FORENSIC CATEGORIES: A FRAMEWORK FOR SQISIGN-LIKE PRIMITIVES

ANDREA BASSO, LUCA DE FEO, SIKHAR PATRANABIS, **ILINCA RADULESCU***, BENJAMIN WESOLOWSKI

*ENS de Lyon & CNRS

OVERVIEW

- Motivation
- Background
- Effective Categories
- The Fingerprint
- Instantiation
- Protocols
- Conclusion

Motivation

MOTIVATION

- There are several versions of SQIsign, all with the same structure, so we want to extract what's essential

MOTIVATION

- There are several versions of SQIsign, all with the same structure, so we want to extract what's essential
- SQIsign without all the algebraic machinery - more accessible conceptually

MOTIVATION

- There are several versions of SQIsign, all with the same structure, so we want to extract what's essential
- SQIsign without all the algebraic machinery - more accessible conceptually
- A new perspective to formalize such concepts using category theory

Background

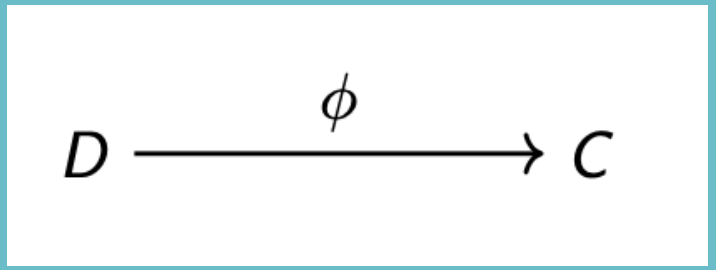
INTRODUCTION TO CATEGORIES

A (small) category is defined by a set of **objects** and a set of **morphisms** with the following properties:

INTRODUCTION TO CATEGORIES

A (small) category is defined by a set of **objects** and a set of **morphisms** with the following properties:

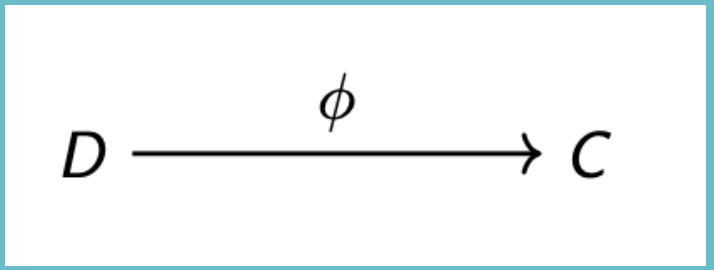
- Every morphism ϕ has a **domain** object D and a **codomain** object C, denoted by $\phi : D \rightarrow C$.

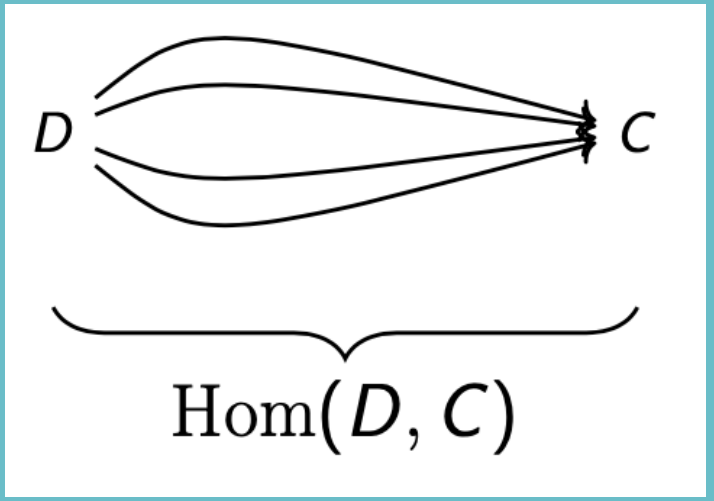


INTRODUCTION TO CATEGORIES

A (small) category is defined by a set of **objects** and a set of **morphisms** with the following properties:

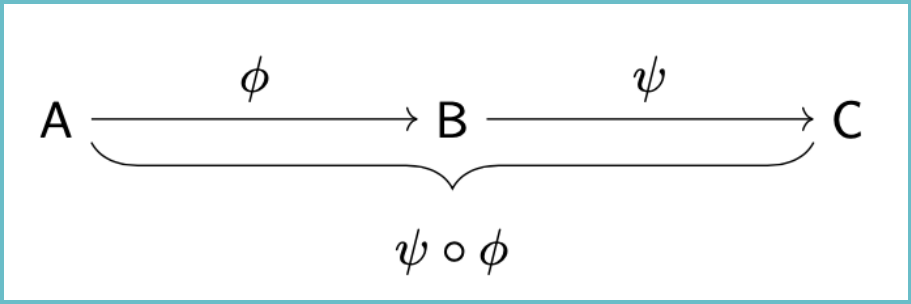
- Every morphism ϕ has a **domain** object D and a **codomain** object C, denoted by $\phi : D \to C$.
- The set of all morphisms with domain D and codomain C is denoted by Hom(D,C); this is called a **homset**





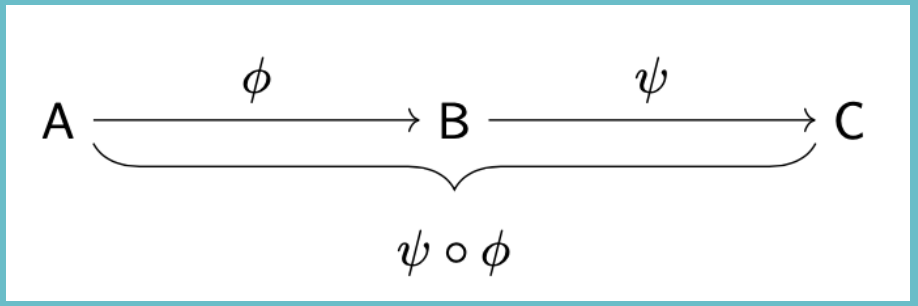
INTRODUCTION TO CATEGORIES

- There exists a **composition law**, written ∘, that maps a morphism $\phi : A \to B$ and a morphism $\psi : B \to C$ to a morphism $\psi \circ \phi : A \to C$, and that is *associative*: $(\psi \circ \phi) \circ \chi = \psi \circ (\phi \circ \chi)$.

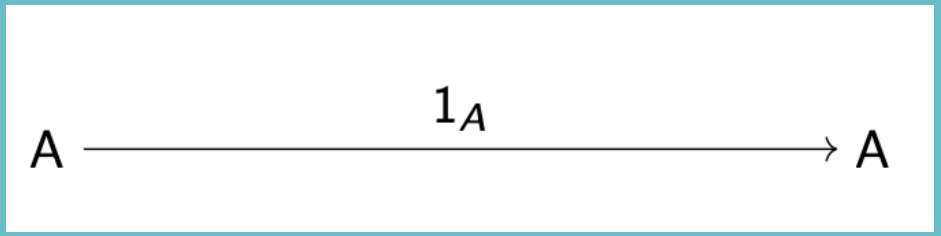


INTRODUCTION TO CATEGORIES

- There exists a **composition law**, written $\circ$, that maps a morphism $\phi : A \to B$ and a morphism $\psi : B \to C$ to a morphism $\psi \circ \phi : A \to C$, and that is *associative*: $(\psi \circ \phi) \circ \chi = \psi \circ (\phi \circ \chi)$.



- For every object A, there exists a morphism $1_A : A \to A$ such that $1_A \circ \phi = \phi$ and $\psi \circ 1_A = \psi$ for every $\phi : Z \to A$ and every $\psi : A \to B$



Effective Categories

PRE-EFFECTIVE CATEGORY FAMILY

PRE-EFFECTIVE CATEGORY FAMILY

Definition: A **pre-effective category family** is a sequence $(\mathscr{C}_{n)_{n\in\mathbb{N}}}$ where each $\mathscr{C}_n$ is a category, together with algorithms such that the following holds. For each n, we write $\lambda(n) = \log(n)$. Every object in $\mathscr{C}_n$ has a unique representation as a binary string of length polynomial in $\lambda(n)$, and there exist (probabilistic) polynomial time algorithms (in the length of the input) that can solve the following tasks:

PRE-EFFECTIVE CATEGORY FAMILY

Definition: A **pre-effective category family** is a sequence $(\mathscr{C}_{n)_{n\in\mathbb{N}}}$ where each $\mathscr{C}_n$ is a category, together with algorithms such that the following holds. For each n, we write $\lambda(n) = \log(n)$. Every object in $\mathscr{C}_n$ has a unique representation as a binary string of length polynomial in $\lambda(n)$, and there exist (probabilistic) polynomial time algorithms (in the length of the input) that can solve the following tasks:

- Given n and a binary string, decide whether it encodes an object or a morphism in $\mathscr{C}_n$.

PRE-EFFECTIVE CATEGORY FAMILY

Definition: A **pre-effective category family** is a sequence $(\mathscr{C}_{n)_{n\in\mathbb{N}}}$ where each $\mathscr{C}_n$ is a category, together with algorithms such that the following holds. For each n, we write $\lambda(n) = \log(n)$. Every object in $\mathscr{C}_n$ has a unique representation as a binary string of length polynomial in $\lambda(n)$, and there exist (probabilistic) polynomial time algorithms (in the length of the input) that can solve the following tasks:

- Given n and a binary string, decide whether it encodes an object or a morphism in $\mathscr{C}_n$.
- Given n and a morphism $\varphi : D \to C$ in $\mathscr{C}_n$, compute the domain D and codomain C.

PRE-EFFECTIVE CATEGORY FAMILY

Definition: A **pre-effective category family** is a sequence $(\mathscr{C}_{n)_{n\in\mathbb{N}}}$ where each $\mathscr{C}_n$ is a category, together with algorithms such that the following holds. For each n, we write $\lambda(n) = \log(n)$. Every object in $\mathscr{C}_n$ has a unique representation as a binary string of length polynomial in $\lambda(n)$, and there exist (probabilistic) polynomial time algorithms (in the length of the input) that can solve the following tasks:

- Given n and a binary string, decide whether it encodes an object or a morphism in $\mathscr{C}_n$.
- Given n and a morphism $\varphi : D \to C$ in $\mathscr{C}_n$, compute the domain D and codomain C.
- Given n and two morphisms $\varphi : A \to B$ and $\psi : B \to C$ in $\mathscr{C}_n$, return the composition $\psi \circ \varphi : A \to C$.

