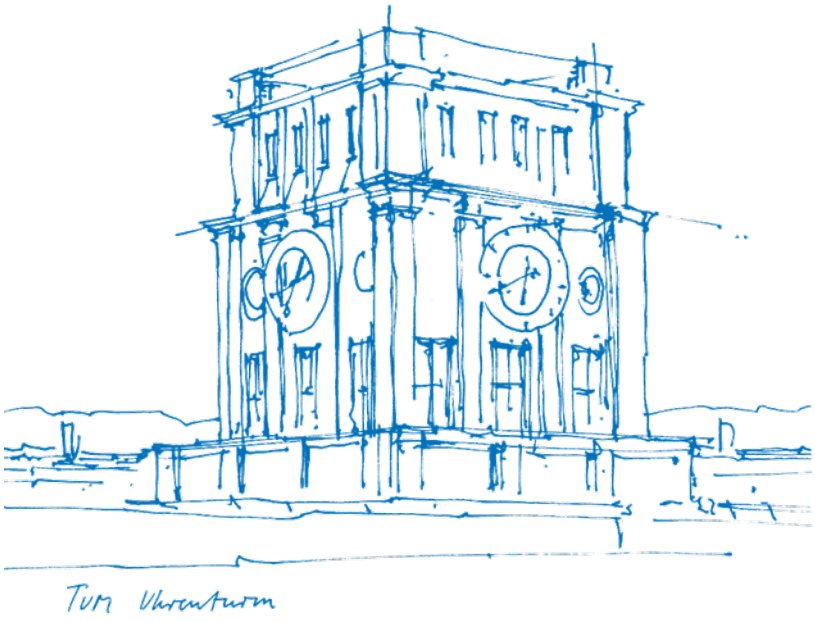
Formally Verifying the KZG Polynomial Commitment Scheme

Tobias Rothmann
Technical University of Munich
School of Computation, Information and Technology
Chair for Logic and Verification
28. Mai 2024



Outline

- Motivation
- Polynomial Commitment Schemes
- KZG
- Formal Verification

Motivation

What is formal verification?

What is formal verification?

- providing a formal proof

What is formal verification?

- providing a formal proof
  $\Rightarrow$ providing a machine-checkable proof

What is formal verification?

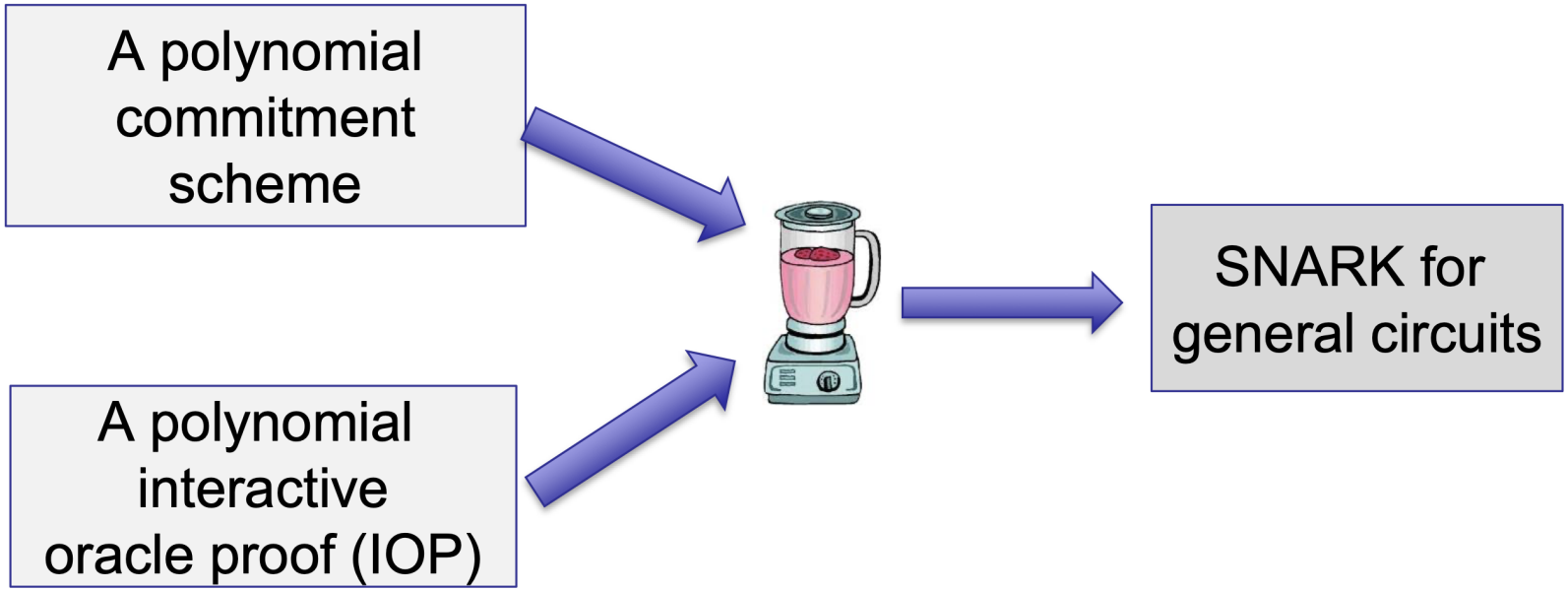
- providing a formal proof
  $\Rightarrow$ providing a machine-checkable proof
- undecidable

What is formal verification?

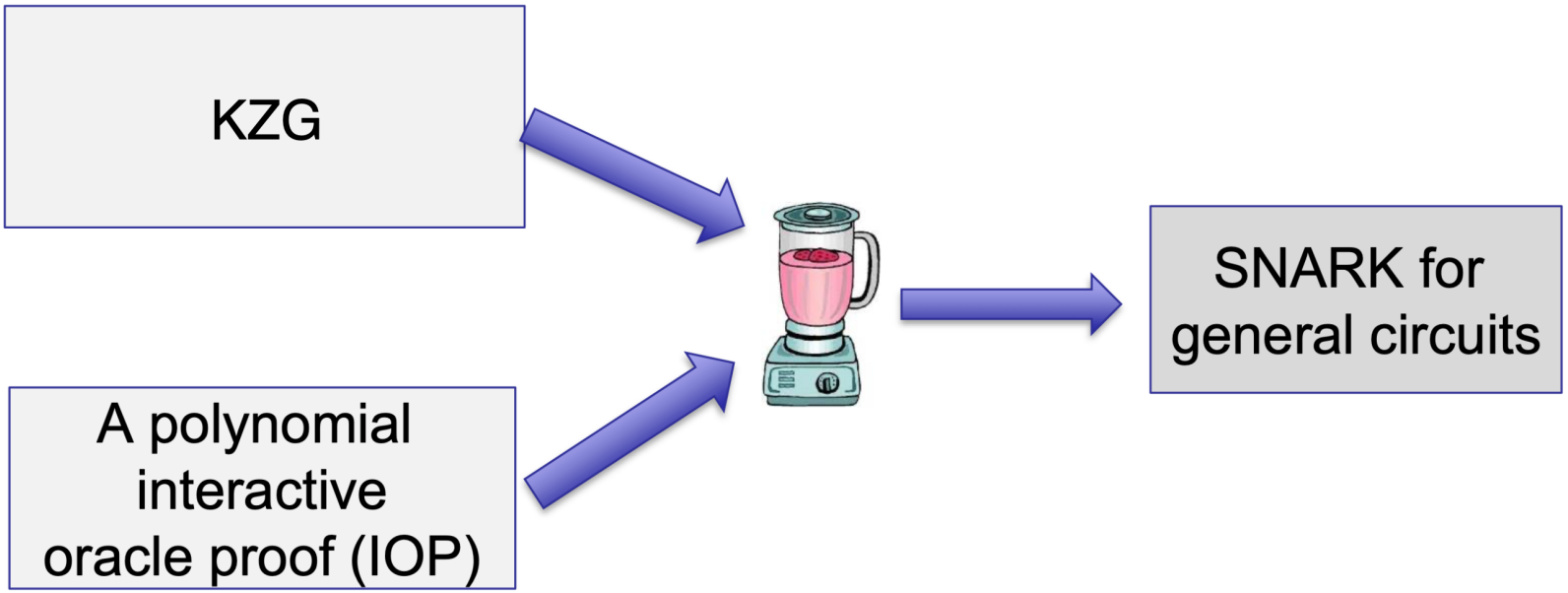
- providing a formal proof
  $\Rightarrow$ providing a machine-checkable proof
- undecidable
  $\Rightarrow$ *Pain in the A***

Why formally verify the KZG?

