

A FULLY COLLISION-RESISTANT CHAMELEON HASH FUNCTION BASED ON ISOGENIES

Joint work (in progress) with
Thomas den Hollander and Leon Weingarten

Sebastian A. Spindler • August 13th, 2026



Research Institute
Cyber Defence
University of the Bundeswehr Munich



CHAMELEON HASH FUNCTIONS (AKA TRAPDOOR COMMITMENTS)

Sometimes we would like to censor/update certain information in a signed message m ...

CHAMELEON HASH FUNCTIONS (AKA TRAPDOOR COMMITMENTS)

Sometimes we would like to censor/update certain information in a signed message m ...

... but what if we can't re-sign the new message m^* ?

CHAMELEON HASH FUNCTIONS (AKA TRAPDOOR COMMITMENTS)

Sometimes we would like to censor/update certain information in a signed message m ...

... but what if we can't re-sign the new message m^* ?

Recall hash-and-sign:

$$\sigma = \text{Sign}_{\text{sk}}(H(m, r))$$

CHAMELEON HASH FUNCTIONS (AKA TRAPDOOR COMMITMENTS)

Sometimes we would like to censor/update certain information in a signed message m ...

... but what if we can't re-sign the new message m^* ?

Recall hash-and-sign:

$$\sigma = \text{Sign}_{\text{sk}}(H(m, r))$$

Maybe we can find a collision $H(m, r) = H(m^*, r^*)$?

CHAMELEON HASH FUNCTIONS (AKA TRAPDOOR COMMITMENTS)

Sometimes we would like to censor/update certain information in a signed message m ...

... but what if we can't re-sign the new message m^* ?

Recall hash-and-sign:

$$\sigma = \text{Sign}_{\text{sk}}(H(m, r))$$

Maybe we can find a collision $H(m, r) = H(m^*, r^*)$?

Need additional functionality, and this should be 'controlled'. How?

CHAMELEON HASH FUNCTIONS (AKA TRAPDOOR COMMITMENTS)

Sometimes we would like to censor/update certain information in a signed message m ...

... but what if we can't re-sign the new message m^* ?

Recall hash-and-sign:

$$\sigma = \text{Sign}_{\text{sk}}(H(m, r))$$

Maybe we can find a collision $H(m, r) = H(m^*, r^*)$?

Need additional functionality, and this should be 'controlled'. How?

Chameleon hash functions!

Applications: Sanitizable Signatures, EUF-CMA transformation, Online/offline signatures, ...

THE CHAMELEON HASH ALGORITHMS

- $\text{CHParGen}(1^\lambda)$: Returns pp

THE CHAMELEON HASH ALGORITHMS

- $\text{CHParGen}(1^\lambda)$: Returns pp
- $\text{CHKeyGen}(\text{pp})$: Returns (sk, pk)

THE CHAMELEON HASH ALGORITHMS

- $\text{CHParGen}(1^\lambda)$: Returns pp
- $\text{CHKeyGen}(\text{pp})$: Returns (sk, pk)
- $\text{CHash}(\text{pk}, m)$: Returns hash-check pair (h, r)

THE CHAMELEON HASH ALGORITHMS

- $\text{CHParGen}(1^\lambda)$: Returns pp
- $\text{CHKeyGen}(\text{pp})$: Returns (sk, pk)
- $\text{CHash}(\text{pk}, m)$: Returns hash-check pair (h, r)
- $\text{CHCheck}(\text{pk}, h, m, r)$: Checks that (h, r) verifies for (pk, m)

THE CHAMELEON HASH ALGORITHMS

- $\text{CHParGen}(1^\lambda)$: Returns pp
- $\text{CHKeyGen}(\text{pp})$: Returns (sk, pk)
- $\text{CHash}(\text{pk}, m)$: Returns hash-check pair (h, r)
- $\text{CHCheck}(\text{pk}, h, m, r)$: Checks that (h, r) verifies for (pk, m)
- $\text{CHAdapt}(\text{sk}, h, m, m^*, r)$: Checks (pk, h, m, r)

THE CHAMELEON HASH ALGORITHMS

- $\text{CHParGen}(1^\lambda)$: Returns pp
- $\text{CHKeyGen}(\text{pp})$: Returns (sk, pk)
- $\text{CHash}(\text{pk}, m)$: Returns hash-check pair (h, r)
- $\text{CHCheck}(\text{pk}, h, m, r)$: Checks that (h, r) verifies for (pk, m)
- $\text{CHAdapt}(\text{sk}, h, m, m^*, r)$: Checks (pk, h, m, r) , then returns new r^* such that

$$\text{CHCheck}(\text{pk}, h, m^*, r^*) = 1$$

COLLISION RESISTANCE

Have multiple notions of collision resistance:

