Code-based Signatures: New Approaches and Research Directions

Edoardo Persichetti

1 February 2022

Charles E. Schmidt College of Science
Florida Atlantic University

IN THIS TALK

- Code-based Cryptography
- Traditional Approach: Hash-and-sign
- Stern's Scheme and Zero-Knowledge
- Code Equivalence and LESS

Part I

CODE-BASED CRYPTOGRAPHY

Error-Correcting Codes

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

A subspace of dimension k of $\mathbb{F}_q^n$. Value n is called length.

Error-Correcting Codes

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

A subspace of dimension k of $\mathbb{F}_q^n$. Value n is called length.

Hamming metric

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.

Minimum distance (of $\mathcal{C}$): $\min\{d(x, y) : x, y \in \mathcal{C}\}$.

Error-Correcting Codes

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

A subspace of dimension k of $\mathbb{F}_q^n$. Value n is called length.

Hamming metric

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.

Minimum distance (of $\mathcal{C}$): $\min\{d(x, y) : x, y \in \mathcal{C}\}$.

Generator matrix

$G \in \mathbb{F}_q^{k \times n}$ defines the code as : $x \in \mathcal{C} \iff x = uG$ for $u \in \mathbb{F}_q^k$.
Not unique: $SG, S \in GL(k, q)$; Systematic form: $(I_k|M)$.

Error-Correcting Codes

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

A subspace of dimension k of $\mathbb{F}_q^n$. Value n is called length.

Hamming metric

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.

Minimum distance (of $\mathcal{C}$): $\min\{d(x, y) : x, y \in \mathcal{C}\}$.

Generator matrix

$G \in \mathbb{F}_q^{k \times n}$ defines the code as : $x \in \mathcal{C} \iff x = uG$ for $u \in \mathbb{F}_q^k$.
Not unique: $SG, S \in GL(k, q)$; Systematic form: $(I_k|M)$.

Parity-check matrix

$H \in \mathbb{F}_q^{(n-k) \times n}$ defines the code as: $x \in \mathcal{C} \iff Hx^T = 0$ (syndrome).
Not unique: $SH, S \in GL(n - k, q)$; Systematic form: $(M^T|I_{n-k})$.

Error-Correcting Codes

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

A subspace of dimension k of $\mathbb{F}_q^n$. Value n is called length.

Hamming metric

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.
Minimum distance (of $\mathcal{C}$): $\min\{d(x, y) : x, y \in \mathcal{C}\}$.

Generator matrix

$G \in \mathbb{F}_q^{k\times n}$ defines the code as : $x \in \mathcal{C} \iff x = uG$ for $u \in \mathbb{F}_q^k$.
Not unique: $SG, S \in GL(k, q)$; Systematic form: $(I_k|M)$.

Parity-check matrix

$H \in \mathbb{F}_q^{(n-k)\times n}$ defines the code as: $x \in \mathcal{C} \iff Hx^T = 0$ (syndrome).
Not unique: $SH, S \in GL(n - k, q)$; Systematic form: $(M^T|I_{n-k})$.

t-error correcting: $\exists$ algorithm that corrects up to t errors.

Example: Goppa Codes

Select $g(X) \in \mathbb{F}_{q^m}[X]$ and non-zero $\alpha_1, \dots, \alpha_n \in \mathbb{F}_{q^m}$ with $g(\alpha_i) \neq 0$.

Parity-check given by $H = \{H_{ij}\} = \{\alpha_j^{i-1}/g(\alpha_j)\}$. Codewords over $\mathbb{F}_q$.

Let noisy codeword be $y = x + e$, $x \in \mathcal{C}, wt(e) \leq t$.

For Goppa codes, $t = r/2$ (or $t = r$ if binary), where $r = deg(g)$.

Example: Goppa Codes

Select $g(X) \in \mathbb{F}_{q^m}[X]$ and non-zero $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^m}$ with $g(\alpha_i) \neq 0$.

Parity-check given by $H = \{H_{ij}\} = \{\alpha_j^{i-1}/g(\alpha_j)\}$. Codewords over $\mathbb{F}_q$.

Let noisy codeword be $y = x + e$, $x \in \mathcal{C}, wt(e) \leq t$.

For Goppa codes, $t = r/2$ (or $t = r$ if binary), where $r = deg(g)$.

To decode:

1. Compute syndrome $s = Hy^T = (s_0, \ldots, s_{r-1})$.
2. Obtain *error locator* poly $\sigma(X)$ and *error evaluator* poly $\omega(X)$ by solving *key equation*
$$\frac{\omega(X)}{\sigma(X)} \equiv s(X) \mod X^r.$$
3. Find roots; error positions are reciprocals (values from $\omega(X)$).

In general, it is hard to decode random codes.

Decoding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ and $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq t$ such that $y - e = x \in \mathcal{C}_G$.

Decoding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k\times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e = x \in \mathcal{C}_G$.

Easy to see this is equivalent to the following.

Decoding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ and $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq t$ such that $y - e = x \in \mathcal{C}_G$.

Easy to see this is equivalent to the following.

Problem (Syndrome Decoding)

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

Decoding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k\times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e = x \in \mathcal{C}_G$.

Easy to see this is equivalent to the following.

Problem (Syndrome Decoding)

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

NP-Complete (Berlekamp, McEliece and Van Tilborg, 1978; Barg, 1994).

Decoding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e = x \in \mathcal{C}_G$.

Easy to see this is equivalent to the following.

Problem (Syndrome Decoding)

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

NP-Complete (Berlekamp, McEliece and Van Tilborg, 1978; Barg, 1994).
Unique solution when t is below a certain threshold.

Decoding Problems

In general, it is hard to decode random codes.

Problem (General Decoding)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ *and* $t \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ *with* $wt(e) \leq t$ *such that* $y - e = x \in \mathcal{C}_G$.

Easy to see this is equivalent to the following.

Problem (Syndrome Decoding)

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

NP-Complete (Berlekamp, McEliece and Van Tilborg, 1978; Barg, 1994).
Unique solution when t is below a certain threshold.

Gilbert-Varshamov (GV) Bound

The largest integer d_0 such that

$$|\mathcal{B}(0, d_0 - 1)| \leq q^{n-k}.$$

What is Code-Based Cryptography?

The family of primitives based on hard problems from coding theory.

What is Code-Based Cryptography?

The family of primitives based on hard problems from coding theory.

To get trapdoor, need one more ingredient.

What is Code-Based Cryptography?