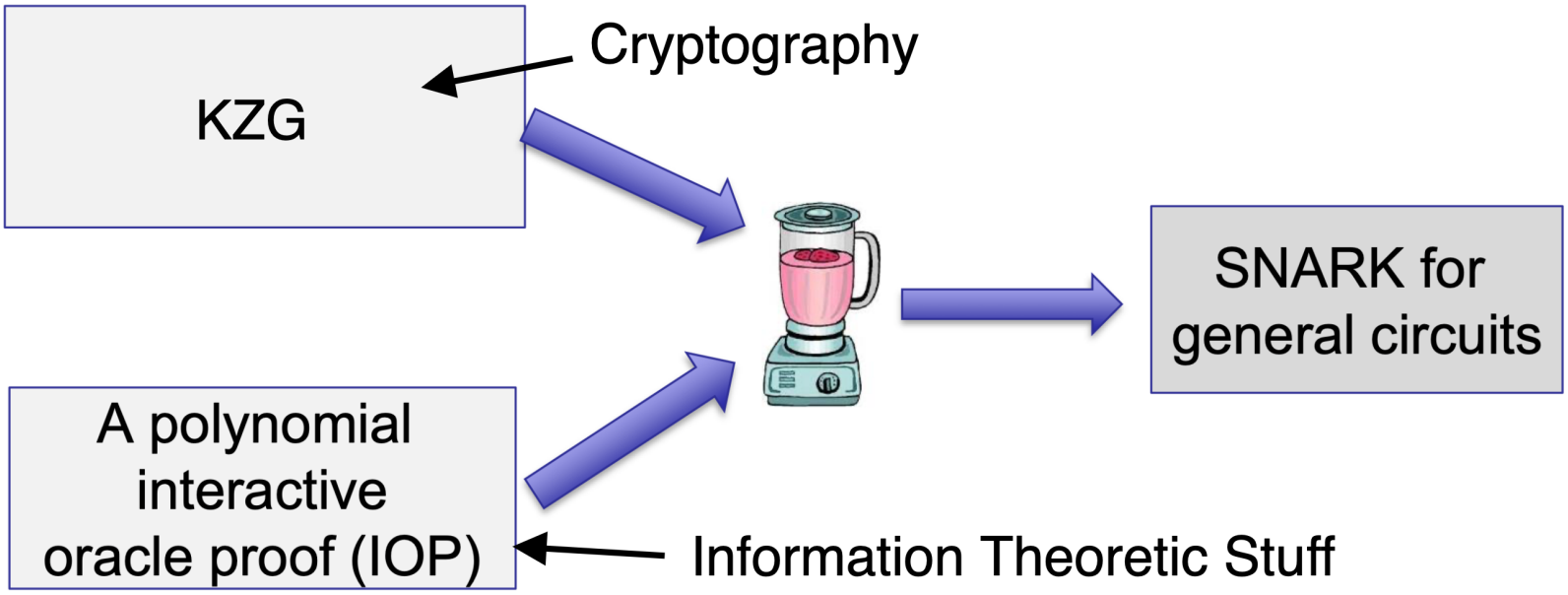
Why formally verify the KZG?



Why formally verify the KZG?



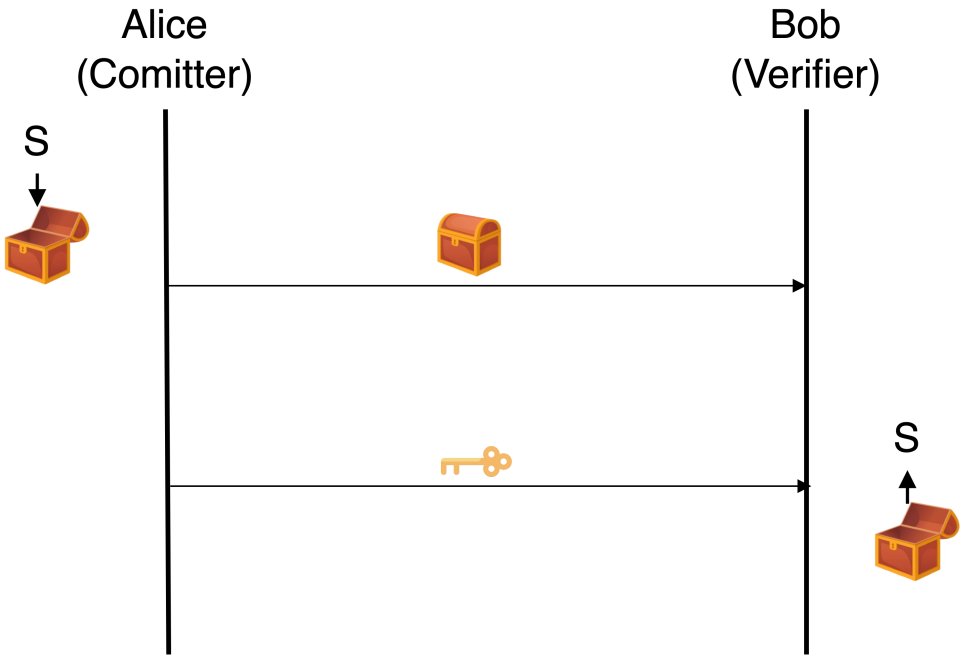
Why formally verify the KZG?