PRE-EFFECTIVE CATEGORY FAMILY

Definition: A **pre-effective category family** is a sequence $(\mathscr{C}_{n)_{n\in\mathbb{N}}}$ where each $\mathscr{C}_n$ is a category, together with algorithms such that the following holds. For each n, we write $\lambda(n) = \log(n)$. Every object in $\mathscr{C}_n$ has a unique representation as a binary string of length polynomial in $\lambda(n)$, and there exist (probabilistic) polynomial time algorithms (in the length of the input) that can solve the following tasks:

- Given n and a binary string, decide whether it encodes an object or a morphism in $\mathscr{C}_n$.
- Given n and a morphism $\varphi : D \to C$ in $\mathscr{C}_n$, compute the domain D and codomain C.
- Given n and two morphisms $\varphi : A \to B$ and $\psi : B \to C$ in $\mathscr{C}_n$, return the composition $\psi \circ \varphi : A \to C$.
- Given n and an object A in $\mathscr{C}_n$, compute the identity morphism $1_A : A \to A$.

WALKING IN A PRE-EFFECTIVE CATEGORY

WALKING IN A PRE-EFFECTIVE CATEGORY

WALKING IN A PRE-EFFECTIVE CATEGORY

Def (Walk): A **walk** in a pre-effective category is defined as a pair $(\mathbf{Walk}, \mu)$, where:

WALKING IN A PRE-EFFECTIVE CATEGORY

Def (Walk): A **walk** in a pre-effective category is defined as a pair (**Walk**, μ), where:

- **Walk** is a deterministic polynomial time algorithm which takes as input an object A and a sequence of random coins $r \in R^*$ (for some finite set R, e.g., $R = \{0,1\}$ and produces a pair (B, ψ), where ψ is a morphism $\psi : A \to B$, and

WALKING IN A PRE-EFFECTIVE CATEGORY

Def (Walk): A **walk** in a pre-effective category is defined as a pair (**Walk**, μ), where:

- **Walk** is a deterministic polynomial time algorithm which takes as input an object A and a sequence of random coins $r \in R^*$ (for some finite set R, e.g., $R = \{0,1\}$ and produces a pair (B, ψ), where ψ is a morphism $\psi : A \to B$, and
- μ is a distribution on objects (supported everywhere) such that when r is a sequence of n uniformly random coins, the statistical distance between μ and the distribution of the output's codomain B is a *negligible function* of n.

EFFECTIVE CATEGORY

EFFECTIVE CATEGORY

Pre-effective category

EFFECTIVE CATEGORY

Pre-effective category +

EFFECTIVE CATEGORY

Pre-effective category + Walk

EFFECTIVE CATEGORY

Pre-effective category + Walk +

EFFECTIVE CATEGORY

Pre-effective category + Walk + Origin object

EFFECTIVE CATEGORY

Pre-effective category + Walk + Origin object =

EFFECTIVE CATEGORY

EFFECTIVE CATEGORY

Pre-effective category + Walk + Origin object = Effective Category

Def (Effective category): An **effective category** is a *pre-effective category* together with a *walk* **Walk** and an *origin object* O whose representation can be computed in polynomial time.

The Fingerprint

DEFINITION

Def (Fingerprint): A **fingerprint** fp on a category $\mathscr{C}$ is a collection of maps

$$\mathsf{fp}: \mathsf{Hom}(-,-) \to \mathscr{M} \cup \{ \perp \},$$

such that $\mathscr{M}$ is finite and containing at least two fingerprints, i.e., $|\mathscr{M}| \geq 2$. The special symbol $\perp$ is used to denote those cases when fp is undefined.

Forensic Categories

DEFINITION

DEFINITION

Effective category

DEFINITION

Effective category

+

DEFINITION

Effective category

+

Fingerprint
(with special properties)

DEFINITION

Effective category + Fingerprint (with special properties) =

DEFINITION

Effective category + Fingerprint (with special properties) =

DEFINITION

Effective category + Fingerprint (with special properties) = Forensic category

DEFINITION

Effective category + Fingerprint (with special properties) = Forensic category

Def (Forensic category): A **forensic category** is an *effective category* where the *fingerprint* satisfies the following properties:

- Effective
- Hard (on average)
- Triangularizable
- (Computationally) Walk-Indistinguishable

EFFECTIVE

Given φ, computing $\mathrm{fp}(\varphi)$ is efficient.

HARD (ON AVERAGE)

The language

$$\mathscr{L} = \left\{ (A, (\varphi, \psi)) \mid \exists B \text{ s.t. } \left\{ \begin{array}{l} \varphi, \psi : A \to B, \\ \mathsf{fp}(\varphi) \neq \mathsf{fp}(\psi), \\ \mathsf{fp}(\varphi), \mathsf{fp}(\psi) \neq \bot \end{array} \right\} \right.$$

is hard on average for A following distribution μ.

HARD (ON AVERAGE)

The language

$$\mathscr{L} = \left\{ (A, (\varphi, \psi)) \mid \exists B \text{ s.t. } \left\{ \begin{array}{l} \varphi, \psi : A \to B, \\ \mathsf{fp}(\varphi) \neq \mathsf{fp}(\psi), \\ \mathsf{fp}(\varphi), \mathsf{fp}(\psi) \neq \bot \end{array} \right\} \right.$$

is hard on average for A following distribution μ.

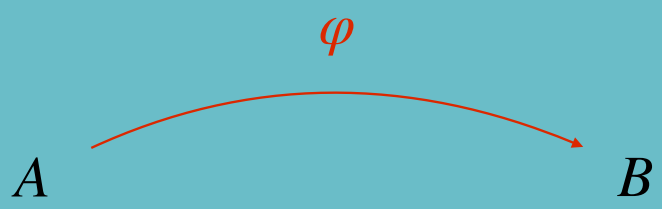
A

HARD (ON AVERAGE)

The language

$$\mathscr{L} = \left\{ (A, (\varphi, \psi)) \mid \exists B \text{ s.t. } \left\{ \begin{array}{l} \varphi, \psi : A \to B, \\ \mathsf{fp}(\varphi) \neq \mathsf{fp}(\psi), \\ \mathsf{fp}(\varphi), \mathsf{fp}(\psi) \neq \bot \end{array} \right\} \right.$$

is hard on average for A following distribution μ.

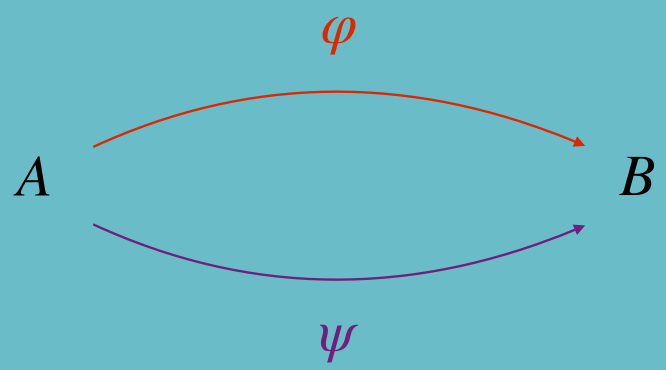


HARD (ON AVERAGE)

The language

$$\mathscr{L} = \left\{ (A, (\varphi, \psi)) \mid \exists B \text{ s.t. } \left\{ \begin{array}{l} \varphi, \psi : A \to B, \\ \mathsf{fp}(\varphi) \neq \mathsf{fp}(\psi), \\ \mathsf{fp}(\varphi), \mathsf{fp}(\psi) \neq \bot \end{array} \right\} \right.$$

is hard on average for A following distribution μ.

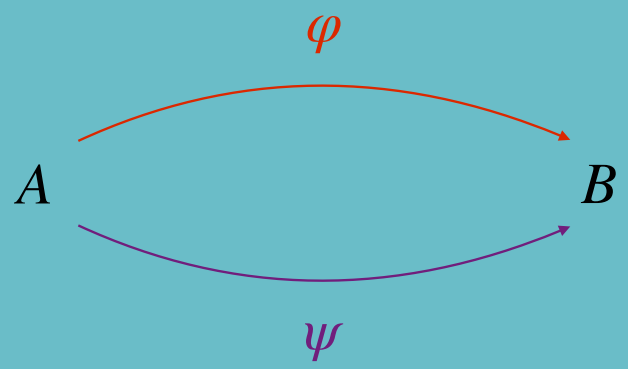


HARD (ON AVERAGE)

The language

$$\mathscr{L} = \left\{ (A, (\varphi, \psi)) \mid \exists B \text{ s.t. } \left\{ \begin{array}{l} \varphi, \psi : A \to B, \\ \mathsf{fp}(\varphi) \neq \mathsf{fp}(\psi), \\ \mathsf{fp}(\varphi), \mathsf{fp}(\psi) \neq \perp \end{array} \right\} \right.$$

is hard on average for A following distribution μ.



TRIANGULARIZABLE

There exists a PPT algorithm, **Triangle**, that on inputs two morphisms $\varphi\colon O \to A, \psi\colon O \to B$ and a fingerprint $m \in \mathcal{M}$, returns $\chi\colon A \to B$ such that $\mathsf{fp}(\chi) = m$. Furthermore, the distribution of χ only depends on A, B and m.

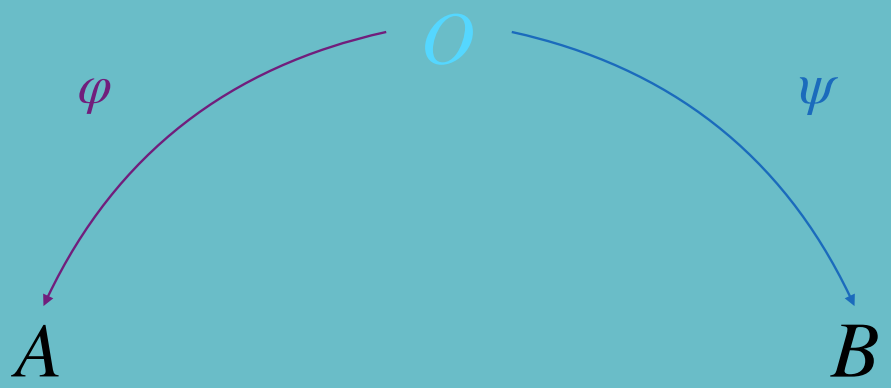
TRIANGULARIZABLE

There exists a PPT algorithm, **Triangle**, that on inputs two morphisms $\varphi\colon O \to A, \psi\colon O \to B$ and a fingerprint $m \in \mathcal{M}$, returns $\chi\colon A \to B$ such that $\mathsf{fp}(\chi) = m$. Furthermore, the distribution of χ only depends on A, B and m.

