Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs

Thibauld Feneuil[1,2] Antoine Joux[3] Matthieu Rivain[1]

1. CryptoExperts, Paris, France

2. Sorbonne Université, CNRS, INRIA, Institut de Mathématiques de Jussieu-Paris Rive Gauche, Ouragan, Paris, France

3. CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

March 14, 2022

Table of Contents

Zero-Knowledge Protocol of Knowledge

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$in_{\mathcal{P}}$		$in_{\mathcal{V}}$
[...]	$\xrightarrow{\textsc{Com}}$	
	$\xleftarrow{\textsc{Ch}_1}$	
	$\xrightarrow{\textsc{Rsp}_1}$	
	$\vdots$	
	$\xleftarrow{\textsc{Ch}_n}$	
	$\xrightarrow{\textsc{Rsp}_n}$	
		Return $out \in \{0,1\}$

The prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ of the correctness of a *statement*. He can cheat with a probability up to the *soundness error*.

Which code-based assumption?

Syndrome Decoding Problem on Random Linear Code

Let H, x and y be such that:

- ☞ H is uniformly sampled from $\mathbb{F}^{(m-k)\times m}$,
- ☞ x is uniformly sampled from $\{x \in \mathbb{F}^m : \mathrm{wt}(x) = w\}$,
- ☞ y is defined as $y := Hx$.

From (H, y), find x.

Which code-based assumption?

Syndrome Decoding Problem on Random Linear Code

Let H, x and y be such that:

- ☞ H is uniformly sampled from $\mathbb{F}^{(m-k)\times m}$,
- ☞ x is uniformly sampled from $\{x \in \mathbb{F}^m : \mathrm{wt}(x) = w\}$,
- ☞ y is defined as $y := Hx$.

From (H, y), find x.

The prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that he knows the solution x... without revealing any information about x.

State of the art about ZK PoK for SD

Protocol	**Year**	**Assumption**	**Soundness err.**
Stern's	1993	SD	$2/3$
Véron's	1997	SD	$2/3$
CVE's	2010	SD on $\mathbb{F}_q$	$\approx 1/2$
AGS's	2011	QCSD	$\approx 1/2$

State of the art about ZK PoK for SD

Protocol	Year	Assumption	Soundness err.
Stern's	1993	SD	$2/3$
Véron's	1997	SD	$2/3$
CVE's	2010	SD on $\mathbb{F}_q$	$\approx 1/2$
AGS's	2011	QCSD	$\approx 1/2$
GPS's	2021	SD on $\mathbb{F}_q$	$\approx 1/N$

[GPS21] Shay Gueron, Edoardo Persichetti, and Paolo Santini. *Designing a Practical Code-based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup*. Eprint 2021/1020.

State of the art about ZK PoK for SD

Protocol	Year	Assumption	Soundness err.
Stern's	1993	SD	$2/3$
Véron's	1997	SD	$2/3$
CVE's	2010	SD on $\mathbb{F}_q$	$\approx 1/2$
AGS's	2011	QCSD	$\approx 1/2$
GPS's	2021	SD on $\mathbb{F}_q$	$\approx 1/N$
BGKS's	2021	QCSD	$\approx 1/2$

[BGKS21] Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, Nicolas Sendrier. *Quasi-Cyclic Stern Proof of Knowledge.* arXiv 2110.05005.

State of the art about ZK PoK for SD

Protocol	**Year**	**Assumption**	**Soundness err.**
Stern's	1993	SD	$2/3$
Véron's	1997	SD	$2/3$
CVE's	2010	SD on $\mathbb{F}_q$	$\approx 1/2$
AGS's	2011	QCSD	$\approx 1/2$
GPS's	2021	SD on $\mathbb{F}_q$	$\approx 1/N$
BGKS's	2021	QCSD	$\approx 1/2$
FJR21's	2021	SD	$\approx 1/N$

$$\sigma = \sigma_N \circ \sigma_{N-1} \circ \ldots \circ \sigma_3 \circ \sigma_2 \circ \sigma_1$$

[FJR21] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. *Shared Permutation for Syndrome Decoding: New Zero-Knowledge Protocol and Code-Based Signature.* Eprint 2021/1576.