- **Weak collision resistance:** It is hard to find (h, m, r, m^*, r^*) with $m \neq m^*$ such that

$$\text{CHCheck}(\text{pk}, h, m, r) = 1 = \text{CHCheck}(\text{pk}, h, m^*, r^*)$$

COLLISION RESISTANCE

Have multiple notions of collision resistance:

- **Weak collision resistance:** It is hard to find (h, m, r, m^*, r^*) with $m \neq m^*$ such that

$$\text{CHCheck}(\text{pk}, h, m, r) = 1 = \text{CHCheck}(\text{pk}, h, m^*, r^*)$$

without having access to sk

COLLISION RESISTANCE

Have multiple notions of collision resistance:

- **Weak collision resistance:** It is hard to find (h, m, r, m^*, r^*) with $m \neq m^*$ such that

$$\text{CHCheck}(\text{pk}, h, m, r) = 1 = \text{CHCheck}(\text{pk}, h, m^*, r^*)$$

without having access to sk

- **Full collision resistance:** Having query-access to an sk -keyed CHAdapt -oracle

COLLISION RESISTANCE

Have multiple notions of collision resistance:

- **Weak collision resistance:** It is hard to find (h, m, r, m^*, r^*) with $m \neq m^*$ such that

$$\text{CHCheck}(\text{pk}, h, m, r) = 1 = \text{CHCheck}(\text{pk}, h, m^*, r^*)$$

without having access to sk

- **Full collision resistance:** Having query-access to an sk -keyed CHAdapt -oracle, it is hard to find (without access to sk) a collision with $m \neq m^*$ if

$$(h = \text{CHash}(\text{pk}, m), m)$$

was not queried previously

COLLISION RESISTANCE

Have multiple notions of collision resistance:

- **Weak collision resistance:** It is hard to find (h, m, r, m^*, r^*) with $m \neq m^*$ such that

$$\text{CHCheck}(\text{pk}, h, m, r) = 1 = \text{CHCheck}(\text{pk}, h, m^*, r^*)$$

without having access to sk

- **Full collision resistance:** Having query-access to an sk -keyed CHAdapt -oracle, it is hard to find (without access to sk) a collision with $m \neq m^*$ if

$$(h = \text{CHash}(\text{pk}, m), m)$$

was not queried previously

- **Standard CR** and **Enhanced CR:** h respectively m not queried previously

COLLISION RESISTANCE

Have multiple notions of collision resistance:

- **Weak collision resistance:** It is hard to find (h, m, r, m^*, r^*) with $m \neq m^*$ such that

$$\text{CHCheck}(\text{pk}, h, m, r) = 1 = \text{CHCheck}(\text{pk}, h, m^*, r^*)$$

without having access to sk

- **Full collision resistance:** Having query-access to an sk -keyed CHAdapt -oracle, it is hard to find (without access to sk) a collision with $m \neq m^*$ if

$$(h = \text{CHash}(\text{pk}, m), m)$$

was not queried previously

- **Standard CR** and **Enhanced CR:** h respectively m not queried previously
- Have full CR CHF's from **LPN** [Der+20] and from **SIS** [Cle+25]

CHAMELEON HASHING FROM ISOGENIES [BAS+26]

- Setup: $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$, E_{pk} supersingular

CHAMELEON HASHING FROM ISOGENIES [BAS+26]

- Setup: $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$, E_{pk} supersingular
- $N \in \mathbb{N}$ and a prime $\ell \in \mathbb{N}$ with $\gcd(\ell, N) = 1$

CHAMELEON HASHING FROM ISOGENIES [BAS+26]

- Setup: $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$, E_{pk} supersingular
- $N \in \mathbb{N}$ and a prime $\ell \in \mathbb{N}$ with $\gcd(\ell, N) = 1$
- Randomness: $\psi_r : E_{\text{pk}} \rightarrow E_h$ of degree $N \cdot \ell^k$

CHAMELEON HASHING FROM ISOGENIES [BAS+26]

- Setup: $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}}), E_{\text{pk}}$ supersingular
- $N \in \mathbb{N}$ and a prime $\ell \in \mathbb{N}$ with $\gcd(\ell, N) = 1$
- Randomness: $\psi_r : E_{\text{pk}} \rightarrow E_h$ of degree $N \cdot \ell^k$
- Hash: E_h and m -dependent masking of $\ker(\psi_r)[N]$