O

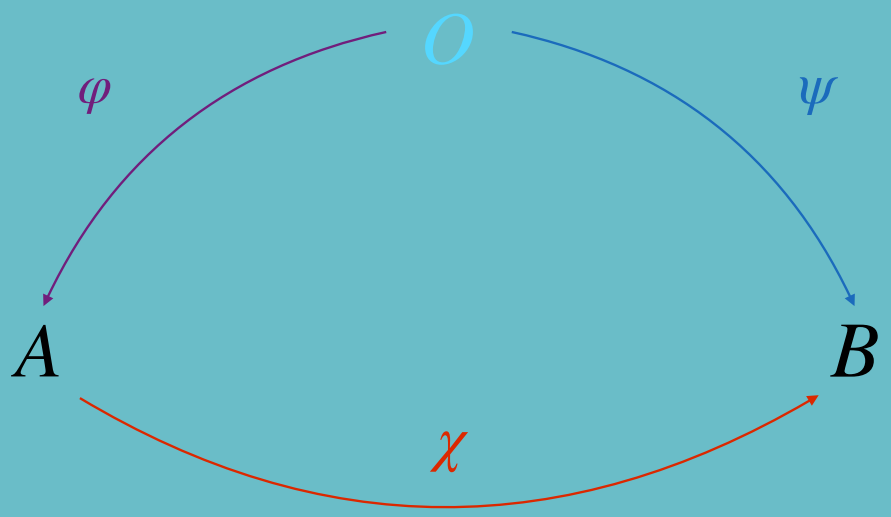
TRIANGULARIZABLE

There exists a PPT algorithm, **Triangle**, that on inputs two morphisms $\varphi\colon O \to A, \psi\colon O \to B$ and a fingerprint $m \in \mathcal{M}$, returns $\chi\colon A \to B$ such that $\mathsf{fp}(\chi) = m$. Furthermore, the distribution of χ only depends on A, B and m.



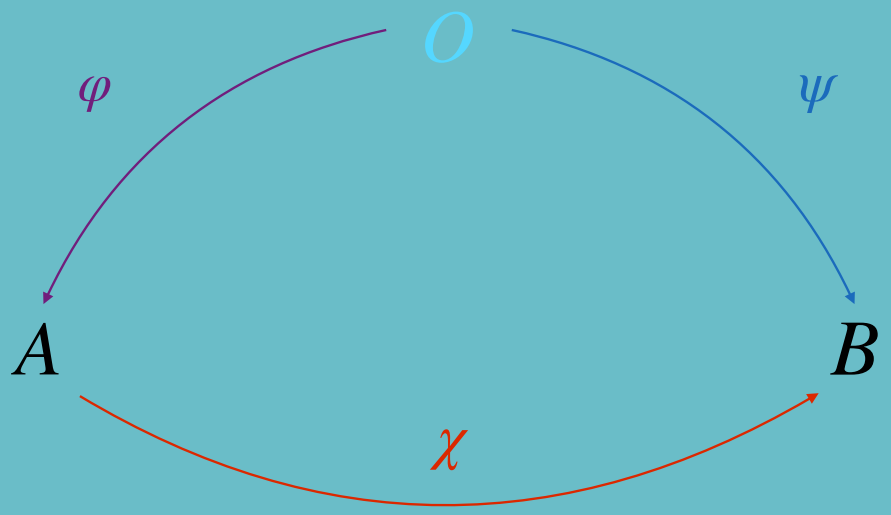
TRIANGULARIZABLE

There exists a PPT algorithm, **Triangle**, that on inputs two morphisms $\varphi\colon O \to A, \psi\colon O \to B$ and a fingerprint $m \in \mathcal{M}$, returns $\chi\colon A \to B$ such that $\mathsf{fp}(\chi) = m$. Furthermore, the distribution of χ only depends on A, B and m.



TRIANGULARIZABLE

There exists a PPT algorithm, **Triangle**, that on inputs two morphisms $\varphi\colon O \to A, \psi\colon O \to B$ and a fingerprint $m \in \mathcal{M}$, returns $\chi\colon A \to B$ such that $\mathsf{fp}(\chi) = m$. Furthermore, the distribution of χ only depends on A, B and m.



$= m$

(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

There exists a PPT algorithm **SimWalk** which takes an object A as input and returns a morphism $\chi : A \to B$ indistinguishable from the output of **Triangle**.

(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

There exists a PPT algorithm **SimWalk** which takes an object A as input and returns a morphism $\chi : A \to B$ indistinguishable from the output of **Triangle**.

Algorithm 1 Oracles of the walk-distinguishing experiment

1: **oracle** $\mathcal{O}^{\varphi:O\to A}_{\mathsf{SimWalk}}$	1: **oracle** $\mathcal{O}^{\varphi:O\to A}_{\mathsf{Triangle}}$
2: $\quad \chi \leftarrow \mathsf{SimWalk}(A)$	2: $\quad (B,\psi) \leftarrow \mathsf{Walk}(O)$
3: $\quad$ **return** χ	3: $\quad m \leftarrow_{\$} \mathcal{M}$
	4: $\quad$ **return** $\chi \leftarrow \mathsf{Triangle}(\varphi,\psi,m)$

An Instantiation of a Forensic Category

CHOOSING AN EFFECTIVE CATEGORY

CHOOSING AN EFFECTIVE CATEGORY

CHOOSING AN EFFECTIVE CATEGORY

Pre-effective category + Walk + Origin object = Effective Category

The category of *supersingular elliptic curves in characteristic* p, written $\mathscr{C}_p$, is the category where

- **objects** are isomorphism classes of supersingular elliptic curves over $\overline{\mathbb{F}}_p$, and
- **morphisms** $\psi\colon E_1 \to E_2$ are isogenies between them.

A PRE-EFFECTIVE CATEGORY?

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation 
- Deciding validity of encodings

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation
- Deciding validity of encodings

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation
- Deciding validity of encodings
- Computing domain and codomain

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation
- Deciding validity of encodings
- Computing domain and codomain

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation
- Deciding validity of encodings
- Computing domain and codomain
- Composing morphisms

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation
- Deciding validity of encodings
- Computing domain and codomain
- Composing morphisms

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation
- Deciding validity of encodings
- Computing domain and codomain
- Composing morphisms
- Computing identity morphisms

A PRE-EFFECTIVE CATEGORY?

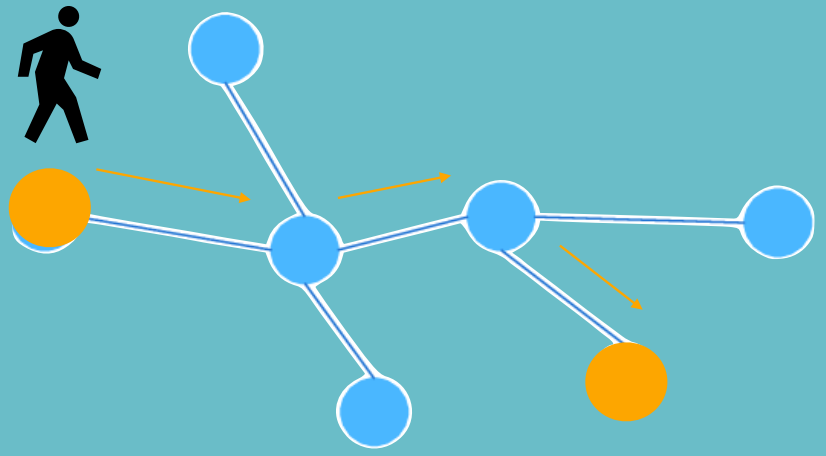
- Objects have unique representation
- Deciding validity of encodings
- Computing domain and codomain
- Composing morphisms
- Computing identity morphisms

A PRE-EFFECTIVE CATEGORY?

- Objects have unique representation ✓
- Deciding validity of encodings ✓
- Computing domain and codomain ✓
- Composing morphisms ✓
- Computing identity morphisms ✓

A Walk

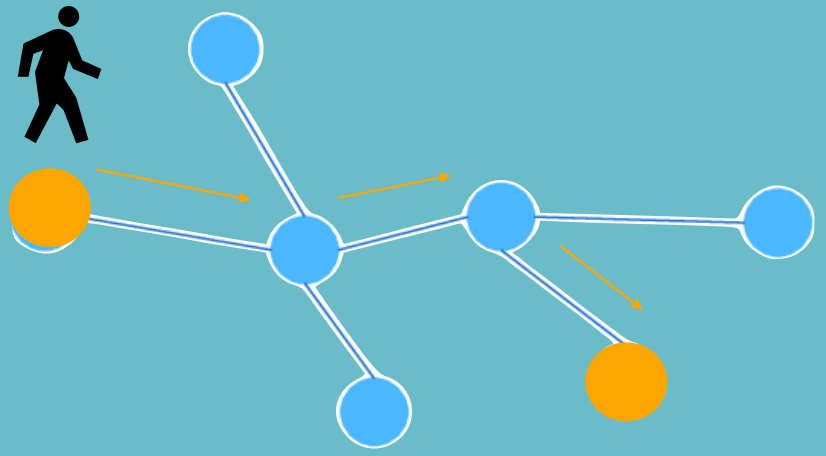
Walk is instantiated by following a (non-backtracking) random walk of length n from the input curve in the ℓ-isogeny graph.



A Walk

Walk is instantiated by following a (non-backtracking) random walk of length n from the input curve in the ℓ-isogeny graph.

Since this graph is *Ramanujan* the distance between the distribution of the output of the walk and the stationary distribution is a negligible function of the walk's length n.



AN ORIGIN OBJECT $\mathcal{O}$

There exists an algorithm to find a supersingular curve E_0 of known endomorphism ring.

