PIsignHD: A New Structure for the SQIsign Family with Flexible Applicability

Kaizhan Lin Weize Wang Chang-An Zhao Yunlei Zhao✉

August 14, 2025

Agenda

1 Introduction

2 SQIsignHD

3 PIsignHD

4 Implementation and Application

Isogeny-based Cryptography

- CRS scheme 1997-2006
- Inefficient, Attacked in subexponential time
- SIDH 2011
- Efficient, Short key size
- SQIsign 2020
- Parctical, Compact
- SIDH attack 2022
- Polynomial time, Isogenies in high dimension
- SQIsignHD 2023
- Fast signing, Compact, Isogenies in high dimension

Isogeny-based Cryptography

- CRS scheme 1997-2006
- Inefficient, Attacked in subexponential time
- SIDH 2011
- Efficient, Short key size
- SQIsign 2020
- Parctical, Compact
- SIDH attack 2022
- Polynomial time, Isogenies in high dimension
- SQIsignHD 2023
- Fast signing, Compact, Isogenies in high dimension

Isogeny-based Cryptography

- CRS scheme — 1997-2006
- Inefficient, Attacked in subexponential time
- SIDH — 2011
- Efficient, Short key size
- SQIsign — 2020
- Parctical, Compact
- SIDH attack — 2022
- Polynomial time, Isogenies in high dimension
- SQIsignHD — 2023
- Fast signing, Compact, Isogenies in high dimension

Isogeny-based Cryptography

- CRS scheme 1997-2006
- Inefficient, Attacked in subexponential time
- SIDH 2011
- Efficient, Short key size
- SQIsign 2020
- Parctical, Compact
- SIDH attack 2022
- Polynomial time, Isogenies in high dimension
- SQIsignHD 2023
- Fast signing, Compact, Isogenies in high dimension

Isogeny-based Cryptography

- CRS scheme 1997-2006
- Inefficient, Attacked in subexponential time
- SIDH 2011
- Efficient, Short key size
- SQIsign 2020
- Parctical, Compact
- SIDH attack 2022
- Polynomial time, Isogenies in high dimension
- SQIsignHD 2023
- Fast signing, Compact, Isogenies in high dimension

Agenda

1 Introduction

2 SQIsignHD

3 PIsignHD

4 Implementation and Application

Σ-protocol

Prover knows w with $(x, w) \in \mathcal{R}$, where $\mathcal{R}$ is an $\mathcal{NP}$-relation.

Prover		*Verifier*
generate commitment a	$\xrightarrow{a}$	
	$\xleftarrow{e}$	generate challenge e
generate response z	$\xrightarrow{z}$	
		output *accept* or *reject*

Fiat-Shamir paradigm: Σ-protocol⇒ Σ-signature
Main idea: $e = h(a||m)$, where h is a hash function and m is the message to be signed.

Σ-signature

- **Key Generation**: The signer

  generates $x = F(w)$ such that $(x, w) \in \mathcal{R}$
- F: one-way and polynomial-time computable function sk: w pk: x
- **Signature**: The signer
  1. randomly selects $r_P \in R_P$
  2. computes $a = f_a(r_P, x)$
  3. computes $e = h(a||m)$
  4. generates z w.r.t. (a, e)
  5. transmits (a, z) as the signature
- f_a: polynomial-time computable function h: hash function m: message
- **Verification**: The verifier
  1. computes $e = h(a||m)$
  2. accepts if (a, e, z) is a valid conversation

Σ-signature

- **Key Generation**: The signer

  generates $x = F(w)$ such that $(x, w) \in \mathcal{R}$
- F: one-way and polynomial-time computable function   sk: w pk: x
- **Signature**: The signer
  1. randomly selects $r_P \in R_P$
  2. computes $a = f_a(r_P, x)$
  3. computes $e = h(a||m)$
  4. generates z w.r.t. (a, e)
  5. transmits (a, z) as the signature
- f_a: polynomial-time computable function h: hash function m: message
- **Verification**: The verifier
  1. computes $e = h(a||m)$
  2. accepts if (a, e, z) is a valid conversation