CHAMELEON HASHING FROM ISOGENIES [BAS+26]

- Setup: $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$, E_{pk} supersingular
- $N \in \mathbb{N}$ and a prime $\ell \in \mathbb{N}$ with $\gcd(\ell, N) = 1$
- Randomness: $\psi_r : E_{\text{pk}} \rightarrow E_h$ of degree $N \cdot \ell^k$
- Hash: E_h and m -dependent masking of $\ker(\psi_r)[N]$

Any collision reveals a non-trivial endomorphism

$$\begin{array}{ccc} & \xrightarrow{\psi_r} & \\ E_{\text{pk}} & & E_h \\ & \xleftarrow{\widehat{\psi_{r'}}} & \end{array}$$

CHAMELEON HASHING FROM ISOGENIES [BAS+26]

- Setup: $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$, E_{pk} supersingular
- $N \in \mathbb{N}$ and a prime $\ell \in \mathbb{N}$ with $\gcd(\ell, N) = 1$
- Randomness: $\psi_r : E_{\text{pk}} \rightarrow E_h$ of degree $N \cdot \ell^k$
- Hash: E_h and m -dependent masking of $\ker(\psi_r)[N]$

Any collision reveals a non-trivial **endomorphism**

$$\begin{array}{ccc} & \xrightarrow{\psi_r} & \\ E_{\text{pk}} & & E_h \\ & \xleftarrow{\widehat{\psi_{r'}}} & \end{array}$$

CHAMELEON HASHING FROM ISOGENIES [BAS+26]

- Setup: $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$, E_{pk} supersingular
- $N \in \mathbb{N}$ and a prime $\ell \in \mathbb{N}$ with $\gcd(\ell, N) = 1$
- Randomness: $\psi_r : E_{\text{pk}} \rightarrow E_h$ of degree $N \cdot \ell^k$
- Hash: E_h and m -dependent masking of $\ker(\psi_r)[N]$

Any collision reveals a non-trivial **endomorphism**

$$\begin{array}{ccc} & \psi_r & \\ & \curvearrowright & \\ E_{\text{pk}} & & E_h \\ & \curvearrowleft & \\ & \overline{\psi_{r'}} & \end{array}$$

$\leadsto$ Only achieves weak collision resistance!

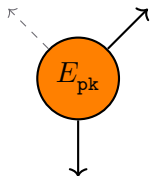
THE CGL HASH FUNCTION [CLG09]

General approach: Pick a starting curve E_{pk} and hash bit-string $(b_0, \dots, b_{t-1}) \in \{0, 1\}^t$ as follows:

THE CGL HASH FUNCTION [CLG09]

General approach: Pick a starting curve E_{pk} and hash bit-string $(b_0, \dots, b_{t-1}) \in \{0, 1\}^t$ as follows:

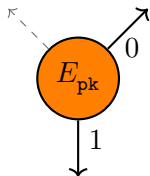
- At first step: Have 3 starting 2-isogenies from E_{pk} , one of which gets discarded in setup



THE CGL HASH FUNCTION [CLG09]

General approach: Pick a starting curve E_{pk} and hash bit-string $(b_0, \dots, b_{t-1}) \in \{0, 1\}^t$ as follows:

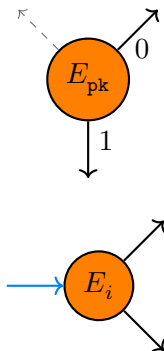
- At first step: Have 3 starting 2-isogenies from E_{pk} , one of which gets discarded in setup
- Choose initial 2-isogeny (in deterministic way) according to b_0



THE CGL HASH FUNCTION [CLG09]

General approach: Pick a starting curve E_{pk} and hash bit-string $(b_0, \dots, b_{t-1}) \in \{0, 1\}^t$ as follows:

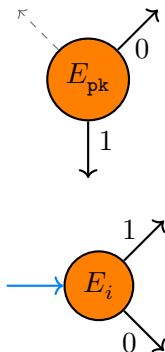
- At first step: Have 3 starting 2-isogenies from E_{pk} , one of which gets discarded in setup
- Choose initial 2-isogeny (in deterministic way) according to b_0
- At each following step, have incoming 2-isogeny



THE CGL HASH FUNCTION [CLG09]