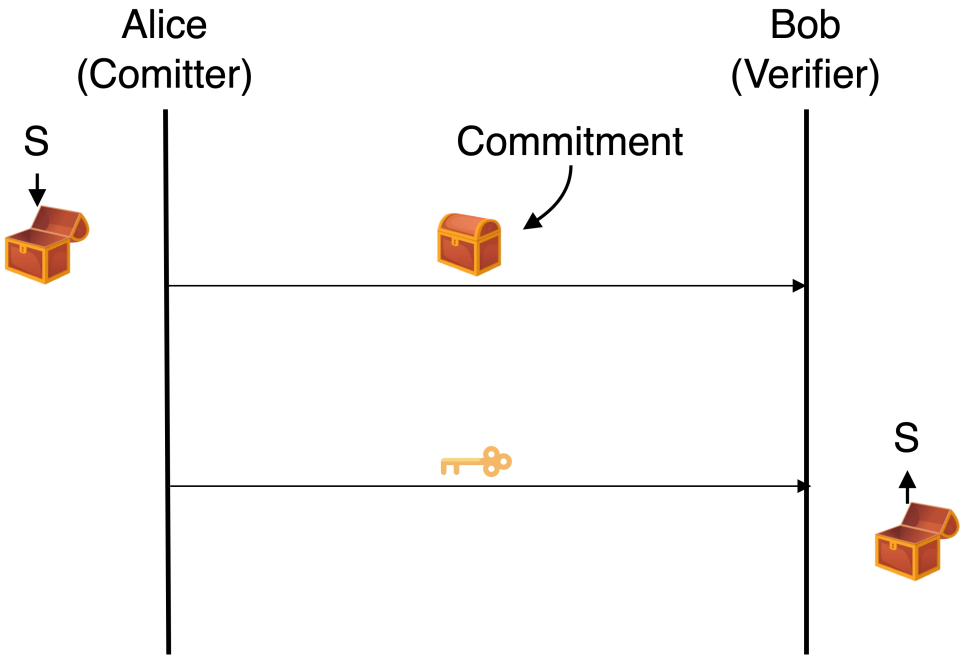
Polynomial Commitment Scheme

Commitment Scheme

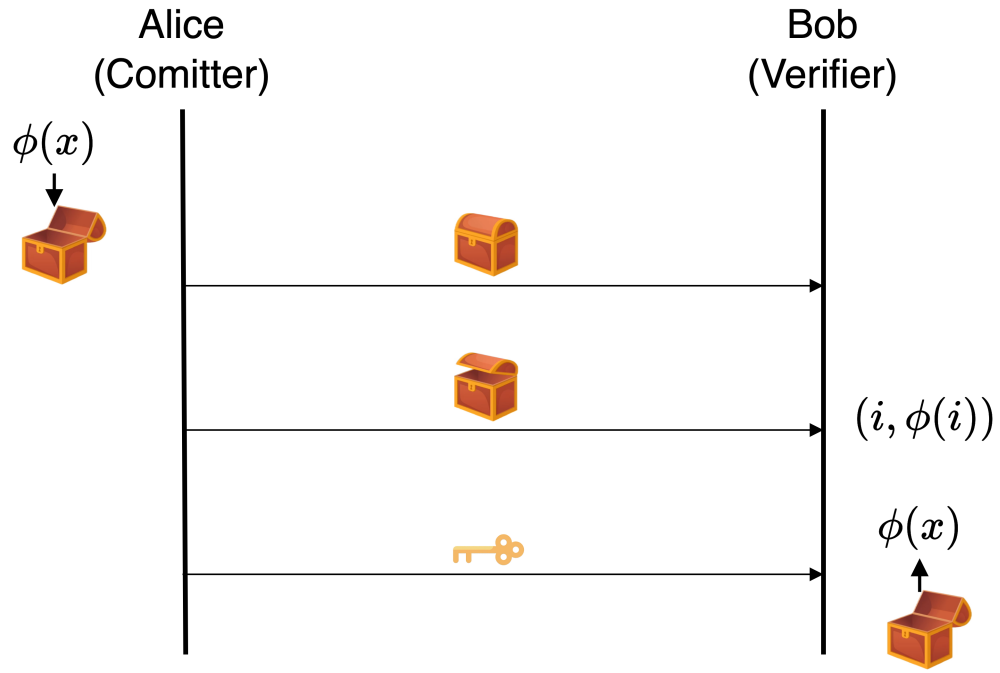
Commitment Scheme



Commitment Scheme

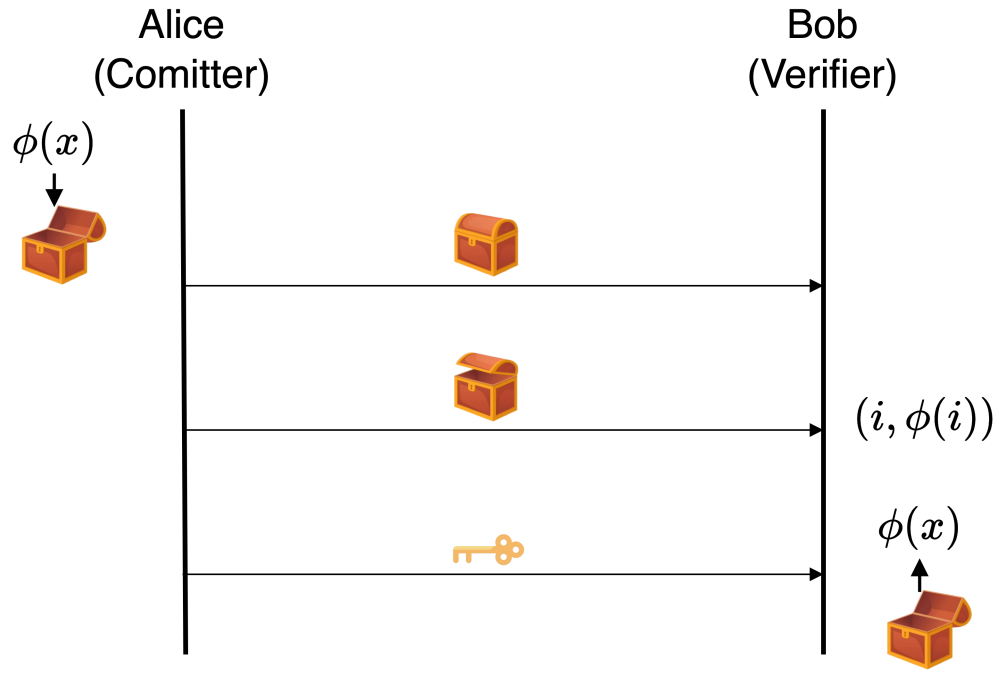


Commitment Scheme

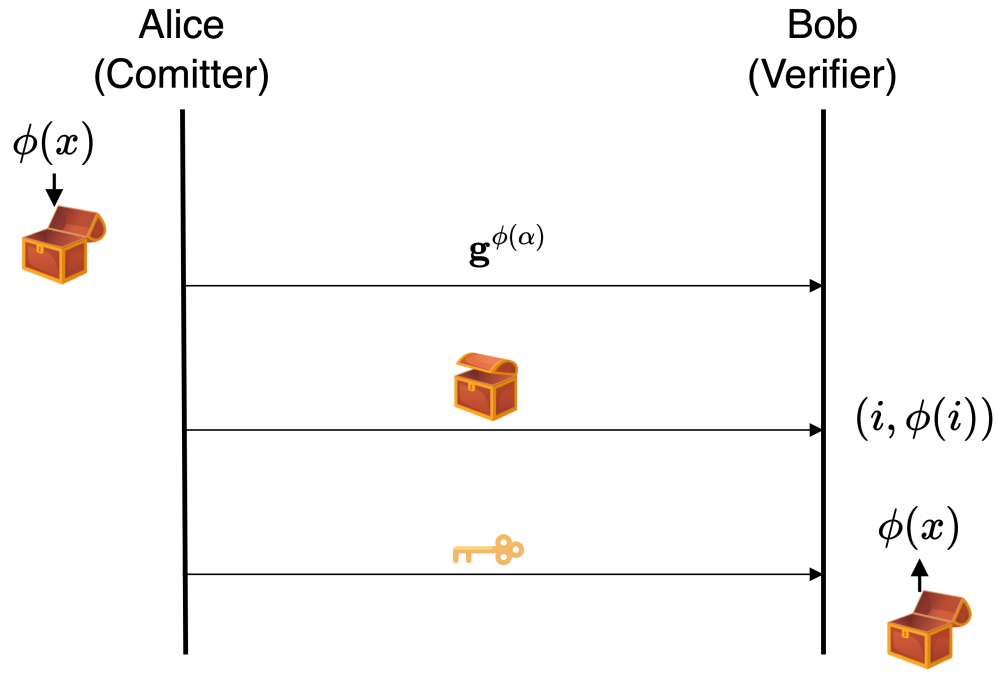


The KZG

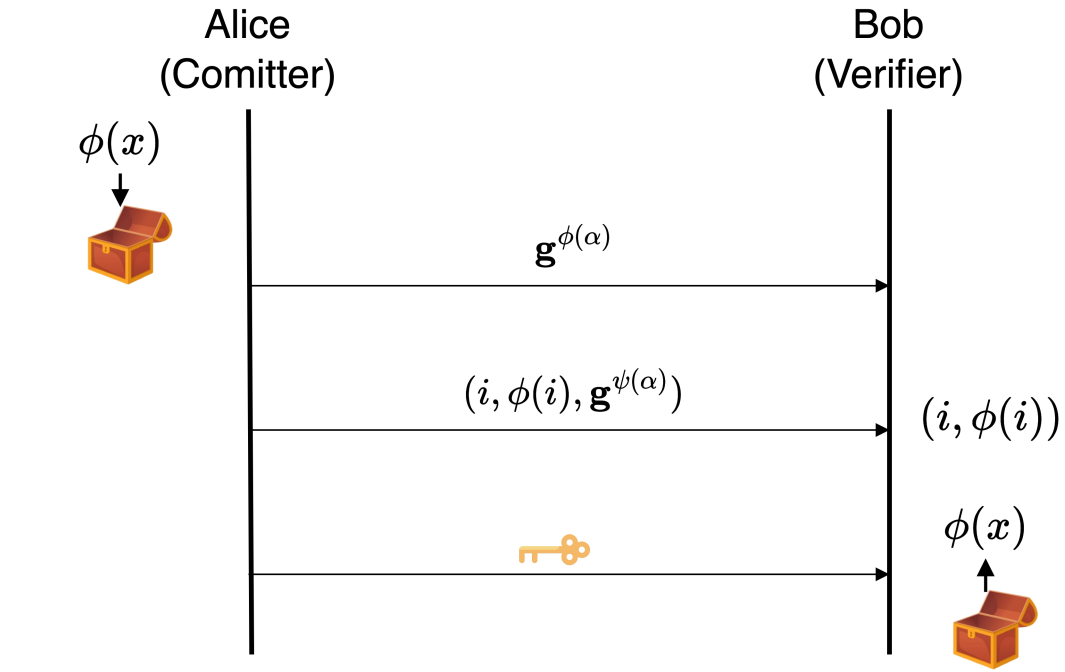
KZG - Polynomial Commitment Scheme



KZG - Polynomial Commitment Scheme



KZG - Polynomial Commitment Scheme



KZG - how to prove a peek? (Intuition)

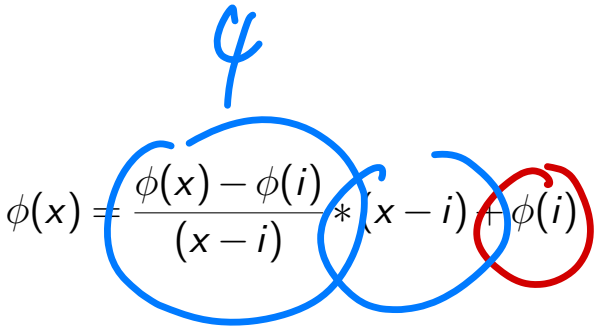
Commitment - $\mathbf{g}^{\phi(\alpha)}$

Peek - $(i, \phi(i), \mathbf{g}^{\psi(\alpha)})$

KZG - how to prove a peek? (Intuition)

Commitment - $\mathbf{g}^{\phi(\alpha)}$

Peek - $(i, \phi(i), \mathbf{g}^{\psi(\alpha)})$



KZG - how to prove a peek? (Intuition)

Commitment - $\mathbf{g}^{\phi(\alpha)}$

Peek - $(i, \phi(i), \mathbf{g}^{\psi(\alpha)})$