Σ-signature

- **Key Generation**: The signer
  generates $x = F(w)$ such that $(x, w) \in \mathcal{R}$
- F: one-way and polynomial-time computable function sk: w pk: x
- **Signature**: The signer
  1 randomly selects $r_P \in R_P$
  2 computes $a = f_a(r_P, x)$
  3 computes $e = h(a||m)$
  4 generates z w.r.t. (a, e)
  5 transmits (a, z) as the signature
- f_a: polynomial-time computable function h: hash function m: message
- **Verification**: The verifier
  1. computes $e = h(a||m)$
  2. accepts if (a, e, z) is a valid conversation

SQIsignHD identification protocol

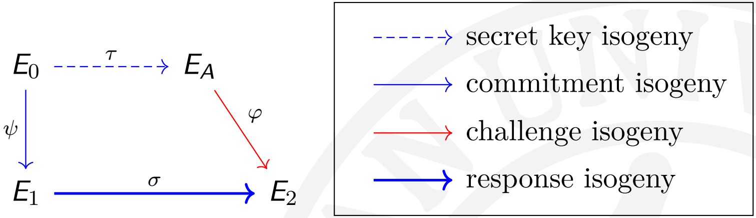

Figure: A sketch of the SQIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$
- transmit E_1
- Challenge: $\varphi : E_A \to E_2$
- transmit φ

- Response: $\sigma : E_1 \to E_2$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_1 to E_2

SQIsignHD identification protocol

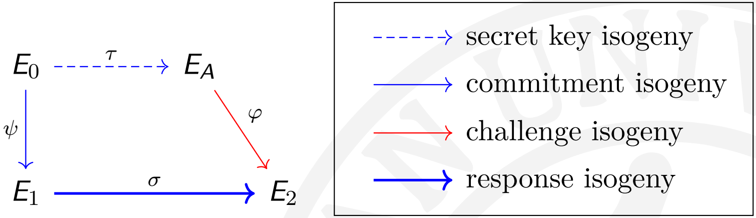

Figure: A sketch of the SQIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$
- transmit E_1
- Challenge: $\varphi : E_A \to E_2$
- transmit φ

- Response: $\sigma : E_1 \to E_2$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_1 to E_2

SQIsignHD identification protocol

$$
\begin{tikzcd}
E_0 \arrow[r, dashed, "\tau"] \arrow[d, "\psi"'] & E_A \arrow[dr, "\varphi"] & \\
E_1 \arrow[rr, "\sigma"] & & E_2
\end{tikzcd}
$$

- - -→ secret key isogeny
—→ commitment isogeny
—→ challenge isogeny
—→ response isogeny

Figure: A sketch of the SQIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$
- transmit E_1
- Challenge: $\varphi : E_A \to E_2$
- transmit φ

- Response: $\sigma : E_1 \to E_2$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_1 to E_2

SQIsignHD identification protocol

$$\begin{tikzcd}
E_0 \arrow[r, dashed, "\tau"] \arrow[d, "\psi"'] & E_A \arrow[d, "\varphi"] \\
E_1 \arrow[r, "\sigma"] & E_2
\end{tikzcd}$$

$\dashrightarrow$ secret key isogeny

$\longrightarrow$ commitment isogeny

$\longrightarrow$ challenge isogeny

$\longrightarrow$ response isogeny

Figure: A sketch of the SQIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$
- transmit E_1
- Challenge: $\varphi : E_A \to E_2$
- transmit φ

- Response: $\sigma : E_1 \to E_2$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_1 to E_2

SQIsignHD identification protocol

$$\begin{tikzcd}
E_0 \arrow[r, dashed, blue, "\tau"] \arrow[d, blue, "\psi"'] & E_A \arrow[d, red, "\varphi"] \\
E_1 \arrow[r, blue, thick, "\sigma"] & E_2
\end{tikzcd}$$

$\dashrightarrow$ secret key isogeny

$\longrightarrow$ commitment isogeny

$\longrightarrow$ challenge isogeny

$\longrightarrow$ response isogeny

Figure: A sketch of the SQIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$
- transmit E_1
- Challenge: $\varphi : E_A \to E_2$
- transmit φ
- Response: $\sigma : E_1 \to E_2$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_1 to E_2