CHOOSING A FINGERPRINT

CHOOSING A FINGERPRINT

Let $\varphi : E \to E'$ be an isogeny and define $\ker(\varphi)[N] = \ker(\varphi) \cap E[N]$. Then,

$$\mathsf{prefp}(\varphi) = \begin{cases} \ker(\varphi)[N], & \text{if } \ker(\varphi)[N] \cong \mathbb{Z}/N\mathbb{Z} \\ \perp, & \text{otherwise.} \end{cases}$$

CHOOSING A FINGERPRINT

Let $\varphi : E \to E'$ be an isogeny and define $\ker(\varphi)[N] = \ker(\varphi) \cap E[N]$. Then,

$$\text{prefp}(\varphi) = \begin{cases} \ker(\varphi)[N], & \text{if } \ker(\varphi)[N] \cong \mathbb{Z}/N\mathbb{Z} \\ \perp, & \text{otherwise.} \end{cases}$$

$$\ker(\varphi)[\ell^n] = \langle P \rangle = \langle aP_1 + bQ_1 \rangle$$

CHOOSING A FINGERPRINT

Let $\varphi : E \to E'$ be an isogeny and define $\ker(\varphi)[N] = \ker(\varphi) \cap E[N]$. Then,

$$\mathsf{prefp}(\varphi) = \begin{cases} \ker(\varphi)[N], & \text{if } \ker(\varphi)[N] \cong \mathbb{Z}/N\mathbb{Z} \\ \perp, & \text{otherwise.} \end{cases}$$

$$\ker(\varphi)[N] = \langle P \rangle = \langle aP_E + bQ_E \rangle$$

$$\mathsf{fp}_{\mathscr{K}}(\varphi) = \begin{cases} (1,b), & \text{if } \ker(\varphi)[N] = \langle P_E + bQ_E \rangle \text{ for } b \in (\mathbb{Z}/N\mathbb{Z}) \\ (a,1), & \text{if } \ker(\varphi)[N] = \langle aP_E + Q_E \rangle \text{ for } a \in (\mathbb{Z}/N\mathbb{Z}) \backslash (\mathbb{Z}/N\mathbb{Z})^{\times} \\ \perp, & \text{otherwise,} \end{cases}$$

EFFECTIVE

EFFECTIVE

1. Given an efficient representation for φ, we first compute **prefp**(φ) by evaluating it on $E[N]$.

EFFECTIVE

1. Given an efficient representation for φ, we first compute $\mathsf{prefp}(\varphi)$ by evaluating it on $E[N]$.

2. Then we compute $\mathsf{fp}_{\mathscr{K}}(\varphi)$ by a generalised discrete logarithm computation, taking time polynomial in $\log(N)$

HARD (ON AVERAGE)

HARD (ON AVERAGE)

Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then ONEEND is not hard on average either (hence ISOGENY is not hard in the worst case)

HARD (ON AVERAGE)

Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then ONEEND is not hard on average either (hence ISOGENY is not hard in the worst case)

- Assume that $\mathsf{fp}_{\mathscr{K}}$ is not hard on average: there exists a PPT algorithm $\mathscr{A}$ which, on input a uniformly random E_1, returns $\varphi, \psi \colon E_1 \to E_2$ s.t. $(E_1, (\varphi, \psi)) \in \mathscr{L}$ with non-negligible probability p.

HARD (ON AVERAGE)

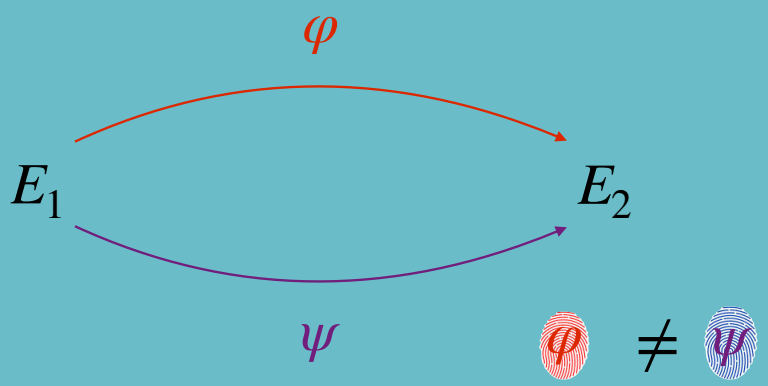
Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then ONEEND is not hard on average either (hence ISOGENY is not hard in the worst case)

- Assume that $\mathsf{fp}_{\mathscr{K}}$ is not hard on average: there exists a PPT algorithm $\mathscr{A}$ which, on input a uniformly random E_1, returns $\varphi, \psi \colon E_1 \to E_2$ s.t. $(E_1, (\varphi, \psi)) \in \mathscr{L}$ with non-negligible probability p.
- **Solve** ONEEND: Let E_1 be uniformly random.

HARD (ON AVERAGE)

Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then **ONEEND** is not hard on average either (hence **ISOGENY** is not hard in the worst case)

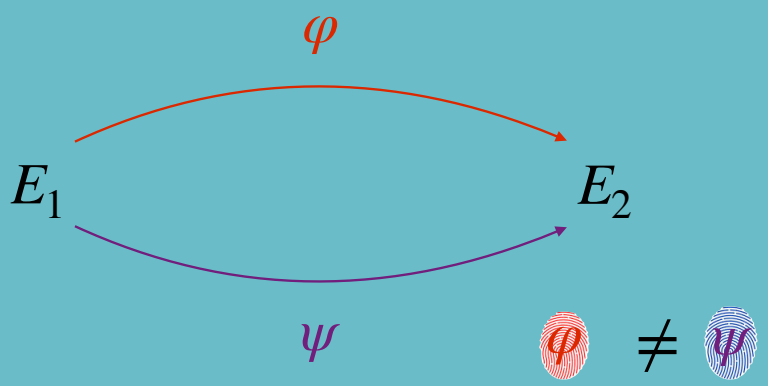
- Assume that $\mathsf{fp}_{\mathscr{K}}$ is not hard on average: there exists a PPT algorithm $\mathscr{A}$ which, on input a uniformly random E_1, returns $\varphi, \psi \colon E_1 \to E_2$ s.t. $(E_1, (\varphi, \psi)) \in \mathscr{L}$ with non-negligible probability p.
- **Solve** ONEEND: Let E_1 be uniformly random.
  - run $(\varphi, \psi) \leftarrow \mathscr{A}(E_1)$



HARD (ON AVERAGE)

Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then **ONEEND** is not hard on average either (hence **ISOGENY** is not hard in the worst case)

- Assume that $\mathsf{fp}_{\mathscr{K}}$ is not hard on average: there exists a PPT algorithm $\mathscr{A}$ which, on input a uniformly random E_1, returns $\varphi, \psi \colon E_1 \to E_2$ s.t. $(E_1, (\varphi, \psi)) \in \mathscr{L}$ with non-negligible probability p.
- **Solve** ONEEND: Let E_1 be uniformly random.
  - run $(\varphi, \psi) \leftarrow \mathscr{A}(E_1)$
  - return the composition $\hat{\psi} \circ \varphi$

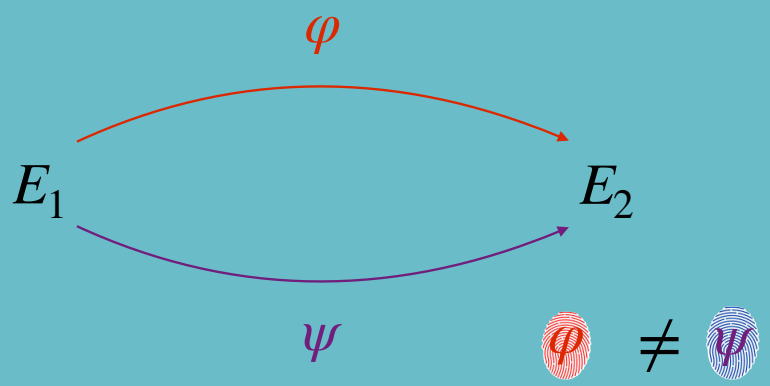


HARD (ON AVERAGE)

Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then **ONEEND** is not hard on average either (hence **ISOGENY** is not hard in the worst case)

- Assume that $\mathsf{fp}_{\mathscr{K}}$ is not hard on average: there exists a PPT algorithm $\mathscr{A}$ which, on input a uniformly random E_1, returns $\varphi, \psi \colon E_1 \to E_2$ s.t. $(E_1, (\varphi, \psi)) \in \mathscr{L}$ with non-negligible probability p.
- **Solve** ONEEND: Let E_1 be uniformly random.
  - run $(\varphi, \psi) \leftarrow \mathscr{A}(E_1)$
  - return the composition $\hat{\psi} \circ \varphi$

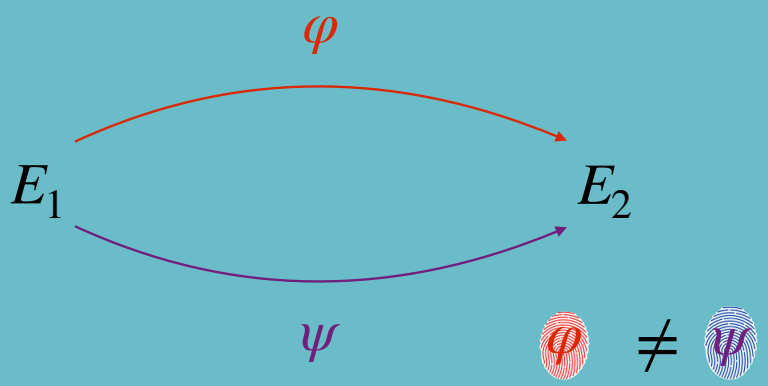
HARD (ON AVERAGE) ⇔ ONEEND



HARD (ON AVERAGE)

Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then **ONEEND** is not hard on average either (hence **ISOGENY** is not hard in the worst case)

- Assume that $\mathsf{fp}_{\mathscr{K}}$ is not hard on average: there exists a PPT algorithm $\mathscr{A}$ which, on input a uniformly random E_1, returns $\varphi, \psi \colon E_1 \to E_2$ s.t. $(E_1, (\varphi, \psi)) \in \mathscr{L}$ with non-negligible probability p.
- **Solve** ONEEND: Let E_1 be uniformly random.
  - run $(\varphi, \psi) \leftarrow \mathscr{A}(E_1)$
  - return the composition $\hat{\psi} \circ \varphi$



