# Constant-Size zk-SNARKs in ROM from Falsifiable Assumptions

Helger Lipmaa, University of Tartu

Roberto Parisella, Simula UiB

Janno Siim, Simula UiB

# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$

# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public

# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private

# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- <u>Prover claims:</u> there is $w$ such that $(x, w) \in R$

# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- <u>Prover claims:</u> there is $w$ such that $(x, w) \in R$

**Prover** $(x, w)$

# Interactive Argument

- ❑Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- ❑Prover claims: there is $w$ such that $(x, w) \in R$

**Prover** $(x, w)$

**Verifier** $(x)$

# Interactive Argument

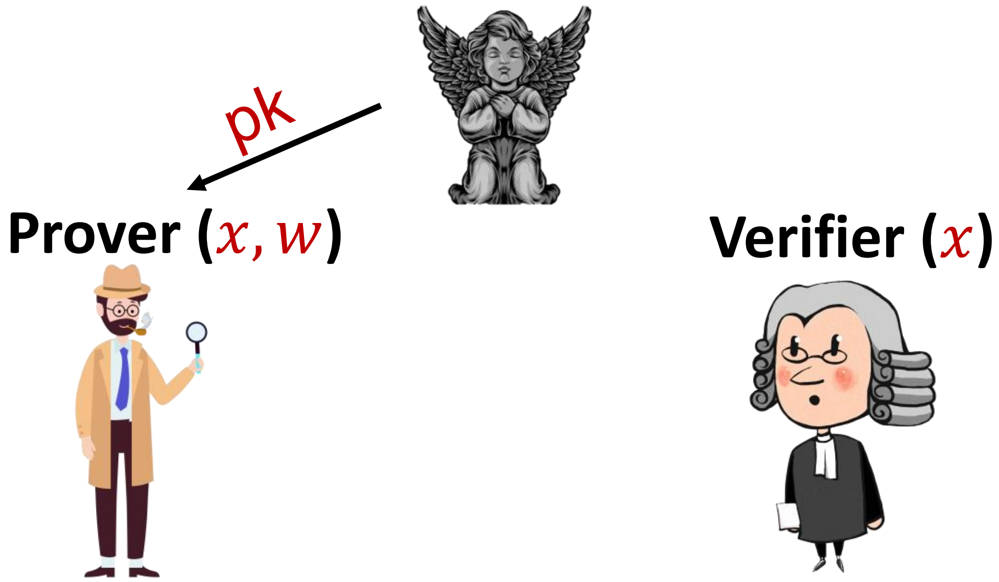
- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- Prover claims: there is $w$ such that $(x, w) \in R$

**Prover** $(x, w)$

**Verifier** $(x)$

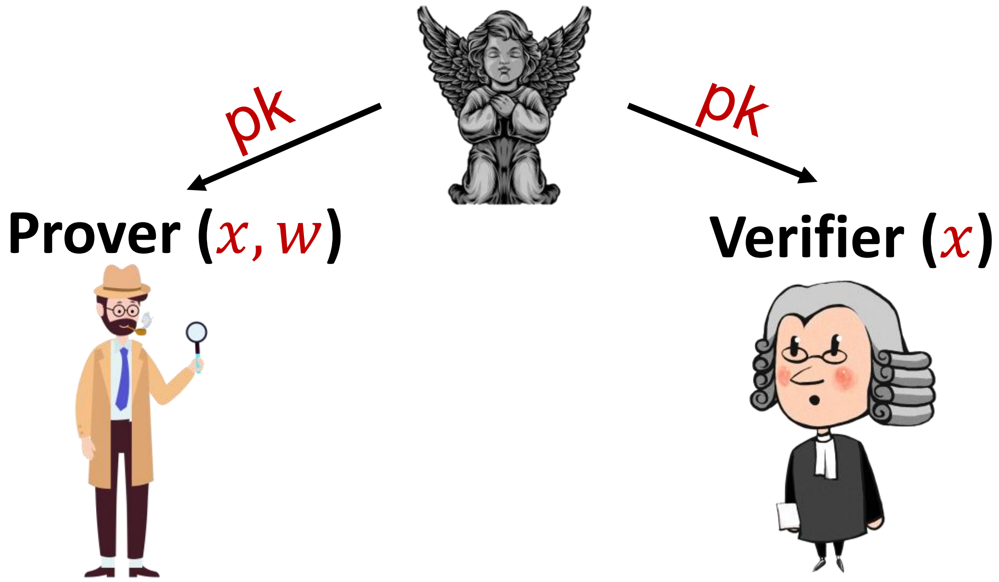
# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- <u>Prover claims:</u> there is $w$ such that $(x, w) \in R$



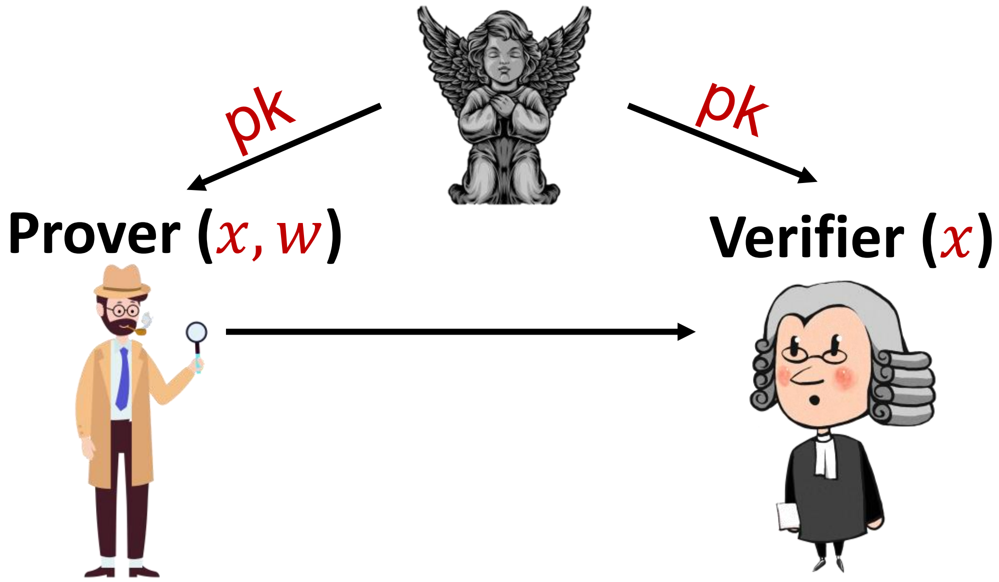
# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- <u>Prover claims:</u> there is $w$ such that $(x, w) \in R$



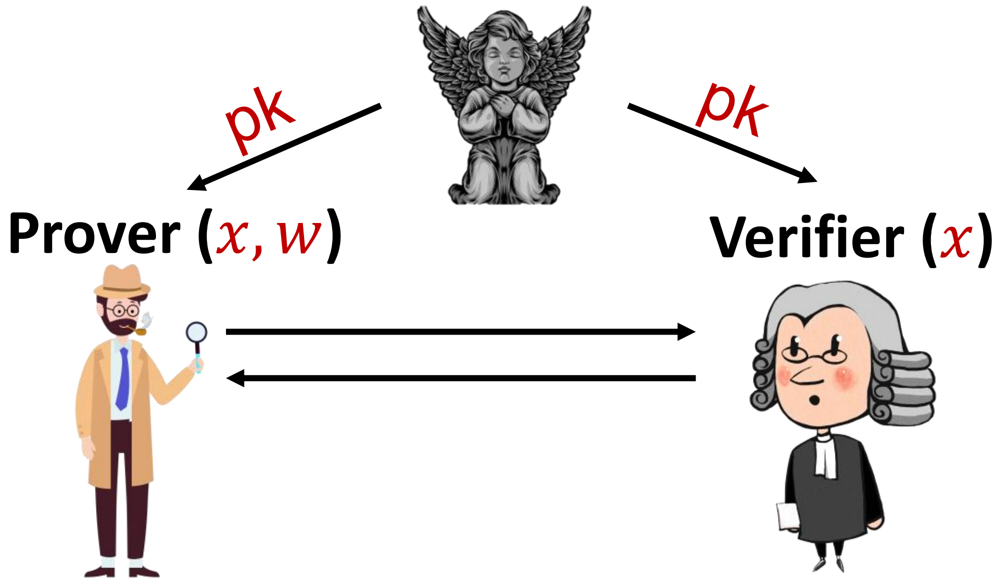
# Interactive Argument

- Relation $R = \{(x,w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- <u>Prover claims:</u> there is $w$ such that $(x,w) \in R$