SQIsignHD

- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. generate $\psi : E_0 \to E_1$
  2. generate $\varphi : E_A \to E_2$ w.r.t. $h(E_1 || m)$
  3. generate $\sigma : E_1 \to E_2$ w.r.t. (τ, ψ, φ)
  4. transmit (E_1, R) as the signature, where R is an efficient representation of σ
- Verify:
  1. generate $\varphi : E_A \to E_2$ w.r.t. $h(E_1 || m)$
  2. accept if σ is an isogeny from E_1 to E_2

SQIsignHD

- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. generate $\psi : E_0 \to E_1$
  2. generate $\varphi : E_A \to E_2$ w.r.t. $h(E_1||m)$
  3. generate $\sigma : E_1 \to E_2$ w.r.t. (τ, ψ, φ)
  4. transmit (E_1, R) as the signature, where R is an efficient representation of σ
- Verify:
  1. generate $\varphi : E_A \to E_2$ w.r.t. $h(E_1||m)$
  2. accept if σ is an isogeny from E_1 to E_2

SQIsignHD

- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. generate $\psi : E_0 \to E_1$
  2. generate $\varphi : E_A \to E_2$ w.r.t. $h(E_1||m)$
  3. generate $\sigma : E_1 \to E_2$ w.r.t. (τ, ψ, φ)
  4. transmit (E_1, R) as the signature, where R is an efficient representation of σ
- Verify:
  1. generate $\varphi : E_A \to E_2$ w.r.t. $h(E_1||m)$
  2. accept if σ is an isogeny from E_1 to E_2

Agenda

1 Introduction

2 SQIsignHD

3 PIsignHD

4 Implementation and Application

Γ-protocol

Prover knows w with $(x, w) \in \mathcal{R}$, where $\mathcal{R}$ is an $\mathcal{NP}$-relation.

Prover		*Verifier*
generate commitment a generate random string d	$\xrightarrow{a, d}$	
	$\xleftarrow{e}$	generate challenge e
generate response z	$\xrightarrow{z}$	
		output *accept* or *reject*

Γ-transformation: Γ-protocol $\Rightarrow$ Γ-signature

Main idea: $d = h_1(a)$ and $e = h_2(m)$, where h_1, h_2 are hash functions and m is the message to be signed.

Γ-signature

- **Key Generation**: The signer

  generates $x = F(w)$ such that $(x, w) \in \mathcal{R}$
- F: one-way and polynomial-time computable function sk: w pk: x
- **Signature**: The signer
  1. randomly selects $r_P \in R_P$
  2. computes $a = f_a(r_P, x)$
  3. computes $d = h_1(a)$ and $e = h_2(m)$
  4. generates z w.r.t. (a, d, e)
  5. transmits (a, z) as the signature.
- f_a: polynomial-time computable function h_1, h_2: hash functions m: message
- **Verification**: The verifier
  1. computes $e = h_2(m)$
  2. computes $d = h_1(a)$
  3. accepts if (a, d, e, z) is a valid conversation

Γ-signature

- **Key Generation**: The signer

  generates $x = F(w)$ such that $(x, w) \in \mathcal{R}$

- F: one-way and polynomial-time computable function sk: w pk: x

- **Signature**: The signer
  1. randomly selects $r_P \in R_P$
  2. computes $a = f_a(r_P, x)$
  3. computes $d = h_1(a)$ and $e = h_2(m)$
  4. generates z w.r.t. (a, d, e)
  5. transmits (a, z) as the signature.

- f_a: polynomial-time computable function h_1, h_2: hash functions m: message

- **Verification**: The verifier
  1. computes $e = h_2(m)$
  2. computes $d = h_1(a)$
  3. accepts if (a, d, e, z) is a valid conversation

Γ-signature

- **Key Generation**: The signer

  generates $x = F(w)$ such that $(x, w) \in \mathcal{R}$
- F: one-way and polynomial-time computable function sk: w pk: x
- **Signature**: The signer
  1 randomly selects $r_P \in R_P$
  2 computes $a = f_a(r_P, x)$
  3 computes $d = h_1(a)$ and $e = h_2(m)$
  4 generates z w.r.t. (a, d, e)
  5 transmits (a, z) as the signature.
- f_a: polynomial-time computable function h_1, h_2: hash functions m: message
- **Verification**: The verifier
  1. computes $e = h_2(m)$
  2. computes $d = h_1(a)$
  3. accepts if (a, d, e, z) is a valid conversation