State of the art about ZK PoK for SD

Protocol	**Year**	**Assumption**	**Soundness err.**
Stern's	1993	SD	$2/3$
Véron's	1997	SD	$2/3$
CVE's	2010	SD on $\mathbb{F}_q$	$\approx 1/2$
AGS's	2011	QCSD	$\approx 1/2$
GPS's	2021	SD on $\mathbb{F}_q$	$\approx 1/N$
BGKS's	2021	QCSD	$\approx 1/2$
FJR21's	2021	SD	$\approx 1/N$
BGKM's	2022	SD	$\approx 1/N$

[BGKM22] Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, Victor Mateu. *Code-based Signatures from New Proofs of Knowledge for the Syndrome Decoding Problem.* arXiv 2110.05005.

State of the art about ZK PoK for SD

Protocol	Year	Assumption	Soundness err.
Stern's	1993	SD	$2/3$
Véron's	1997	SD	$2/3$
CVE's	2010	SD on $\mathbb{F}_q$	$\approx 1/2$
AGS's	2011	QCSD	$\approx 1/2$
GPS's	2021	SD on $\mathbb{F}_q$	$\approx 1/N$
BGKS's	2021	QCSD	$\approx 1/2$
FJR21's	2021	SD	$\approx 1/N$
BGKM's	2022	SD	$\approx 1/N$
FJR22's	2022	SD	$\approx 1/N$

Prove $\mathrm{wt}(x) \leq w$, not $\mathrm{wt}(x) = w$.

Table of Contents

Definition for sharing

Let have $v \in \mathbb{F}_q^m$.

Sample $[\![v]\!] = ([\![v]\!]_1, \ldots, [\![v]\!]_N) \in (\mathbb{F}_q^m)^N$ such that

$$v = [\![v]\!]_1 + [\![v]\!]_2 + \ldots + [\![v]\!]_N$$

In practice,

$$\begin{cases} [\![v]\!]_i \xleftarrow{\$} \mathbb{F}_q^m & \text{for } i < N \\ [\![v]\!]_N = v - \sum_{i<N} [\![v]\!]_i & \end{cases}$$

Multi-Party Computation

In the MPC context, an N-sharing is usually distributed to N parties.

$$\begin{array}{cccc} \mathcal{P}_1(\llbracket v \rrbracket_1) & \mathcal{P}_2(\llbracket v \rrbracket_2) & \dots & \mathcal{P}_N(\llbracket v \rrbracket_N) \\ \vdots & \vdots & & \vdots \end{array}$$

From those shares, the parties can perform distributed computation.

Multi-Party Computation

Addition: $[\![x+y]\!] = [\![x]\!] + [\![y]\!]$

$$\forall i,\ [\![x+y]\!]_i := [\![x]\!]_i + [\![y]\!]_i$$

Multi-Party Computation

Addition: $[\![x+y]\!] = [\![x]\!] + [\![y]\!]$

$$\forall i,\ [\![x+y]\!]_i := [\![x]\!]_i + [\![y]\!]_i$$

Addition with a constant: $[\![x+\alpha]\!] = [\![x]\!] + \alpha$

$$\begin{cases} [\![x+\alpha]\!]_1 := [\![x]\!]_1 + \alpha \\ [\![x+\alpha]\!]_i := [\![x]\!]_i \quad \text{for } i \neq 1 \end{cases}$$

Multi-Party Computation

Addition: $[\![x+y]\!] = [\![x]\!] + [\![y]\!]$

$$\forall i,\ [\![x+y]\!]_i := [\![x]\!]_i + [\![y]\!]_i$$

Addition with a constant: $[\![x+\alpha]\!] = [\![x]\!] + \alpha$

$$\begin{cases} [\![x+\alpha]\!]_1 := [\![x]\!]_1 + \alpha \\ [\![x+\alpha]\!]_i := [\![x]\!]_i \ \text{ for } i \neq 1 \end{cases}$$

Multiplication by a constant: $[\![\alpha \cdot x]\!] = \alpha \cdot [\![x]\!]$

$$\forall i,\ [\![\alpha \cdot x]\!]_i := \alpha \cdot [\![x]\!]_i$$

Sharing for polynomials

Let have $P \in \mathbb{F}[X]$ of degree at most d.