$$\phi(x) = \psi(x) * (x - i) + \phi(i)$$

KZG - how to prove a peek? (Intuition)

Commitment - $\mathbf{g}^{\phi(\alpha)}$

Peek - $(i, \phi(i), \mathbf{g}^{\psi(\alpha)})$

$x = i$

$$\phi(x) = \psi(x) * (x - i) + \phi(i)$$

$\phi(i) = \phi(i)$

KZG - how to prove a peek? (Intuition)

Commitment - $\mathbf{g}^{\phi(\alpha)}$

Peek - $(i, \phi(i), \mathbf{g}^{\psi(\alpha)})$

g^{α}

$$\phi(x) = \psi(x) * (x - i) + \phi(i)$$

$G \times G \to G_T$

$e(g^a, g^b) = g^{a \cdot b}$

$\Downarrow$ Schwarz-Zippel

$$\phi(\alpha) = \psi(\alpha) * (\alpha - i) + \phi(i)$$

$g^{\phi(\alpha)^{\alpha}}$

KZG - how to prove a peek? (Intuition)

Commitment - $\mathbf{g}^{\phi(\alpha)}$

Peek - $(i, \phi(i), \mathbf{g}^{\psi(\alpha)})$

$$\phi(x) = \psi(x) * (x - i) + \phi(i)$$

$$\Downarrow \text{ Schwarz-Zippel}$$

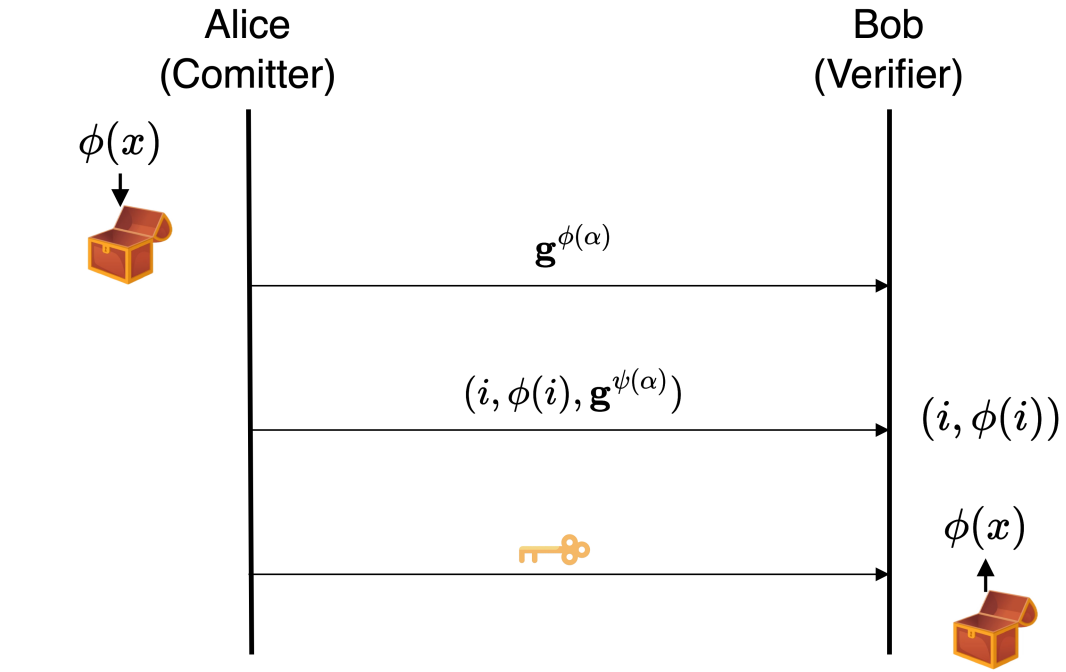
$$\phi(\alpha) = \psi(\alpha) * (\alpha - i) + \phi(i)$$