PIsignHD identification protocol

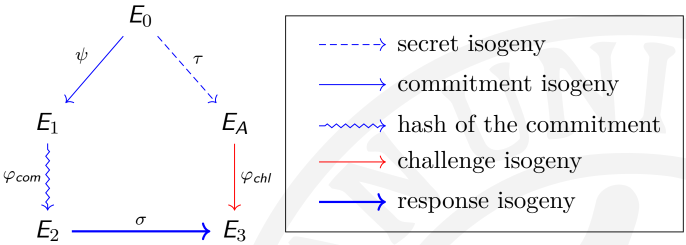

Figure: A sketch of the PIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$, $\varphi_{com} : E_1 \to E_2$
- transmit E_1, φ_{com}
- Challenge: $\varphi_{chl} : E_A \to E_3$
- transmit φ_{chl}
- Response: $\sigma : E_2 \to E_3$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_2 to E_3

PIsignHD identification protocol

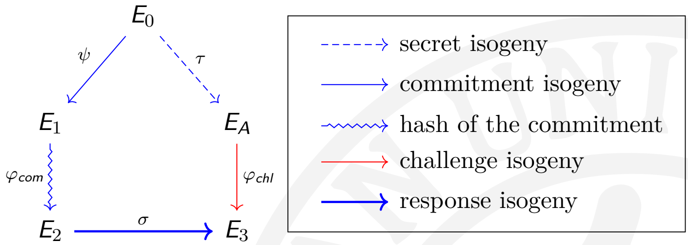

Figure: A sketch of the PIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$, $\varphi_{com} : E_1 \to E_2$
- transmit E_1, φ_{com}
- Challenge: $\varphi_{chl} : E_A \to E_3$
- transmit φ_{chl}
- Response: $\sigma : E_2 \to E_3$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_2 to E_3

PIsignHD identification protocol

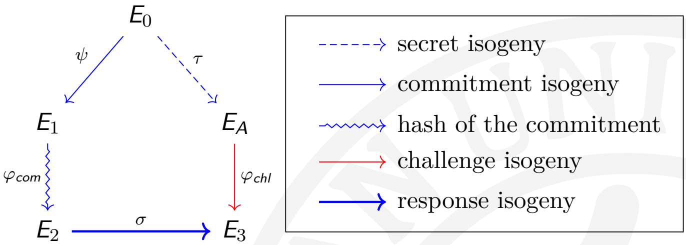

Figure: A sketch of the PIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$, $\varphi_{com} : E_1 \to E_2$
- transmit E_1, φ_{com}
- Challenge: $\varphi_{chl} : E_A \to E_3$
- transmit φ_{chl}
- Response: $\sigma : E_2 \to E_3$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_2 to E_3

PIsignHD identification protocol

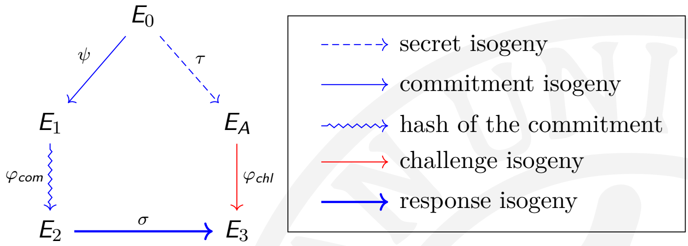

Figure: A sketch of the PIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$, $\varphi_{com} : E_1 \to E_2$
- transmit E_1, φ_{com}
- Challenge: $\varphi_{chl} : E_A \to E_3$
- transmit φ_{chl}
- Response: $\sigma : E_2 \to E_3$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_2 to E_3

PIsignHD identification protocol

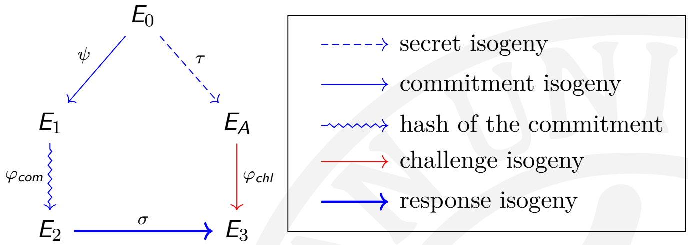

Figure: A sketch of the PIsignHD identification protocol

- Key Generation: $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Commitment: $\psi : E_0 \to E_1$, $\varphi_{com} : E_1 \to E_2$
- transmit E_1, φ_{com}
- Challenge: $\varphi_{chl} : E_A \to E_3$
- transmit φ_{chl}
- Response: $\sigma : E_2 \to E_3$
- transmit an efficient representation of σ
- Verify: σ is an isogeny from E_2 to E_3