The family of primitives based on hard problems from coding theory.

To get trapdoor, need one more ingredient.

Assumption (Code Indistinguishability)

Let M be a matrix defining a code. Then M is indistinguishable from a randomly generated matrix of the same size.

What is Code-Based Cryptography?

The family of primitives based on hard problems from coding theory.

To get trapdoor, need one more ingredient.

Assumption (Code Indistinguishability)

Let M be a matrix defining a code. Then M is indistinguishable from a randomly generated matrix of the same size.

Hardness of assumption depends on chosen code family.

What is Code-Based Cryptography?

The family of primitives based on hard problems from coding theory.

To get trapdoor, need one more ingredient.

Assumption (Code Indistinguishability)

Let M be a matrix defining a code. Then M is indistinguishable from a randomly generated matrix of the same size.

Hardness of assumption depends on chosen code family.

Choose a code family with efficient decoding algorithm associated to description Δ and hide the structure.

Code-Based Cryptosystems

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Plaintext is encrypted as noisy codeword (scheme is probabilistic).

McEliece: first proposal (1978), based on GDP.

Chosen code family: binary Goppa codes.

KeyGen chooses generator matrix G and forms public key as SGP.

Plaintext is encrypted as noisy codeword (scheme is probabilistic).

Niederreiter: “dual”/equivalent version (1985), based on SDP.

Chosen code family: Generalized Reed-Solomon (GRS) codes.

KeyGen chooses parity-check matrix H and forms public key as SHP.

Plaintext is encrypted as syndrome (scheme is deterministic).

Part II

Traditional Approach: Hash-and-sign

Intuition

Rely directly on SDP.

Intuition

Rely directly on SDP.

Use hash-and-sign framework as in e.g. Full Domain Hash (RSA).

Intuition

Rely directly on SDP.

Use hash-and-sign framework as in e.g. Full Domain Hash (RSA).

Given message μ, trapdoor OW function *f* and hash function **H**.

Intuition

Rely directly on SDP.

Use hash-and-sign framework as in e.g. Full Domain Hash (RSA).

Given message μ, trapdoor OW function *f* and hash function **H**.

Create signature $\sigma = f^{-1}(\mathbf{H}(\mu))$.

INTUITION

Rely directly on SDP.

Use hash-and-sign framework as in e.g. Full Domain Hash (RSA).

Given message μ, trapdoor OW function f and hash function $\mathbf{H}$.

Create signature $\sigma = f^{-1}(\mathbf{H}(\mu))$.

Verify signature if $f(\sigma) = \mathbf{H}(\mu)$.

Intuition

Rely directly on SDP.

Use hash-and-sign framework as in e.g. Full Domain Hash (RSA).

Given message μ, trapdoor OW function f and hash function $\mathbf{H}$.

Create signature $\sigma = f^{-1}(\mathbf{H}(\mu))$.

Verify signature if $f(\sigma) = \mathbf{H}(\mu)$.

For code-based, trapdoor is decoding: CFS scheme
(Courtois, Finiasz, Sendrier, 2001).

THE CFS SCHEME

Select hash function $\mathbf{H} : \{0,1\}^* \rightarrow \mathbb{F}_q^{(n-k)}$.

The CFS Scheme

Select hash function $\mathbf{H} : \{0,1\}^* \rightarrow \mathbb{F}_q^{(n-k)}$.

Key Generation

- Choose a code $\mathcal{C}$ (e.g. Goppa).
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

The CFS Scheme

Select hash function $\mathbf{H} : \{0,1\}^* \to \mathbb{F}_q^{(n-k)}$.

Key Generation

- Choose a code $\mathcal{C}$ (e.g. Goppa).
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Sign

- Compute $y = \mathbf{H}(\mu)$.
- Set $e = \mathit{Decode}_\Delta(y)$, with $e \in \mathbb{F}_q^n$ of weight t.
- Signature is $\sigma = e$.

The CFS Scheme

Select hash function $\mathbf{H} : \{0,1\}^* \to \mathbb{F}_q^{(n-k)}$.

Key Generation

- Choose a code $\mathcal{C}$ (e.g. Goppa).
- SK: code description Δ for $\mathcal{C}$.
- PK: parity-check matrix H in systematic form for $\mathcal{C}$.

Sign

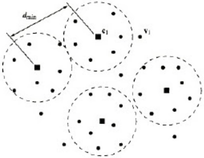
- Compute $y = \mathbf{H}(\mu)$.
- Set $e = Decode_\Delta(y)$, with $e \in \mathbb{F}_q^n$ of weight t.
- Signature is $\sigma = e$.

Verify

- Compute $y' = H\sigma^T$.
- Accept if $y' = \mathbf{H}(\mu)$, otherwise reject.

About CFS

Decoding a random word is not always possible!



About CFS

Decoding a random word is not always possible!

Solution: try again.

About CFS

Decoding a random word is not always possible!

Solution: try again.

Add counter c to the input of $\mathbf{H}$ and try to decode $y = \mathbf{H}(\mu, c)$. Send c along with signature.

About CFS

Decoding a random word is not always possible!

Solution: try again.

Add counter c to the input of $\mathbf{H}$ and try to decode $y = \mathbf{H}(\mu, c)$. Send c along with signature.

With McEliece parameters $(1024, 524, 50)$ average of 2^{216} decoding attempts.

About CFS

Decoding a random word is not always possible!

Solution: try again.

Add counter c to the input of $\mathbf{H}$ and try to decode $y = \mathbf{H}(\mu, c)$. Send c along with signature.

With McEliece parameters $(1024, 524, 50)$ average of 2^{216} decoding attempts.

Adjust parameters: choose (m, t), then $(q = 2, n = 2^m, k = n - mt)$ can correct t errors.

About CFS

Decoding a random word is not always possible!

Solution: try again.

Add counter c to the input of $\mathbf{H}$ and try to decode $y = \mathbf{H}(\mu, c)$. Send c along with signature.

With McEliece parameters $(1024, 524, 50)$ average of 2^{216} decoding attempts.

Adjust parameters: choose (m, t), then $(q = 2, n = 2^m, k = n - mt)$ can correct t errors.

E.g. $(16, 9)$ leads to $(2^{16}, 2^{16} - 144, 9)$ code: $9!$ decoding attempts.

Decoding a random word is not always possible!

Solution: try again.

Add counter c to the input of $\mathbf{H}$ and try to decode $y = \mathbf{H}(\mu, c)$. Send c along with signature.

