Signature Scheme from Restricted Errors

Violetta Weger

CBCrypto 2023
International Workshop on Code-Based Cryptography
April 23, 2023

Marco Baldi, Sebastian Bitzer
Alessio Pavoni, Paolo Santini
Antonia Wachter-Zeh
Motivation

2016 NIST standardization call for post-quantum PKE/KEM and signatures
Motivation

2016 NIST standardization call for post-quantum PKE/KEM and signatures

- PKE/KEM: 1 lattice-based, round 4: 3 code-based
- Signature schemes: 1 hash-based and 2 based on ideal lattices
Motivation

2016  NIST standardization call for post-quantum PKE/KEM and signatures

- PKE/KEM: 1 lattice-based, round 4: 3 code-based
- Signature schemes: 1 hash-based and 2 based on ideal lattices

2022  NIST reopened standardization call for signature schemes
Idea of Signature Schemes

Signer

Key Generation

Secret key $S$, public key $P$

Signing

Message $m$, signature $\sigma$

Verifier

$P$ →

$\overrightarrow{m,\sigma}$ →

Verification

Verify $\sigma$
Idea of Signature Schemes

Signer

Key Generation

Secret key $S$, public key $P$

Signing

Message $m$, signature $\sigma$

Verifier

$P \rightarrow m,\sigma$ 

Verification

Verify $\sigma$

Two approaches to get a code-based signature scheme:

- Hash-and-sign
- Through ZK protocol
Idea of Signature Schemes

Signer

Key Generation

Secret key \( S \), public key \( P \)

Signing

Message \( m \), signature \( \sigma \)

Verifier

\( P \)

\( m, \sigma \)

Verification

Verify \( \sigma \)

Two approaches to get a code-based signature scheme:

- Hash-and-sign
  - large public key sizes
- Through ZK protocol
  - large signature sizes
Idea of Signature Schemes

Signer

Key Generation

Secret key $S$, public key $P$

Signing

Message $m$, signature $\sigma$

Verifier

Two approaches to get a code-based signature scheme:

- **Hash-and-sign**
  - large public key sizes
  - Stefan’s talk: FuLeeca

- **Through ZK protocol**
  - large signature sizes
  - this talk: restricted errors
Idea of Signature Schemes

**Signer**

**Key Generation**
- Secret key $S$, public key $P$

**Signing**
- Message $m$, signature $\sigma$

**Verifier**

Two approaches to get a code-based signature scheme:

- **Hash-and-sign**
  - large public key sizes
  - Stefan’s talk: FuLeeca

- **Through ZK protocol**
  - large signature sizes
  - this talk: restricted errors
Idea of ZK Protocol

**Prover**

- $S$: secret, $P$: related public key
- $c$: commitments to secret
- $r_b$: response to challenge $b$

**Verifier**

- $b$: challenge
  - Recover $c$ from $r_b$ and $P$

- Complete: a honest prover gets accepted
- Zero-knowledge: verifier does not gain information on $S$
- Sound: small probability of an impersonator getting accepted
- $\alpha$: cheating probability, $\lambda$: bit security level
- Rounds: have to repeat ZK protocol $N$ times: $2^\lambda < (1/\alpha)^N$
Idea of ZK Protocol

### Prover

- $S$: secret, $\mathcal{P}$: related public key
- $c$: commitments to secret
- $r_b$: response to challenge $b$

### Verifier

- $b$: challenge
- Recover $c$ from $r_b$ and $\mathcal{P}$

- **complete**: a honest prover gets accepted
- **zero-knowledge**: verifier does not gain information on $S$
- **sound**: small probability of an impersonator getting accepted
Idea of ZK Protocol

**Prover**

- $\mathcal{S}$: secret, $\mathcal{P}$: related public key
- $c$: commitments to secret
- $r_b$: response to challenge $b$

**Interaction**

- $\mathcal{P}, c \rightarrow b$
- $b \leftarrow r_b$

**Verifier**

- $b$: challenge
- Recover $c$ from $r_b$ and $\mathcal{P}$

- **complete**: a honest prover gets accepted
- **zero-knowledge**: verifier does not gain information on $\mathcal{S}$
- **sound**: small probability of an impersonator getting accepted

---

**Violetta Weger — Signature Scheme from Restricted Errors**

2/11
Idea of ZK Protocol

**Prover**
- $S$: secret, $P$: related public key
- $c$: commitments to secret
- $b$: Hash of message, $c$
- $r_b$: response to challenge $b$

**Verifier**
- Recover $c$ from $r_b$ and $P$
- Verify $b = \text{Hash}(m, c)$

- **complete**: a honest prover gets accepted
- **zero-knowledge**: verifier does not gain information on $S$
- **sound**: small probability of an impersonator getting accepted
Idea of ZK Protocol

**Prover**

- $S$: secret
- $P$: related public key
- $c$: commitments to secret
- $b$: Hash of message, $c$
- $r_b$: response to challenge $b$

**Verifier**

- Recover $c$ from $r_b$ and $P$
- Verify $b = \text{Hash}(m, c)$

- **complete**: a honest prover gets accepted
- **zero-knowledge**: verifier does not gain information on $S$
- **sound**: small probability of an impersonator getting accepted
- **$\alpha$ cheating probability**, $\lambda$ bit security level
- **Rounds**: have to repeat ZK protocol $N$ times: $2^\lambda < (1/\alpha)^N$
Code-based ZK Protocols

ZK protocol \(\xrightarrow{\text{Fiat-Shamir}}\) Signature scheme


Syndrome Decoding Problem

Given parity-check matrix \(H\), syndrome \(s\), weight \(t\), find \(e\) s.t.