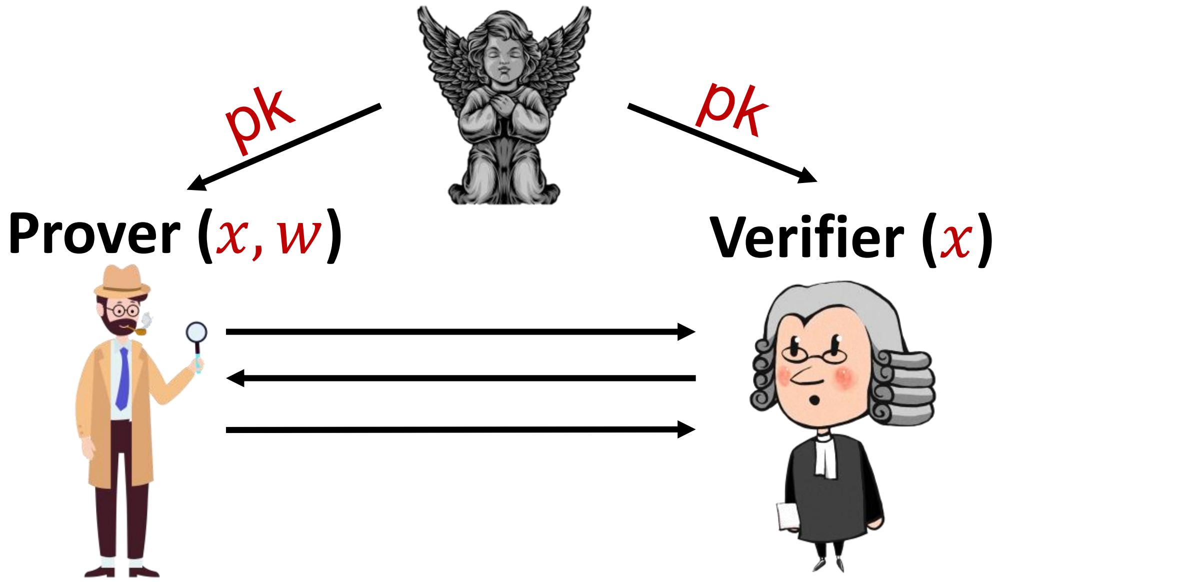
# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- <u>Prover claims:</u> there is $w$ such that $(x, w) \in R$



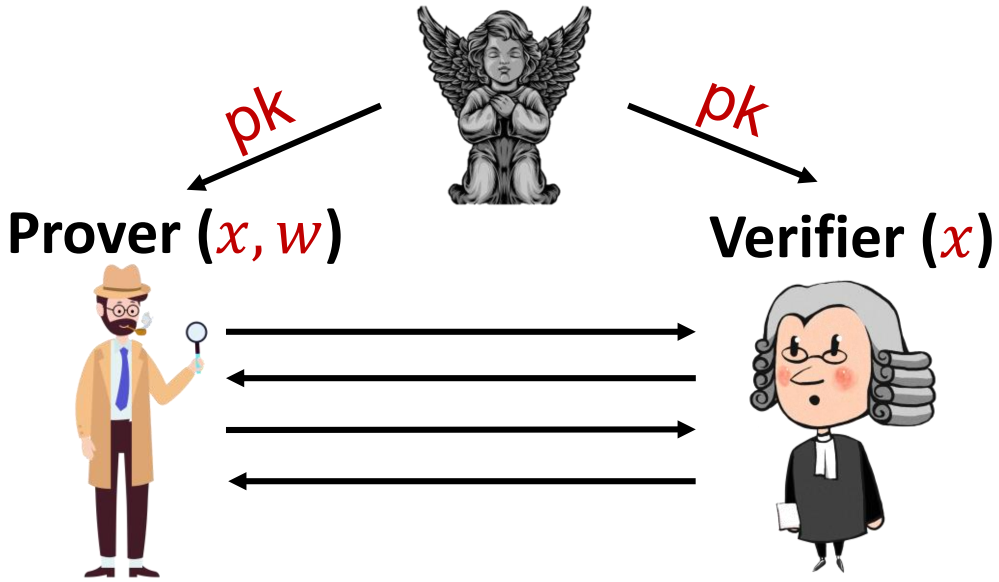
# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- <u>Prover claims:</u> there is $w$ such that $(x, w) \in R$



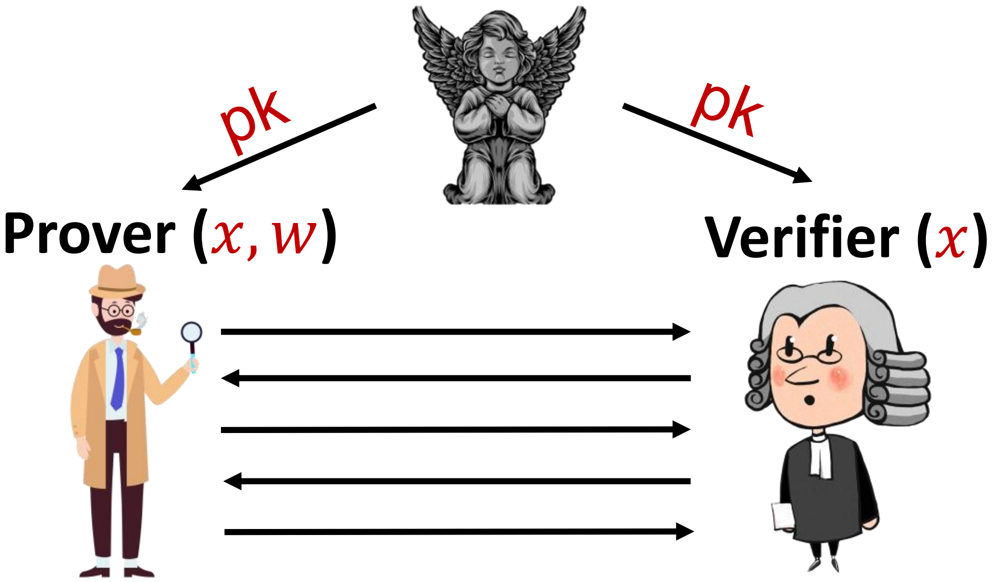
# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- Prover claims: there is $w$ such that $(x, w) \in R$