With McEliece parameters $(1024, 524, 50)$ average of 2^{216} decoding attempts.

Adjust parameters: choose (m, t), then $(q = 2, n = 2^m, k = n - mt)$ can correct t errors.

E.g. $(16, 9)$ leads to $(2^{16}, 2^{16} - 144, 9)$ code: 9! decoding attempts.

CFS parameters:

q	m	n	t	PK Size (KB)	Sig Size (bits)	Security
2	16	65536	9	1152	144	80

CONSIDERATIONS

Signing is very slow: in the order of seconds.

Considerations

Signing is very slow: in the order of seconds.

Public-key size can be reduced (e.g. using Quasi-Dyadic codes) (Barreto, Cayrel, Misoczki, Niebuhr, 2010), but subject to algebraic attacks.

Considerations

Signing is very slow: in the order of seconds.

Public-key size can be reduced (e.g. using Quasi-Dyadic codes) (Barreto, Cayrel, Misoczki, Niebuhr, 2010), but subject to algebraic attacks.

Additional security concerns: very high rate leads to distinguishers (Faugère Gauthier-Umana, Otmani, Perret, Tillich, 2013).

Considerations

Signing is very slow: in the order of seconds.

Public-key size can be reduced (e.g. using Quasi-Dyadic codes) (Barreto, Cayrel, Misoczki, Niebuhr, 2010), but subject to algebraic attacks.

Additional security concerns: very high rate leads to distinguishers (Faugère Gauthier-Umana, Otmani, Perret, Tillich, 2013).

Generalized Birthday Attack by Bleichenbacher (2009) invalidates previous parameters: minimum $(m, t) = (15, 12)$ for 80 sec bits.

CONSIDERATIONS

Signing is very slow: in the order of seconds.

Public-key size can be reduced (e.g. using Quasi-Dyadic codes) (Barreto, Cayrel, Misoczki, Niebuhr, 2010), but subject to algebraic attacks.

Additional security concerns: very high rate leads to distinguishers (Faugère Gauthier-Umana, Otmani, Perret, Tillich, 2013).

Generalized Birthday Attack by Bleichenbacher (2009) invalidates previous parameters: minimum $(m, t) = (15, 12)$ for 80 sec bits.

Improved repetition approach: Parallel CFS. Sign multiple hashes (e.g. 2) partially mitigates attack at the cost of increased (e.g. double) signature size and signing time (Finiasz, 2010).

Considerations

Signing is very slow: in the order of seconds.

Public-key size can be reduced (e.g. using Quasi-Dyadic codes) (Barreto, Cayrel, Misoczki, Niebuhr, 2010), but subject to algebraic attacks.

Additional security concerns: very high rate leads to distinguishers (Faugère Gauthier-Umana, Otmani, Perret, Tillich, 2013).

Generalized Birthday Attack by Bleichenbacher (2009) invalidates previous parameters: minimum $(m, t) = (15, 12)$ for 80 sec bits.

Improved repetition approach: Parallel CFS. Sign multiple hashes (e.g. 2) partially mitigates attack at the cost of increased (e.g. double) signature size and signing time (Finiasz, 2010).

Still completely impractical: implementations show GB of public key, and several seconds to sign.
(Landais, Sendrier, 2012; Bernstein, Chou, Schwabe, 2013).

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

Solution is not unique, but that is ok.

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

Solution is not unique, but that is ok.

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

Solution is not unique, but that is ok.

Can use CFS approach if we could guarantee syndrome of high-weight vector.

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

Solution is not unique, but that is ok.

Can use CFS approach if we could guarantee syndrome of high-weight vector.

Decoding algorithm doesn't work anymore.

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

Solution is not unique, but that is ok.

Can use CFS approach if we could guarantee syndrome of high-weight vector.

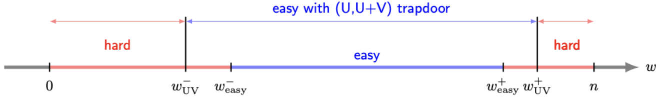
Decoding algorithm doesn't work anymore.

With $(U \mid U+V)$ codes can use generic decoding algorithm (e.g. ISD, Prange) with advantage.

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

Solution is not unique, but that is ok.



Can use CFS approach if we could guarantee syndrome of high-weight vector.

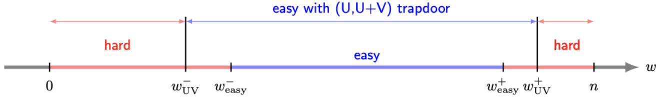
Decoding algorithm doesn't work anymore.

With $(U \mid U+V)$ codes can use generic decoding algorithm (e.g. ISD, Prange) with advantage.

High-Weight Decoding

SDP is hard when weight is low (obvious), but also when very high!

Solution is not unique, but that is ok.



Can use CFS approach if we could guarantee syndrome of high-weight vector.

Decoding algorithm doesn't work anymore.

With $(U \mid U + V)$ codes can use generic decoding algorithm (e.g. ISD, Prange) with advantage.

"Manifacture" high-weight syndrome: rejection sampling.

Surf and Wave

SURF scheme uses binary $(U \mid U+V)$ codes

(Debris-Alazard, Sendrier, Tillich, 2018).

Surf and Wave

SURF scheme uses binary $(U \mid U + V)$ codes
(Debris-Alazard, Sendrier, Tillich, 2018).

Broken by authors (!) before publication: there exists distinguisher.

Surf and Wave

SURF scheme uses binary $(U \mid U+V)$ codes
(Debris-Alazard, Sendrier, Tillich, 2018).

Broken by authors (!) before publication: there exists distinguisher.

New variant WAVE using ternary, generalized $(U \mid U+V)$ codes.

SURF AND WAVE

SURF scheme uses binary $(U \mid U + V)$ codes
(Debris-Alazard, Sendrier, Tillich, 2018).

Broken by authors (!) before publication: there exists distinguisher.

New variant WAVE using ternary, generalized $(U \mid U + V)$ codes.

Initial concerns about information leakage, later addressed
(Barreto, P., 2019).

SURF AND WAVE

SURF scheme uses binary $(U \mid U + V)$ codes
(Debris-Alazard, Sendrier, Tillich, 2018).

Broken by authors (!) before publication: there exists distinguisher.

New variant WAVE using ternary, generalized $(U \mid U + V)$ codes.

Initial concerns about information leakage, later addressed
(Barreto, P., 2019).

Great improvement over CFS, but still shows many similar features (large key, small signature, slow signing etc.).