1. \(s = eH^\top\)
2. \(\text{wt}_H(e) \leq t\)

Prover

\(S:\) \(e\) of weight \(t\),
\(\mathcal{P}:\) random \(H, s = eH^\top, t\)
\(c_1:\) commitment to syndrome equation 1.
\(c_2:\) commitment to weight 2.
response: \(r_1 = \varphi, r_2 = \varphi(e)\)

Verifier

\(\xrightarrow{\mathcal{P}}\)

\(b \in \{1, 2\}\)

\(\xleftarrow{b}\)

\(r_b\) recover \(c_b\) from \(r_b\) and \(\mathcal{P}\)
Code-based ZK Protocols

- ZK protocol → Fiat-Shamir → Signature scheme


Syndrome Decoding Problem

Given parity-check matrix $H$, syndrome $s$, weight $t$, find $e$ s.t.
1. $s = eH^\top$
2. $\text{wt}_H(e) \leq t$

Prover
- $S$: $e$ of weight $t$,
- $P$: random $H$, $s = eH$
- $c_1$: commitment to syndrome
- $c_2$: commitment to weight 2.
- response: $r_1 = \varphi$, $r_2 = \varphi(e)$

Verifier
- Problem: large cheating probability $\rightarrow$ big signature sizes
- $r_b$ recover $c_b$ from $r_b$ and $P$
Performance of Classical Approach

Example

- $\lambda = 128$ bit security level $\rightarrow N = 135$ $\rightarrow$ public key size: 832 b
- $q = 31, n = 256, k = 204$ $\rightarrow$ signature size: 43 kB
## Performance of Classical Approach

### Example

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>128 bit security level</td>
<td>$N = 135$</td>
</tr>
<tr>
<td>$q$</td>
<td>31</td>
<td>$n = 256$</td>
</tr>
<tr>
<td>$n$</td>
<td>256</td>
<td>$k = 204$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public key size</td>
<td>832 b</td>
<td></td>
</tr>
<tr>
<td>Signature size</td>
<td>43 kB</td>
<td></td>
</tr>
</tbody>
</table>

For a long time not been considered practical.


## Performance of Classical Approach

### Example

- $\lambda = 128$ bit security level $\rightarrow N = 135$ $\rightarrow$ public key size: 832 b
- $q = 31, n = 256, k = 204$ $\rightarrow$ signature size: 43 kB

for a long time not been considered practical

Recent improvements through in-the-head computations

$\rightarrow$ smaller signature sizes $\sim 10$ kB

---


Performance of Classical Approach

Example

- $\lambda = 128$ bit security level $\rightarrow N = 135$ $\rightarrow$ public key size: 832 b
- $q = 31$, $n = 256$, $k = 204$ $\rightarrow$ signature size: 43 kB

for a long time not been considered practical

Recent improvements through in-the-head computations $\rightarrow$ smaller signature sizes $\sim 10$ kB

Problem of Classical Approach

Classical CVE (1 round)

- public key size: seed of $H$, $s$; $\log_2(q)(n - k) < 0.1$ kB
- signature size: $\text{Hash}(m, c)$ and response: transformation $\varphi$ or $\varphi(e)$
Problem of Classical Approach

### Classical CVE (1 round)

- public key size: seed of $H$, $s$; $\log_2(q)(n - k) < 0.1$ kB
- signature size: $\text{Hash}(m, c)$ and response: transformation $\varphi$ or $\varphi(e)$

Which $\varphi$ are allowed?
Problem of Classical Approach

Classical CVE (1 round)

- public key size: seed of $H$, $s$; $\log_2(q)(n-k) < 0.1$ kB
- signature size: Hash$(m,c)$ and response: transformation $\varphi$ or $\varphi(e)$

Which $\varphi$ are allowed?

Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, weight $t$, find $e \in \mathbb{F}_q^n$ such that $s = eH^\top$ and $\text{wt}_H(e) \leq t$.

\[
\begin{array}{cccc}
e & 0 & 0 & 0 & 0 \\
\end{array}
\xrightarrow{\varphi} \begin{array}{cccc}
0 & 0 & 0 & 0 & e'
\end{array}
\]
Problem of Classical Approach

Classical CVE (1 round)

- public key size: seed of $H$, $s$; $\log_2(q)(n - k) < 0.1$ kB
- signature size: $\text{Hash}(m, c)$ and response: transformation $\varphi$ or $\varphi(e)$

Which $\varphi$ are allowed?

Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_{q}^{n-k}$, weight $t$, find $e \in \mathbb{F}_q^n$ such that $s = eH^\top$ and $\text{wt}_H(e) \leq t$.