A sharing $[\![P]\!]$ for P is a N-tuple of $(\mathbb{F}[X])^N$ such that $P = \sum_{i=1}^{N} [\![P]\!]_i$, where each $[\![P]\!]_i$ is of degree at most d.

Sharing for polynomials

Let have $P \in \mathbb{F}[X]$ of degree at most d.

A sharing $[\![P]\!]$ for P is a N-tuple of $(\mathbb{F}[X])^N$ such that $P = \sum_{i=1}^{N} [\![P]\!]_i$, where each $[\![P]\!]_i$ is of degree at most d.

Evaluation: given r, $[\![P(r)]\!] = [\![P]\!](r)$

$$\forall i,\ [\![P(r)]\!]_i := [\![P]\!]_i(r) = \sum_{j=0}^{d} [\![P_j]\!]_i \cdot r^j\ ,$$

MPC Protocol

Let have a SD instance (H, y).
In the article, we propose a MPC protocol π where parties take shares of a vector x as input,

$$\begin{array}{cccc} \mathcal{P}_1(\llbracket x \rrbracket_1) & \mathcal{P}_2(\llbracket x \rrbracket_2) & \dots & \mathcal{P}_N(\llbracket x \rrbracket_N) \\ \vdots & \vdots & & \vdots \end{array}$$

and which outputs

$$\begin{cases} \textsc{Accept} \text{ if } y = Hx \text{ and } \mathrm{wt}(x) \leq w, \\ \textsc{Reject} \text{ otherwise.} \end{cases}$$

MPC-in-the-Head paradigm

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
H, y, x such that $y = Hx$ and $\mathrm{wt}(x) \leq w$		H, y
Run the MPC protocol π for each party. $\textsc{Com}_i \leftarrow \mathrm{Com}(\text{view } V_i)$		
	$\xrightarrow{\textsc{Com}_1, \ldots, \textsc{Com}_N}$	$i^* \xleftarrow{\$} \{1, \ldots, N\}$
	$\xleftarrow{i^*}$	
	$\xrightarrow{\text{all } V_i \text{ for } i \neq i^*}$	
		Check that the views are consistent Check that the MPC output is $\textsc{Accept}$

$$\text{View } V_i \text{ of the party } \mathcal{P}_i = \begin{cases} \text{party's input share,} \\ \text{secret random tape,} \\ \text{sent and received messages.} \end{cases}$$

Construction

Let $x \in \mathbb{F}_{\mathrm{SD}}^m$.
To prove that $\mathrm{wt}(x) \leq w$, we prove there exists $Q \in \mathbb{F}_{\mathrm{poly}}[X]$ s.t.

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \circ \begin{pmatrix} Q(\gamma_1) \\ Q(\gamma_2) \\ \vdots \\ Q(\gamma_m) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

where

- $\mathbb{F}_{\mathrm{poly}}$ is a field extension of $\mathbb{F}_{\mathrm{SD}}$,
- the degree of Q is exactly w,
- $\gamma_1, \ldots, \gamma_m$ are distinct elements of $\mathbb{F}_{\mathrm{poly}}$.

$\Rightarrow$ there are m multiplications.