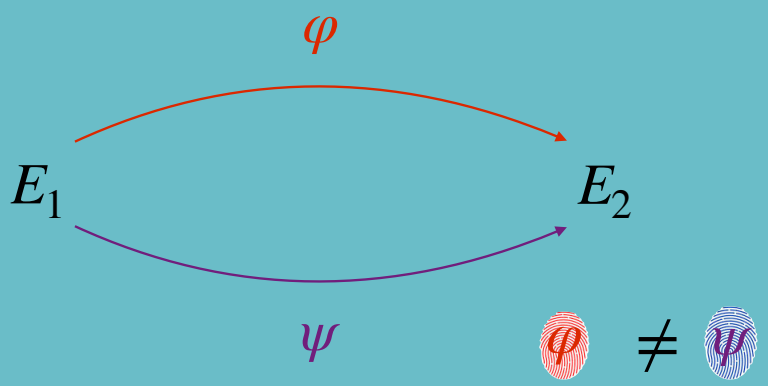
HARD (ON AVERAGE) ⇔ ONEEND

Note: If $\hat{\psi} \circ \varphi \in \mathbb{Z}$, there exist $a, b \in \mathbb{Z}$ such that $[a]\varphi = [b]\psi$. Then,

HARD (ON AVERAGE)

Strategy: Prove that if $\mathsf{fp}_{\mathscr{K}}$ does not satisfy the Hard on average property, then **ONEEND** is not hard on average either (hence **ISOGENY** is not hard in the worst case)

- Assume that $\mathsf{fp}_{\mathscr{K}}$ is not hard on average: there exists a PPT algorithm $\mathscr{A}$ which, on input a uniformly random E_1, returns $\varphi, \psi\colon E_1 \to E_2$ s.t. $(E_1, (\varphi, \psi)) \in \mathscr{L}$ with non-negligible probability p.
- **Solve** ONEEND: Let E_1 be uniformly random.
  - run $(\varphi, \psi) \leftarrow \mathscr{A}(E_1)$
  - return the composition $\hat{\psi} \circ \varphi$



HARD (ON AVERAGE) ⇔ ONEEND

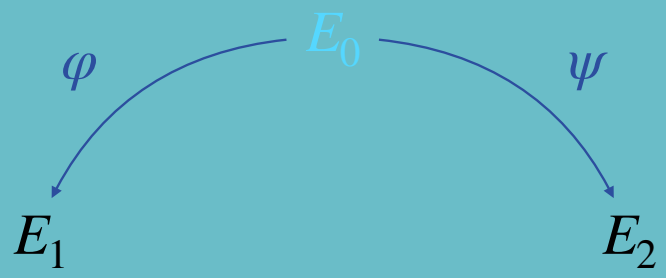
Note: If $\hat{\psi} \circ \varphi \in \mathbb{Z}$, there exist $a, b \in \mathbb{Z}$ such that $[a]\varphi = [b]\psi$. Then,

$$\ker(\varphi)[N] = \ker([a]\varphi)[N] = \ker([b]\psi)[N] = \ker(\psi)[N]$$

TRIANGULARIZABLE

TRIANGULARIZABLE

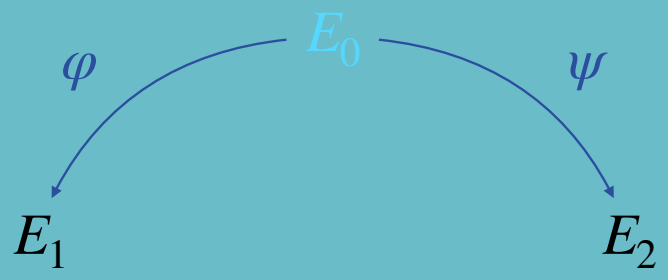
Input: $\varphi\colon E_0 \to E_1, \psi\colon E_0 \to E_2, (a, b)$



TRIANGULARIZABLE

Input: $\varphi\colon E_0 \to E_1, \psi\colon E_0 \to E_2, (a, b)$

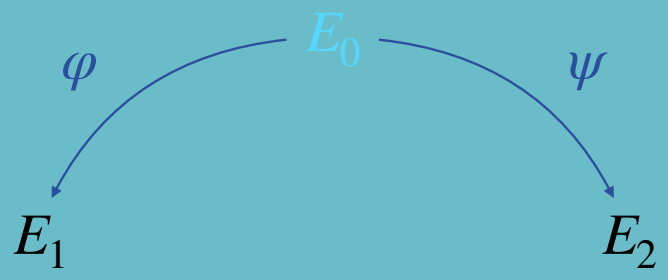
1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$



TRIANGULARIZABLE

Input: $\varphi\colon E_0 \to E_1, \psi\colon E_0 \to E_2, (a, b)$

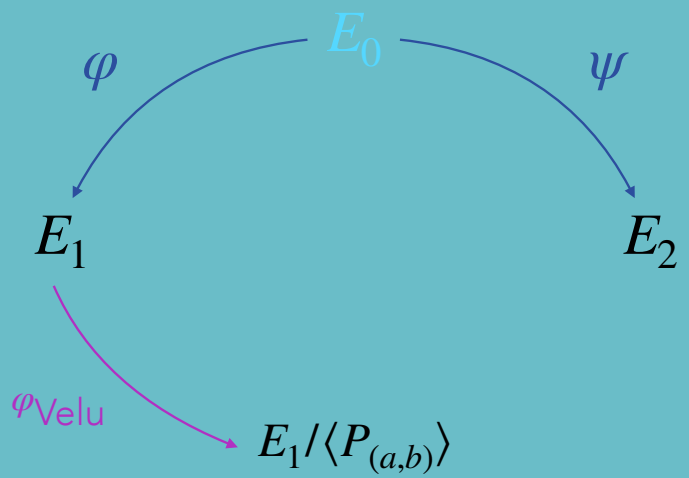
1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$
2. Let $P_{(a,b)} = aP_0 + bQ_0$



TRIANGULARIZABLE

Input: $\varphi\colon E_0 \to E_1, \psi\colon E_0 \to E_2, (a, b)$

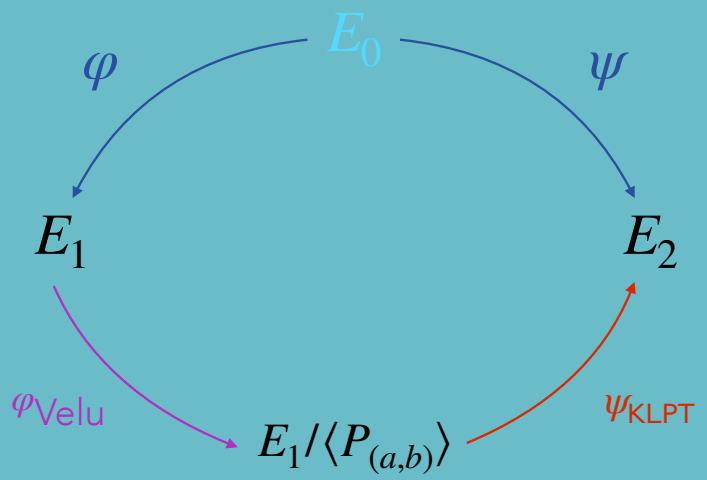
1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$
2. Let $P_{(a,b)} = aP_0 + bQ_0$
3. Compute the quotient isogeny $\varphi_{\text{Velu}}\colon E_1 \to E_1/\langle P_{(a,b)} \rangle$ using Velu's formula



TRIANGULARIZABLE

Input: $\varphi\colon E_0 \to E_1, \psi\colon E_0 \to E_2, (a, b)$

1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$
2. Let $P_{(a,b)} = aP_0 + bQ_0$
3. Compute the quotient isogeny $\varphi_{\text{Velu}}\colon E_1 \to E_1/\langle P_{(a,b)}\rangle$ using Velu's formula
4. Use **RSigningKLPT**$_{\ell'}(\varphi_{\text{Velu}} \circ \varphi, \psi)$ to compute $\psi_{\text{KLPT}}\colon E_1/\langle P_{(a,b)}\rangle \to E_2$ of degree a power of ℓ'

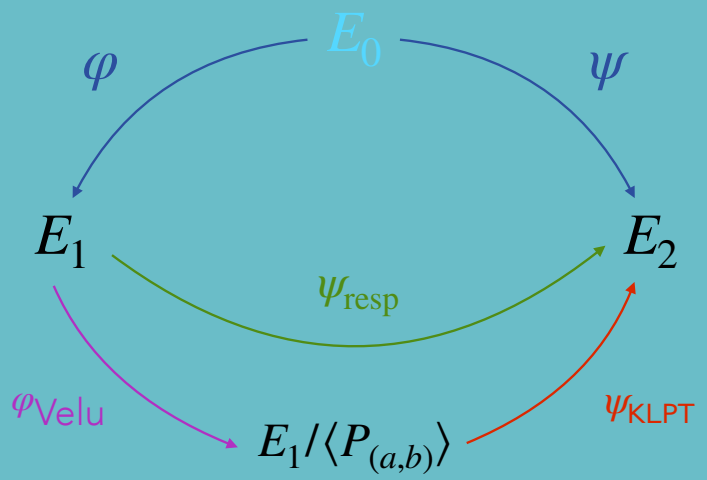


TRIANGULARIZABLE

Input: $\varphi: E_0 \to E_1, \psi: E_0 \to E_2, (a, b)$

1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$
2. Let $P_{(a,b)} = aP_0 + bQ_0$
3. Compute the quotient isogeny $\varphi_{\text{Velu}}: E_1 \to E_1/\langle P_{(a,b)} \rangle$ using Velu's formula
4. Use $\text{RSigningKLPT}_{\ell'}(\varphi_{\text{Velu}} \circ \varphi, \psi)$ to compute $\psi_{\text{KLPT}}: E_1/\langle P_{(a,b)} \rangle \to E_2$ of degree a power of ℓ'

Output: $\psi_{\text{resp}} = \psi_{\text{KLPT}} \circ \varphi_{\text{Velu}}$, where $\psi_{\text{resp}}: E_1 \to E_2$ such that $\text{fp}_{\mathcal{K}}(\psi_{\text{resp}}) = (a, b)$



(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

Input: E_1

E_1

(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

Input: E_1

1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$ and let $P_{(a,b)} = aP_0 + bQ_0$

