $h = \mathit{Hash}(\tilde{G}_0, \dots, \tilde{G}_{t-1}, m)$.
- Parse h as challenge vector with $h_j \in \mathbb{Z}_2^\ell$.
- Signature $\sigma = (\mu_0, \dots, \mu_{t-1}, h)$ with $\mu_j = Q_{h_j}^{-1}\tilde{Q}_j$.

LESS-M: Multi-bit Challenges

Idea: use multiple public keys and use challenge(s) to pick one.

Key Generation

- SK: monomial matrices $Q_0 \dots Q_{r-1}$.
- PK: generator matrices $G_0 \dots G_{r-1}$ where $G_i = \mathit{SystForm}(GQ_i)$.

Signer

- Choose random monomial matrices $\tilde{Q}_j$.
- Set $\tilde{G}_j = \mathit{SystForm}(G\tilde{Q}_j)$ and $h = \mathit{Hash}(\tilde{G}_0, \dots, \tilde{G}_{t-1}, m)$.
- Parse h as challenge vector with $h_j \in \mathbb{Z}_2^\ell$.
- Signature $\sigma = (\mu_0, \dots, \mu_{t-1}, h)$ with $\mu_j = Q_{h_j}^{-1}\tilde{Q}_j$.

Verifier

- Set $\hat{G}_j = \mathit{SystForm}(G_{h_j}\mu_j)$.
- Accept if $\mathit{Hash}(\hat{G}_0, \dots, \hat{G}_{t-1}, m) = h$.

About the -M Variant

Security proof based on a variant of the Code Equivalence problem.

Security proof based on a variant of the Code Equivalence problem.

Multiple Codes Linear Equivalence Problem

Consider linearly equivalent $[n, k]$-linear codes $\mathfrak{C}_0 \dots \mathfrak{C}_{r-1}$, with generator matrices $G_0, \dots, G_{r-1}$ of the form $S_0 G Q_0, \dots, S_{r-1} G Q_{r-1}$. Find matrices $S^* \in \mathrm{GL}$ and $Q^* \in M_n(q)$ such that $G_{j'} = S^* G_j Q^*$, for some $j \neq j'$.

About the -M Variant

Security proof based on a variant of the Code Equivalence problem.

Multiple Codes Linear Equivalence Problem

Consider linearly equivalent $[n, k]$-linear codes $\mathfrak{C}_0 \dots \mathfrak{C}_{r-1}$, with generator matrices $G_0, \dots, G_{r-1}$ of the form $S_0 G Q_0, \dots, S_{r-1} G Q_{r-1}$. Find matrices $S^* \in \mathrm{GL}$ and $Q^* \in M_n(q)$ such that $G_{j'} = S^* G_j Q^*$, for some $j \neq j'$.

MCLE reduces to CE tightly with a simple probabilistic argument.

About the -M Variant

Security proof based on a variant of the Code Equivalence problem.

Multiple Codes Linear Equivalence Problem

Consider linearly equivalent $[n, k]$-linear codes $\mathfrak{C}_0 \dots \mathfrak{C}_{r-1}$, with generator matrices $G_0, \dots, G_{r-1}$ of the form $S_0 G Q_0, \dots, S_{r-1} G Q_{r-1}$. Find matrices $S^* \in \mathrm{GL}$ and $Q^* \in M_n(q)$ such that $G_{j'} = S^* G_j Q^*$, for some $j \neq j'$.

MCLE reduces to CE tightly with a simple probabilistic argument.

Advantage: multiple challenge bits require fewer rounds.

About the -M Variant

Security proof based on a variant of the Code Equivalence problem.

Multiple Codes Linear Equivalence Problem

Consider linearly equivalent $[n, k]$-linear codes $\mathfrak{C}_0 \dots \mathfrak{C}_{r-1}$, with generator matrices $G_0, \dots, G_{r-1}$ of the form $S_0 G Q_0, \dots, S_{r-1} G Q_{r-1}$. Find matrices $S^* \in \mathrm{GL}$ and $Q^* \in M_n(q)$ such that $G_{j'} = S^* G_j Q^*$, for some $j \neq j'$.

MCLE reduces to CE tightly with a simple probabilistic argument.

Advantage: multiple challenge bits require fewer rounds.

Tradeoff between public key and signature size.

LESS-F: Fixed-weight Challenges

Idea: use fixed-weight challenge vector to reduce transmission cost.

LESS-F: Fixed-weight Challenges

Idea: use fixed-weight challenge vector to reduce transmission cost.

Key Generation

- Call $G_0 = \mathit{SystForm}(G)$ and $Q_0 = I_n$.
- SK: monomial matrix Q_1.
- PK: generator matrix $G_1 = \mathit{SystForm}(GQ_1)$.

LESS-F: Fixed-weight Challenges

Idea: use fixed-weight challenge vector to reduce transmission cost.