In terms of polynomials

Let $x \in \mathbb{F}_{\mathrm{SD}}^m$.
To prove that $\mathrm{wt}(x) \leq w$, we prove there exists $Q \in \mathbb{F}_{\mathrm{poly}}[X]$ s.t.

$$\begin{pmatrix} S(\gamma_1) \\ S(\gamma_2) \\ \vdots \\ S(\gamma_m) \end{pmatrix} \circ \begin{pmatrix} Q(\gamma_1) \\ Q(\gamma_2) \\ \vdots \\ Q(\gamma_m) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

where

- $\mathbb{F}_{\mathrm{poly}}$ is a field extension of $\mathbb{F}_{\mathrm{SD}}$,
- the degree of Q is exactly w,
- $\gamma_1, \ldots, \gamma_m$ are distinct elements of $\mathbb{F}_{\mathrm{poly}}$,
- S is built by interpolation such that

$$\forall i, \; S(\gamma_i) = x_i.$$

In terms of polynomials

Let $x \in \mathbb{F}_{\mathrm{SD}}^m$.
To prove that $\mathrm{wt}(x) \leq w$, we prove there exists $Q \in \mathbb{F}_{\mathrm{poly}}[X]$ s.t.

$$S \cdot Q \text{ is equal to zero on } \{\gamma_1, \ldots, \gamma_m\}.$$

where

- $\mathbb{F}_{\mathrm{poly}}$ is a field extension of $\mathbb{F}_{\mathrm{SD}}$,
- the degree of Q is exactly w,
- $\gamma_1, \ldots, \gamma_m$ are distinct elements of $\mathbb{F}_{\mathrm{poly}}$,
- S is built by interpolation such that

$$\forall i,\ S(\gamma_i) = x_i.$$

In terms of polynomials

If the prover convinces the verifier that there exists $Q, P \in \mathbb{F}_{\text{poly}}[X]$ s.t.

$$S \cdot Q = P \cdot F$$

where

the degree of Q is exactly w,

S is built by interpolation such that $\forall i,\ S(\gamma_i) = x_i$,

$F := \prod_{i=1}^{m}(X - \gamma_i)$,

then, the verifier deduces that

$$\forall i \leq m,\ (Q \cdot S)(\gamma_i) = P(\gamma_i) \cdot F(\gamma_i) = 0$$
$$\Rightarrow\ \forall i \leq m,\ Q(\gamma_i) = 0 \ \text{ or } \ S(\gamma_i) = x_i = 0$$

In terms of polynomials

If the prover convinces the verifier that there exists $Q, P \in \mathbb{F}_{\text{poly}}[X]$ s.t.

$$S \cdot Q = P \cdot F$$

where

the degree of Q is exactly w,

S is built by interpolation such that $\forall i,\ S(\gamma_i) = x_i$,

$F := \prod_{i=1}^{m}(X - \gamma_i)$,

then, the verifier deduces that

$$\forall i \leq m,\ (Q \cdot S)(\gamma_i) = P(\gamma_i) \cdot F(\gamma_i) = 0$$

$$\Rightarrow\ \forall i \leq m,\ Q(\gamma_i) = 0 \ \text{ or } \ S(\gamma_i) = x_i = 0$$

i.e.

$$\mathrm{wt}(x) \leq w$$

The MPC Protocol

Inputs of the party $\mathcal{P}_i$: $[\![x]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$.

1. Check that $y = H[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$ thanks to

$$[\![S(X)]\!] = \sum_i [\![x_i]\!] \cdot \prod_{\ell \neq i} \frac{X - \gamma_\ell}{\gamma_i - \gamma_\ell} .$$

3. Check that $S \cdot Q = P \cdot F$ with $F := \prod_{i=1}^{m}(X - \gamma_i)$.

Linear constraint

Let us assume $H = (H' \mid I)$. We split x as $\begin{pmatrix} x_A \\ x_B \end{pmatrix}$.
We have $y = Hx = x_B + H'x_A$. So

$$x_B = y - H'x_A.$$

The MPC Protocol

Inputs of the party $\mathcal{P}_i$: $[\![x_A]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$.

1. Compute $[\![x_B]\!] = y - H'[\![x_A]\!]$, and then deduce $[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$ thanks to

$$[\![S(X)]\!] = \sum_i [\![x_i]\!] \cdot \prod_{\ell \neq i} \frac{X - \gamma_\ell}{\gamma_i - \gamma_\ell} \ .$$

3. Check that $S \cdot Q = P \cdot F$ with $F := \prod_{i=1}^{m} (X - \gamma_i)$.

The MPC Protocol

Inputs of the party $\mathcal{P}_i$: $[\![x_A]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$.