SURF AND WAVE

SURF scheme uses binary $(U \mid U + V)$ codes
(Debris-Alazard, Sendrier, Tillich, 2018).

Broken by authors (!) before publication: there exists distinguisher.

New variant WAVE using ternary, generalized $(U \mid U + V)$ codes.

Initial concerns about information leakage, later addressed
(Barreto, P., 2019).

Great improvement over CFS, but still shows many similar features (large key, small signature, slow signing etc.).

Wave parameters:

q	n	k_u	k_v	t	PK (MB)	Sig (kB)	Security
3	8492	3558	2047	7980	3.2	1.6	128

Part III

STERN'S SCHEME AND ZERO-KNOWLEDGE

Intuition

Avoid decoding: rely indirectly on SDP.

Intuition

Avoid decoding: rely indirectly on SDP.

Use zero-knowledge framework to build identification protocol.

INTUITION

Avoid decoding: rely indirectly on SDP.

Use zero-knowledge framework to build identification protocol.

PROVER (SK)		VERIFIER (PK)
Commit to random data	$\xrightarrow{\textit{Com}}$	
	$\xleftarrow{\textit{Ch}}$	Select random challenge
Produce response	$\xrightarrow{\textit{Rsp}}$	
		Accept/Reject

Avoid decoding: rely indirectly on SDP.

Use zero-knowledge framework to build identification protocol.

PROVER (SK)		VERIFIER (PK)
Commit to random data	$\xrightarrow{\quad \textit{Com} \quad}$	
	$\xleftarrow{\quad \textit{Ch} \quad}$	Select random challenge
Produce response	$\xrightarrow{\quad \textit{Rsp} \quad}$	
		Accept/Reject

Signature scheme can be obtained using Fiat-Shamir transformation.

Intuition

Avoid decoding: rely indirectly on SDP.

Use zero-knowledge framework to build identification protocol.

Prover (SK)		**Verifier (PK)**
Commit to random data	$\xrightarrow{\quad Com \quad}$	
	$\xleftarrow{\quad Ch \quad}$	Select random challenge
Produce response	$\xrightarrow{\quad Rsp \quad}$	
		Accept/Reject

Signature scheme can be obtained using Fiat-Shamir transformation.

Replace random challenge with $\mathbf{H}(Com, \mu)$, obtain signature as $\sigma = (Com, Rsp)$.

Intuition

Avoid decoding: rely indirectly on SDP.

Use zero-knowledge framework to build identification protocol.

PROVER (SK)		VERIFIER (PK)
Commit to random data	$\xrightarrow{\textit{Com}}$	
	$\xleftarrow{\textit{Ch}}$	Select random challenge
Produce response	$\xrightarrow{\textit{Rsp}}$	
		Accept/Reject

Signature scheme can be obtained using Fiat-Shamir transformation.

Replace random challenge with $\mathbf{H}(\textit{Com}, \mu)$, obtain signature as $\sigma = (\textit{Com}, \textit{Rsp})$.

Verify as in identification protocol.

Intuition

Avoid decoding: rely indirectly on SDP.

Use zero-knowledge framework to build identification protocol.

Prover (SK)		**Verifier (PK)**
Commit to random data	$\xrightarrow{\textit{Com}}$	
	$\xleftarrow{\textit{Ch}}$	Select random challenge
Produce response	$\xrightarrow{\textit{Rsp}}$	
		Accept/Reject

Signature scheme can be obtained using Fiat-Shamir transformation.

Replace random challenge with $\mathbf{H}(\textit{Com}, \mu)$, obtain signature as $\sigma = (\textit{Com}, \textit{Rsp})$.

Verify as in identification protocol.

For code-based, exploit hardness of finding low-weight words (Stern, 1993).

Stern's ZKID Protocol

Select hash function **H**.

Stern's ZKID Protocol

Select hash function **H**.

Key Generation

- Choose random binary code $\mathcal{C}$, given by parity-check matrix H.
- SK: $e \in \mathbb{F}_2^n$ of weight t.
- PK: the syndrome $s = He^T$.

Stern's ZKID Protocol

Select hash function $\mathbf{H}$.

Key Generation

- Choose random binary code $\mathcal{C}$, given by parity-check matrix H.
- SK: $e \in \mathbb{F}_2^n$ of weight t.
- PK: the syndrome $s = He^T$.

Prover | Verifier

Prover		Verifier
Choose $y \in \mathbb{F}_2^n$ and permutation π. Set $c_1 = \mathbf{H}(\pi, Hy^T), c_2 = \mathbf{H}(\pi(y))$ $c_3 = \mathbf{H}(\pi(y+e))$		
	$\xrightarrow{c_1, c_2, c_3}$	
	$\xleftarrow{b}$	Select random $b \in \{0, 1, 2\}$.
If $b = 0$ set $Rsp = (y, \pi)$		Verify c_1, c_2.
If $b = 1$ set $Rsp = (y + e, \pi)$	$\xrightarrow{Rsp}$	Verify c_1, c_3.
If $b = 2$ set $Rsp = (\pi(y), \pi(e))$		Verify c_2, c_3 and $wt(\pi(e)) = t$.

ZKID protocols characterized by the presence of soundness error.

About Stern's ZKID

ZKID protocols characterized by the presence of soundness error.

Malicious adversary can cheat with non-trivial probability: here 2/3.

ZKID protocols characterized by the presence of soundness error.

Malicious adversary can cheat with non-trivial probability: here 2/3.

For instance: choose random y and π, then $x \in \mathbb{F}_2^n$ with $Hx^T = Hy^T + s$. Build c_1 and c_2 normally and $c_3 = \mathbf{H}(\pi(x))$. Then $Rsp = (y, \pi)$ and $Rsp = (x, \pi)$ pass verification for $b = 0$ and $b = 1$ (strategy fails for $b = 2$). Similarly for other combinations.

About Stern's ZKID

ZKID protocols characterized by the presence of soundness error.

Malicious adversary can cheat with non-trivial probability: here 2/3.

For instance: choose random y and π, then $x \in \mathbb{F}_2^n$ with $Hx^T = Hy^T + s$. Build c_1 and c_2 normally and $c_3 = \mathbf{H}(\pi(x))$. Then $Rsp = (y, \pi)$ and $Rsp = (x, \pi)$ pass verification for $b = 0$ and $b = 1$ (strategy fails for $b = 2$). Similarly for other combinations.

This means several repetitions are necessary to amplify error and reach target authentication level.