PIsignHD

- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. generate $\psi : E_0 \to E_1$
  2. generate $\varphi_{com} : E_1 \to E_2$ w.r.t. $h_1(E_1)$
  3. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  4. generate $\sigma : E_2 \to E_3$ w.r.t. $(\tau, \psi, \varphi_{com}, \varphi_{chl})$
  5. transmit (E_1, R) as the signature, where R is an efficient representation of σ
- Verify:
  1. generate $\varphi_{com} : E_1 \to E_2$ w.r.t. $h_1(E_1)$
  2. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  3. accept if σ is an isogeny from E_2 to E_3

PIsignHD

- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. generate $\psi : E_0 \to E_1$
  2. generate $\varphi_{com} : E_1 \to E_2$ w.r.t. $h_1(E_1)$
  3. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  4. generate $\sigma : E_2 \to E_3$ w.r.t. $(\tau, \psi, \varphi_{com}, \varphi_{chl})$
  5. transmit (E_1, R) as the signature, where R is an efficient representation of σ
- Verify:
  1. generate $\varphi_{com} : E_1 \to E_2$ w.r.t. $h_1(E_1)$
  2. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  3. accept if σ is an isogeny from E_2 to E_3

PIsignHD

- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. generate $\psi : E_0 \to E_1$
  2. generate $\varphi_{com} : E_1 \to E_2$ w.r.t. $h_1(E_1)$
  3. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  4. generate $\sigma : E_2 \to E_3$ w.r.t. $(\tau, \psi, \varphi_{com}, \varphi_{chl})$
  5. transmit (E_1, R) as the signature, where R is an efficient representation of σ
- Verify:
  1. generate $\varphi_{com} : E_1 \to E_2$ w.r.t. $h_1(E_1)$
  2. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  3. accept if σ is an isogeny from E_2 to E_3

PIsignHD

Observation

1. The isogeny $\varphi_{com} : E_1 \to E_2$ can be recovered by:
   - E_1
   - E_2 and the kernel of $\hat{\varphi}_{com}$
2. E_2 can be public without the knowledge of m $(d = h_1(a))$

The signature could be $(E_2, \ker(\hat{\varphi}_{com}), R)$ (E_2 can be public and precomputed)

PIsignHD (Compressed)

- Setup: generate a list of $\psi : E_0 \to E_1$ and $\varphi_{com} : E_1 \to E_2$.
- Public: a list of E_2 Secret: a list of $\varphi_{com} \circ \psi$ and $\ker(\hat{\varphi}_{com})$
- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. select $\varphi_{com} \circ \psi : E_0 \to E_2$ and $\ker(\hat{\varphi}_{com})$
  2. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  2. generate $\sigma : E_2 \to E_3$ w.r.t. $(\tau, \varphi_{com} \circ \psi, \varphi_{chl})$
  5. transmit $(ind, \ker(\hat{\varphi}_{com}), R)$ as the signature, where R is an efficient representation of σ
- Verify:
  1. find out E_2 w.r.t. ind
  2. recover φ_{com} and check if φ_{com} can be generated w.r.t. E_1
  3. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  4. accept if σ is an isogeny from E_2 to E_3

PIsignHD (Compressed)

- Setup: generate a list of $\psi : E_0 \to E_1$ and $\varphi_{com} : E_1 \to E_2$.
- Public: a list of E_2 Secret: a list of $\varphi_{com} \circ \psi$ and $\ker(\hat{\varphi}_{com})$
- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1. select $\varphi_{com} \circ \psi : E_0 \to E_2$ and $\ker(\hat{\varphi}_{com})$
  2. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  2. generate $\sigma : E_2 \to E_3$ w.r.t. $(\tau, \varphi_{com} \circ \psi, \varphi_{chl})$
  5. transmit $(ind, \ker(\hat{\varphi}_{com}), R)$ as the signature, where R is an efficient representation of σ
- Verify:
  1. find out E_2 w.r.t. ind
  2. recover φ_{com} and check if φ_{com} can be generated w.r.t. E_1
  3. generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  4. accept if σ is an isogeny from E_2 to E_3