$$\Downarrow \text{ Pairing}$$

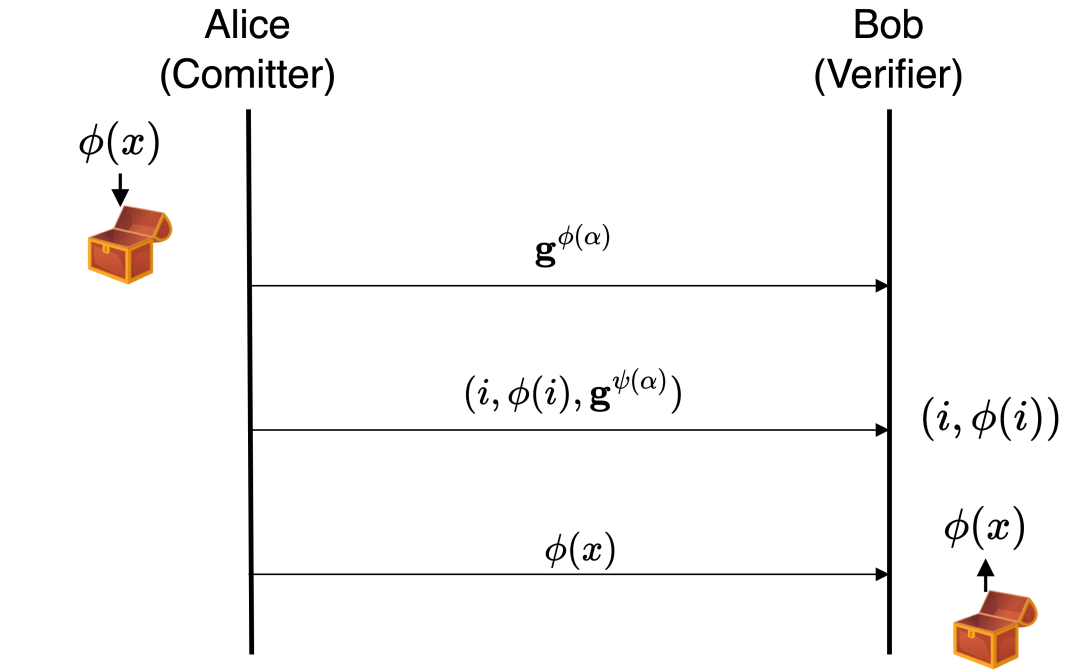
$$e\left(\mathbf{g}^{\phi(\alpha)}, \mathbf{g}\right) = e\left(\mathbf{g}^{\psi(\alpha)}, \frac{\mathbf{g}^{\alpha}}{\mathbf{g}^{i}}\right) \cdot e\left(\mathbf{g}^{\phi(i)}, \mathbf{g}\right)$$

Disclaimer: This was soley intuition.
Not security proofs.

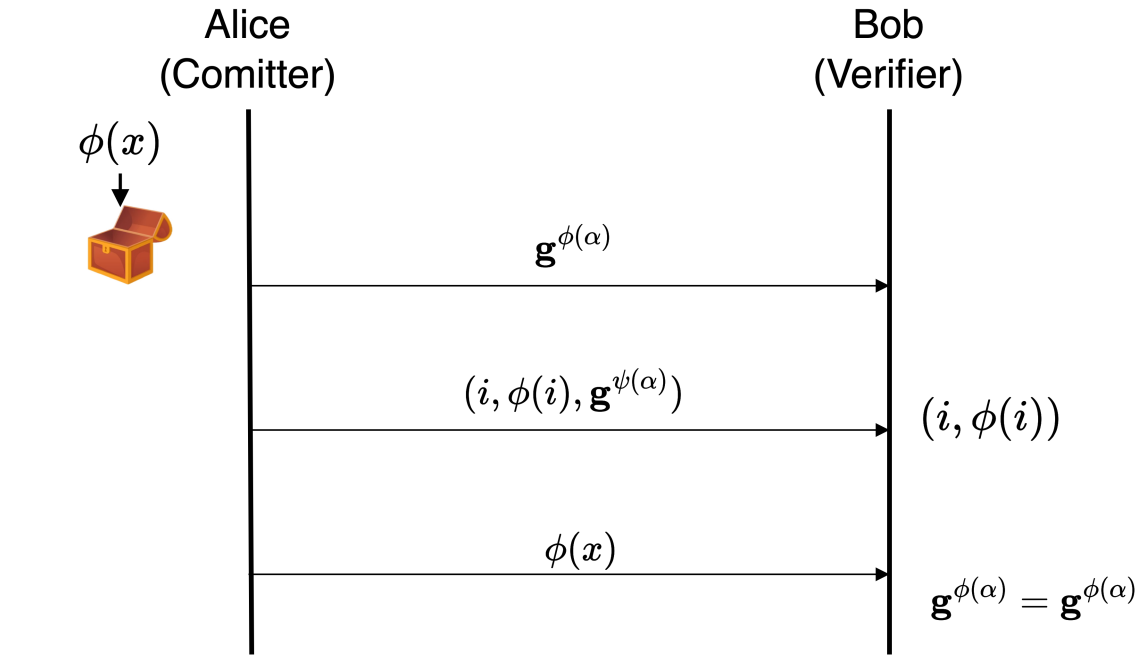
KZG - Polynomial Commitment Scheme



KZG - Polynomial Commitment Scheme



KZG - Polynomial Commitment Scheme



Formal Verification

What do we formally verify? (Highlevel)

What do we formally verify? (Highlevel)

Security proofs for the KZG:

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 
- evaluation binding

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 📦
- evaluation binding $\rightarrow$ Alice cannot reveal 📦$:=(i, \phi(i))$ and 📦$' := (i, \phi(i)')$ with 📦$\neq$ 📦$' \equiv \phi(i) \neq \phi(i)'$

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 📦
- evaluation binding $\rightarrow$ Alice cannot reveal 📦$:=(i, \phi(i))$ and 📦$' := (i, \phi(i)')$ with 📦$\neq$ 📦$' \equiv \phi(i) \neq \phi(i)'$
- hiding

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 🧰
- evaluation binding $\rightarrow$ Alice cannot reveal 🧰$:=(i, \phi(i))$ and 🧰$' := (i, \phi(i)')$ with 🧰$\neq$ 🧰$' \equiv \phi(i) \neq \phi(i)'$
- hiding $\rightarrow$ Bob cannot infer $\phi(x)$ from 🧰

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 🧰
- evaluation binding $\rightarrow$ Alice cannot reveal 🧰$:=(i, \phi(i))$ and 🧰$' := (i, \phi(i)')$ with 🧰$\neq$ 🧰$' \equiv \phi(i) \neq \phi(i)'$
- hiding $\rightarrow$ Bob cannot infer $\phi(x)$ from 🧰
- knowledge soundness

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 🧰
- evaluation binding $\rightarrow$ Alice cannot reveal 🧰$:=(i, \phi(i))$ and 🧰$' := (i, \phi(i)')$ with 🧰$\neq$ 🧰$' \equiv \phi(i) \neq \phi(i)'$
- hiding $\rightarrow$ Bob cannot infer $\phi(x)$ from 🧰
- knowledge soundness $\rightarrow$ Alice cannot reveal points without knowing $\phi(x)$ in 🧰

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynom
- evaluatic $(i, \phi(i)')$ with 📦≠📦
- hiding –
- knowledge soundness → Alice cannot reveal points without knowing $\phi(x)$ in 📦

Standard PCS Security Properties

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynom
- evaluatic $(i, \phi(i)')$ with 📦≠📦
- hiding –
- knowledge soun nowing $\phi(x)$ in 📦

Standard PCS Security Properties SNARK precondition

What do we formally verify? (Highlevel)

Security proofs for the KZG:

- polynomial binding $\rightarrow$ Alice cannot change $\phi(x)$ in 🧰
- evaluation binding $\rightarrow$ Alice cannot reveal 🧰$:=(i, \phi(i))$ and 🧰$' := (i, \phi(i)')$ with 🧰$\neq$ 🧰$' \equiv \phi(i) \neq \phi(i)'$
- hiding $\rightarrow$ Bob cannot infer $\phi(x)$ from 🧰
- knowledge soundness $\rightarrow$ Alice cannot reveal points without knowing $\phi(x)$ in 🧰

What do we formally verify?

Games against adversary Eve

What do we formally verify?

Games against adversary Eve

$\rightarrow$ games as probabilistic algorithms

What do we formally verify?

Games against adversary Eve

$\rightarrow$ games as probabilistic algorithms

$\rightarrow$ Eve modeled as arbitrary probabilistic algorithm

What do we formally verify?