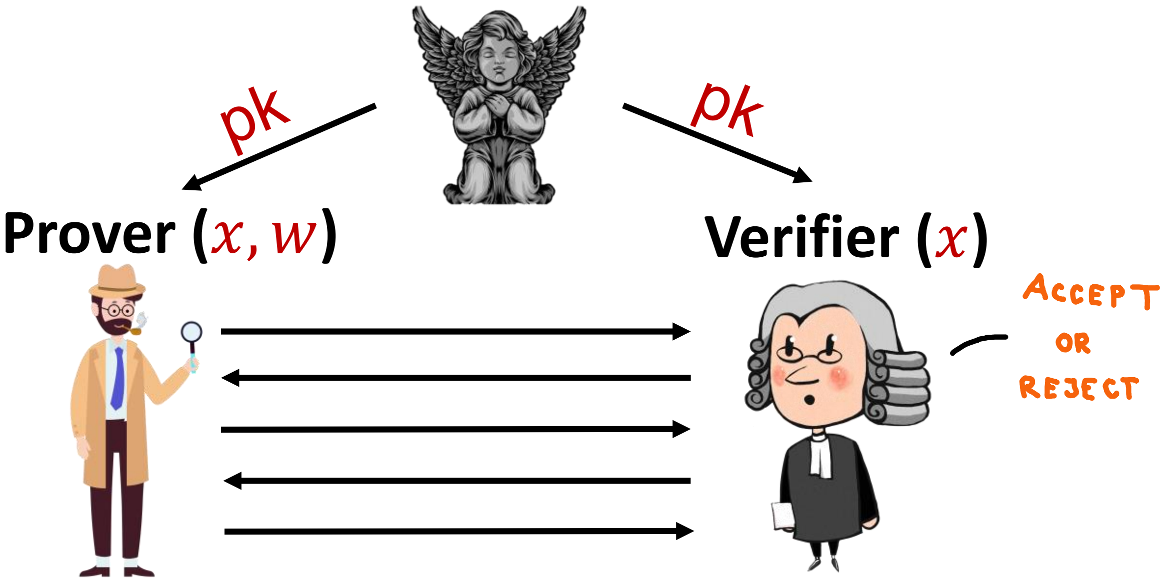
# Interactive Argument

- Relation $R = \{(x,w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- Prover claims: there is $w$ such that $(x,w) \in R$



# Interactive Argument

- Relation $R = \{(x, w)\} \subseteq \{0,1\}^*$
  - $x$ is public
  - $w$ is private
- Prover claims: there is $w$ such that $(x, w) \in R$



# SNARK

- Succinct Non-interactive Argument of Knowledge

# SNARK

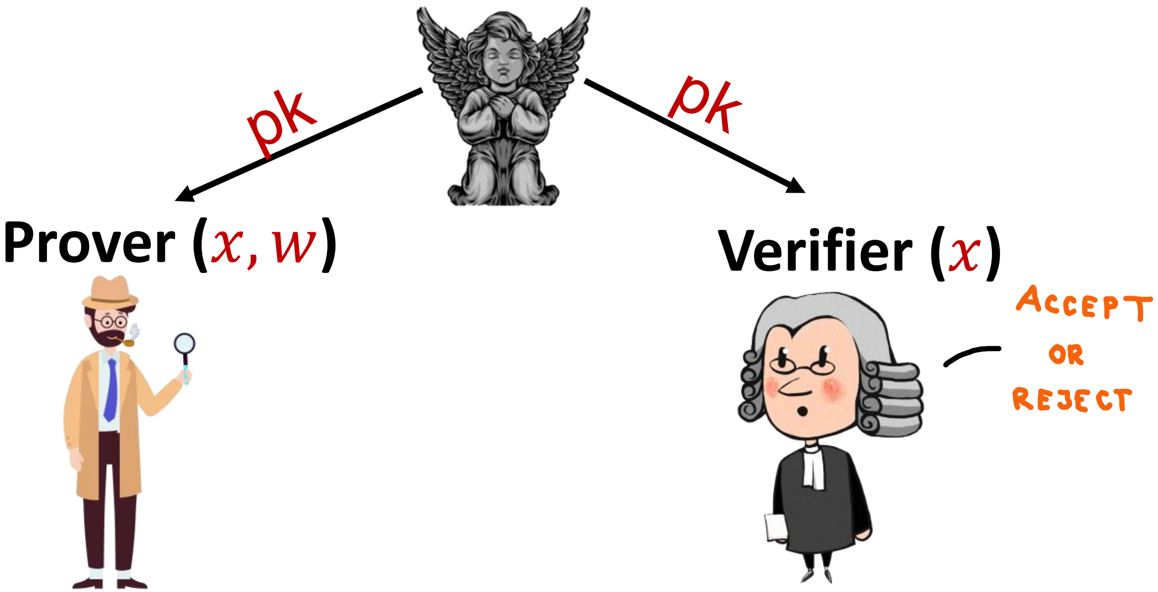
- Succinct Non-interactive Argument of Knowledge
  - **<u>Succinct:</u>** argument size is sublinear in $w$

# SNARK

- Succinct Non-interactive Argument of Knowledge
  - **Succinct:** argument size is sublinear in $w$
  - **Non-interactive:** single message from Prover to Verifier

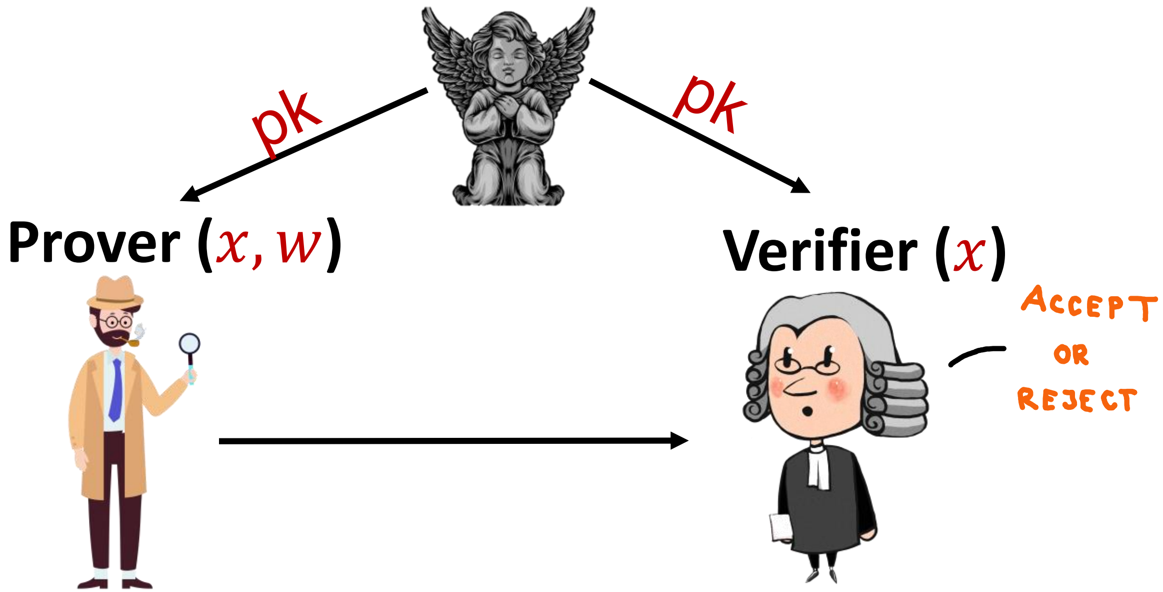
# SNARK

- Succinct Non-interactive Argument of Knowledge
  - **Succinct:** argument size is sublinear in $w$
  - **Non-interactive:** single message from Prover to Verifier



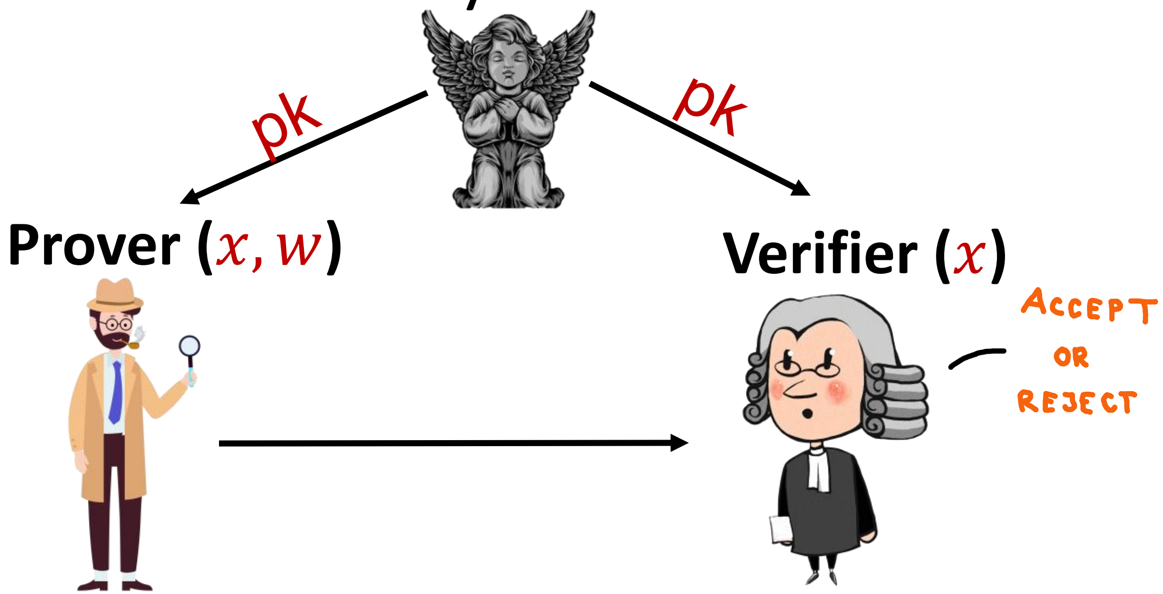
# SNARK

- Succinct Non-interactive Argument of Knowledge
  - **<u>Succinct:</u>** argument size is sublinear in $w$
  - **<u>Non-interactive:</u>** single message from Prover to Verifier