$e \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \xrightarrow{\varphi} \begin{array}{cccc} 0 & \color{#e67e22}0 & \color{#e67e22}0 & 0 \end{array} e'$

$\rightarrow \varphi$: linear isometries of Hamming metric:
permutation + scalar multiplication
Problem of Classical Approach

Classical CVE (1 round)

- public key size: seed of $H, s$; $\log_2(q)(n - k) < 0.1$ kB
- signature size: $\varphi(e) : t \log_2(q - 1) + t \log_2(n)$ or $\varphi : n \log_2(q - 1) + n \log_2(n)$

Which $\varphi$ are allowed?

Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k)\times n}, s \in \mathbb{F}_q^{n-k},$ weight $t$, find $e \in \mathbb{F}_q^n$ such that $s = eH^\top$ and $\text{wt}_H(e) \leq t$.

$\rightarrow \varphi : \text{linear isometries of Hamming metric:}
\text{permutation + scalar multiplication}$
Restricted Errors

Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, weight $t$, find $e \in \mathbb{F}_q^n$ such that $s = eH^\top$ and $\text{wt}(e) \leq t$.

Can we avoid permutations - but keep the hardness of the problem?
Restricted Errors

Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, weight $t$, find $e \in \mathbb{F}_q^n$ such that $s = eH^\top$ and $\text{wt}(e) \leq t$.

Can we avoid permutations - but keep the hardness of the problem?

↓

Restricted Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, syndrome $s \in \mathbb{F}_q^{n-k}$, $E \subseteq \mathbb{F}_q^\ast$, find $e \in E^n$ such that $s = eH^\top$. 
Restricted Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, $E \subseteq \mathbb{F}_q^*$, find $e \in E^n$ such that $s = eH^\top$. 
Restricted Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$, $E \subseteq \mathbb{F}_q^*$, find $e \in \mathbb{F}^n$ such that $s = eH^\top$.

Idea

- $g \in \mathbb{F}_q^*$ of order $z$,
- $E = \{g^i \mid i \in \{1, \ldots, z\}\}$
Restricted Errors


Restricted Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, $E \subseteq \mathbb{F}_q^*$, find $e \in \mathbb{E}^n$ such that $s = eH^\top$.

**Idea**

- $g \in \mathbb{F}_q^*$ of order $z$,
  $E = \{g^i \mid i \in \{1, \ldots, z\}\}$
- transf. $\varphi : \mathbb{E}^n \rightarrow \mathbb{E}^n$, $e \mapsto e \ast e'$ for $e' \in \mathbb{E}^n$
- size of $\varphi$ is $n \log_2(z)$
  (instead of $n \log_2((q-1)n)$)
Benefits of Restricted Errors

- Larger cost of solvers than for classical SDP
  → Recall talk of Sebastian
- Size of $\varphi$ and $\varphi(e)$ is smaller
- Computations are easier (in $\mathbb{F}_z$ instead of $\mathbb{F}_q$)
  → can choose smaller parameters
  → smaller signature sizes
  → smaller running times

We can replace SDP with Restricted SDP in any code-based ZK protocol

Example GPS for $\lambda = 128$, $q = 128$, $n = 220$, $k = 101$, $t = 90$
→ signature size: 24.6 kB

Example Rest. GPS for $\lambda = 128$, $q = 67$, $n = 147$, $k = 63$, $z = 11$
→ signature size: 14.8 kB

S. Gueron, E. Persichetti, P. Santini. “Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup”
Benefits of Restricted Errors

- Larger cost of solvers than for classical SDP → Recall talk of Sebastian
- Size of $\varphi$ and $\varphi(e)$ is smaller → smaller signature sizes
- Computations are easier (in $\mathbb{F}_z$ instead of $\mathbb{F}_q$) → smaller running times

We can replace SDP with Restricted SDP in any code-based ZK protocol

---

**Example GPS for $\lambda = 128$**

$$q = 128, n = 220, k = 101, t = 90$$

→ signature size: 24.6 kB

**Example Rest. GPS for $\lambda = 128$**

$$q = 67, n = 147, k = 63, z = 11$$

→ signature size: 14.8 kB

---

S. Gueron, E. Persichetti, P. Santini. “Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup”
Benefits of Restricted Errors

- Larger cost of solvers than for classical SDP
  → Recall talk of Sebastian
- Size of $\varphi$ and $\varphi(e)$ is smaller
- Computations are easier (in $\mathbb{F}_z$ instead of $\mathbb{F}_q$)
  → can choose smaller parameters
  → smaller signature sizes
  → smaller running times

We can replace SDP with Restricted SDP in any code-based ZK protocol

Example GPS for $\lambda = 128$

$q = 128, n = 220, k = 101, t = 90$
→ signature size: 24.6 kB

Example Rest. GPS for $\lambda = 128$

$q = 67, n = 147, k = 63, z = 11$
→ signature size: 14.8 kB

S. Gueron, E. Persichetti, P. Santini. “Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup”

But we can do even better: Restricted SDP in a subgroup $G$
Restricted-G SDP

$(\mathbb{E}^n, \star)$ is an abelian group isomorphic to $(\mathbb{F}_z^n, +)$

**Restricted Syndrome Decoding Problem**

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, $\mathbb{E} \subseteq \mathbb{F}_q^\star$, find $e \in \mathbb{E}^n$ s.t. $s = eH^\top$. 
Restricted-G SDP