About Stern's ZKID

ZKID protocols characterized by the presence of soundness error.

Malicious adversary can cheat with non-trivial probability: here 2/3.

For instance: choose random y and π, then $x \in \mathbb{F}_2^n$ with $Hx^T = Hy^T + s$. Build c_1 and c_2 normally and $c_3 = \mathbf{H}(\pi(x))$. Then $Rsp = (y, \pi)$ and $Rsp = (x, \pi)$ pass verification for $b = 0$ and $b = 1$ (strategy fails for $b = 2$). Similarly for other combinations.

This means several repetitions are necessary to amplify error and reach target authentication level.

Trasmitting the entire transcript means very long signature.

About Stern's ZKID

ZKID protocols characterized by the presence of soundness error.

Malicious adversary can cheat with non-trivial probability: here 2/3.

For instance: choose random y and π, then $x \in \mathbb{F}_2^n$ with $Hx^T = Hy^T + s$. Build c_1 and c_2 normally and $c_3 = \mathbf{H}(\pi(x))$. Then $Rsp = (y, \pi)$ and $Rsp = (x, \pi)$ pass verification for $b = 0$ and $b = 1$ (strategy fails for $b = 2$). Similarly for other combinations.

This means several repetitions are necessary to amplify error and reach target authentication level.

Trasmitting the entire transcript means very long signature.

Stern's ZKID parameters:

q	n	t	ℓ	PK (bits)	Sig (KB)	Security	Auth.
2	512	56	35	256	5	60	20
2	620	68	137	310	93.3	80	80
2	1024	112	219	512	245	128	128

CONSIDERATIONS

Several variants proposed over the years:

Several variants proposed over the years:

- Stern, 1993.
- Véron, 1996.
- Gaborit, Girault, 2007.
- Cayrel, Véron, El Yousfi, 2010.
- Aguilar, Gaborit, Schrek, 2011.
- ...

CONSIDERATIONS

Several variants proposed over the years:

- Stern, 1993.
- Véron, 1996.
- Gaborit, Girault, 2007.
- Cayrel, Véron, El Yousfi, 2010.
- Aguilar, Gaborit, Schrek, 2011.
- ...

Considerable improvements (soundness $\approx 1/2$, signatures below 40KB etc.)

CONSIDERATIONS

Several variants proposed over the years:

- Stern, 1993.
- Véron, 1996.
- Gaborit, Girault, 2007.
- Cayrel, Véron, El Yousfi, 2010.
- Aguilar, Gaborit, Schrek, 2011.
- ...

Considerable improvements (soundness $\approx 1/2$, signatures below 40KB etc.)

...yet, sizes remain very large.

CVE's Protocol

Select hash function **H**.

CVE's Protocol

Select hash function **H**.

Key Generation

- Choose random q-ary code $\mathcal{C}$, given by parity-check matrix H.
- SK: $e \in \mathbb{F}_q^n$ of weight t.
- PK: the syndrome $s = He^T$.

CVE's Protocol

Select hash function $\mathbf{H}$.

Key Generation

- Choose random q-ary code $\mathcal{C}$, given by parity-check matrix H.
- SK: $e \in \mathbb{F}_q^n$ of weight t.
- PK: the syndrome $s = He^T$.

Prover / Verifier

Prover		Verifier
Choose $y \in \mathbb{F}_q^n$ and monomial τ. Set $c_1 = \mathbf{H}(\tau, Hy^T)$, $c_2 = \mathbf{H}(\tau(y), \tau(e))$	$\xrightarrow{c_1, c_2}$	
	$\xleftarrow{c}$	Select random $c \in \mathbb{F}_q^*$.
$z = \tau(y + ce)$	$\xrightarrow{z}$	
	$\xleftarrow{b}$	Select random $b \in \{0, 1\}$.
If $b = 0$ set $Rsp = \tau$		Verify $c_1 = \mathbf{H}(\tau, H\tau^{-1}(z))^T - cs)$.
If $b = 1$ set $Rsp = \tau(e)$	$\xrightarrow{Rsp}$	Verify $c_2 = \mathbf{H}(z - c\tau(e), \tau(e))$ and $wt(\tau(e)) = t$.

ALTERNATIVE APPROACH

Design ZKID protocol with high soundness.

Alternative Approach

Design ZKID protocol with high soundness.

Set $SK = s, PK = S = \mathbf{H}(s)$, for a collision-resistant hash function $\mathbf{H}$.

ALTERNATIVE APPROACH

Design ZKID protocol with high soundness.

Set $SK = s$, $PK = S = \mathbf{H}(s)$, for a collision-resistant hash function $\mathbf{H}$.

PROVER		VERIFIER
Sample y	$\xrightarrow{Y=\mathbf{H}(y)}$	
	$\xleftarrow{c}$	Select random challenge c
compute $z = y + cs$	$\xrightarrow{z}$	
		Check $\mathbf{H}(z) = Y + cS$

Design ZKID protocol with high soundness.

Set $SK = s$, $PK = S = \mathbf{H}(s)$, for a collision-resistant hash function $\mathbf{H}$.

Prover		Verifier
Sample y	$\xrightarrow{Y=\mathbf{H}(y)}$	
	$\xleftarrow{c}$	Select random challenge c
compute $z = y + cs$	$\xrightarrow{z}$	
		Check $\mathbf{H}(z) = Y + cS$

Works well for lattices using vectors of small norm (Euclidean)
(Lyubashevsky, 2009).

In the Code-Based Setting

Use $\mathbf{H}(e) = He^T$, where H is a parity-check matrix for a random code, e is a vector of small norm (=Hamming weight) and $c \in \mathbb{F}_q$.

In the Code-Based Setting

Use $\mathbf{H}(e) = He^T$, where H is a parity-check matrix for a random code, e is a vector of small norm (=Hamming weight) and $c \in \mathbb{F}_q$.

Problem: the metric is too weak.

In the Code-Based Setting

Use $\mathbf{H}(e) = He^T$, where H is a parity-check matrix for a random code, e is a vector of small norm (=Hamming weight) and $c \in \mathbb{F}_q$.

Problem: the metric is too weak.

Note: y also has to have low weight, else easy to forge z.

In the Code-Based Setting

Use $\mathbf{H}(e) = He^T$, where H is a parity-check matrix for a random code, e is a vector of small norm (=Hamming weight) and $c \in \mathbb{F}_q$.

Problem: the metric is too weak.

Note: y also has to have low weight, else easy to forge z.