1. Compute $[\![x_B]\!] = y - H'[\![x_A]\!]$, and then deduce $[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$ thanks to

$$[\![S(X)]\!] = \sum_i [\![x_i]\!] \cdot \prod_{\ell \neq i} \frac{X - \gamma_\ell}{\gamma_i - \gamma_\ell} \ .$$

3. Check that $S \cdot Q = P \cdot F$ with $F := \prod_{i=1}^{m} (X - \gamma_i)$.

To check $S \cdot Q = P \cdot F$, we check the relation on a random point.

The MPC Protocol

Inputs of the party $\mathcal{P}_i$: $[\![x_A]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$.

1. Compute $[\![x_B]\!] = y - H'[\![x_A]\!]$, and then deduce $[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$ thanks to

$$[\![S(X)]\!] = \sum_i [\![x_i]\!] \cdot \prod_{\ell \neq i} \frac{X - \gamma_\ell}{\gamma_i - \gamma_\ell} .$$

3. Get a random point $r \in \mathbb{F}_{\text{poly}}$ (from a trusted source) and check that $S(r) \cdot Q(r) = P(r) \cdot F(r)$.

The MPC Protocol

Inputs of the party $\mathcal{P}_i$: $[\![x_A]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$.

1. Compute $[\![x_B]\!] = y - H'[\![x_A]\!]$, and then deduce $[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$ thanks to

$$[\![S(X)]\!] = \sum_i [\![x_i]\!] \cdot \prod_{\ell \neq i} \frac{X - \gamma_\ell}{\gamma_i - \gamma_\ell} .$$

3. Get a random point $r \in \mathbb{F}_{\text{poly}}$ (from a trusted source) and check that $S(r) \cdot Q(r) = P(r) \cdot F(r)$.

Schwartz-Zippel Lemma: If $S \cdot Q \neq P \cdot F$, then

$$\Pr_{r \xleftarrow{\$} \mathbb{F}_{\text{poly}}} [S(r) \cdot Q(r) = P(r) \cdot F(r)] \leq \frac{m + w - 1}{|\mathbb{F}_{\text{poly}}|}$$

The MPC Protocol

Inputs of the party $\mathcal{P}_i$: $[\![x_A]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$.

1. Compute $[\![x_B]\!] = y - H'[\![x_A]\!]$, and then deduce $[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$ thanks to

$$[\![S(X)]\!] = \sum_i [\![x_i]\!] \cdot \prod_{\ell \neq i} \frac{X - \gamma_\ell}{\gamma_i - \gamma_\ell} .$$

3. Get a random point $r \in \mathbb{F}_{\text{points}}$ (from a trusted source) and check that $S(r) \cdot Q(r) = P(r) \cdot F(r)$.

Schwartz-Zippel Lemma: If $S \cdot Q \neq P \cdot F$, then

$$\Pr_{r \xleftarrow{\$} \mathbb{F}_{\text{points}}} [S(r) \cdot Q(r) = P(r) \cdot F(r)] \leq \frac{m + w - 1}{|\mathbb{F}_{\text{points}}|}$$

$\mathbb{F}_{points}$ is a field extension of $\mathbb{F}_{poly}$.

The MPC Protocol

Inputs of the party $\mathcal{P}_i$: $[\![x_A]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$.

1. Compute $[\![x_B]\!] = y - H'[\![x_A]\!]$, and then deduce $[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$.
3. Get a random point $r \in \mathbb{F}_{\text{points}}$.
4. Compute
$$\begin{cases} [\![S(r)]\!] = [\![S]\!](r) \\ [\![Q(r)]\!] = [\![Q]\!](r) \\ [\![P(r)]\!] = [\![P]\!](r) \end{cases}$$
5. Using [BN20], check that $S(r) \cdot Q(r) = P(r) \cdot F(r)$.

[BN20] Carsten Baum and Ariel Nof. *Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography.* PKC 2020.

BN20 Checking Protocol

Inputs: $([\![x]\!], [\![y]\!], [\![z]\!])$ and $([\![a]\!], [\![b]\!], [\![c]\!])$.