$(\mathbb{E}^n, \star)$ is an abelian group isomorphic to $(\mathbb{F}_z^n, +)$ $\rightarrow$ Subgroup $(G, \star) \leq (\mathbb{E}^n, \star)$

\[ G = \langle x_1, \ldots, x_m \rangle = \left\{ \prod_{i=1}^{m} x_i^{u_i} \mid u_i \in \{1, \ldots, z\} \right\} \]

Restricted Syndrome Decoding Problem

Given $H \in \mathbb{F}_q^{(n-k)\times n}$, $s \in \mathbb{F}_q^{n-k}$, $E \subseteq \mathbb{F}_q^*$, find $e \in \mathbb{E}^n$ s.t. $s = eH^\top$. 
Restricted-$G$ SDP

$(\mathbb{E}^n, \star)$ is an abelian group isomorphic to $(\mathbb{F}_z^n, +) \rightarrow \text{Subgroup (} G, \star \leq (\mathbb{E}^n, \star) \text{)}$

$$G = \langle x_1, \ldots, x_m \rangle = \left\{ \prod_{i=1}^{m} x_i^{u_i} \mid u_i \in \{1, \ldots, z\} \right\}$$

**Restricted-$G$ Syndrome Decoding Problem**

Given $H \in \mathbb{F}_q^{(n-k) \times n}$, $s \in \mathbb{F}_q^{n-k}$, $\mathbb{E} \subseteq \mathbb{F}_q^*$, $G = \langle x_1, \ldots, x_m \rangle \leq \mathbb{E}^n$ find $e \in G$ s.t. $s = eH^\top$. 
**Restricted-\(G\) SDP**

\((\mathbb{E}^n, \star)\) is an abelian group isomorphic to \((\mathbb{F}_z^n, +)\) \(\rightarrow\) Subgroup \((G, \star) \leq (\mathbb{E}^n, \star)\)

\[
G = \langle x_1, \ldots, x_m \rangle = \left\{ \prod_{i=1}^{m} x_i^{u_i} \mid u_i \in \{1, \ldots, z\} \right\}
\]

**Restricted-\(G\) Syndrome Decoding Problem**

Given \(H \in \mathbb{F}_q^{(n-k) \times n}\), \(s \in \mathbb{F}_q^{n-k}\), \(E \subseteq \mathbb{F}_q^*\), \(G = \langle x_1, \ldots, x_m \rangle \leq \mathbb{E}^n\) find \(e \in G\) s.t. \(s = eH^\top\).

<table>
<thead>
<tr>
<th>Classical</th>
<th>Rest.</th>
<th>Rest.-(G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n \log_2((q - 1)n))</td>
<td>(n \log_2(z))</td>
<td>(m \log_2(z))</td>
</tr>
</tbody>
</table>
Example

- \( q = 13, n = 4, g = 3 \), \( \rightarrow \) multiplicative order \( z = 3 \);
  \[ \mathbb{E} = \{g^0 = 1, g^1 = 3, g^2 = 9\} \]

- E.g. \( e = (1, 9, 3, 3) \in \mathbb{E}^n \)

- \( m = 3 \), generators
  \[ x_1 = (g^2, g^0, g^2, g^0), \quad x_2 = (g^2, g^2, g^0, g^2, g^2), \quad x_3 = (g^0, g^2, g^2, g^1). \]

- \( G = \langle x_1, x_2, x_3 \rangle \)

- E.g. \( x_1^2 \star x_2^1 \star x_3^0 = (g^0, g^2, g^1, g^2) = (1, 9, 3, 9) \in G \), but \( e = (1, 9, 3, 3) \notin G \)

- \( |G| = z^m = 9 \), easy check:
  \[ M_G = \begin{pmatrix} 2 & 0 & 2 & 0 \\ 2 & 2 & 0 & 2 \\ 0 & 2 & 2 & 1 \end{pmatrix} \in \mathbb{F}_z^{m \times n} \]
Performance of Restricted SDP in $G$ Signatures

**Example GPS for $\lambda = 128$**

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
<th>Signature Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical GPS</td>
<td>$q = 128, n = 220, k = 101, t = 90$</td>
<td>$\rightarrow$ signature size: 24.6 kB</td>
</tr>
<tr>
<td>Restricted GPS</td>
<td>$q = 67, n = 147, k = 63, z = 11$</td>
<td>$\rightarrow$ signature size: 14.8 kB</td>
</tr>
<tr>
<td>Restricted-$G$ GPS</td>
<td>$q = 53, n = 82, k = 47, z = 13, m = 54$</td>
<td>$\rightarrow$ signature size: 12.7 kB</td>
</tr>
</tbody>
</table>
Performance of Restricted SDP in $G$ Signatures