But then, the response $z = y + ce$ leaks secret positions.

In the Code-Based Setting

Use $\mathbf{H}(e) = He^T$, where H is a parity-check matrix for a random code, e is a vector of small norm (=Hamming weight) and $c \in \mathbb{F}_q$.

Problem: the metric is too weak.

Note: y also has to have low weight, else easy to forge z.

But then, the response $z = y + ce$ leaks secret positions.

Collecting samples of the form $c^{-1}z$ is enough to reveal the secret.

In the Code-Based Setting

Use $\mathbf{H}(e) = He^T$, where H is a parity-check matrix for a random code, e is a vector of small norm (=Hamming weight) and $c \in \mathbb{F}_q$.

Problem: the metric is too weak.

Note: y also has to have low weight, else easy to forge z.

But then, the response $z = y + ce$ leaks secret positions.

Collecting samples of the form $c^{-1}z$ is enough to reveal the secret.

Switching to QC-codes and polynomial operations: still vulnerable to similar statistical analysis.

(Santini, Baldi, Chiaraluce, 2019; Deneuville, Gaborit, 2020)

IN THE CODE-BASED SETTING

Use $\mathbf{H}(e) = He^T$, where H is a parity-check matrix for a random code, e is a vector of small norm (=Hamming weight) and $c \in \mathbb{F}_q$.

Problem: the metric is too weak.

Note: y also has to have low weight, else easy to forge z.

But then, the response $z = y + ce$ leaks secret positions.

Collecting samples of the form $c^{-1}z$ is enough to reveal the secret.

Switching to QC-codes and polynomial operations: still vulnerable to similar statistical analysis.
(Santini, Baldi, Chiaraluce, 2019; Deneuville, Gaborit, 2020)

Unlikely to overcome the metric* limitations.

The idea is still valid - need to use a different protocol.

Decreasing the Soundness Error

The idea is still valid - need to use a different protocol.

Take for example the CVE scheme: soundness error $\dfrac{q}{2(q-1)}$.

Decreasing the Soundness Error

The idea is still valid - need to use a different protocol.

Take for example the CVE scheme: soundness error $\dfrac{q}{2(q-1)}$.

After response, perform only first check.

Decreasing the Soundness Error

The idea is still valid - need to use a different protocol.

Take for example the CVE scheme: soundness error $\frac{q}{2(q-1)}$.

After response, perform only first check.

The other check can be offloaded to a trusted setup ("helper").

Decreasing the Soundness Error

The idea is still valid - need to use a different protocol.

Take for example the CVE scheme: soundness error $\dfrac{q}{2(q-1)}$.

After response, perform only first check.

The other check can be offloaded to a trusted setup (“helper”).

Preprocessing phase prepares auxiliary collection of samples of the form* $y + ce$.

Decreasing the Soundness Error

The idea is still valid - need to use a different protocol.

Take for example the CVE scheme: soundness error $\dfrac{q}{2(q-1)}$.

After response, perform only first check.

The other check can be offloaded to a trusted setup (“helper”).

Preprocessing phase prepares auxiliary collection of samples of the form* $y + ce$.

Verification depends now on c and soundness is $1/q$.

Decreasing the Soundness Error

The idea is still valid - need to use a different protocol.

Take for example the CVE scheme: soundness error $\frac{q}{2(q-1)}$.

After response, perform only first check.

The other check can be offloaded to a trusted setup ("helper").

Preprocessing phase prepares auxiliary collection of samples of the form* $y + ce$.

Verification depends now on c and soundness is $1/q$.

"MPC-in-the-head" approach used e.g. in Picnic.

(Ishai, Kushilevitz, Ostrovsky, Sahai, 2007; Katz, Kolesnikov, Wang, 2018)

GPS's Protocol

KeyGen: as in CVE, using a commitment scheme **Com**.

KeyGen: as in CVE, using a commitment scheme **Com**.

HELPER

- Generate random $y, \tilde{e} \in \mathbb{F}_q^n$, with $\tilde{e}$ of weight t, from seed.
- Compute $aux = \{\mathbf{Com}(y + c\tilde{e})\}_{c \in \mathbb{F}_q}$.
- Send seed to prover and aux to verifier.

KeyGen: as in CVE, using a commitment scheme **Com**.

Helper

- Generate random $y, \tilde{e} \in \mathbb{F}_q^n$, with $\tilde{e}$ of weight t, from seed.
- Compute $aux = \{\mathbf{Com}(y + c\tilde{e})\}_{c \in \mathbb{F}_q}$.
- Send seed to prover and aux to verifier.

Prover / Verifier

Prover		Verifier
Regenerate $y, \tilde{e}$ from seed.		
Determine τ s.t. $e = \tau(\tilde{e})$		
$\alpha = \mathbf{Com}(\tau, H(\tau(y))^T)$	$\xrightarrow{\alpha}$	
	$\xleftarrow{c}$	Select random $c \in \mathbb{F}_q$.
$z = y + c\tilde{e}$	$\xrightarrow{z}$	
		Verify $\alpha = \mathbf{Com}(\tau, H(\tau(z))^T - cs)$.
		Verify $\mathbf{Com}(z)$ with corresponding value from *aux*.

Use “cut-and-choose” technique to remove preprocessing.

Producing a Signature Scheme

Use “cut-and-choose” technique to remove preprocessing.

Executing 1 out of N setups produces soundness error $max\left(\frac{1}{N}, \frac{1}{q}\right)$.

Producing a Signature Scheme

Use "cut-and-choose" technique to remove preprocessing.

Executing 1 out of N setups produces soundness error $max\left(\frac{1}{N}, \frac{1}{q}\right)$.

Iterate as needed, then apply Fiat-Shamir.

Use “cut-and-choose” technique to remove preprocessing.

Executing 1 out of N setups produces soundness error $max\left(\frac{1}{N}, \frac{1}{q}\right)$.

Iterate as needed, then apply Fiat-Shamir.

Several optimizations are possible (e.g. Merkle trees).

Producing a Signature Scheme

Use “cut-and-choose” technique to remove preprocessing.

Executing 1 out of N setups produces soundness error $max\left(\frac{1}{N}, \frac{1}{q}\right)$.

Iterate as needed, then apply Fiat-Shamir.

Several optimizations are possible (e.g. Merkle trees).

Can potentially yield smaller signatures, at the cost of increased computation (signing/verification time).

Producing a Signature Scheme

Use "cut-and-choose" technique to remove preprocessing.