1. The parties get a random $\varepsilon \in \mathbb{F}_{\text{points}}$.
2. The parties locally set $[\![\alpha]\!] = \varepsilon[\![x]\!] + [\![a]\!]$ and $[\![\beta]\!] = [\![y]\!] + [\![b]\!]$
3. The parties broadcast $[\![\alpha]\!]$ and $[\![\beta]\!]$ to obtain α and β.
4. The parties locally set
   $[\![v]\!] = \varepsilon[\![z]\!] - [\![c]\!] + \alpha \cdot [\![b]\!] + \beta \cdot [\![a]\!] - \alpha \cdot \beta$.
5. The parties broadcast $[\![v]\!]$ to obtain v.
6. The parties output Accept if $v = 0$ and Reject otherwise.

BN20 Checking Protocol

Inputs: $([\![x]\!], [\![y]\!], [\![z]\!])$ and $([\![a]\!], [\![b]\!], [\![c]\!])$.

1. The parties get a random $\varepsilon \in \mathbb{F}_{\text{points}}$.
2. The parties locally set $[\![\alpha]\!] = \varepsilon[\![x]\!] + [\![a]\!]$ and $[\![\beta]\!] = [\![y]\!] + [\![b]\!]$
3. The parties broadcast $[\![\alpha]\!]$ and $[\![\beta]\!]$ to obtain α and β.
4. The parties locally set $[\![v]\!] = \varepsilon[\![z]\!] - [\![c]\!] + \alpha \cdot [\![b]\!] + \beta \cdot [\![a]\!] - \alpha \cdot \beta$.
5. The parties broadcast $[\![v]\!]$ to obtain v.
6. The parties output Accept if $v = 0$ and Reject otherwise.

$$(z = x \cdot y) \quad \text{and} \quad (c = a \cdot b) \Longrightarrow v = 0$$

$$(z \neq x \cdot y) \quad \text{or} \quad (c \neq a \cdot b) \Longrightarrow v = 0 \quad \text{with proba } \frac{1}{|\mathbb{F}_{\text{points}}|}$$

The MPC Protocol

Inputs of the party $\mathcal{P}_i$:

$[\![x_A]\!]_i$, $[\![Q]\!]_i$ and $[\![P]\!]_i$

$([\![a]\!]_i, [\![b]\!]_i, [\![c]\!]_i)$ such that $c = a \cdot b$

MPC Protocol:

1. Compute $[\![x_B]\!] = y - H'[\![x_A]\!]$, and then deduce $[\![x]\!]$.
2. Compute $[\![S]\!]$ from $[\![x]\!]$.
3. Get a random point $r, \varepsilon \in \mathbb{F}_{\text{points}}$.
4. Compute
$$\begin{cases} [\![S(r)]\!] = [\![S]\!](r) \\ [\![Q(r)]\!] = [\![Q]\!](r) \\ [\![P(r)]\!] = [\![P]\!](r) \end{cases}$$
5. Using [BN20], check that $S(r) \cdot Q(r) = P(r) \cdot F(r)$ using $([\![a]\!], [\![b]\!], [\![c]\!])$ and ε.

Summary

The MPC protocol π checks that $([\![x_A]\!], [\![Q]\!], [\![P]\!])$ describes a solution of the SD instance (H, y).

	Output of π	
	Accept	Reject
A good witness	1	0
Not a good witness	p	$1-p$

where

$$p = \underbrace{\frac{m+w-1}{|\mathbb{F}_{\text{points}}|}}_{\substack{\text{false positive} \\ \text{from Schwartz-Zippel}}} + \left(1 - \frac{m+w-1}{|\mathbb{F}_{\text{points}}|}\right) \cdot \underbrace{\frac{1}{|\mathbb{F}_{\text{points}}|}}_{\substack{\text{false positive} \\ \text{from [BN20]}}}$$