E_1

(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

Input: E_1

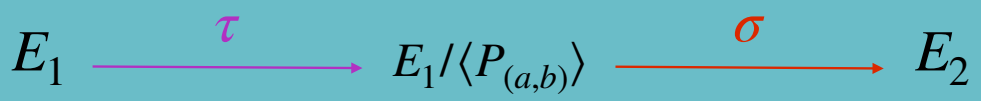
1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$ and let $P_{(a,b)} = aP_0 + bQ_0$
2. Use Velu's formulas to compute $\tau : E_1 \to E_1/P_{(a,b)}$

$$E_1 \xrightarrow{\tau} E_1/\langle P_{(a,b)} \rangle$$

(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

Input: E_1

1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$ and let $P_{(a,b)} = aP_0 + bQ_0$
2. Use Velu's formulas to compute $\tau : E_1 \to E_1/P_{(a,b)}$
3. Take a random walk in the ℓ'-isogeny graph, where ℓ' coprime to ℓ, from $E_1/P_{(a,b)}$: $(E_2, \sigma) \leftarrow \mathsf{Walk}_{\ell'}(E_1/P_{(a,b)}, n)$

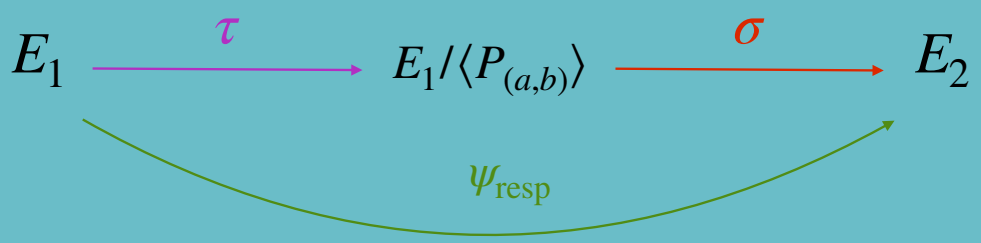


(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

Input: E_1

1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$ and let $P_{(a,b)} = aP_0 + bQ_0$
2. Use Velu's formulas to compute $\tau : E_1 \to E_1/P_{(a,b)}$
3. Take a random walk in the ℓ'-isogeny graph, where ℓ' coprime to ℓ, from $E_1/P_{(a,b)}$: $(E_2, \sigma) \leftarrow \mathsf{Walk}_{\ell'}(E_1/P_{(a,b)}, n)$

Return: $\psi_{rsp} = \sigma \circ \tau$, where $\psi_{rsp} : E_1 \to E_2$

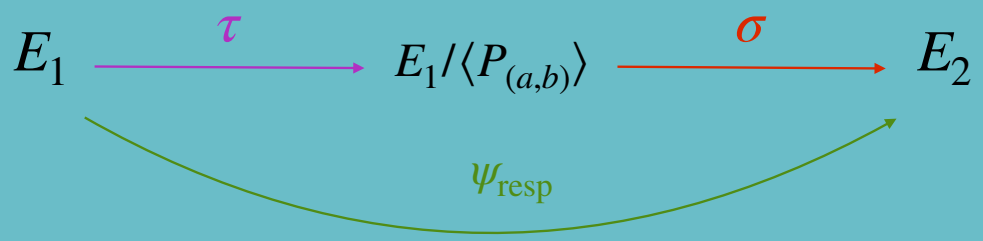


(COMPUTATIONALLY) WALK-INDISTINGUISHABLE

Input: E_1

1. Let (P_0, Q_0) be the chosen basis of $E_0[N]$ and let $P_{(a,b)} = aP_0 + bQ_0$
2. Use Velu's formulas to compute $\tau : E_1 \to E_1/P_{(a,b)}$
3. Take a random walk in the ℓ'-isogeny graph, where ℓ' coprime to ℓ, from $E_1/P_{(a,b)}$: $(E_2, \sigma) \leftarrow \mathsf{Walk}_{\ell'}(E_1/P_{(a,b)}, n)$

Return: $\psi_{rsp} = \sigma \circ \tau$, where $\psi_{rsp} : E_1 \to E_2$



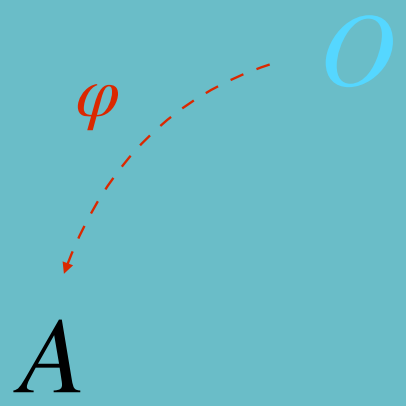
Assumption: *The output of* RSigningKLPT*, is computationally indistinguishable from uniformly random cyclic isogenies of same degree and same domain.*

Digital Signature

THE SIGMA PROTOCOL

THE SIGMA PROTOCOL

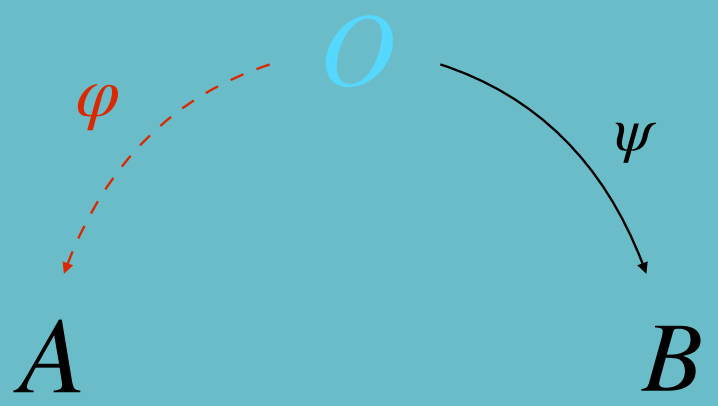
InstanceGen. Run $(A, \varphi) \leftarrow \mathsf{Walk}(O)$. A will serve as public key, while φ will be the secret key *used to produce tuples* $(A, (\psi, \chi)) \in \mathscr{L}$.



THE SIGMA PROTOCOL

InstanceGen. Run $(A, \varphi) \leftarrow \mathsf{Walk}(O)$. A will serve as public key, while φ will be the secret key *used to produce tuples* $(A, (\psi, \chi)) \in \mathscr{L}$.

Commitment. The prover generates $(B, \psi) \leftarrow \mathsf{Walk}(O)$, and sends B as the commitment object.

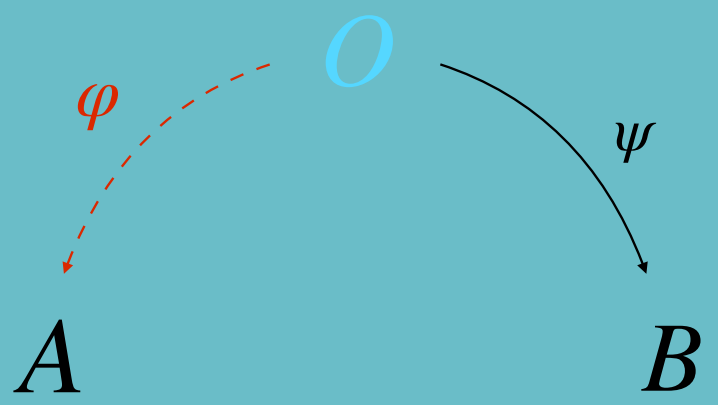


THE SIGMA PROTOCOL

InstanceGen. Run $(A, \varphi) \leftarrow \mathsf{Walk}(O)$. A will serve as public key, while φ will be the secret key *used to produce tuples* $(A, (\psi, \chi)) \in \mathscr{L}$.

Commitment. The prover generates $(B, \psi) \leftarrow \mathsf{Walk}(O)$, and sends B as the commitment object.

Challenge. The verifier samples a fingerprint $m \in \mathscr{M}$ uniformly at random and sends it to the prover.



m

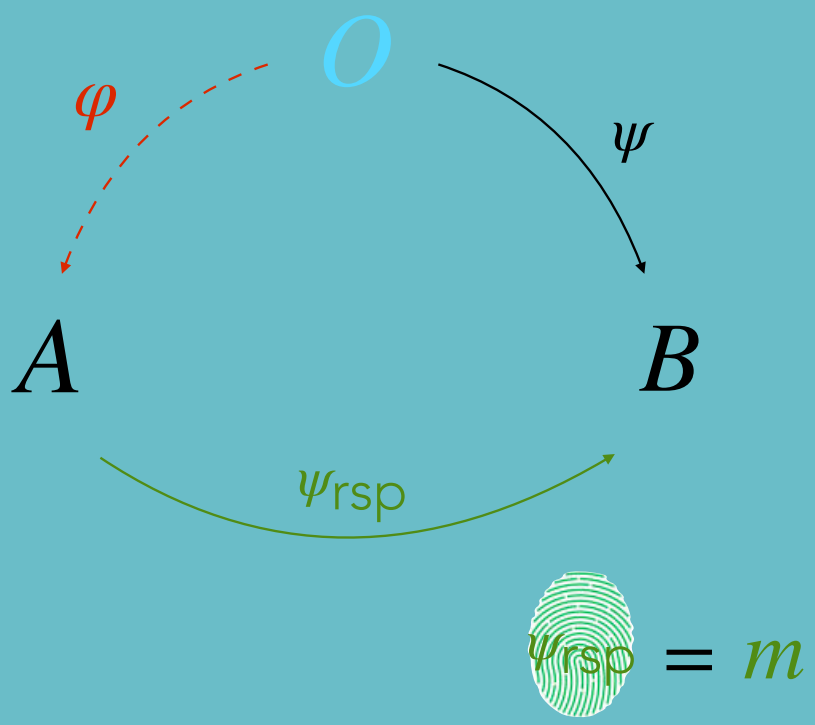
THE SIGMA PROTOCOL

InstanceGen. Run $(A, \varphi) \leftarrow \mathsf{Walk}(O)$. A will serve as public key, while φ will be the secret key *used to produce tuples* $(A, (\psi, \chi)) \in \mathscr{L}$.

Commitment. The prover generates $(B, \psi) \leftarrow \mathsf{Walk}(O)$, and sends B as the commitment object.

Challenge. The verifier samples a fingerprint $m \in \mathscr{M}$ uniformly at random and sends it to the prover.