Key Generation

- Call $G_0 = SystForm(G)$ and $Q_0 = I_n$.
- SK: monomial matrix Q_1.
- PK: generator matrix $G_1 = SystForm(GQ_1)$.

Signer

- Choose random monomial matrices $\tilde{Q}_j$.
- Set $\tilde{G}_j = SystForm(G\tilde{Q}_j)$ and $h = Hash(\tilde{G}_0, \ldots, \tilde{G}_{t-1}, m)$. (Here *Hash* is a weight-restricted hash function).
- Parse h as challenge vector with $h_j \in \mathbb{Z}_2$.
- Signature $\sigma = (\mu_0, \ldots, \mu_{t-1}, h)$ with $\mu_j = Q_{h_j}^{-1}\tilde{Q}_j$.

LESS-F: Fixed-weight Challenges

Idea: use fixed-weight challenge vector to reduce transmission cost.

Key Generation

- Call $G_0 = SystForm(G)$ and $Q_0 = I_n$.
- SK: monomial matrix Q_1.
- PK: generator matrix $G_1 = SystForm(GQ_1)$.

Signer

- Choose random monomial matrices $\tilde{Q}_j$.
- Set $\tilde{G}_j = SystForm(G\tilde{Q}_j)$ and $h = Hash(\tilde{G}_0, \dots, \tilde{G}_{t-1}, m)$. (Here *Hash* is a weight-restricted hash function).
- Parse h as challenge vector with $h_j \in \mathbb{Z}_2$.
- Signature $\sigma = (\mu_0, \dots, \mu_{t-1}, h)$ with $\mu_j = Q_{h_j}^{-1}\tilde{Q}_j$.

Verifier

- Set $\hat{G}_j = SystForm(G_{h_j}\mu_j)$.
- Accept if $Hash(\hat{G}_0, \dots, \hat{G}_{t-1}, m) = h$.

About the -F Variant

Idea not new in literature, goes back to original Fiat-Shamir work.

About the -F Variant

Idea not new in literature, goes back to original Fiat-Shamir work.

Security proof as original, with small adjustment in Forking Lemma.

About the -F Variant

Idea not new in literature, goes back to original Fiat-Shamir work.

Security proof as original, with small adjustment in Forking Lemma.

Switch challenge set from $\{0,1\}^t$ to set of fixed-weight strings $\mathbb{Z}^t_{2,\omega}$.

About the -F Variant

Idea not new in literature, goes back to original Fiat-Shamir work.

Security proof as original, with small adjustment in Forking Lemma.

Switch challenge set from $\{0,1\}^t$ to set of fixed-weight strings $\mathbb{Z}^t_{2,\omega}$.

While normally $t = \lambda$, we now choose $\log_2 \binom{t}{\omega} \geq \lambda$.

About the -F Variant

Idea not new in literature, goes back to original Fiat-Shamir work.

Security proof as original, with small adjustment in Forking Lemma.

Switch challenge set from $\{0,1\}^t$ to set of fixed-weight strings $\mathbb{Z}^t_{2,\omega}$.

While normally $t = \lambda$, we now choose $\log_2 \binom{t}{\omega} \geq \lambda$.

Advantage: 0 challenge bits correspond to random monomial matrix, can send using PRNG seed.

About the -F Variant

Idea not new in literature, goes back to original Fiat-Shamir work.

Security proof as original, with small adjustment in Forking Lemma.

Switch challenge set from $\{0,1\}^t$ to set of fixed-weight strings $\mathbb{Z}^t_{2,\omega}$.

While normally $t = \lambda$, we now choose $\log_2 \binom{t}{\omega} \geq \lambda$.

Advantage: 0 challenge bits correspond to random monomial matrix, can send using PRNG seed.

Considerably reduce signature size.

Previous techniques can be combined (proofs modified accordingly).

LESS-FM: Fine-tuning Signatures

Previous techniques can be combined (proofs modified accordingly).

Use multiple keys, then choose challenge string of elements of $\mathbb{Z}_2^\ell$.

LESS-FM: Fine-tuning Signatures

Previous techniques can be combined (proofs modified accordingly).

Use multiple keys, then choose challenge string of elements of $\mathbb{Z}_2^\ell$.

Tune parameters such that $\left|\mathbb{Z}^t_{2^\ell,\omega}\right| = \binom{t}{\omega}(2^\ell - 1)^\omega \geq 2^\lambda$.

LESS-FM: Fine-tuning Signatures

Previous techniques can be combined (proofs modified accordingly).

Use multiple keys, then choose challenge string of elements of $\mathbb{Z}_2^\ell$.

Tune parameters such that $\left|\mathbb{Z}^t_{2^\ell,\omega}\right| = \binom{t}{\omega}(2^\ell - 1)^\omega \geq 2^\lambda$.

Allows for great flexibility in signature design.

LESS-FM: Fine-tuning Signatures

Previous techniques can be combined (proofs modified accordingly).