PIsignHD (Compressed)

- Setup: generate a list of $\psi : E_0 \to E_1$ and $\varphi_{com} : E_1 \to E_2$.
- Public: a list of E_2 Secret: a list of $\varphi_{com} \circ \psi$ and $\ker(\hat{\varphi}_{com})$
- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1 select $\varphi_{com} \circ \psi : E_0 \to E_2$ and $\ker(\hat{\varphi}_{com})$
  2 generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  2 generate $\sigma : E_2 \to E_3$ w.r.t. $(\tau, \varphi_{com} \circ \psi, \varphi_{chl})$
  5 transmit $(\mathit{ind}, \ker(\hat{\varphi}_{com}), R)$ as the signature, where R is an efficient representation of σ
- Verify:
  1 find out E_2 w.r.t. ind
  2 recover φ_{com} and check if φ_{com} can be generated w.r.t. E_1
  3 generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  4 accept if σ is an isogeny from E_2 to E_3

PIsignHD (Compressed)

- Setup: generate a list of $\psi : E_0 \to E_1$ and $\varphi_{com} : E_1 \to E_2$.
- Public: a list of E_2 Secret: a list of $\varphi_{com} \circ \psi$ and $\ker(\hat{\varphi}_{com})$
- Key Generation: generate $\tau : E_0 \to E_A$
- sk: τ pk: E_A
- Sign:
  1 select $\varphi_{com} \circ \psi : E_0 \to E_2$ and $\ker(\hat{\varphi}_{com})$
  2 generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  2 generate $\sigma : E_2 \to E_3$ w.r.t. $(\tau, \varphi_{com} \circ \psi, \varphi_{chl})$
  5 transmit $(ind, \ker(\hat{\varphi}_{com}), R)$ as the signature, where R is an efficient representation of σ
- Verify:
  1 find out E_2 w.r.t. ind
  2 recover φ_{com} and check if φ_{com} can be generated w.r.t. E_1
  3 generate $\varphi_{chl} : E_A \to E_3$ w.r.t. $h_2(m)$
  4 accept if σ is an isogeny from E_2 to E_3

Agenda

1 Introduction

2 SQIsignHD

3 PIsignHD

4 Implementation and Application

Implementation

Table: Signing comparisons of SQIsignHD and PIsignHD targeting the NIST-I security level. For the performance results (expressed in millions of clock cycles), we execute 1000 times for a 256-bit message and record the average time.

Implementation		SQIsignHD	PIsignHD
Sig. size (bits)	Original	870	870
	Compressed	-	519
Clock cycles ($\times 10^6$)	Original	70.1	89.8
	Offline (Uncom.)	57.9	77.8
	Online (Uncom.)	12.0	11.8
	Offline (Com.)	-	89.6
	Online (Com.)	-	11.8

Application

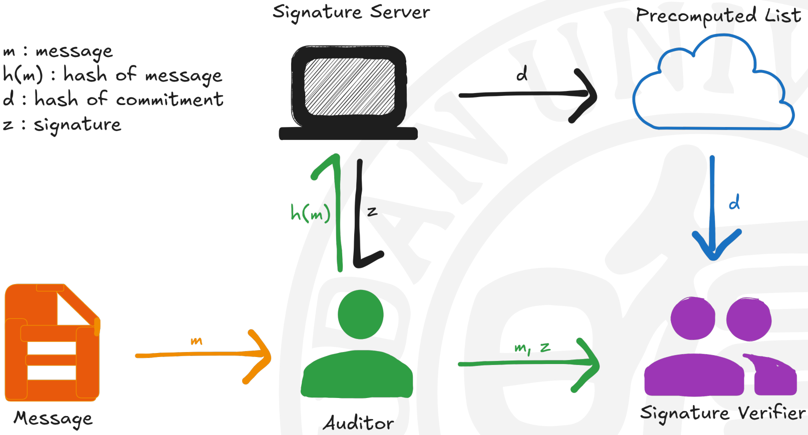

Figure: Application.

Thanks

Thanks!

https://eprint.iacr.org/2024/1404