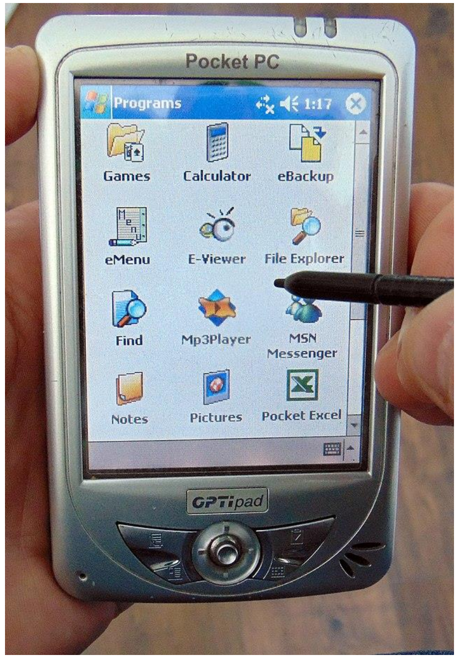


# SNARK

- Succinct Non-interactive Argument of Knowledge
  - **Succinct:** argument size is sublinear in $w$
  - **Non-interactive:** single message from Prover to Verifier
  - **Argument of Knowledge:** prover knows $w$ if verifier accepts (formally: $w$ can be efficiently extracted in the security proof)

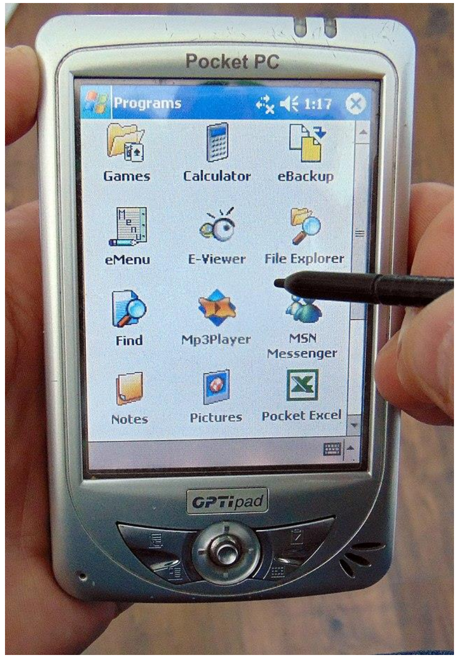


# Many Applications



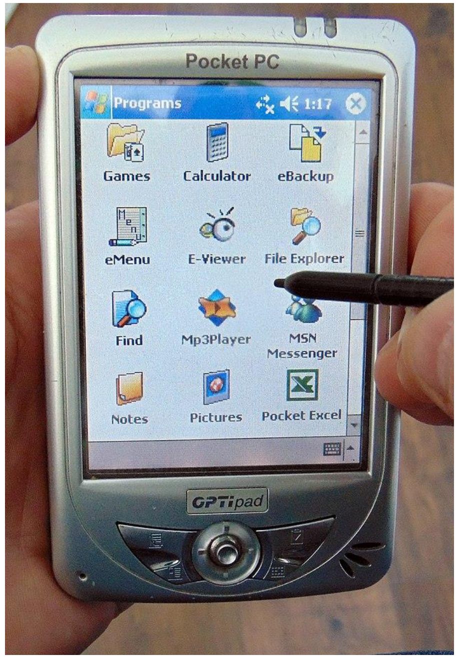
# Many Applications

- ❑Verifiable outsourced computation



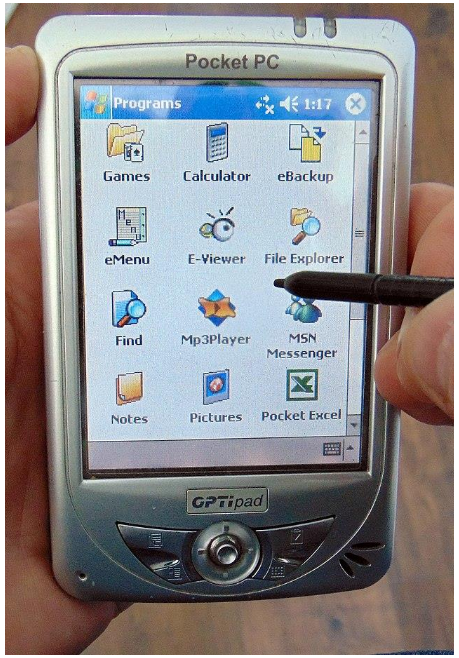
# Many Applications

- ❑Verifiable outsourced computation
- ❑Blockchain scalability (ZK rollups)



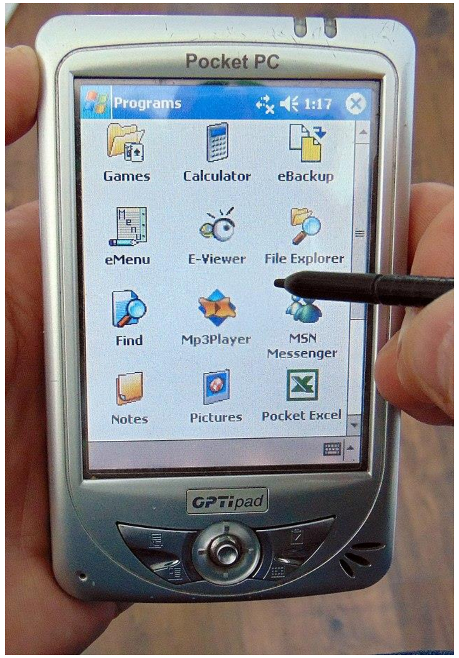
# Many Applications

- ❑Verifiable outsourced computation
- ❑Blockchain scalability (ZK rollups)
- ❑Verifiable electronic voting



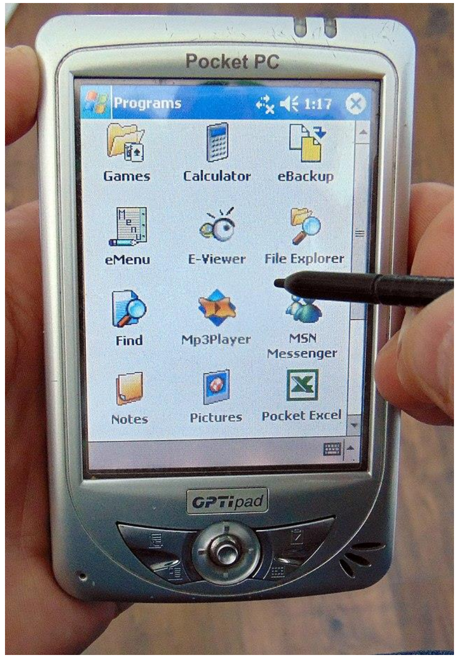
# Many Applications

- ❑Verifiable outsourced computation
- ❑Blockchain scalability (ZK rollups)
- ❑Verifiable electronic voting
- ❑Verifiable fully homomorphic encryption