Use multiple keys, then choose challenge string of elements of $\mathbb{Z}_2^\ell$.

Tune parameters such that $\left|\mathbb{Z}^t_{2^\ell,\omega}\right| = \binom{t}{\omega}(2^\ell - 1)^\omega \geq 2^\lambda$.

Allows for great flexibility in signature design.

Code parameters recalibrated after latest developments on code equivalence.
(Beullens, 2020)

LESS-FM: Fine-tuning Signatures

Previous techniques can be combined (proofs modified accordingly).

Use multiple keys, then choose challenge string of elements of $\mathbb{Z}_2^\ell$.

Tune parameters such that $\left|\mathbb{Z}^t_{2^\ell,\omega}\right| = \binom{t}{\omega}(2^\ell - 1)^\omega \geq 2^\lambda$.

Allows for great flexibility in signature design.

Code parameters recalibrated after latest developments on code equivalence.
(Beullens, 2020)

Full security analysis in extended version of this work.
(ePrint 2021/396)

Part IV

Performance

Parameters

Parameters for $\lambda = 128$ security bits, optimized for various scenarios.

PARAMETERS

Parameters for $\lambda = 128$ security bits, optimized for various scenarios.

LESS parameters for 128 bits of security.

Criterion	Type	n	k	q	ℓ	t	ω	Pk (kB)	Sig (kB)
Min Pk	F - MONO	198	94	251	1	283	28	9.77	15.2
Min Sig	FM - PERM	305	127	31	4	66	19	205.74	5.25
Min Pk + Sig	F - PERM	280	117	149	1	233	31	11.57	10.39

Parameters for $\lambda = 128$ security bits, optimized for various scenarios.

LESS parameters for 128 bits of security.

Criterion	Type	n	k	q	ℓ	t	ω	Pk (kB)	Sig (kB)
Min Pk	F - MONO	198	94	251	1	283	28	9.77	15.2
Min Sig	FM - PERM	305	127	31	4	66	19	205.74	5.25
Min Pk + Sig	F - PERM	280	117	149	1	233	31	11.57	10.39

Can push public key below 10 kB, or signature to about 5 kB, or their sum below 22 kB.

Parameters

Parameters for $\lambda = 128$ security bits, optimized for various scenarios.

LESS parameters for 128 bits of security.

Criterion	Type	n	k	q	ℓ	t	ω	Pk (kB)	Sig (kB)
Min Pk	F - MONO	198	94	251	1	283	28	9.77	15.2
Min Sig	FM - PERM	305	127	31	4	66	19	205.74	5.25
Min Pk + Sig	F - PERM	280	117	149	1	233	31	11.57	10.39

Can push public key below 10 kB, or signature to about 5 kB, or their sum below 22 kB.

This compares favorably with other ZK-based code-based signatures.

Parameters

Parameters for $\lambda = 128$ security bits, optimized for various scenarios.

LESS parameters for 128 bits of security.

Criterion	Type	n	k	q	ℓ	t	ω	Pk (kB)	Sig (kB)
Min Pk	F - MONO	198	94	251	1	283	28	9.77	15.2
Min Sig	FM - PERM	305	127	31	4	66	19	205.74	5.25
Min Pk + Sig	F - PERM	280	117	149	1	233	31	11.57	10.39

Can push public key below 10 kB, or signature to about 5 kB, or their sum below 22 kB.

This compares favorably with other ZK-based code-based signatures.

Wave (trapdoor-based) has very large public key (3.2 MB) and short signature (1 kB).

PARAMETERS

Parameters for $\lambda = 128$ security bits, optimized for various scenarios.

LESS parameters for 128 bits of security.

Criterion	Type	n	k	q	ℓ	t	ω	Pk (kB)	Sig (kB)
Min Pk	F - MONO	198	94	251	1	283	28	9.77	15.2
Min Sig	FM - PERM	305	127	31	4	66	19	205.74	5.25
Min Pk + Sig	F - PERM	280	117	149	1	233	31	11.57	10.39

Can push public key below 10 kB, or signature to about 5 kB, or their sum below 22 kB.

This compares favorably with other ZK-based code-based signatures.

Wave (trapdoor-based) has very large public key (3.2 MB) and short signature (1 kB).

Same ballpark as Durandal (rank-based), Pk + Sig between 19 kB and 24 kB.

CONCLUSIONS

LESS is the first code-based signature not based directly on syndrome decoding.

Conclusions

LESS is the first code-based signature not based directly on syndrome decoding.

We have provided variants aimed at optimizing performance.

Conclusions

LESS is the first code-based signature not based directly on syndrome decoding.

We have provided variants aimed at optimizing performance.

The result is a flexible and practical scheme, suitable for various scenarios.

Conclusions

LESS is the first code-based signature not based directly on syndrome decoding.

We have provided variants aimed at optimizing performance.

The result is a flexible and practical scheme, suitable for various scenarios.

Follow-up work is currently underway (e.g. implementation).

Thank you!