Games against adversary Eve

$\rightarrow$ games as probabilistic algorithms

$\rightarrow$ Eve modeled as arbitrary probabilistic algorithm

$\rightarrow$ Goal: prove

$$Pr(\text{Eve wins game}) \approx 0$$

What do we formally verify?

Games against adversary Eve

→ games as probabilistic algorithms

→ Eve modeled as arbitrary probabilistic algorithm

→ Goal: prove

$$Pr(\text{Eve wins game}) \approx 0$$

Example Hiding Game:

$$\left(\begin{array}{c} C \leftarrow \text{📦}, \\ wtnss \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow Eve(C, wtnss), \\ : \ \phi = \phi' \end{array}\right)$$

How do we prove this?

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\left(\begin{array}{c} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow Eve(\mathbf{g}^a) \\ : a = a' \end{array}\right)$$

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\left(\begin{array}{c} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{array}\right)$$

remember: Hiding game:

$$\left(\begin{array}{c} C \leftarrow \text{🧰}, \\ \mathit{wtnss} \leftarrow \text{🧰🧰🧰🧰🧰🧰}\ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{array}\right)$$

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\left(\begin{array}{c} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{array}\right)$$

remember: Hiding game:

$$\left(\begin{array}{c} C \leftarrow \text{📦} := \mathbf{g}^{\phi(\alpha)}, \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{array}\right)$$

How do we prove this?

Reduction:

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\left(\begin{array}{c} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{array}\right)$$

remember: Hiding game:

$$\left(\begin{array}{c} C \leftarrow \text{📦}, \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{array}\right)$$

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\left(\begin{array}{c} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{array}\right)$$

remember: Hiding game:

$$\left(\begin{array}{c} C \leftarrow \text{  }, \\ \mathit{wtnss} \leftarrow \text{  }\ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{array}\right)$$

Reduction:

let $I = [(y, \mathbf{g}^b), \ldots]$

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix}$$

remember: Hiding game:

$$\begin{pmatrix} C \leftarrow \text{📦}, \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{pmatrix}$$

Reduction:

let $I = [(y, \mathbf{g}^b), \ldots]$

$$\begin{pmatrix} \mathbf{g}^{\phi(\alpha)} \leftarrow \text{interpolate}((x, \mathbf{g}^a) \# I), \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(\mathbf{g}^{\phi(\alpha)}, \mathit{wtnss}), \\ : \ \phi = \phi', \\ \phi(x) \simeq \alpha \end{pmatrix}$$

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\left(\begin{array}{l} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ \quad : a = a' \end{array}\right)$$

remember: Hiding game:

$$\left(\begin{array}{l} C \leftarrow \text{📦}, \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ \quad : \ \phi = \phi' \end{array}\right)$$

Reduction:

let $I = [(y, \mathbf{g}^b), \ldots]$

$$\left(\begin{array}{l} \mathbf{g}^{\phi(\alpha)} \leftarrow \text{interpolate}((x, \mathbf{g}^a) \# I), \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(\mathbf{g}^{\phi(\alpha)}, \mathit{wtnss}), \\ \quad : \ \phi = \phi', \\ \phi(x) \end{array}\right)$$

$\Rightarrow$ Eve efficient $\Longrightarrow$ DL broken

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix}$$

remember: Hiding game:

$$\begin{pmatrix} C \leftarrow \text{🧰}, \\ \mathit{wtnss} \leftarrow \text{🧰🧰🧰🧰🧰🧰} \ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{pmatrix}$$

Reduction:

let $I = [(y, \mathbf{g}^b), \ldots]$

$$\begin{pmatrix} \mathbf{g}^{\phi(\alpha)} \leftarrow \text{interpolate}((x, \mathbf{g}^a) \# I), \\ \mathit{wtnss} \leftarrow \text{🧰🧰🧰🧰🧰🧰} \ldots, \\ \phi' \leftarrow \mathit{Eve}(\mathbf{g}^{\phi(\alpha)}, \mathit{wtnss}), \\ : \ \phi = \phi', \\ \phi(x) \end{pmatrix}$$

$\Rightarrow$ Eve efficient $\implies$ DL broken
$\Rightarrow$ Is this reduction cryptographically sound?

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix}$$

remember: Hiding game:

$$\begin{pmatrix} C \leftarrow \text{🧰}, \\ \mathit{wtnss} \leftarrow \text{🧰🧰🧰🧰🧰🧰} \ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{pmatrix}$$

Reduction:

let $I = [(y, \mathbf{g}^b), \ldots]$

$$\begin{pmatrix} \mathbf{g}^{\phi(\alpha)} \leftarrow \text{interpolate}((x, \mathbf{g}^a) \# I), \\ \mathit{wtnss} \leftarrow \text{🧰🧰🧰🧰🧰🧰} \ldots, \\ \phi' \leftarrow \mathit{Eve}(\mathbf{g}^{\phi(\alpha)}, \mathit{wtnss}), \\ : \ \phi = \phi', \\ \phi(x) \end{pmatrix}$$

$\Rightarrow$ Eve efficient $\Longrightarrow$ DL broken
$\Rightarrow$ Is this reduction cryptographically sound?
NO!

How do we prove this?

$\Rightarrow$ Reduction from Hard Problem

Example Discrete Log:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z}_p \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix}$$

remember: Hiding game:

$$\begin{pmatrix} C \leftarrow \text{📦}, \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(C, \mathit{wtnss}), \\ : \ \phi = \phi' \end{pmatrix}$$

Reduction:

let $I = [(y, \mathbf{g}^b), \ldots]$

$$\begin{pmatrix} \mathbf{g}^{\phi(\alpha)} \leftarrow \text{interpolate}((x, \mathbf{g}^a) \# I), \\ \mathit{wtnss} \leftarrow \text{📦📦📦📦📦📦}\ldots, \\ \phi' \leftarrow \mathit{Eve}(\mathbf{g}^{\phi(\alpha)}, \mathit{wtnss}), \\ : \ \phi = \phi', \\ \phi(x) \end{pmatrix}$$

$\Rightarrow$ Eve efficient $\Longrightarrow$ DL broken
$\Rightarrow$ Is this reduction cryptographically sound?
NO!
$\Rightarrow$ "no adversary can break hiding for arbitrary polynomials"
vs "for arbitrary polynomials no adversary can break hiding"

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops $\Rightarrow$ show security game $=$ hardness game for some adversary

$$\text{Security Game} = \left(\quad \dots \quad \right) = \left(\quad \dots \quad \right) \leq \left(\quad \dots \quad \right) = \cdots = \left(\quad \dots \quad \right) = \text{Hardness Assumption Game}$$

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops $\Rightarrow$ show security game $=$ hardness game for some adversary

$$\text{Security Game} = \left(\quad \dots \quad \right) = \left(\quad \dots \quad \right) \leq \left(\quad \dots \quad \right) = \cdots = \left(\quad \dots \quad \right) = \text{Hardness Assumption Game}$$

1. Bridging step:

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops $\Rightarrow$ show security game $=$ hardness game for some adversary

$$\text{Security Game} = \left(\quad \dots \quad \right) = \left(\quad \dots \quad \right) \leq \left(\quad \dots \quad \right) = \cdots = \left(\quad \dots \quad \right) = \text{Hardness Assumption Game}$$

1. Bridging step:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} = \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a' = a \end{pmatrix}$$

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops $\Rightarrow$ show security game $=$ hardness game for some adversary

$$\text{Security Game} = \left(\quad \dots \quad \right) = \left(\quad \dots \quad \right) \leq \left(\quad \dots \quad \right) = \cdots = \left(\quad \dots \quad \right) = \text{Hardness Assumption Game}$$

1. Bridging step:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} = \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a' = a \end{pmatrix}$$