# Many Applications

- ❑Verifiable outsourced computation
- ❑Blockchain scalability (ZK rollups)
- ❑Verifiable electronic voting
- ❑Verifiable fully homomorphic encryption
- ❑…



# Recipe for SNARKs

**<u>Ingredient 1</u>**: Polynomial Interactive Oracle Proof (PIOP)

**Prover ($x, w$)**

**Verifier ($x$)**

# Recipe for SNARKs

**<u>Ingredient 1</u>**: Polynomial Interactive Oracle Proof (PIOP)

$R$

**Prover** ($x, w$)

**Verifier** ($x$)

# Recipe for SNARKs

**<u>Ingredient 1</u>**: Polynomial Interactive Oracle Proof (PIOP)

$R$

oracles $i_1(X), \ldots, i_\ell(X)$

**Prover** ($x, w$)

**Verifier** ($x$)

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)

$R$

oracles $i_1(X), \dots, i_\ell(X)$

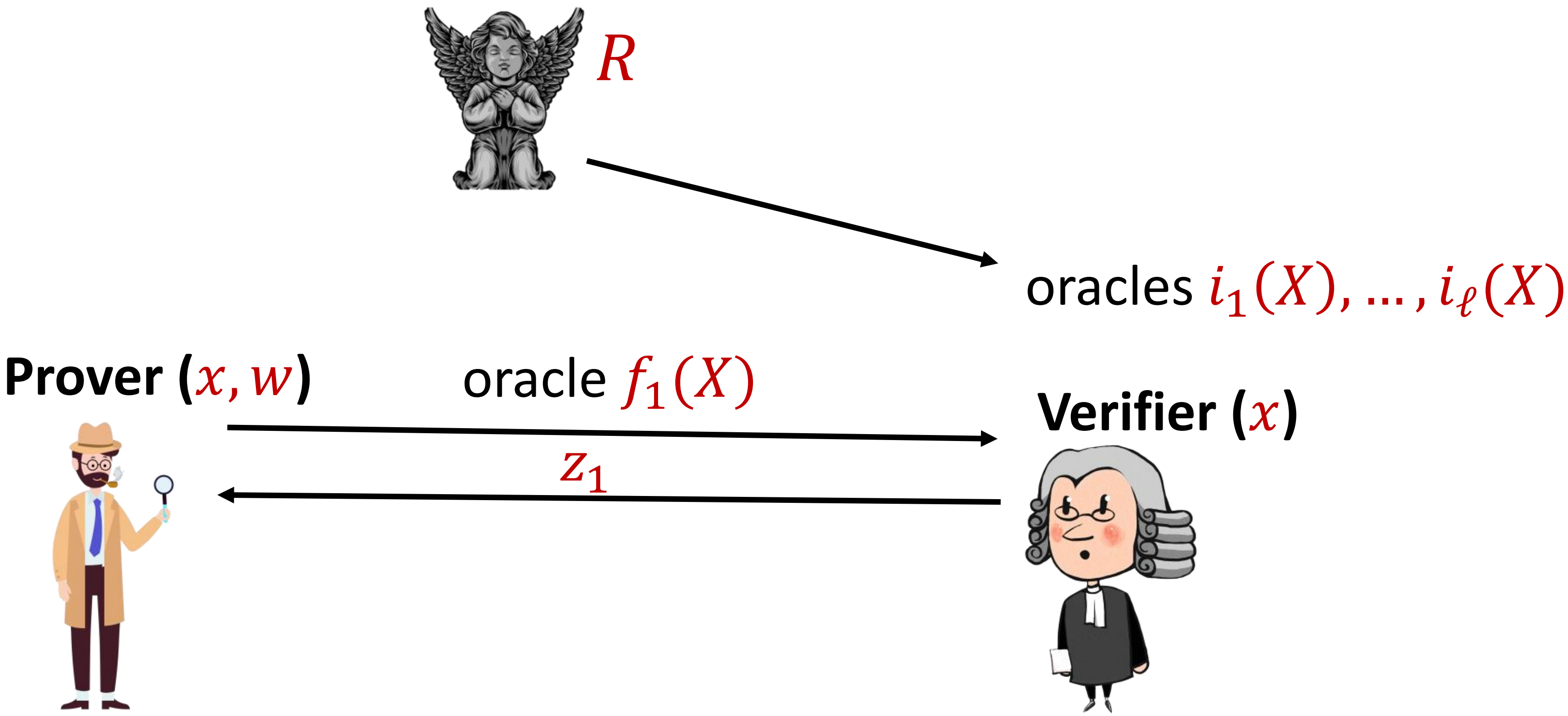
**Prover** ($x, w$)

oracle $f_1(X)$

**Verifier** ($x$)

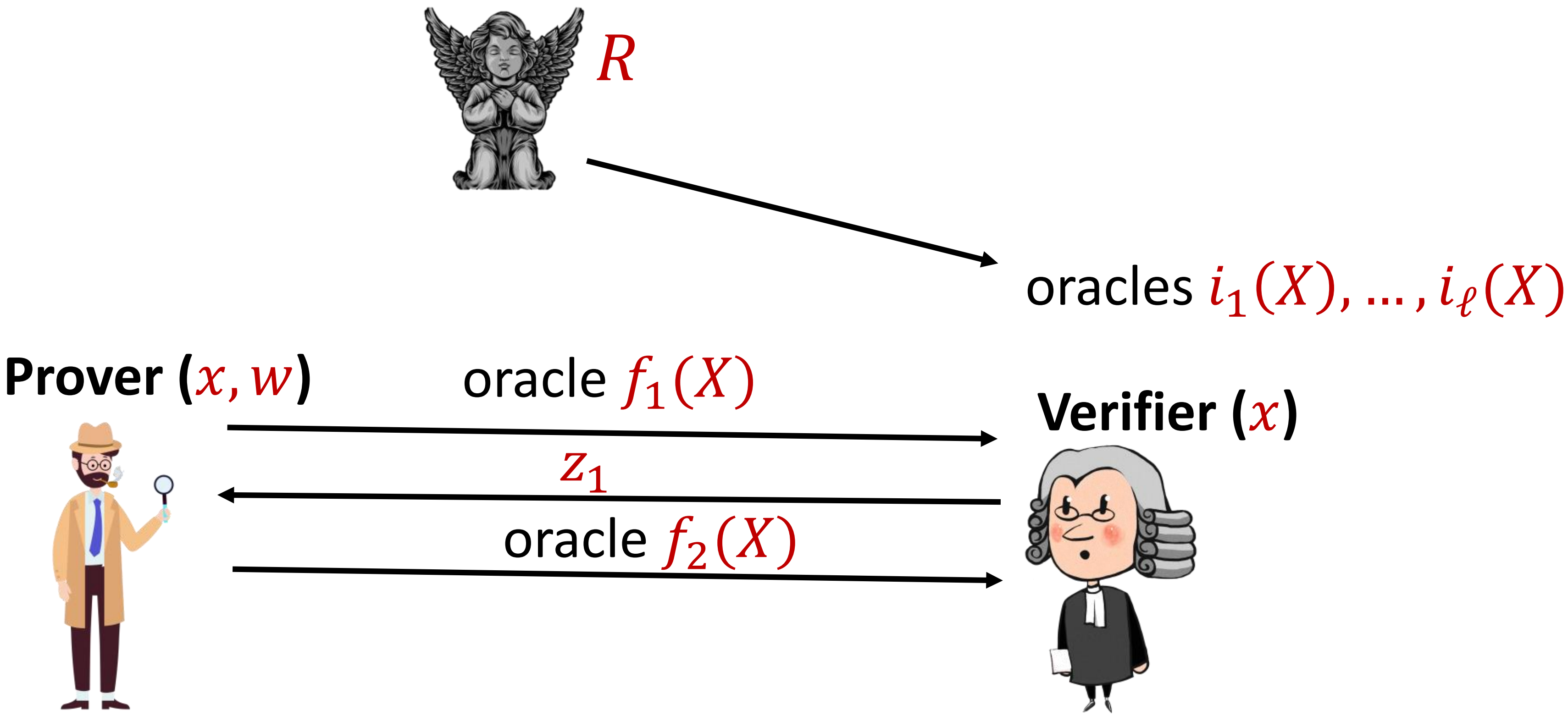
# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)