**Example BG for $\lambda = 128$**

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
<th>Signature Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical BG:</td>
<td>$q = 997, n = 61, k = 33, t = 31$</td>
<td>$8.9$ kB</td>
</tr>
<tr>
<td>Restricted BG:</td>
<td>$q = 991, n = 77, k = 38, z = 33$</td>
<td>$9.5$ kB</td>
</tr>
<tr>
<td>Restricted-G BG:</td>
<td>$q = 1019, n = 40, k = 16, z = 509, m = 18$</td>
<td>$7.2$ kB</td>
</tr>
</tbody>
</table>

---

L. Bidoux, P. Gaborit. “Shorter Signatures from Proofs of Knowledge for the SD, MQ, PKP and RSD Problems”
Performance of Restricted SDP in $G$ Signatures

Example BG for $\lambda = 128$

- Classical BG: $q = 997, n = 61, k = 33, t = 31$ → signature size: 8.9 kB
- Restricted BG: $q = 991, n = 77, k = 38, z = 33$ → signature size: 9.5 kB
- Restricted-G BG: $q = 1019, n = 40, k = 16, z = 509, m = 18$ → signature size: 7.2 kB

L. Bidoux, P. Gaborit. “Shorter Signatures from Proofs of Knowledge for the SD, MQ, PKP and RSD Problems”

Conclusion/Open Questions

- Can replace classical SDP with Restricted SDP/ Restricted-G SDP in any code-based ZK protocol.
- Achieve smaller signature sizes, smaller running times
- Can we exploit the commutativity of the restricted transformations?
Questions?

CROSS

Codes & Restricted Objects Signature Scheme
http://cross-crypto.com/

Thank you!
Running times

Running time given in kCycles, CROSS has only PoC, no optimization, parallelization

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Key gen.</th>
<th>Signature gen.</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS</td>
<td>1794</td>
<td>5802</td>
<td>6506</td>
</tr>
<tr>
<td>Dilithium</td>
<td>49</td>
<td>140</td>
<td>61</td>
</tr>
<tr>
<td>CROSS</td>
<td>19</td>
<td>187</td>
<td>184</td>
</tr>
</tbody>
</table>
Solving Restricted SDP in subgroup $G$

- Recall Sebastian’s talk: we want $q, z$ such that $\mathbb{E}$ has no additive structure.
- Publicly known: $x_1, \ldots, x_m$ generators of multiplicative group $G$.
- $x_\ell = (g_{i_1, \ell}, \ldots, g_{i_n, \ell})$.
- Define $M_G \in \mathbb{F}_z^{m \times n}$ having rows $(i_1, \ell, \ldots, i_n, \ell)$.

$$M_G = \begin{bmatrix} i_{1, \ell} & \cdots & i_{n, \ell} \end{bmatrix}$$

$m' \geq \min \left\{ |J|, \frac{\lambda}{\log_2(z)} \right\} \rightarrow$ no improvement over enumerating all possible errors in these positions.
## Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Public Key size</th>
<th>Signature size</th>
<th>Total size</th>
<th>Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS+</td>
<td>&lt;0.1</td>
<td>16.7</td>
<td>16.7</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>&lt;0.1</td>
<td>7.7</td>
<td>7.7</td>
<td>Short</td>
</tr>
<tr>
<td>Falcon</td>
<td>0.9</td>
<td>0.6</td>
<td>1.5</td>
<td>-</td>
</tr>
<tr>
<td>Dilithium</td>
<td>1.3</td>
<td>2.4</td>
<td>3.7</td>
<td>-</td>
</tr>
<tr>
<td>CROSS</td>
<td>0.1</td>
<td>7.7</td>
<td>7.8</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>7.2</td>
<td>7.3</td>
<td>Short</td>
</tr>
<tr>
<td>GPS</td>
<td>0.1</td>
<td>24.0</td>
<td>24.1</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>19.8</td>
<td>19.9</td>
<td>Short</td>
</tr>
<tr>
<td>FJR</td>
<td>0.1</td>
<td>22.6</td>
<td>22.7</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>16.0</td>
<td>16.1</td>
<td>Short</td>
</tr>
<tr>
<td>SDItH</td>
<td>0.1</td>
<td>11.5</td>
<td>11.6</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>8.3</td>
<td>8.4</td>
<td>Short</td>
</tr>
<tr>
<td>Ret. of SDItH</td>
<td>0.1</td>
<td>12.1</td>
<td>12.1</td>
<td>Fast, V3</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>5.7</td>
<td>5.8</td>
<td>Shortest, V3</td>
</tr>
</tbody>
</table>
# Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Public Key size</th>
<th>Signature size</th>
<th>Total size</th>
<th>Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAVE</td>
<td>3200</td>
<td>2.1</td>
<td>3202</td>
<td>-</td>
</tr>
<tr>
<td>Durandal</td>
<td>15.2</td>
<td>4.1</td>
<td>19.3</td>
<td>-</td>
</tr>
<tr>
<td>Ideal Rank BG</td>
<td>0.5</td>
<td>8.4</td>
<td>8.9</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>6.1</td>
<td>6.6</td>
<td>Short</td>
</tr>
<tr>
<td>MinRank Fen</td>
<td>18.2</td>
<td>9.3</td>
<td>27.5</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>18.2</td>
<td>7.1</td>
<td>25.3</td>
<td>Short</td>
</tr>
<tr>
<td>Rank SDP Fen</td>
<td>0.9</td>
<td>7.4</td>
<td>8.3</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>5.9</td>
<td>6.8</td>
<td>Short</td>
</tr>
<tr>
<td>Beu</td>
<td>0.1</td>
<td>18.4</td>
<td>18.5</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>12.1</td>
<td>12.2</td>
<td>Short</td>
</tr>
<tr>
<td>PKP BG</td>
<td>0.1</td>
<td>9.8</td>
<td>9.9</td>
<td>Fast</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>8.8</td>
<td>8.9</td>
<td>Short</td>
</tr>
<tr>
<td>FuLeeca</td>
<td>0.4</td>
<td>0.3</td>
<td>0.7</td>
<td>-</td>
</tr>
</tbody>
</table>
Hash-and-Sign: CFS