General approach: Pick a starting curve E_{pk} and hash bit-string $(b_0, \dots, b_{t-1}) \in \{0, 1\}^t$ as follows:

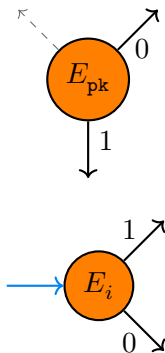
- At first step: Have 3 starting 2-isogenies from E_{pk} , one of which gets discarded in setup
- Choose initial 2-isogeny (in deterministic way) according to b_0
- At each following step, have incoming 2-isogeny
- Choose non-backtracking 2-isogeny deterministically according to b_i



THE CGL HASH FUNCTION [CLG09]

General approach: Pick a starting curve E_{pk} and hash bit-string $(b_0, \dots, b_{t-1}) \in \{0, 1\}^t$ as follows:

- At first step: Have 3 starting 2-isogenies from E_{pk} , one of which gets discarded in setup
- Choose initial 2-isogeny (in deterministic way) according to b_0
- At each following step, have incoming 2-isogeny
- Choose non-backtracking 2-isogeny deterministically according to b_i
- Output: Endpoint E_t after t steps



A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$

A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$

A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$
- CHCheck : Verify the hash via recomputation

A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$
- CHCheck : Verify the hash via recomputation

CHAdapt works as follows:

$$\begin{array}{ccc} E_{\text{pk}} & \xrightarrow{m} & \text{CGL}(E_{\text{pk}}, m) \\ m^* \downarrow & & \downarrow r \\ \text{CGL}(E_{\text{pk}}, m^*) & & h = \text{CGL}(E_{\text{pk}}, m || r) \end{array}$$

A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$
- CHCheck : Verify the hash via recomputation

CHAdapt works as follows:

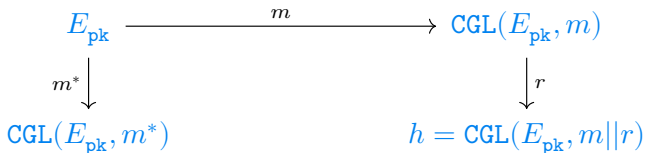
$$\begin{array}{ccc} E_{\text{pk}} & \xrightarrow{m} & \text{CGL}(E_{\text{pk}}, m) \\ m^* \downarrow & & \downarrow r \\ \text{CGL}(E_{\text{pk}}, m^*) & & h = \text{CGL}(E_{\text{pk}}, m || r) \end{array}$$

A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$
- CHCheck : Verify the hash via recomputation

CHAdapt works as follows:



A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$
- CHCheck : Verify the hash via recomputation

CHAdapt works as follows:

$$\begin{array}{ccc} E_{\text{pk}} & \xrightarrow{m} & \text{CGL}(E_{\text{pk}}, m) \\ \downarrow m^* & & \downarrow r \\ \text{CGL}(E_{\text{pk}}, m^*) & \xrightarrow[\text{IdealToIsogeny}]{\text{KLPT}} & h = \text{CGL}(E_{\text{pk}}, m || r) \end{array}$$

A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$
- CHCheck : Verify the hash via recomputation

CHAdapt works as follows:

$$\begin{array}{ccc} E_{\text{pk}} & \xrightarrow{m} & \text{CGL}(E_{\text{pk}}, m) \\ \downarrow m^* & & \downarrow r \\ \text{CGL}(E_{\text{pk}}, m^*) & \xrightarrow[\text{IdealToIsogeny}]{\text{KLPT}} & h = \text{CGL}(E_{\text{pk}}, m || r) \end{array}$$

Any collision reveals an endomorphism, so again only weak CR

A CGL-BASED CHAMELEON HASH: A FIRST ATTEMPT

We use the CGL hash function for a chameleon hash:

- $(\text{sk}, \text{pk}) = (\text{End}(E_{\text{pk}}), E_{\text{pk}})$
- $\text{CHash}(E_{\text{pk}}, m)$: Sample bitstring r , return $\text{CGL}(E_{\text{pk}}, m || r)$
- CHCheck : Verify the hash via recomputation

CHAdapt works as follows:

$$\begin{array}{ccc} E_{\text{pk}} & \xrightarrow{m} & \text{CGL}(E_{\text{pk}}, m) \\ \downarrow m^* & & \downarrow r \\ \text{CGL}(E_{\text{pk}}, m^*) & \xrightarrow[\text{IdealToIsogeny}]{\text{KLPT}} & h = \text{CGL}(E_{\text{pk}}, m || r) \end{array}$$