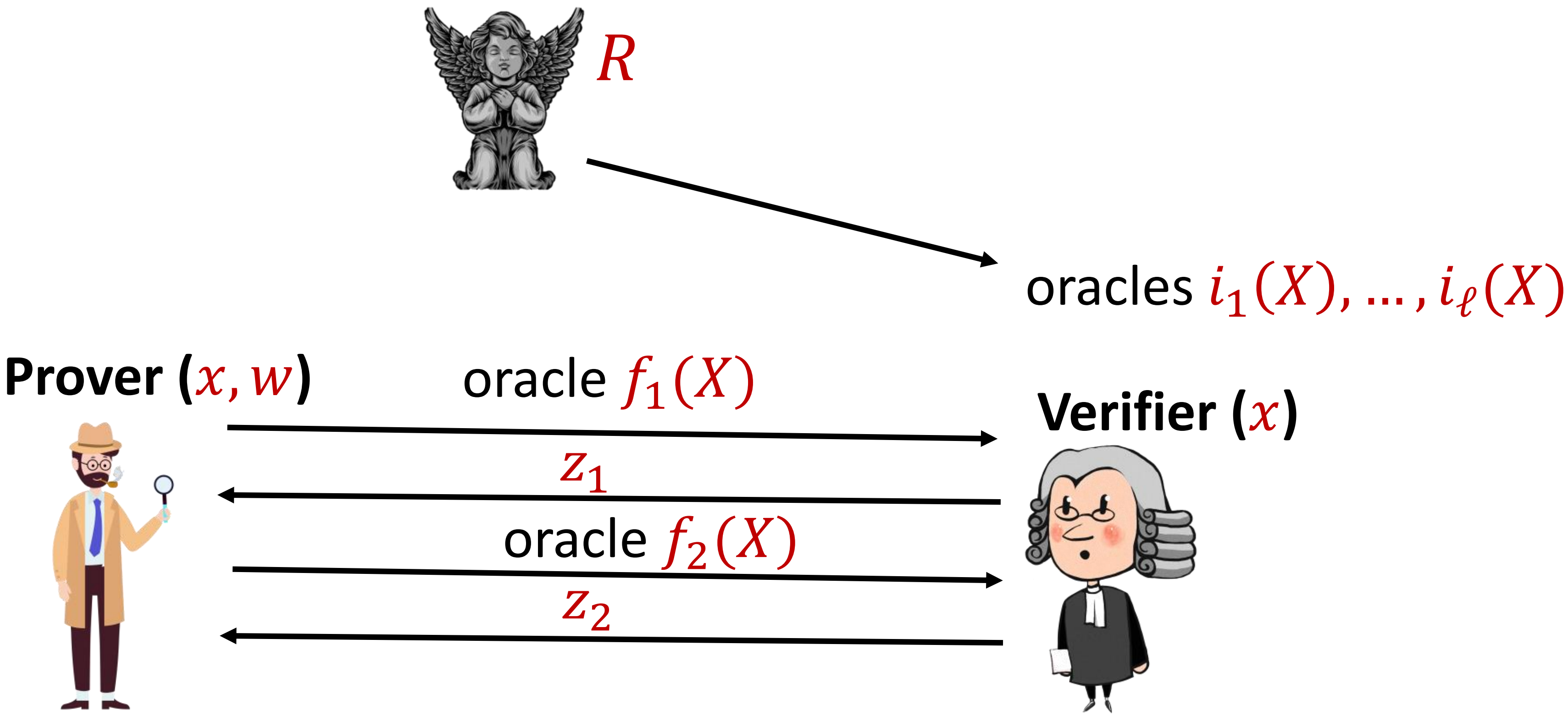
# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)