Response. The prover runs $\psi_{\mathsf{rsp}} \leftarrow \mathsf{Triangle}(\varphi, \psi, m)$ to obtain a morphism ψ_{rsp} such that $\mathsf{fp}(\psi_{\mathsf{rsp}}) = m$. It sends ψ_{rsp} to the verifier.



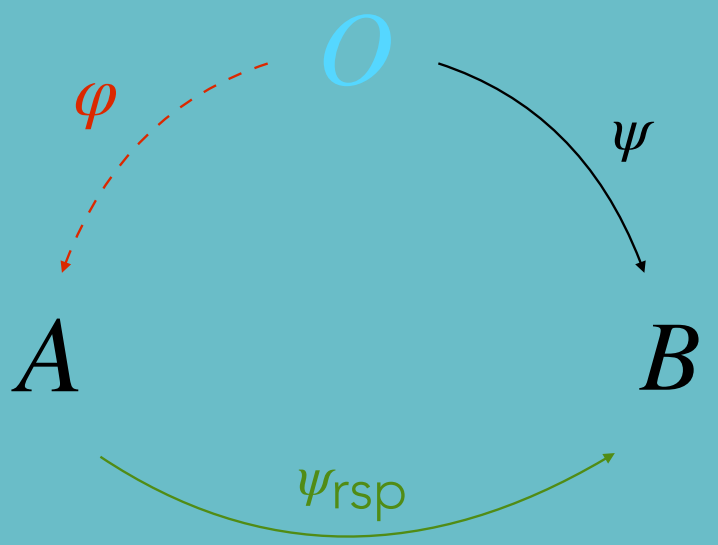
THE SIGMA PROTOCOL

InstanceGen. Run $(A, \varphi) \leftarrow \mathsf{Walk}(O)$. A will serve as public key, while φ will be the secret key *used to produce tuples* $(A, (\psi, \chi)) \in \mathscr{L}$.

Commitment. The prover generates $(B, \psi) \leftarrow \mathsf{Walk}(O)$, and sends B as the commitment object.

Challenge. The verifier samples a fingerprint $m \in \mathscr{M}$ uniformly at random and sends it to the prover.

Response. The prover runs $\psi_{\mathsf{rsp}} \leftarrow \mathsf{Triangle}(\varphi, \psi, m)$ to obtain a morphism ψ_{rsp} such that $\mathsf{fp}(\psi_{\mathsf{rsp}}) = m$. It sends ψ_{rsp} to the verifier.



Verification. The verifier accepts if ψ_{rsp} has A for domain, B for codomain, and if $\mathsf{fp}(\psi_{\mathsf{rsp}}) = m$.

SPECIAL SOUNDNESS

SPECIAL SOUNDNESS

We say that $\mathbf{FCSigma}_{\mathscr{C}}$ is special-sound if, given two accepting transcripts $(B, m, \psi_{\text{rsp}})$ and $(B, m', \psi'_{\text{rsp}})$, with the same commitment B and distinct challenges $m \neq m'$, one can find a witness (χ, χ') for A in $\mathscr{L}$.

SPECIAL SOUNDNESS

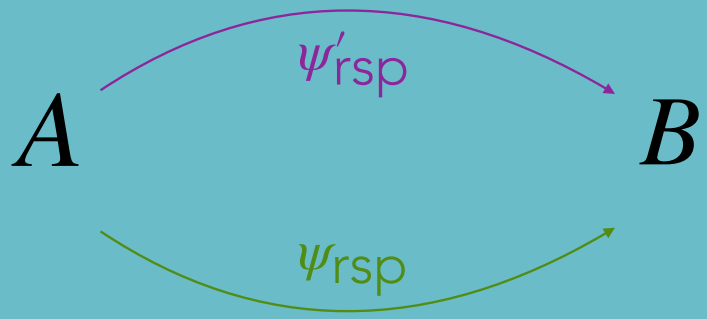
We say that $\mathbf{FCSigma}_{\mathscr{C}}$ is special-sound if, given two accepting transcripts $(B, m, \psi_{\text{rsp}})$ and $(B, m', \psi'_{\text{rsp}})$, with the same commitment B and distinct challenges $m \neq m'$, one can find a witness (χ, χ') for A in $\mathscr{L}$.

- By construction, $(\psi_{\text{rsp}}, \psi'_{\text{rsp}})$ is indeed a witness for A in $\mathscr{L}$.

SPECIAL SOUNDNESS

We say that $\mathbf{FCSigma}_{\mathscr{C}}$ is special-sound if, given two accepting transcripts $(B, m, \psi_{\text{rsp}})$ and $(B, m', \psi'_{\text{rsp}})$, with the same commitment B and distinct challenges $m \neq m'$, one can find a witness (χ, χ') for A in $\mathscr{L}$.

- By construction, $(\psi_{\text{rsp}}, \psi'_{\text{rsp}})$ is indeed a witness for A in $\mathscr{L}$.



HONEST-VERIFIER ZERO-KNOWLEDGE

HONEST-VERIFIER ZERO-KNOWLEDGE

Let $\varphi : O \to A$ be a secret key

HONEST-VERIFIER ZERO-KNOWLEDGE

Let $\varphi : O \to A$ be a secret key

The simulator, on input the public key A:

HONEST-VERIFIER ZERO-KNOWLEDGE

Let $\varphi : O \to A$ be a secret key

The simulator, on input the public key A:

- $\mathrm{Run}(B, \chi) \leftarrow \mathsf{SimWalk}(A)$

HONEST-VERIFIER ZERO-KNOWLEDGE

Let $\varphi : O \to A$ be a secret key

The simulator, on input the public key A:

- Run$(B, \chi) \leftarrow \mathsf{SimWalk}(A)$
- Compute $m \leftarrow \mathsf{fp}(\chi)$

HONEST-VERIFIER ZERO-KNOWLEDGE

Let $\varphi : O \to A$ be a secret key

The simulator, on input the public key A:

- Run$(B, \chi) \leftarrow \mathsf{SimWalk}(A)$
- Compute $m \leftarrow \mathsf{fp}(\chi)$
- Return (B, m, χ)

HONEST-VERIFIER ZERO-KNOWLEDGE

Let $\varphi : O \to A$ be a secret key

The simulator, on input the public key A:

- Run$(B, \chi) \leftarrow$ SimWalk(A)
- Compute $m \leftarrow$ fp(χ)
- Return (B, m, χ)

By **Walk-Indistinguishable,** (B, m, χ) is indistinguishable from a transcript obtained from an honest run of the protocol

HONEST-VERIFIER ZERO-KNOWLEDGE

Let $\varphi : O \to A$ be a secret key

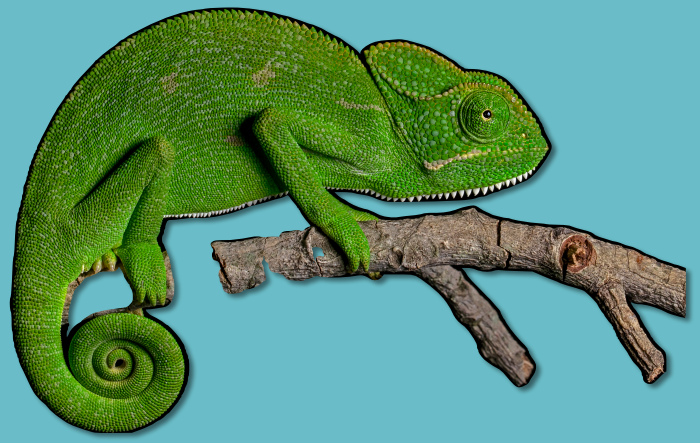
The simulator, on input the public key A:

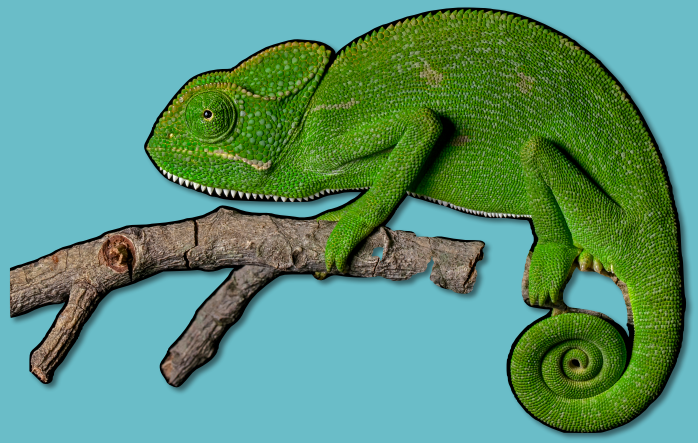
- Run$(B, \chi) \leftarrow \mathsf{SimWalk}(A)$
- Compute $m \leftarrow \mathsf{fp}(\chi)$
- Return (B, m, χ)

By **Walk-Indistinguishable,** (B, m, χ) is indistinguishable from a transcript obtained from an honest run of the protocol

Comment: This construction does not quite capture the HD variants of SQIsign

Chameleon Hash Function





WHAT IS A CHAMELEON HASH FUNCTION?

WHAT IS A CHAMELEON HASH FUNCTION?

Chameleon hash functions (aka **trapdoor commitments**) are trapdoor collision-resistant hash functions parametrized by a public key

WHAT IS A CHAMELEON HASH FUNCTION?

Chameleon hash functions (aka **trapdoor commitments**) are trapdoor collision-resistant hash functions parametrized by a public key

CHPG: On input a security parameter λ, outputs public parameters **pp**.

WHAT IS A CHAMELEON HASH FUNCTION?

Chameleon hash functions (aka **trapdoor commitments**) are trapdoor collision-resistant hash functions parametrized by a public key

CHPG: On input a security parameter λ, outputs public parameters **pp**.

CHKG: On input the public parameters **pp** outputs a private-public key pair (**sk, pk**).

WHAT IS A CHAMELEON HASH FUNCTION?

Chameleon hash functions (aka **trapdoor commitments**) are trapdoor collision-resistant hash functions parametrized by a public key

CHPG: On input a security parameter λ, outputs public parameters **pp**.

CHKG: On input the public parameters **pp** outputs a private-public key pair (**sk, pk**).

CHash: On input a public key **pk** and a message **msg**, outputs a hash h and a *check value* (aka *randomness*) c.

WHAT IS A CHAMELEON HASH FUNCTION?