2. Failure-event:

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops $\Rightarrow$ show security game $=$ hardness game for some adversary

$$\text{Security Game} = \left(\quad \dots \quad \right) = \left(\quad \dots \quad \right) \leq \left(\quad \dots \quad \right) = \cdots = \left(\quad \dots \quad \right) = \text{Hardness Assumption Game}$$

1. Bridging step:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} = \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a' = a \end{pmatrix}$$

2. Failure-event:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} \approx \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^{\frac{1}{a}}) \\ : \frac{1}{a} = a' \end{pmatrix}$$

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops $\Rightarrow$ show security game $=$ hardness game for some adversary

$$\text{Security Game} = \left(\quad \dots \quad \right) = \left(\quad \dots \quad \right) \leq \left(\quad \dots \quad \right) = \cdots = \left(\quad \dots \quad \right) = \text{Hardness Assumption Game}$$

1. Bridging step:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} = \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a' = a \end{pmatrix}$$

2. Failure-event:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} \approx \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^{\frac{1}{a}}) \\ : \frac{1}{a} = a' \end{pmatrix}$$

3. Over-estimation:

A Better Way - Sequences of Games [Shoup04]

Transformation via Game-hops $\Rightarrow$ show security game $=$ hardness game for some adversary

$$\text{Security Game} = \left(\quad \dots \quad \right) = \left(\quad \dots \quad \right) \leq \left(\quad \dots \quad \right) = \cdots = \left(\quad \dots \quad \right) = \text{Hardness Assumption Game}$$

1. Bridging step:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} = \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a' = a \end{pmatrix}$$

2. Failure-event:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} \approx \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^{\frac{1}{a}}) \\ : \frac{1}{a} = a' \end{pmatrix}$$

3. Over-estimation:

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \\ : a = a' \end{pmatrix} \leq \begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow \mathit{Eve}(\mathbf{g}^a) \end{pmatrix}$$

How do we formally VERIFY this?

$\Rightarrow$ turn formal proof into machine checkable proof:

How do we formally VERIFY this?

$\Rightarrow$ turn formal proof into machine checkable proof:

```
TRY do {
    a  ← sample_uniform p;
    a' ← Eve(g^a);
    return_spmf (a = a')
} ELSE return_spmf False
```

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow Eve(\mathbf{g}^a) \\ : a = a' \end{pmatrix}$$

How do we formally VERIFY this?

$\Rightarrow$ turn formal proof into machine checkable proof:

```
TRY do {
    a  ← sample_uniform p;
    a' ← Eve(g^a);
    return_spmf (a = a')
} ELSE return_spmf False
```

$$\begin{pmatrix} a \xleftarrow{\$} \mathbb{Z} \\ a' \leftarrow Eve(\mathbf{g}^a) \\ : a = a' \end{pmatrix}$$

$\Rightarrow$ prove game-hop correctness in Higher Order Logic (HOL) (machine checked)

Summary

Summary

- first formalization of a Polynomial Commitment Scheme (PCS)

Summary

- first formalization of a Polynomial Commitment Scheme (PCS)
- show all typical cryptographic security properties

Summary

- first formalization of a Polynomial Commitment Scheme (PCS)
- show all typical cryptographic security properties
- first step to formaliy verify KZG-based SNARKs