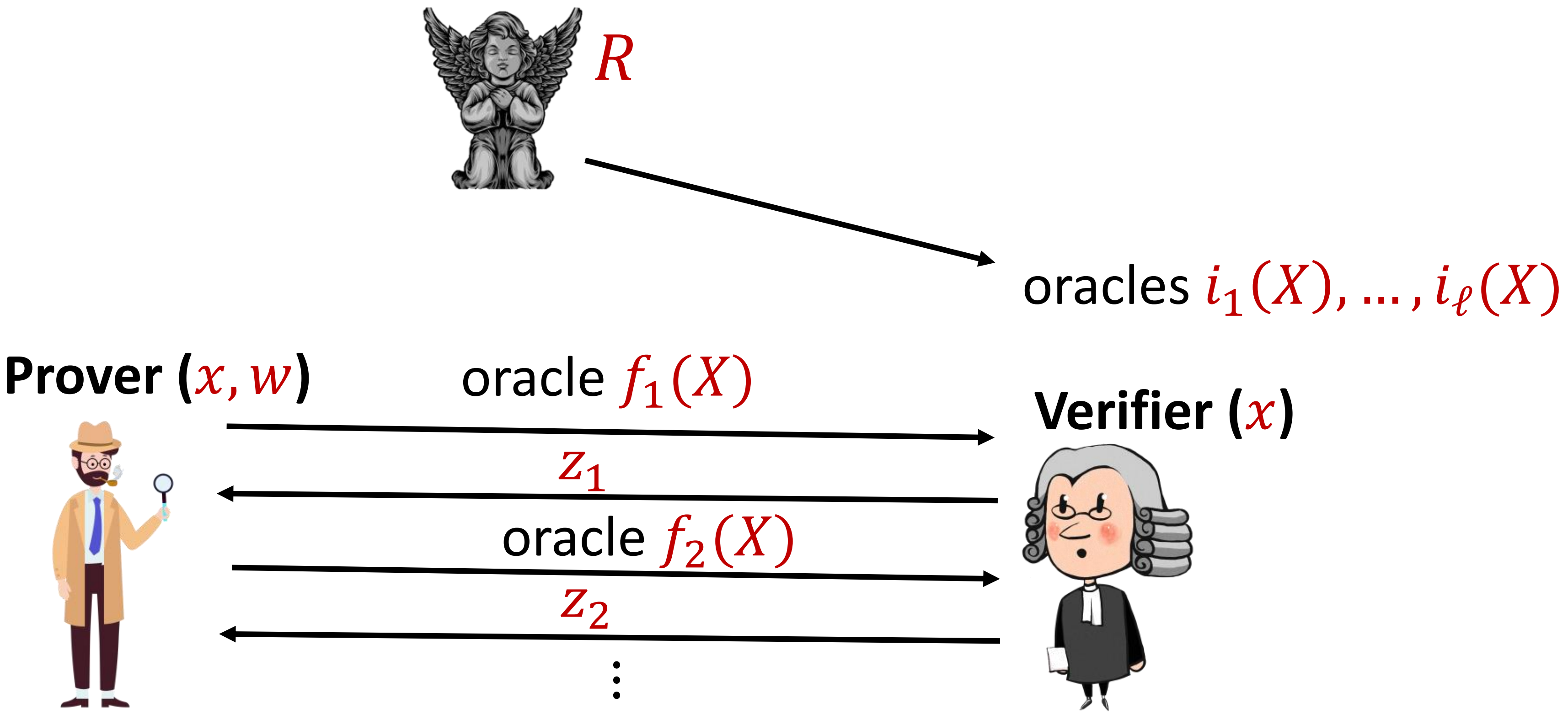
# Recipe for SNARKs

**<u>Ingredient 1</u>**: Polynomial Interactive Oracle Proof (PIOP)



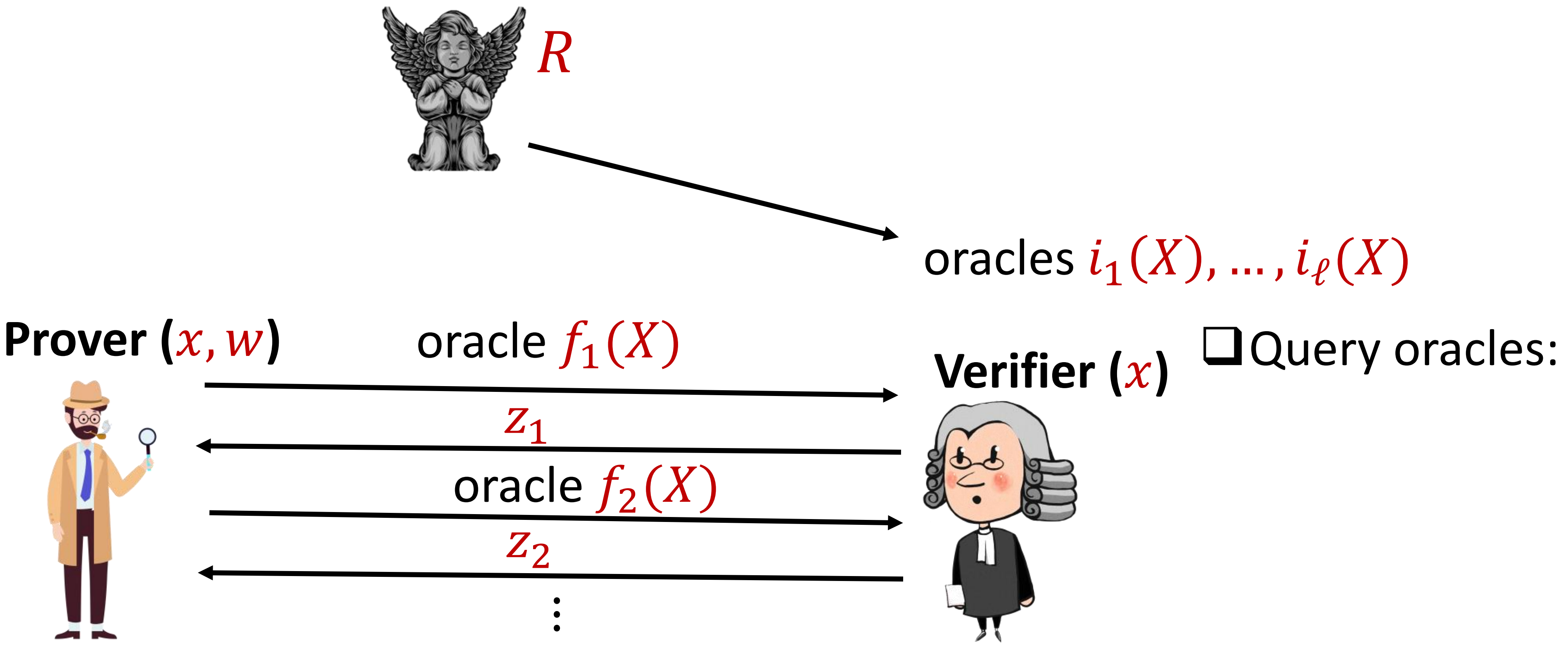
# Recipe for SNARKs

<u>**Ingredient 1**</u>: Polynomial Interactive Oracle Proof (PIOP)



# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)



# Recipe for SNARKs

**<u>Ingredient 1</u>**: Polynomial Interactive Oracle Proof (PIOP)

$R$

oracles $i_1(X), \dots, i_\ell(X)$

**Prover** ($x, w$)

oracle $f_1(X)$

$z_1$

oracle $f_2(X)$

$z_2$

⋮

**Verifier** ($x$)

- ❑ Query oracles:
  - $i_1(\alpha_1), \dots i_\ell(\alpha_\ell)$

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)

$R$

oracles $i_1(X), \ldots, i_\ell(X)$

**Prover ($x, w$)**

oracle $f_1(X)$

$z_1$

oracle $f_2(X)$

$z_2$

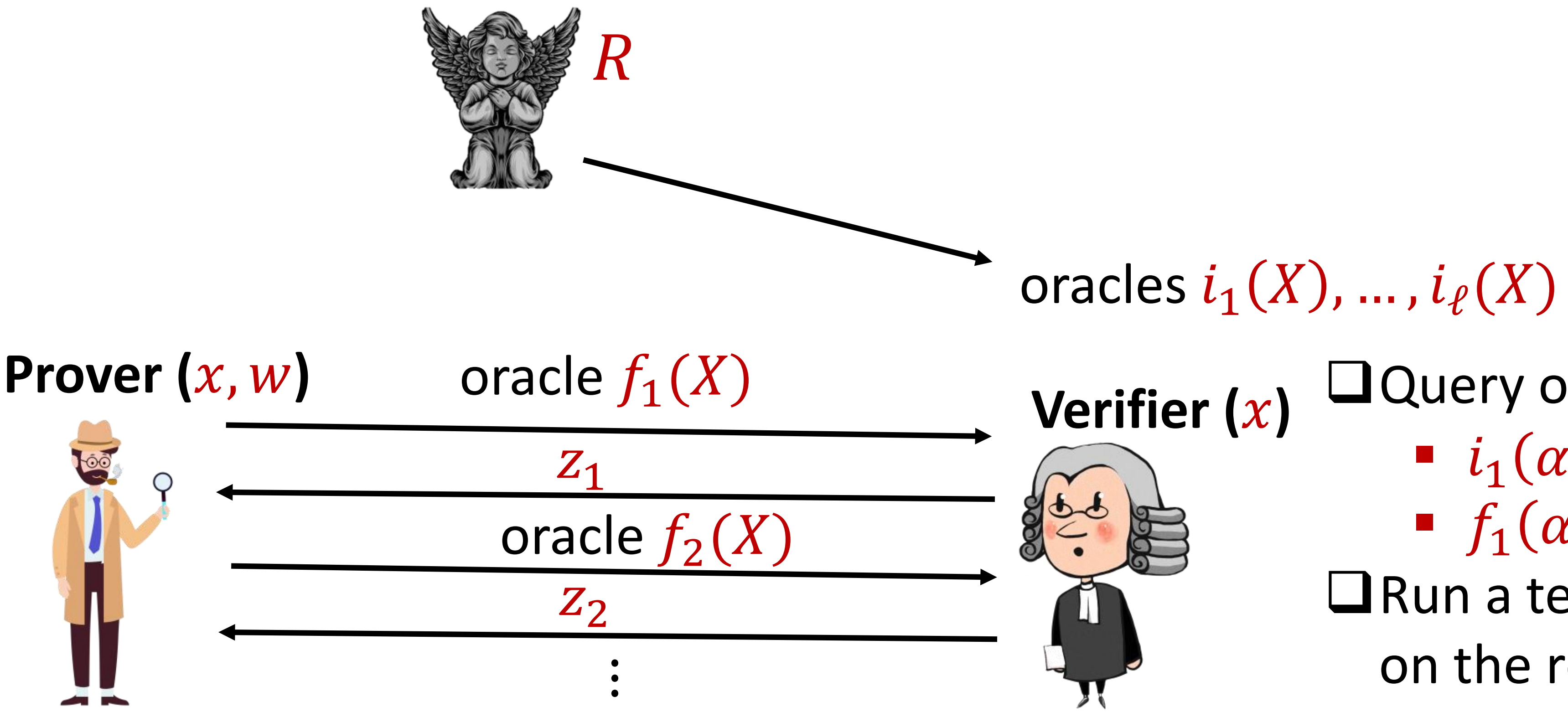
$\vdots$

**Verifier ($x$)**

- Query oracles:
  - $i_1(\alpha_1), \ldots i_\ell(\alpha_\ell)$
  - $f_1(\alpha_{\ell+1}), f_2(\alpha_{\ell+2}), \ldots$

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)