Any collision reveals an endomorphism, so again only weak CR

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ
- A non-interactive proof system Π for the relation

$$\mathcal{R} = \{((h, m), (r)) : \text{Open}_{\Gamma}(h, m, r) = 1\}$$

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ
- A non-interactive proof system Π for the relation

$$\mathcal{R} = \{((h, m), (r)) : \text{Open}_{\Gamma}(h, m, r) = 1\}$$

The main ideas:

- (sk, pk) satisfies $f(\text{sk}) = \text{pk}$

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ
- A non-interactive proof system Π for the relation

$$\mathcal{R} = \{((h, m), (r)) : \text{Open}_{\Gamma}(h, m, r) = 1\}$$

The main ideas:

- (sk, pk) satisfies $f(\text{sk}) = \text{pk}$
- $\text{CHash}(\text{pk}, m)$ returns (h, π) where $(h, r) \leftarrow \text{Commit}_{\Gamma}(m)$

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ
- A non-interactive proof system Π for the relation

$$\mathcal{R} = \{((h, m), (r)) : \text{Open}_{\Gamma}(h, m, r) = 1\}$$

The main ideas:

- (sk, pk) satisfies $f(\text{sk}) = \text{pk}$
- $\text{CHash}(\text{pk}, m)$ returns (h, π) where $(h, r) \leftarrow \text{Commit}_{\Gamma}(m)$ and

$$\pi \leftarrow \text{Prove}_{\Pi}((h, m), (r))$$

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ
- A non-interactive proof system Π for the relation

$$\mathcal{R} = \{((h, m), (r)) : \text{Open}_{\Gamma}(h, m, r) = 1\}$$

The main ideas:

- (sk, pk) satisfies $f(\text{sk}) = \text{pk}$
- $\text{CHash}(\text{pk}, m)$ returns (h, π) where $(h, r) \leftarrow \text{Commit}_{\Gamma}(m)$ and

$$\pi \leftarrow \text{Prove}_{\Pi}((h, m), (r))$$

- $\text{CHCheck}(\text{pk}, h, m, \pi)$ checks that π verifies for (h, m, pk)

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ
- A non-interactive proof system Π for the relation

$$\mathcal{R} = \{((h, m), (r)) : \text{Open}_{\Gamma}(h, m, r) = 1\}$$

The main ideas:

- (sk, pk) satisfies $f(\text{sk}) = \text{pk}$
- $\text{CHash}(\text{pk}, m)$ returns (h, π) where $(h, r) \leftarrow \text{Commit}_{\Gamma}(m)$ and

$$\pi \leftarrow \text{Prove}_{\Pi}((h, m), (r))$$

- $\text{CHCheck}(\text{pk}, h, m, \pi)$ checks that π verifies for (h, m, pk)
- $\text{CHAdapt}(\text{sk}, h, m, m^*, \pi)$: Check $(f(\text{sk}), h, m, \pi)$, then?

A GENERIC CONSTRUCTION [DER+20]

Building blocks:

- A one-way function f and a commitment scheme Γ
- A non-interactive proof system Π for the relation

$$\mathcal{R} = \{((h, m, y), (r, x)) : \text{Open}_{\Gamma}(h, m, r) = 1 \text{ or } f(x) = y\}$$

The main ideas:

- (sk, pk) satisfies $f(\text{sk}) = \text{pk}$
- $\text{CHash}(\text{pk}, m)$ returns (h, π) where $(h, r) \leftarrow \text{Commit}_{\Gamma}(m)$ and

$$\pi \leftarrow \text{Prove}_{\Pi}((h, m, \text{pk}), (r, \perp))$$

- $\text{CHCheck}(\text{pk}, h, m, \pi)$ checks that π verifies for (h, m, pk)
- $\text{CHAdapt}(\text{sk}, h, m, m^*, \pi)$: Check $(f(\text{sk}), h, m, \pi)$, then return

$$\pi^* \leftarrow \text{Prove}_{\Pi}((h, m^*, \text{pk}), (\perp, \text{sk}))$$

FULL COLLISION RESISTANCE: OUR CONTRIBUTION

[Der+20] prove:

f one-way	$\implies$	Generic construction is fully collision-resistant
Γ binding		
Π strong SE		

FULL COLLISION RESISTANCE: OUR CONTRIBUTION

[Der+20] prove:

f one-way
 Γ binding
 Π **strong** SE $\implies$ Generic construction is
fully collision-resistant