<table>
<thead>
<tr>
<th>PROVER</th>
<th>VERIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY GENERATION</td>
<td></td>
</tr>
<tr>
<td>$S = H$ parity-check matrix</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P} = (t, HP)$ permuted $H$</td>
<td></td>
</tr>
<tr>
<td>SIGNING</td>
<td></td>
</tr>
<tr>
<td>Choose message $m$</td>
<td></td>
</tr>
<tr>
<td>$s = \text{Hash}(m)$</td>
<td></td>
</tr>
<tr>
<td>Find $e$: $s = eH^\top = eP(HP)^\top$, and $\text{wt}(e) \leq t$</td>
<td></td>
</tr>
<tr>
<td>$m,eP$</td>
<td></td>
</tr>
<tr>
<td>VERIFICATION</td>
<td></td>
</tr>
<tr>
<td>Check if $\text{wt}(eP) \leq t$ and $eP(HP)^\top = \text{Hash}(m)$</td>
<td></td>
</tr>
</tbody>
</table>
Hash-and-Sign: CFS

<table>
<thead>
<tr>
<th>PROVER</th>
<th>VERIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY GENERATION</td>
<td></td>
</tr>
<tr>
<td>$S = H$ parity-check matrix</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P} = (t, HP)$ permuted $H$</td>
<td></td>
</tr>
<tr>
<td>SIGNING</td>
<td></td>
</tr>
<tr>
<td>Choose message $m$</td>
<td></td>
</tr>
<tr>
<td>$s = \text{Hash}(m)$</td>
<td></td>
</tr>
<tr>
<td>Find $e$: $s = eH^\top = eP(HP)^\top$, and $\text{wt}(e) \leq t$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$m,eP$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERIFICATION</td>
</tr>
<tr>
<td></td>
<td>Check if $\text{wt}(eP) \leq t$</td>
</tr>
<tr>
<td></td>
<td>and $eP(HP)^\top = \text{Hash}(m)$</td>
</tr>
</tbody>
</table>

Problem: Distinguishability
## Hash-and-Sign: CFS

<table>
<thead>
<tr>
<th>PROVER</th>
<th>VERIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KEY GENERATION</strong></td>
<td></td>
</tr>
<tr>
<td>$S = H$ parity-check matrix</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P} = (t, HP)$ permuted $H$</td>
<td></td>
</tr>
<tr>
<td><strong>SIGNING</strong></td>
<td></td>
</tr>
<tr>
<td>Choose message $m$</td>
<td></td>
</tr>
<tr>
<td>$s = \text{Hash}(m)$</td>
<td></td>
</tr>
<tr>
<td>Find $e$: $s = eH^\top = eP(HP)^\top$, and $\text{wt}(e) \leq t$</td>
<td></td>
</tr>
<tr>
<td>$m, eP$</td>
<td></td>
</tr>
<tr>
<td><strong>VERIFICATION</strong></td>
<td></td>
</tr>
<tr>
<td>Check if $\text{wt}(eP) \leq t$ and $eP(HP)^\top = \text{Hash}(m)$</td>
<td></td>
</tr>
</tbody>
</table>

Not any $s$ is syndrome of low weight $e$
ZKID

**PROVER**

<table>
<thead>
<tr>
<th>commitments $c_0, c_1$</th>
<th>$c_0, c_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>response $r_b$</td>
<td>$b$</td>
</tr>
</tbody>
</table>