MPC-in-the-Head paradigm

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
H, y, x such that $y = Hx$ and $\mathrm{wt}(x) \leq w$		H, y
Prepare Q, P and (a, b, c). $\textsc{Com}_i \leftarrow \mathrm{Com}(\text{inputs of } \mathcal{P}_i)$		
	$\xrightarrow{\textsc{Com}_1,\ldots,\textsc{Com}_N}$	$r, \varepsilon \in \mathbb{F}_{\text{points}}$
Run the MPC protocol π for each party.	$\xleftarrow{r,\varepsilon}$	
	$\xrightarrow{\text{broadcast messages}}$	$i^* \xleftarrow{\$} \{1, \ldots, N\}$
	$\xleftarrow{i^*}$	
	$\xrightarrow{\text{all } V_i \text{ for } i \neq i^*}$	
		Check that the views are consistent Check that the MPC output is $\textsc{Accept}$

Zero-Knowledge Protocol

Soundness error:

$$p + (1-p) \cdot \frac{1}{N}$$

Proof size:

- Inputs of $N-1$ parties:
  - Party $i < N$: a seed of λ bits
  - Last party:

$$\underbrace{k \cdot \log_2 |\mathbb{F}_{\mathrm{SD}}|}_{[\![x_A]\!]_N} + \underbrace{2w \cdot \log_2 |\mathbb{F}_{\mathrm{poly}}|}_{[\![Q]\!]_N, [\![P]\!]_N} + \underbrace{\lambda}_{[\![a]\!]_N, [\![b]\!]_N} + \underbrace{\log_2 |\mathbb{F}_{\mathrm{points}}|}_{[\![c]\!]_N}$$

- Communication between parties: 2 elements of $\mathbb{F}_{\mathrm{points}}$.
- 2 hash digests ($2 \times 2\lambda$ bits),
- Some commitment randomness + COM_{i^*}

Comparison Zero-Knowledge Protocol for SD

Name Protocol	Year	Instance 1	Instance 2
Stern	1993	37.4 KB	46.1 KB
Véron	1997	31.7 KB	38.7 KB
CVE10	2010	-	37.4 KB
GPS21 (short)	2021	-	15.2 KB
GPS21 (fast)	2021	-	19.9 KB
FJR21 (short)	2021	13.6 KB	16.4 KB
FJR21 (fast)	2021	20.7 KB	25.6 KB
FJR22 (short)	2022	9.7 KB	6.9 KB
FJR22 (fast)	2022	14.4 KB	9.7 KB
Field size q		2	256
Code length m		1280	208
Code dimension k		$m/2$	$m/2$
Hamming weight w		132	78
Security level λ		128	128

Prove only an inequality

Table of Contents

Fiat-Shamir Transform

Signature algorithm:

Inputs:

- x such that $y = Hx$ and $\mathrm{wt}(x) \leq w$
- the message mess to sign

1. Prepare the witness, *i.e.* the polynomials P and Q.
2. Commit to party's inputs in distinct commitments $\mathrm{COM}_1, \ldots, \mathrm{COM}_N$.
3. $r, \varepsilon = \mathrm{Hash}(\mathsf{mess}, \mathsf{salt}, \mathrm{COM}_1, \ldots, \mathrm{COM}_N)$.
4. Run the MPC protocol π for each party.
5. $i^* = \mathrm{Hash}(\mathsf{mess}, \mathsf{salt}, r, \varepsilon, \text{broadcast messages})$.
6. Build the signature with the views of all the parties except the party i^*.

Security of the signature

$$\boxed{\text{5-round Identification Scheme} \Rightarrow \text{Signature}}$$

Attack of [KZ20]:

$$\mathrm{cost}_{\mathrm{forge}} := \min_{\tau_1,\tau_2 : \tau_1+\tau_2=\tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} p^i (1-p)^{\tau-i}} + N^{\tau_2} \right\}$$

[KZ20] Daniel Kales and Greg Zaverucha. *An attack on some signature schemes constructed from five-pass identification schemes.* CANS 2020.