Chameleon hash functions (aka **trapdoor commitments**) are trapdoor collision-resistant hash functions parametrized by a public key

CHPG: On input a security parameter λ, outputs public parameters **pp**.

CHKG: On input the public parameters **pp** outputs a private-public key pair (**sk, pk**).

CHash: On input a public key **pk** and a message **msg**, outputs a hash h and a *check value* (aka *randomness*) c.

CHCheck: On input a public key **pk**, a message **msg**, a check value c, and a hash h, outputs a bit b, indicating whether the hash h is valid.

WHAT IS A CHAMELEON HASH FUNCTION?

Chameleon hash functions (aka **trapdoor commitments**) are trapdoor collision-resistant hash functions parametrized by a public key

CHPG: On input a security parameter λ, outputs public parameters **pp**.

CHKG: On input the public parameters **pp** outputs a private-public key pair (**sk, pk**).

CHash: On input a public key **pk** and a message **msg**, outputs a hash h and a *check value* (aka *randomness*) c.

CHCheck: On input a public key **pk**, a message **msg**, a check value c, and a hash h, outputs a bit b, indicating whether the hash h is valid.

CHange: On input a key pair (**sk, pk**), a pair of messages **msg, msg′**, a check value c and a hash h, outputs a new check value c'.

DEFINING PROPERTIES

DEFINING PROPERTIES

Correctness - output of **CHash** and **CHange** is always accepted by **CHCheck**

DEFINING PROPERTIES

Correctness - output of CHash and CHange is always accepted by CHCheck

(**Weak) Collision-Resistance**: without knowledge of the secret key, it is hard to find pairs (msg, c) and (msg′, c') with msg ≠ msg′ both passing CHCheck for the same hash h.

DEFINING PROPERTIES

Correctness - output of CHash and CHange is always accepted by CHCheck

(**Weak) Collision-Resistance**: without knowledge of the secret key, it is hard to find pairs (msg, c) and (msg′, c') with msg ≠ msg′ both passing CHCheck for the same hash h.

(**Strong) Indistinguishability**: it is infeasible to distinguish between check values produced by CHash and those produced by CHange

OUR CHAMELEON HASH FUNCTION

OUR CHAMELEON HASH FUNCTION

G be the group camouflaging its fingerprint, and $H : \{0,1\}^* \to G$ a collision-resistant hash function.

OUR CHAMELEON HASH FUNCTION

G be the group camouflaging its fingerprint, and $H : \{0,1\}^* \to G$ a collision-resistant hash function.

- A public key will be an object A in $\mathscr{C}$, with corresponding trapdoor a morphism $\varphi : O \to A$.

OUR CHAMELEON HASH FUNCTION

G be the group camouflaging its fingerprint, and $H : \{0,1\}^* \to G$ a collision-resistant hash function.

- A public key will be an object A in $\mathscr{C}$, with corresponding trapdoor a morphism $\varphi : O \to A$.
- To hash a message **msg** we hash it to $H(\mathbf{msg})$ and use it to camouflage the fingerprint of a random morphism $\psi : A \to B$.

OUR CHAMELEON HASH FUNCTION

G be the group camouflaging its fingerprint, and $H : \{0,1\}^* \to G$ a collision-resistant hash function.

- A public key will be an object A in $\mathscr{C}$, with corresponding trapdoor a morphism $\varphi : O \to A$.
- To hash a message **msg** we hash it to $H(\mathbf{msg})$ and use it to camouflage the fingerprint of a random morphism $\psi : A \to B$.
- To adapt a hash we use the knowledge of φ to invoke **Triangle** and produce a new camouflage $H(\mathbf{msg}')$ of the fingerprint.

OUR CHAMELEON HASH FUNCTION

G be the group camouflaging its fingerprint, and $H : \{0,1\}^* \to G$ a collision-resistant hash function.

- A public key will be an object A in $\mathscr{C}$, with corresponding trapdoor a morphism $\varphi\colon O \to A$.
- To hash a message **msg** we hash it to $H(\mathsf{msg})$ and use it to camouflage the fingerprint of a random morphism $\psi\colon A \to B$.
- To adapt a hash we use the knowledge of φ to invoke **Triangle** and produce a new camouflage $H(\mathsf{msg}')$ of the fingerprint.

Algorithm 4 Chameleon hash function from a camouflaged fingerprint

function CHKG(pp)
  $(A_{\mathsf{pk}}, \varphi_{\mathsf{sk}}) \leftarrow \mathsf{Walk}(O)$
  return $\varphi_{\mathsf{sk}},\ A_{\mathsf{pk}}$

function CHash($A_{\mathsf{pk}}, \mathsf{msg}$)
  $(B_h, \psi_c) \leftarrow \mathsf{SimWalk}(A_{\mathsf{pk}})$
  $m_h \leftarrow H(\mathsf{msg}) \star \mathrm{fp}(\psi_c)$
  return $\psi_c,\ (B_h, m_h)$

function CHeck($A_{\mathsf{pk}}, \mathsf{msg}, \psi_c, (B_h, m_h)$)
  Parse domain and codomain of $\psi_c : A \to B$
  return $H(\mathsf{msg}) \star \mathrm{fp}(\psi_c) = m_h$ **and** $A = A_{\mathsf{pk}}$ **and** $B = B_h$

function CHange($\varphi_{\mathsf{sk}}, A_{\mathsf{pk}}, \mathsf{msg}, \mathsf{msg}', \psi_c, (B_h, m_h)$)
  if not CHeck($A_{\mathsf{pk}}, \mathsf{msg}, \psi_c, (B_h, m_h)$) **then return** $\perp$
  return $\psi_c' \leftarrow \mathsf{Triangle}(\varphi_{\mathsf{sk}}, \psi_c \circ \varphi_{\mathsf{sk}}, H(\mathsf{msg}')^{-1} \star m_h)$

CORRECTNESS

- Follows immediately

CORRECTNESS

- Follows immediately

WEAK COLLISION-RESISTANCE

WEAK COLLISION-RESISTANCE

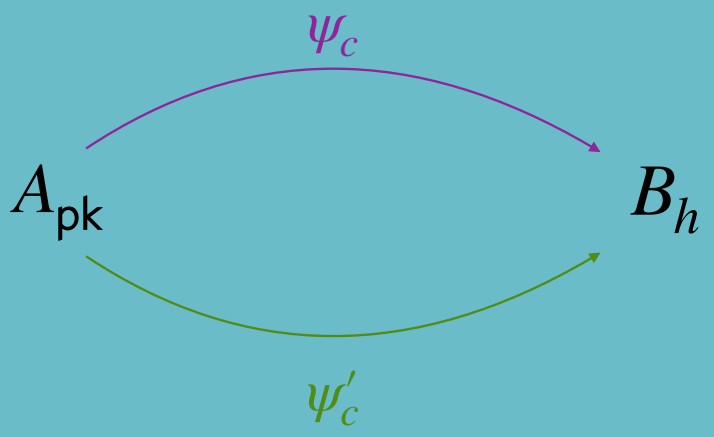
Let A_{pk} be a public key.

WEAK COLLISION-RESISTANCE

Let A_{pk} be a public key.

An adversary **adv** that wins the weak collision-resistance game computes $\mathsf{msg}, \psi_c, \mathsf{msg}', \psi_c', (B_h, m_h)$ such that

$$\mathsf{CHCheck}(A_{\mathsf{pk}}, \mathsf{msg}, \psi_c, (B_h, m_h)) = \mathsf{CHCheck}(A_{\mathsf{pk}}, \mathsf{msg}', \psi_c', (B_h, m_h)) = 1.$$



$$H(\mathsf{msg}) \star \mathsf{fp}(\psi_c) = m_h = H(\mathsf{msg}') \star \mathsf{fp}(\psi_c')$$

1. $H(\mathsf{msg}) = H(\mathsf{msg}')$, but $\mathsf{msg} \neq \mathsf{msg}'$, thus **adv** has produced a collision for H

2. $H(\mathsf{msg}) \neq H(\mathsf{msg}') \implies \mathsf{fp}(\psi_c) \neq \mathsf{fp}(\psi_c')$, witness for the hard language

STRONG INDISTINGUISHABILITY

Algorithm 4 Chameleon hash function from a camouflaged fingerprint

function CHKG(pp)
  $(A_{\mathsf{pk}}, \varphi_{\mathsf{sk}}) \leftarrow \mathsf{Walk}(O)$
  return $\varphi_{\mathsf{sk}},\ A_{\mathsf{pk}}$

function CHash($A_{\mathsf{pk}}, \mathsf{msg}$)
  $(B_h, \psi_c) \leftarrow \mathsf{SimWalk}(A_{\mathsf{pk}})$
  $m_h \leftarrow H(\mathsf{msg}) \star \mathrm{fp}(\psi_c)$
  return $\psi_c,\ (B_h, m_h)$

function CHeck($A_{\mathsf{pk}}, \mathsf{msg}, \psi_c, (B_h, m_h)$)
  Parse domain and codomain of $\psi_c : A \to B$
  return $H(\mathsf{msg}) \star \mathrm{fp}(\psi_c) = m_h$ **and** $A = A_{\mathsf{pk}}$ **and** $B = B_h$

function CHange($\varphi_{\mathsf{sk}}, A_{\mathsf{pk}}, \mathsf{msg}, \mathsf{msg}', \psi_c, (B_h, m_h)$)
  if not CHeck($A_{\mathsf{pk}}, \mathsf{msg}, \psi_c, (B_h, m_h)$) **then return** $\perp$
  return $\psi_c' \leftarrow \mathsf{Triangle}(\varphi_{\mathsf{sk}}, \psi_c \circ \varphi_{\mathsf{sk}}, H(\mathsf{msg}')^{-1} \star m_h)$

The proof relies on the **Walk Indistinguishable** property

Conclusion

FUTURE WORK

FUTURE WORK

- Instantiate the framework in a different setting, such as as lattices

FUTURE WORK

- Instantiate the framework in a different setting, such as as lattices
- Look for other schemes that can be obtained in the framework

FUTURE WORK

- Instantiate the framework in a different setting, such as as lattices
- Look for other schemes that can be obtained in the framework
- Extend the framework to instantiate the HD variants of SQIsign

FUTURE WORK

- Instantiate the framework in a different setting, such as as lattices
- Look for other schemes that can be obtained in the framework
- Extend the framework to instantiate the HD variants of SQIsign

Thank You!