$\leftarrow b \in \{0, 1\}$

$\rightarrow r_b$

$\rightarrow$ Verify $c_b$ using $r_b, \mathcal{P}$

**VERIFIER**

**SIGNING**

Choose message $m$

Construct signature $s$ from $\mathcal{S}, m$

$\rightarrow m, s$

$\rightarrow$ Verify signature $s$ using $\mathcal{P}, m$

Signature Scheme
**PROVER**

commitments $c_0, c_1$

response $r_b$

**VERIFIER**

$\overset{c_0, c_1}{\leftarrow} b$

$b \in \{0, 1\}$

$\overset{r_b}{\rightarrow}$

Verify $c_b$ using $r_b, \mathcal{P}$

---

**SIGNING**

Choose message $m$

Construct signature $s$ from $\mathcal{S}, m$

$\overset{m, s}{\rightarrow}$

**VERIFICATION**

Verify signature $s$ using $\mathcal{P}, m$

---

**Fiat-Shamir**

Signature Scheme
**Fiat-Shamir**

<table>
<thead>
<tr>
<th>PROVER</th>
<th>VERIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KEY GENERATION</strong></td>
<td></td>
</tr>
<tr>
<td>Given $\mathcal{P}, \mathcal{S}$ of some ZKID and message $m$</td>
<td></td>
</tr>
<tr>
<td><strong>SIGNING</strong></td>
<td></td>
</tr>
<tr>
<td>Choose commitment $c$</td>
<td></td>
</tr>
<tr>
<td>$b = \text{Hash}(m, c)$</td>
<td></td>
</tr>
<tr>
<td>Compute response $r_b$</td>
<td></td>
</tr>
<tr>
<td>Signature $s = (b, r_b)$</td>
<td>$m, s$</td>
</tr>
<tr>
<td><strong>VERIFICATION</strong></td>
<td></td>
</tr>
<tr>
<td>Using $r_b, \mathcal{P}$ construct $c$</td>
<td></td>
</tr>
<tr>
<td>check if $b = \text{Hash}(m, c)$</td>
<td></td>
</tr>
</tbody>
</table>
### CVE

<table>
<thead>
<tr>
<th>PROVER</th>
<th>VERIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KEY GENERATION</strong></td>
<td></td>
</tr>
<tr>
<td>Choose $e$ with $\text{wt}(e) \leq t$</td>
<td>$\mathcal{P}=(H,s,t)$</td>
</tr>
<tr>
<td>$H$ parity-check matrix</td>
<td></td>
</tr>
<tr>
<td>Compute $s = eH^\top$</td>
<td></td>
</tr>
</tbody>
</table>

| **VERIFICATION** | |
| Choose $u \in \mathbb{F}_q^n$, $\sigma \in S_n$ | |
| Set $c_1 = \text{Hash}(\sigma, uH^\top)$ | |
| Set $c_2 = \text{Hash}(\sigma(u), \sigma(e))$ | $c_1,c_2$ |
| Choose $z \in \mathbb{F}_q^\times$ | $z$ |
| $y = \sigma(u + ze)$ | $y$ |
| $r_1 = \sigma$ | $b$ |
| $r_2 = \sigma(e)$ | $r_b$ |
| $b = 1$: $c_1 = \text{Hash}(\sigma, \sigma^{-1}(y)H^\top - zs)$ | |
| $b = 2$: $\text{wt}(\sigma(e)) = t$ | |
| and $c_2 = \text{Hash}(y - z\sigma(e), \sigma(e))$ | |
**CVE**

<table>
<thead>
<tr>
<th>PROVER</th>
<th>VERIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KEY GENERATION</strong></td>
<td><strong>Recall SDP: (1) ( s = eH^\top ) (2) ( \text{wt}(e) \leq t )</strong></td>
</tr>
<tr>
<td>Choose ( e ) with ( \text{wt}(e) \leq t ) ( H ) parity-check matrix</td>
<td>( P=(H,s,t) )</td>
</tr>
<tr>
<td>Compute ( s = eH^\top )</td>
<td><strong>VERIFICATION</strong></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose ( u \in \mathbb{F}_q^n, \sigma \in S_n )</td>
<td></td>
</tr>
<tr>
<td>Set ( c_1 = \text{Hash}(\sigma, uH^\top) )</td>
<td></td>
</tr>
<tr>
<td>Set ( c_2 = \text{Hash}(\sigma(u), \sigma(e)) )</td>
<td>( c_1, c_2 )</td>
</tr>
<tr>
<td>( z )</td>
<td>Choose ( z \in \mathbb{F}_q^\times )</td>
</tr>
<tr>
<td>( y )</td>
<td></td>
</tr>
<tr>
<td>Set ( y = \sigma(u + ze) )</td>
<td></td>
</tr>
<tr>
<td>( r_1 = \sigma )</td>
<td>( b )</td>
</tr>
<tr>
<td>( r_2 = \sigma(e) )</td>
<td>( r_b )</td>
</tr>
<tr>
<td>( b = 1: c_1 = \text{Hash}(\sigma, \sigma^{-1}(y)H^\top - zs) )</td>
<td></td>
</tr>
<tr>
<td>( b = 2: \text{wt}(\sigma(e)) = t )</td>
<td></td>
</tr>
<tr>
<td>and ( c_2 = \text{Hash}(y - z\sigma(e), \sigma(e)) )</td>
<td></td>
</tr>
<tr>
<td><strong>PROVER</strong></td>
<td><strong>VERIFIER</strong></td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>KEY GENERATION</strong></td>
<td></td>
</tr>
<tr>
<td>Choose ( e ) with ( \text{wt}(e) \leq t )</td>
<td>( H ) parity-check matrix</td>
</tr>
<tr>
<td>Compute ( s = eH^\top )</td>
<td>( P = (H,s,t) )</td>
</tr>
<tr>
<td>( e ) with ( \text{wt}(e) \leq t )</td>
<td>( H ) parity-check matrix</td>
</tr>
<tr>
<td>Compute ( s = eH^\top )</td>
<td>( P = (H,s,t) )</td>
</tr>
</tbody>
</table>