Executing 1 out of N setups produces soundness error $max\left(\frac{1}{N}, \frac{1}{q}\right)$.

Iterate as needed, then apply Fiat-Shamir.

Several optimizations are possible (e.g. Merkle trees).

Can potentially yield smaller signatures, at the cost of increased computation (signing/verification time).

Theoretical work available (ePrint 2021/1020), implementation is being developed.

Part IV

Code Equivalence and LESS

Isometries in the Hamming Metric

Three types:

1. Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.

Three types:

1. Permutations: $\pi\big((a_1, a_2, \dots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \dots, a_{\pi(n)}\big)$.
2. Monomials: permutations + scaling factors: $\tau = (v; \pi)$, with $v \in {\mathbb{F}_q^*}^n$

$$\tau\big((a_1, a_2, \dots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \dots, v_n \cdot a_{\pi(n)}\big)$$

Monomial matrix: permutation $\times$ diagonal.

Isometries in the Hamming Metric

Three types:

1. Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
2. Monomials: permutations + scaling factors: $\tau = (v; \pi)$, with $v \in {\mathbb{F}_q^*}^n$

$$\tau\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

   Monomial matrix: permutation $\times$ diagonal.
3. Monomials + field automorphism.

Three types:

1. Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
2. Monomials: permutations + scaling factors: $\tau = (v; \pi)$, with $v \in {\mathbb{F}_q^*}^n$

$$\tau\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

   Monomial matrix: permutation $\times$ diagonal.
3. Monomials + field automorphism.

For a code $\mathcal{C}$, $\tau(\mathcal{C}) = \{\tau(c) \mid c \in \mathcal{C}\}$.

Isometries in the Hamming Metric

Three types:

1. Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
2. Monomials: permutations + scaling factors: $\tau = (v; \pi)$, with $v \in \mathbb{F}_q^{*n}$

   $$\tau\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

   Monomial matrix: permutation $\times$ diagonal.
3. Monomials + field automorphism.

For a code $\mathcal{C}$, $\tau(\mathcal{C}) = \{\tau(c) \mid c \in \mathcal{C}\}$.

Two codes are equivalent if they are connected by an isometry.

PERMUTATION EQUIVALENCE PROBLEM (PEP)

Given $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$, find a permutation π such that $\pi(\mathcal{C}) = \mathcal{C}'$.
Equivalently, given generators $G, G' \in \mathbb{F}_q^{k\times n}$, find $S \in \mathrm{GL}_k$, permutation matrix P such that

$$SGP = G'$$

If two codes are permutation equivalent, we write $\mathcal{C} \sim_{PE} \mathcal{C}'$.

Computational Problems

Permutation Equivalence Problem (PEP)

Given $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$, find a permutation π such that $\pi(\mathcal{C}) = \mathcal{C}'$.
Equivalently, given generators $G, G' \in \mathbb{F}_q^{k \times n}$, find $S \in \mathrm{GL}_k$, permutation matrix P such that

$$SGP = G'$$

If two codes are permutation equivalent, we write $\mathcal{C} \sim_{PE} \mathcal{C}'$.

Linear Equivalence Problem (LEP)

Given $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^n$, find a monomial τ such that $\tau(\mathcal{C}) = \mathcal{C}'$. Equivalently, given generators $G, G' \in \mathbb{F}_q^{k \times n}$, find $S \in \mathrm{GL}_k$, monomial matrix Q such that

$$SGQ = G'$$

If two codes are monomially equivalent, we write $\mathcal{C} \sim_{LE} \mathcal{C}'$.

General Considerations

PEP is a special case of LEP, which in turn is a special case of Semi-Linear Equivalence (monomial + field automorphism).

$$\text{PEP} \supseteq \text{LEP} \supseteq \text{SLEP}$$

General Considerations

PEP is a special case of LEP, which in turn is a special case of Semi-Linear Equivalence (monomial + field automorphism).

$$\text{PEP} \supseteq \text{LEP} \supseteq \text{SLEP}$$

PEP is not NP-complete, unless the polynomial hierarchy collapses (Petrank, Roth, 1997).

PEP is a special case of LEP, which in turn is a special case of Semi-Linear Equivalence (monomial + field automorphism).

$$\text{PEP} \supseteq \text{LEP} \supseteq \text{SLEP}$$

PEP is not NP-complete, unless the polynomial hierarchy collapses (Petrank, Roth, 1997).

PEP is deeply connected with Graph Isomorphism (GI) (reductions in both ways!), solvable in quasi-polynomial time.

PEP is a special case of LEP, which in turn is a special case of Semi-Linear Equivalence (monomial + field automorphism).

$$\text{PEP} \supseteq \text{LEP} \supseteq \text{SLEP}$$

PEP is not NP-complete, unless the polynomial hierarchy collapses (Petrank, Roth, 1997).

PEP is deeply connected with Graph Isomorphism (GI) (reductions in both ways!), solvable in quasi-polynomial time.

With time $O(q)$, we have

$$\text{PEP} \xleftarrow{\text{Reduces to}} \text{LEP} \xleftarrow{\text{Reduces to}} \text{SLEP}$$

General Considerations

PEP is a special case of LEP, which in turn is a special case of Semi-Linear Equivalence (monomial + field automorphism).

$$\text{PEP} \supseteq \text{LEP} \supseteq \text{SLEP}$$

PEP is not NP-complete, unless the polynomial hierarchy collapses (Petrank, Roth, 1997).

PEP is deeply connected with Graph Isomorphism (GI) (reductions in both ways!), solvable in quasi-polynomial time.

With time $O(q)$, we have

$$\text{PEP} \xleftarrow{\text{Reduces to}} \text{LEP} \xleftarrow{\text{Reduces to}} \text{SLEP}$$

There very efficient solvers for certain specific cases.

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^\perp$ (Sendrier, 2000).

Security Overview

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^{\perp}$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

Security Overview

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^{\perp}$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

If hull is trivial, can fully exploit reduction to graph isomorphism (Bardet, Otmani, Saeed-Taha, 2019) or use algebraic solvers (Saeed-Taha, 2017).

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^{\perp}$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

If hull is trivial, can fully exploit reduction to graph isomorphism (Bardet, Otmani, Saeed-Taha, 2019) or use algebraic solvers (Saeed-Taha, 2017).

Permutation can be revealed by low-weight codewords: use ISD and match (Leon, 1982).

Security Overview

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^\perp$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

If hull is trivial, can fully exploit reduction to graph isomorphism (Bardet, Otmani, Saeed-Taha, 2019) or use algebraic solvers (Saeed-Taha, 2017).