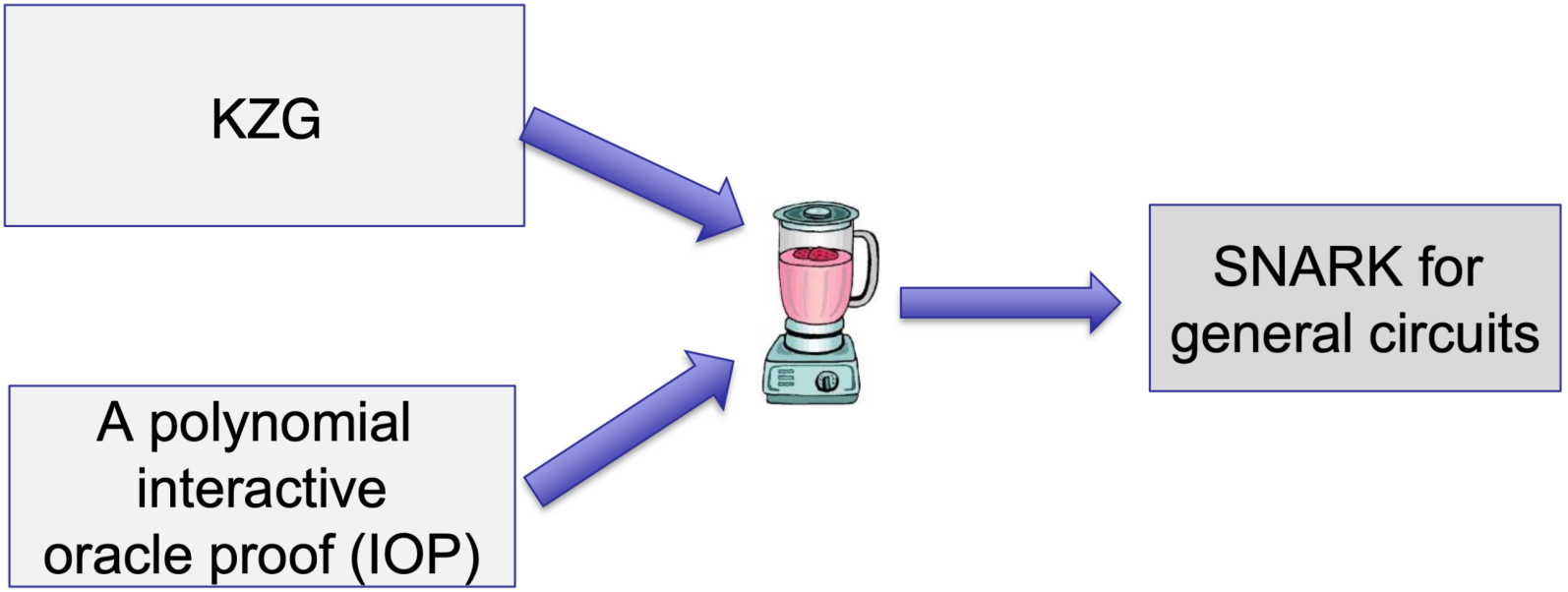
Open Problems

Open Problems

- generate formally verified Go implementation

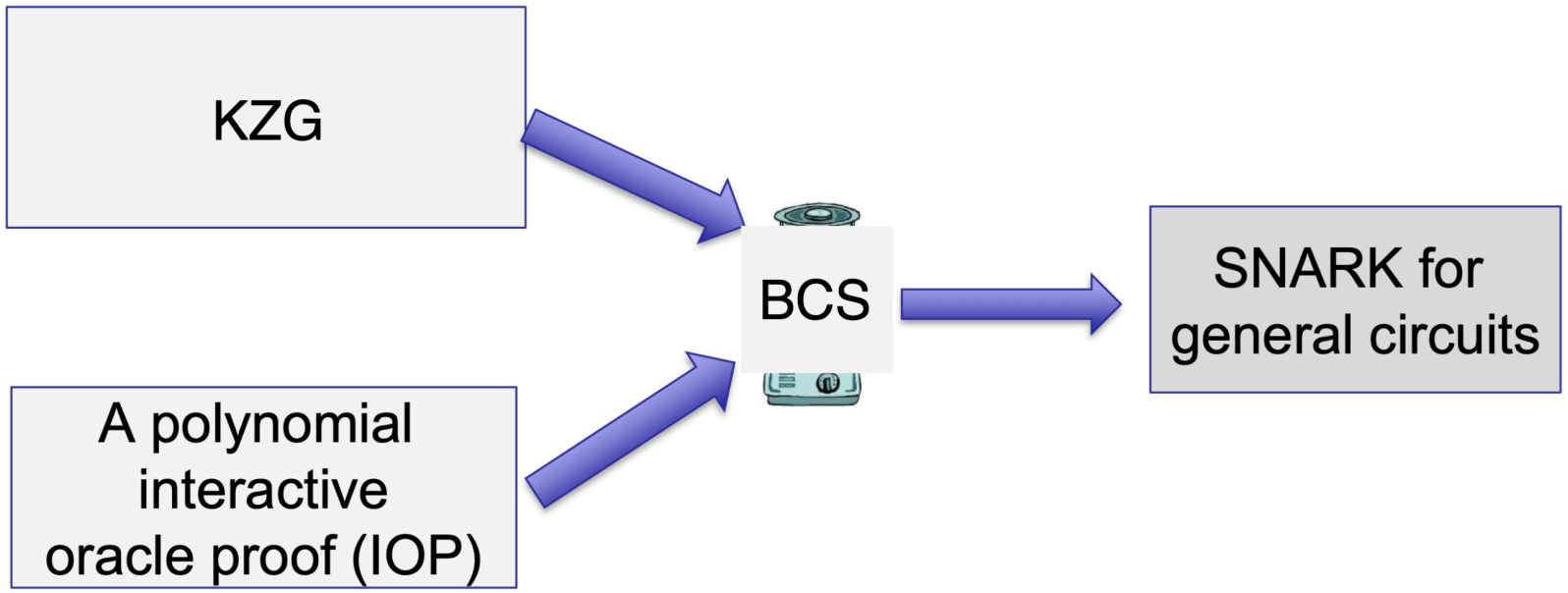
Open Problems

- generate formally verified Go implementation
- SNARKs:



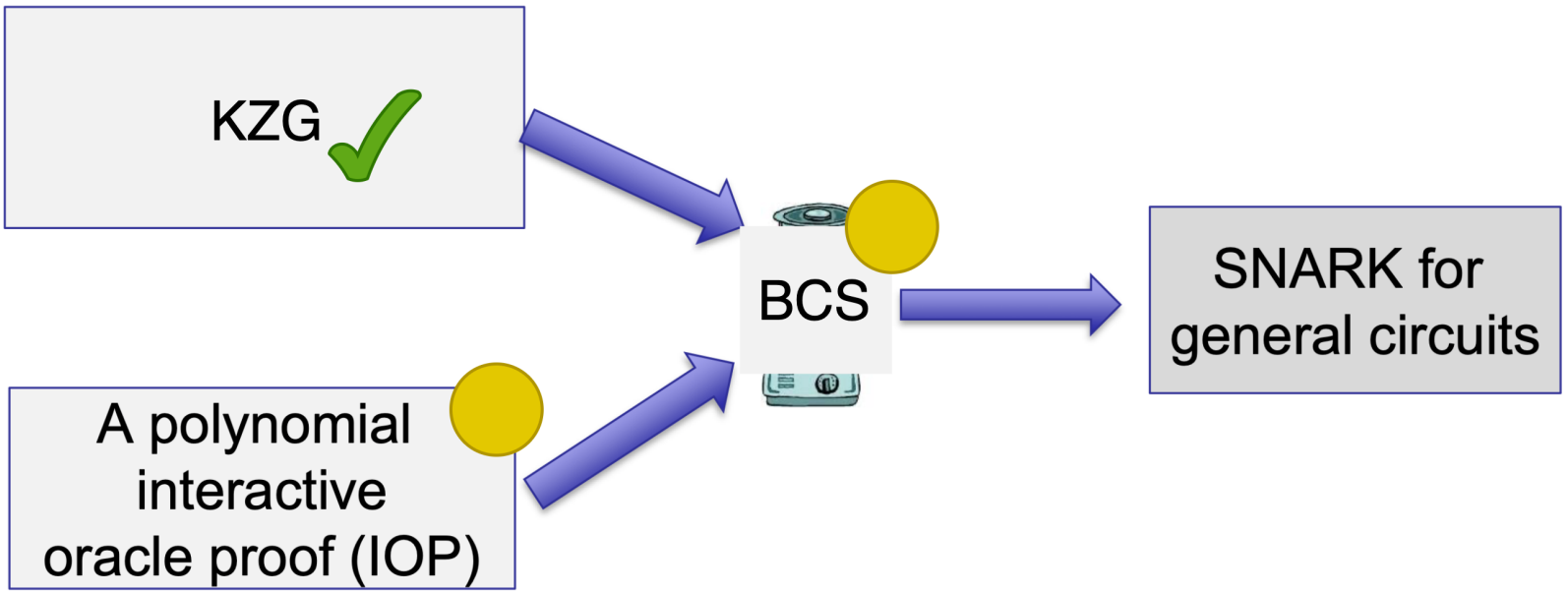
Open Problems

- generate formally verified Go implementation
- SNARKs:



Open Problems

- generate formally verified Go implementation
- SNARKs:



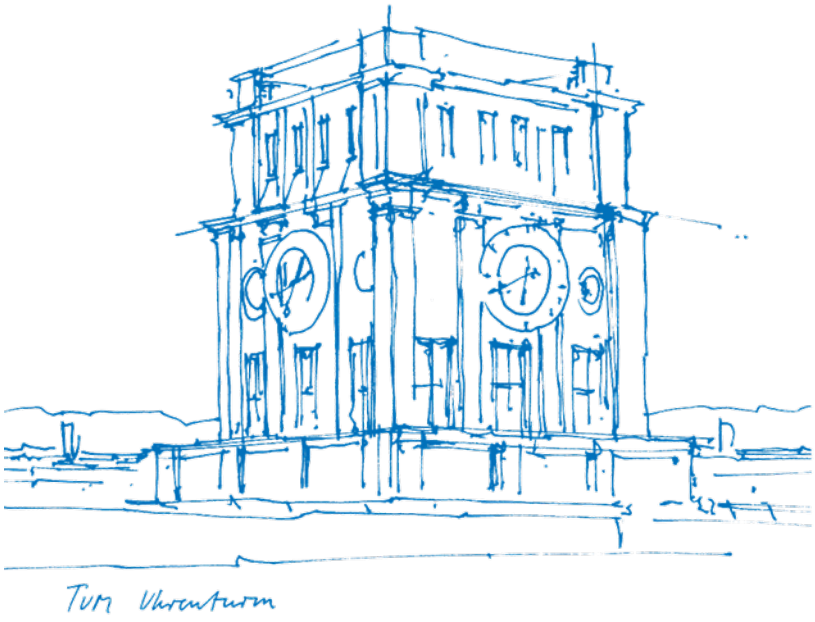
References

- Isabelle formalization of the KZG:
  https://github.com/tobias-rothmann/KZG-Polynomial-Commitment-Scheme
- Kate, Zaverucha, Goldberg - Constant-Size Commitments to Polynomials and Their Applications
  https://www.iacr.org/archive/asiacrypt2010/6477178/6477178.pdf
- Shoup - Sequences of games: a tool for taming complexity in security proofs
  https://eprint.iacr.org/2004/332
- KZG-based SNARKs - Plonk, Halo, Sonic, Marlin, Nova:
  https://eprint.iacr.org/2019/953.pdf, https://eprint.iacr.org/2020/1536,
  https://eprint.iacr.org/2019/099, https://eprint.iacr.org/2019/1047, https://eprint.iacr.org/2021/370
- Ben-Sasson, Chiesa, Spooner - Interactive Oracle Proofs
  https://www.iacr.org/archive/tcc2016b/99850156/99850156.pdf

Chair for Logic and Verification
School of Computation, Information and Technology
Technical University of Munich

Formally Verifying the KZG Polynomial Commitment Scheme

Tobias Rothmann
Technical University of Munich
School of Computation, Information and Technology
Chair for Logic and Verification
28. Mai 2024



Tobias Rothmann
Technical University of Munich
School of Computation, Information and Technology
Chair for Logic and Verification
28. Mai 2024