**VERIFICATION**

Choose \( u \in \mathbb{F}_q^n \), \( \sigma \in S_n \)
Set \( c_1 = \text{Hash}(\sigma, uH^\top) \)
Set \( c_2 = \text{Hash}(\sigma(u), \sigma(e)) \)
Choose \( z \in \mathbb{F}_q^\times \)
Choose \( z \in \mathbb{F}_q^\times \)
Choose \( b \in \{1, 2\} \)

\[ r_1 = \sigma \]
\[ r_2 = \sigma(e) \]

Problem: big signature sizes

\[ y = \sigma(u + ze) \]
\[ r_1 = \sigma \]
\[ r_2 = \sigma(e) \]
Cheating Probability

- Cheating probability = Probability of impersonator getting accepted
- For security level $2^\lambda$ want cheating probability $2^{-\lambda}$
- If cheating probability $\delta$, with $N$ rounds $\rightarrow$ cheating probability $\delta^N$
Cheating Probability

- Cheating probability = Probability of impersonator getting accepted
- For security level $2^\lambda$ want cheating probability $2^{-\lambda}$
- If cheating probability $\delta$, with $N$ rounds $\rightarrow$ cheating probability $\delta^N$
- might need many rounds: large communication cost
Cheating Probability

- Cheating probability = Probability of impersonator getting accepted
- For security level $2^\lambda$ want cheating probability $2^{-\lambda}$
- If cheating probability $\delta$, with $N$ rounds → cheating probability $\delta^N$
- might need many rounds: large communication cost
- solution: compression technique
- do not send $c_0^i, c_1^i$ in each round $i$
- before 1. round send $c = \text{Hash}(c_0^1, c_1^1, \ldots, c_0^N, c_1^N)$
- $i$th round: receiving challenge $b$ prover sends $r_b^i, c_1^i - b$
- end: verifier checks $c = \text{Hash}(c_0^1, c_1^1, \ldots, c_0^N, c_1^N)$

Cheating Probability

- Cheating probability = Probability of impersonator getting accepted
- For security level $2^\lambda$ want cheating probability $2^{-\lambda}$
- If cheating probability $\delta$, with $N$ rounds $\rightarrow$ cheating probability $\delta^N$
- might need many rounds: large communication cost
- other solution: MPC in the head
- third party: trusted helper sends commitments $\rightarrow \delta = 0$
- instead prover sends seeds of commitment: not ZK $\rightarrow$ cut and choose
- $x < N$ times send response, $N - x$ times send the seed of commitment
- to compress: use Merkle root or seed tree

### Comparison

<table>
<thead>
<tr>
<th></th>
<th>ZKID</th>
<th>Hash-and-Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduction to NP-hard</td>
<td></td>
<td></td>
</tr>
<tr>
<td>low public key size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>low signature size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fast verification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>ZKID</th>
<th>Hash-and-Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduction to NP-hard</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Low public key size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low signature size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast verification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Violetta Weger — Signature Scheme from Restricted Errors
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>ZKID</th>
<th>Hash-and-Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduction to NP-hard</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>low public key size</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>low signature size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fast verification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison

<table>
<thead>
<tr>
<th></th>
<th>ZKID</th>
<th>Hash-and-Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduction to NP-hard</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>low public key size</td>
<td>CVE: 70 B</td>
<td>WAVE: 3 MB</td>
</tr>
<tr>
<td>low signature size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fast verification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>ZKID</th>
<th>Hash-and-Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduction to NP-hard</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>low public key size</td>
<td>CVE: 70 B</td>
<td>WAVE: 3 MB</td>
</tr>
<tr>
<td>low signature size</td>
<td>∼</td>
<td>✓</td>
</tr>
<tr>
<td>fast verification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Violetta Weger — Signature Scheme from Restricted Errors
## Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>ZKID</th>
<th>Hash-and-Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduction to NP-hard</td>
<td>✓ ✓</td>
<td>× ×</td>
</tr>
<tr>
<td>low public key size</td>
<td>CVE: 70 B</td>
<td>WAVE: 3 MB</td>
</tr>
<tr>
<td>low signature size</td>
<td>CVE: 43 KB</td>
<td>WAVE: 1 KB</td>
</tr>
<tr>
<td>fast verification</td>
<td></td>
<td>NIST: 2 KB</td>
</tr>
</tbody>
</table>
## Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>ZKID</th>
<th>Hash-and-Sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>reduction to NP-hard</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>low public key size</td>
<td>CVE: 70 B</td>
<td>WAVE: 3 MB</td>
</tr>
<tr>
<td>low signature size</td>
<td>CVE: 43 KB</td>
<td>WAVE: 1 KB</td>
</tr>
<tr>
<td>fast verification</td>
<td>∼</td>
<td>✓</td>
</tr>
</tbody>
</table>