Permutation can be revealed by low-weight codewords: use ISD and match (Leon, 1982).
$\rightarrow$ Choice of weight is crucial for algorithm effectiveness.

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^\perp$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

If hull is trivial, can fully exploit reduction to graph isomorphism (Bardet, Otmani, Saeed-Taha, 2019) or use algebraic solvers (Saeed-Taha, 2017).

Permutation can be revealed by low-weight codewords: use ISD and match (Leon, 1982).
$\rightarrow$ Choice of weight is crucial for algorithm effectiveness.

Can be improved by considering multisets of entries (Beullens, 2020).

Security Overview

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^{\perp}$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

If hull is trivial, can fully exploit reduction to graph isomorphism (Bardet, Otmani, Saeed-Taha, 2019) or use algebraic solvers (Saeed-Taha, 2017).

Permutation can be revealed by low-weight codewords: use ISD and match (Leon, 1982).
$\rightarrow$ Choice of weight is crucial for algorithm effectiveness.

Can be improved by considering multisets of entries (Beullens, 2020).
$\rightarrow$ More efficient when q is large.

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^\perp$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

If hull is trivial, can fully exploit reduction to graph isomorphism (Bardet, Otmani, Saeed-Taha, 2019) or use algebraic solvers (Saeed-Taha, 2017).

Permutation can be revealed by low-weight codewords: use ISD and match (Leon, 1982).
$\rightarrow$ Choice of weight is crucial for algorithm effectiveness.

Can be improved by considering multisets of entries (Beullens, 2020).
$\rightarrow$ More efficient when q is large.

All of the above can be (non-trivially) tweaked to work on LEP.

Security Overview

The weight enumerator is a fully discriminating function. Can compute on the hull, i.e. $\mathcal{C} \cap \mathcal{C}^{\perp}$ (Sendrier, 2000).
$\rightarrow$ Complexity scales with dimension of the hull.

If hull is trivial, can fully exploit reduction to graph isomorphism (Bardet, Otmani, Saeed-Taha, 2019) or use algebraic solvers (Saeed-Taha, 2017).

Permutation can be revealed by low-weight codewords: use ISD and match (Leon, 1982).
$\rightarrow$ Choice of weight is crucial for algorithm effectiveness.

Can be improved by considering multisets of entries (Beullens, 2020).
$\rightarrow$ More efficient when q is large.

All of the above can be (non-trivially) tweaked to work on LEP.

Hard-to-solve Instances

Weakly self-dual codes for PEP.
Random codes over $\mathbb{F}_q$, with $q \geq 5$, for LEP.

Could Code Equivalence be used as a stand-alone problem?

Could Code Equivalence be used as a stand-alone problem?

The situation for isometries recalls that of other group actions, such as for DLP (although without commutativity).

Could Code Equivalence be used as a stand-alone problem?

The situation for isometries recalls that of other group actions, such as for DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence.

Could Code Equivalence be used as a stand-alone problem?

The situation for isometries recalls that of other group actions, such as for DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence.

Possible to construct a ZK protocol based exclusively on the hardness of the code equivalence problem.

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

Could Code Equivalence be used as a stand-alone problem?

The situation for isometries recalls that of other group actions, such as for DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence.

Possible to construct a ZK protocol based exclusively on the hardness of the code equivalence problem.

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

This can be then transformed into a full-fledged signature scheme via Fiat-Shamir.

Applications in Cryptography

Could Code Equivalence be used as a stand-alone problem?

The situation for isometries recalls that of other group actions, such as for DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence.

Possible to construct a ZK protocol based exclusively on the hardness of the code equivalence problem.
(Biasse, Micheli, P., Santini, 2020)

This can be then transformed into a full-fledged signature scheme via Fiat-Shamir.

Protocol can be tweaked to increase efficiency (e.g. multiple public keys, fixed-weight challenges) (Barenghi, Biasse, P., Santini, 2021)

LESS ZK Identification Scheme

Public data: hash function **H**, code $\mathcal{C}$ with generator G

LESS ZK Identification Scheme

Public data: hash function **H**, code $\mathcal{C}$ with generator G

Key Generation

- SK: invertible matrix S and monomial matrix Q.
- PK: matrix $G' = SGQ$ (can be systematic form).

LESS ZK Identification Scheme

Public data: hash function **H**, code $\mathcal{C}$ with generator G

Key Generation

- 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 $h = \mathbf{H}(\tilde{G})$.
  (After receiving challenge bit b).
- If $b = 0$ respond with $\tau = \tilde{Q}$.
- If $b = 1$ respond with $\tau = Q^{-1}\tilde{Q}$.

LESS ZK Identification Scheme

Public data: hash function $\mathbf{H}$, code $\mathcal{C}$ with generator G

Key Generation

- 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 $h = \mathbf{H}(\tilde{G})$.

  (After receiving challenge bit b).
- If $b = 0$ respond with $\tau = \tilde{Q}$.
- If $b = 1$ respond with $\tau = Q^{-1}\tilde{Q}$.

Verifier's Computation

- If $b = 0$ verify that $\mathbf{H}(SystForm(G\tau)) = h$.
- If $b = 1$ verify that $\mathbf{H}(SystForm(G'\tau)) = h$.

The State-of-the-art in Code-based Signatures

Scheme	Security Level	Public Data	Public Key	Sig.	PK + Sig.	Security Assumption
Stern	80	18.43	0.048	113.57	113.62	Decoding with low Hamming
Veron	80	18.43	0.096	109.06	109.16	
CVE	80	5.18	0.072	66.44	66.54	
Wave	128	-	3205	1.04	3206.04	Decoding with high Hamming
cRVDC	125	0.050	0.15	22.48	22.63	Decoding with low rank
Durandal - I	128	307.31	15.24	4.06	19.3	
Durandal - II	128	419.78	18.60	5.01	23.61	
GPS - I	128	9.78	0.11	24.60	24.71	MPC with q-ary SDP
GPS- IV	128	13.71	0.13	19.50	19.63	
LESS-FM - I	128	9.78	9.78	15.2	24.97	Code Equivalence Problem
LESS-FM - II	128	13.71	205.74	5.25	210.99	
LESS-FM - III	128	11.57	11.57	10.39	21.96	

TABLE: A comparison of public keys and signature sizes for code-based signature schemes. All sizes are in Kilobytes (kB).

Grazie per l'attenzione!