FULL COLLISION RESISTANCE: OUR CONTRIBUTION

[Der+20] prove:

f one-way
 Γ binding
 Π **strong** SE $\implies$ Generic construction is
fully collision-resistant

We im-prove:

f one-way
 Γ binding
 Π **weak** SE $\implies$ Generic construction is
fully collision-resistant

OUR INSTANTIATION FROM ISOGENIES

- One-way function: CGL hashing $y = \text{CGL}(E_0, x)$

OUR INSTANTIATION FROM ISOGENIES

- One-way function: CGL hashing $y = \text{CGL}(E_0, x)$
- Commitment scheme: Sterner commitments $h = \text{CGL}(E_0^*, m || r)$

OUR INSTANTIATION FROM ISOGENIES

- One-way function: CGL hashing $y = \text{CGL}(E_0, x)$
- Commitment scheme: Sterner commitments $h = \text{CGL}(E_0^*, m || r)$
- Proof system: Modification of SPRINT [dH+26]

OUR INSTANTIATION FROM ISOGENIES

- One-way function: CGL hashing $y = \text{CGL}(E_0, x)$
- Commitment scheme: Sterner commitments $h = \text{CGL}(E_0^*, m || r)$
- Proof system: Modification of SPRINT [dH+26]
- Security assumption: $\ell^\bullet$ -Isogeny problem

OUR INSTANTIATION FROM ISOGENIES

- One-way function: CGL hashing $y = \text{CGL}(E_0, x)$
- Commitment scheme: Sterner commitments $h = \text{CGL}(E_0^*, m || r)$
- Proof system: Modification of SPRINT [dH+26]
- Security assumption: $\ell^\bullet$ -Isogeny problem

Sterner needs trusted/multi-party setup for binding

OUR INSTANTIATION FROM ISOGENIES

- One-way function: CGL hashing $y = \text{CGL}(E_0, x)$
- Commitment scheme: Sterner commitments $h = \text{CGL}(E_0^*, m || r)$
- Proof system: Modification of SPRINT [dH+26]
- Security assumption: $\ell^\bullet$ -Isogeny problem

Sterner needs trusted/multi-party setup for binding, but:

Binding only needs to hold
for party without sk

OUR INSTANTIATION FROM ISOGENIES

- One-way function: CGL hashing $y = \text{CGL}(E_0, x)$
- Commitment scheme: Sterner commitments $h = \text{CGL}(E_0^*, m || r)$
- Proof system: Modification of SPRINT [dH+26]
- Security assumption: $\ell^\bullet$ -Isogeny problem

Sterner needs trusted/multi-party setup for binding, but:

Binding only needs to hold for party without sk	$\Rightarrow$	No trusted/multi-party setup needed!
--	---------------	---

SPRINT: TIMINGS AND SIZES

Prime p	Path length	Size (kB)	Prove (ms)	Verify (ms)
$5 \cdot 2^{248} - 1$	256	82	27	3.5
$65 \cdot 2^{376} - 1$	384	171	131	15
$27 \cdot 2^{500} - 1$	512	246	166	20

Table: Some DeepSPRINT benchmarks for reference

WORK IN PROGRESS

- Implementation
- Applications
- Writing
- Polishing
- ...

WORK IN PROGRESS

- Implementation
- Applications
- Writing
- Polishing
- ...
- **A multivariate instantiation**

Thank you for your attention!

- [Bas+26] A. Basso, L. De Feo, S. Patranabis, I. Radulescu, and B. Wesolowski. *Forensic categories: a framework for SQLsign-like primitives*.
- [Cle+25] S. Clermont, S. Düzlül, C. Janson, L. Porzenheim, and P. Struck. “Lattice-Based Sanitizable Signature Schemes: Chameleon Hash Functions and More”. In: *PQCrypto 2025*.
- [CLG09] D. X. Charles, K. E. Lauter, and E. Z. Goren. “Cryptographic Hash Functions from Expander Graphs”. In: *Journal of Cryptology* 22.1 (2009).
- [Der+20] D. Derler, S. Krenn, K. Samelin, and D. Slamanig. “Fully Collision-Resistant Chameleon-Hashes from Simpler and Post-quantum Assumptions”. In: *SCN 2020*.
- [dH+26] T. den Hollander, S. Levin, M. Mula, R. Pedersen, D. Slamanig, and S. A. Spindler. *SPRINT: New Isogeny Proofs of Knowledge and Isogeny-Based Signatures*.