Parameters selected

Variant 1: SD over $\mathbb{F}_2$,

$$(m, k, w) = (1280, 640, 132)$$

$$We\ have\ \mathbb{F}_{poly} = \mathbb{F}_{2^{11}}.$$

Parameters selected

Variant 1: SD over $\mathbb{F}_2$,

$$(m, k, w) = (1280, 640, 132)$$

We have $\mathbb{F}_{poly} = \mathbb{F}_{2^{11}}$.

Variant 2: SD over $\mathbb{F}_2$,

$$(m, k, w) = (1536, 888, 120)$$

but we split $x := (x_1 \mid \ldots \mid x_6)$ into 6 chunks and we prove that $\mathrm{wt}(x_i) \leq \frac{w}{6}$ for all i.

We have $\mathbb{F}_{poly} = \mathbb{F}_{2^8}$.

Parameters selected

Variant 3: SD over $\mathbb{F}_{2^8}$,

$$(m, k, w) = (256, 128, 80)$$

We have $\mathbb{F}_{poly} = \mathbb{F}_{2^8}$.

Performances

	Security Assumption	Computation Field
Variant 1	Over $\mathbb{F}_2$	$\mathbb{F}_{2048}$
Variant 2	Over $\mathbb{F}_2$	$\mathbb{F}_{256}$
Variant 3	Over $\mathbb{F}_{256}$	$\mathbb{F}_{256}$

Two trade-offs:

Fast: $N = 32$, $\tau = 27$

Short: $N = 256$, $\tau = 17$

Comparison Code-based Signatures (1/2)

Scheme Name	$\|\mathsf{sgn}\|$	$\|\mathsf{pk}\|$	t_{sgn}	t_{verif}
BGS21	24.1 KB	0.1 KB	-	-
BGS21	22.5 KB	1.7 KB	-	-
GPS21 - 256	22.2 KB	0.11 KB	-	-
GPS21 - 1024	19.5 KB	0.12 KB	-	-
FJR21 (fast)	22.6 KB	0.09 KB	13 ms	12 ms
FJR21 (short)	16.0 KB	0.09 KB	62 ms	57 ms
BGKM22 - Sig1	23.7 KB	0.1 KB	-	-
BGKM22 - Sig2	20.6 KB	0.2 KB	-	-
BGKM22 - Sig3	17.0 KB	0.2 KB	-	-
FJR22 (v1-fast)	15.6 KB	0.09 KB	-	-
FJR22 (v1-short)	10.9 KB	0.09 KB	-	-
FJR22 (v2-fast)	17.0 KB	0.09 KB	13 ms	13 ms
FJR22 (v2-short)	11.8 KB	0.09 KB	64 ms	61 ms
FJR22 (256-fast)	11.5 KB	0.14 KB	6 ms	6 ms
FJR22 (256-short)	8.26 KB	0.14 KB	30 ms	27 ms

Comparison Code-based Signatures (2/2)

Scheme Name	$\lvert\mathsf{sgn}\rvert$	$\lvert\mathsf{pk}\rvert$	t_{sgn}	t_{verif}
Durandal - I	3.97 KB	14.9 KB	4 ms	5 ms
Durandal - II	4.90 KB	18.2 KB	5 ms	6 ms
LESS-FM - I	15.2 KB	9.78 KB	-	-
LESS-FM - II	5.25 KB	205 KB	-	-
LESS-FM - III	10.39 KB	11.57 KB	-	-
Wave	2.07 KB	3.2 MB	300 ms	-
FJR22 (v1-fast)	15.6 KB	0.09 KB	-	-
FJR22 (v1-short)	10.9 KB	0.09 KB	-	-
FJR22 (v2-fast)	17.0 KB	0.09 KB	13 ms	13 ms
FJR22 (v2-short)	11.8 KB	0.09 KB	64 ms	61 ms
FJR22 (256-fast)	11.5 KB	0.14 KB	6 ms	6 ms
FJR22 (256-short)	8.26 KB	0.14 KB	30 ms	27 ms

Conclusion

Summary

- ☞ New signature scheme with Syndrome Decoding
- ☞ Small "signature size + public key size"

Future Work

- ☞ Optimize the signature implementation.
- ☞ Search parameter sets which provide better performances.