- Query oracles:
  - $i_1(\alpha_1), \ldots i_\ell(\alpha_\ell)$
  - $f_1(\alpha_{\ell+1}), f_2(\alpha_{\ell+2}), \ldots$
- Run a testing algorithm on the responses

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)

- ❑PIOP is knowledge-sound

# Recipe for SNARKs

**<u>Ingredient 1</u>**: Polynomial Interactive Oracle Proof (PIOP)

❑ PIOP is knowledge-sound

- $w$ is encoded in the polynomials

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)

- ❑ PIOP is knowledge-sound
  - $w$ is encoded in the polynomials
- ❑ Known how to construct constant-round PIOPs

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)

- PIOP is knowledge-sound
  - $w$ is encoded in the polynomials
- Known how to construct constant-round PIOPs
  - Sonic, Marlin, Plonk, …

# Recipe for SNARKs

<u>**Ingredient 1**</u>: Polynomial Interactive Oracle Proof (PIOP)

- ❑ PIOP is knowledge-sound
  - $w$ is encoded in the polynomials
- ❑ Known how to construct constant-round PIOPs
  - Sonic, Marlin, Plonk, …
- ❑ PIOP is not succinct

# Recipe for SNARKs

**Ingredient 1**: Polynomial Interactive Oracle Proof (PIOP)

- PIOP is knowledge-sound
  - $w$ is encoded in the polynomials
- Known how to construct constant-round PIOPs
  - Sonic, Marlin, Plonk, …
- PIOP is not succinct
  - Polynomials have high degree

# Recipe for SNARKs

**Ingredient 2**: Polynomial Commitment Scheme

# Recipe for SNARKs

**<u>Ingredient 2</u>**: Polynomial Commitment Scheme

- ❑ $\mathrm{Kgen}(n)$: outputs commitment key $\mathrm{ck}$ for degree $\leq n$ polynomials

# Recipe for SNARKs

**Ingredient 2**: Polynomial Commitment Scheme

- ❑ $\mathrm{Kgen}(n)$: outputs commitment key $\mathrm{ck}$ for degree $\leq n$ polynomials
- ❑ $\mathrm{Com}(\mathrm{ck}, f(\mathrm{X}))$: outputs commitment $\mathrm{C}$ for polynomial $f$ of $\deg(f) \leq n$

# Recipe for SNARKs

**Ingredient 2**: Polynomial Commitment Scheme

- ❑ $\mathrm{Kgen}(n)$: outputs commitment key $\mathrm{ck}$ for degree $\leq n$ polynomials
- ❑ $\mathrm{Com}(\mathrm{ck}, f(\mathrm{X}))$: outputs commitment $\mathrm{C}$ for polynomial $f$ of $\deg(f) \leq n$
- ❑ $\mathrm{Open}\left(\mathrm{ck}, \mathrm{C}, \alpha, f(\mathrm{X})\right)$: outputs $\eta = f(\alpha)$ and a proof $\pi$

# Recipe for SNARKs

**Ingredient 2**: Polynomial Commitment Scheme

- $\mathrm{Kgen}(n)$: outputs commitment key $\mathrm{ck}$ for degree $\leq n$ polynomials
- $\mathrm{Com}(\mathrm{ck}, f(\mathrm{X}))$: outputs commitment $\mathrm{C}$ for polynomial $f$ of $\deg(f) \leq n$
- $\mathrm{Open}\left(\mathrm{ck}, \mathrm{C}, \alpha, f(\mathrm{X})\right)$: outputs $\eta = f(\alpha)$ and a proof $\pi$
- $\mathrm{Verify}\left(\mathrm{ck}, \mathrm{C}, \alpha, \eta, \pi\right)$: accepts or rejects

# Recipe for SNARKs

**Ingredient 2**: Polynomial Commitment Scheme

- ❑ $\mathrm{Kgen}(n)$: outputs commitment key $\mathrm{ck}$ for degree $\leq n$ polynomials
- ❑ $\mathrm{Com}(\mathrm{ck}, f(\mathrm{X}))$: outputs commitment $\mathrm{C}$ for polynomial $f$ of $\deg(f) \leq n$
- ❑ $\mathrm{Open}\left(\mathrm{ck}, \mathrm{C}, \alpha, f(\mathrm{X})\right)$: outputs $\eta = f(\alpha)$ and a proof $\pi$
- ❑ $\mathrm{Verify}\left(\mathrm{ck}, \mathrm{C}, \alpha, \eta, \pi\right)$: accepts or rejects

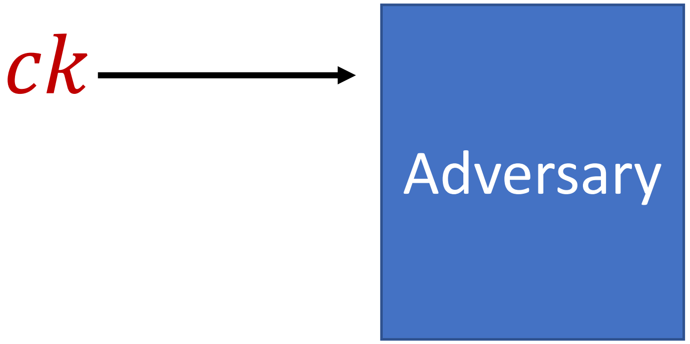
**Succinctness:** Size of $\mathrm{C}$ and $\pi$ is sublinear in $n$

# Non-black-box Extractability

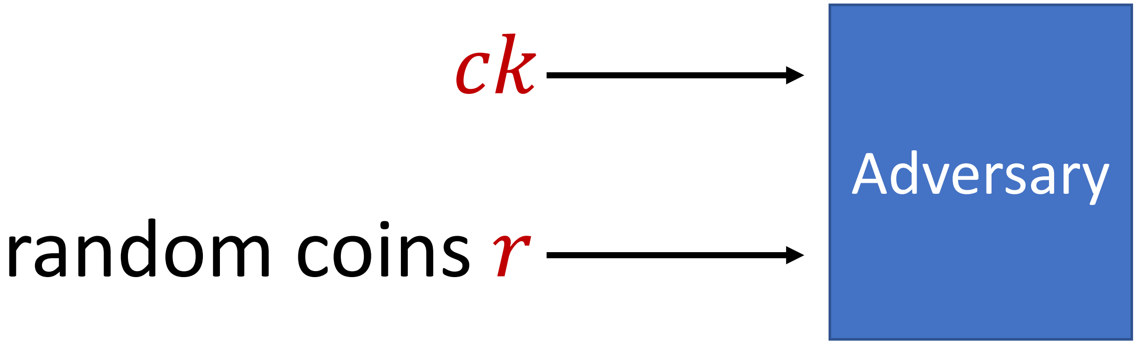
# Non-black-box Extractability

Adversary

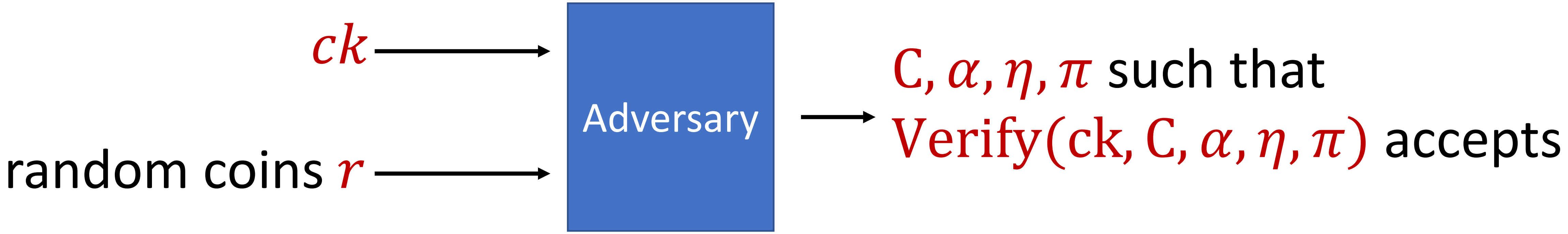
# Non-black-box Extractability



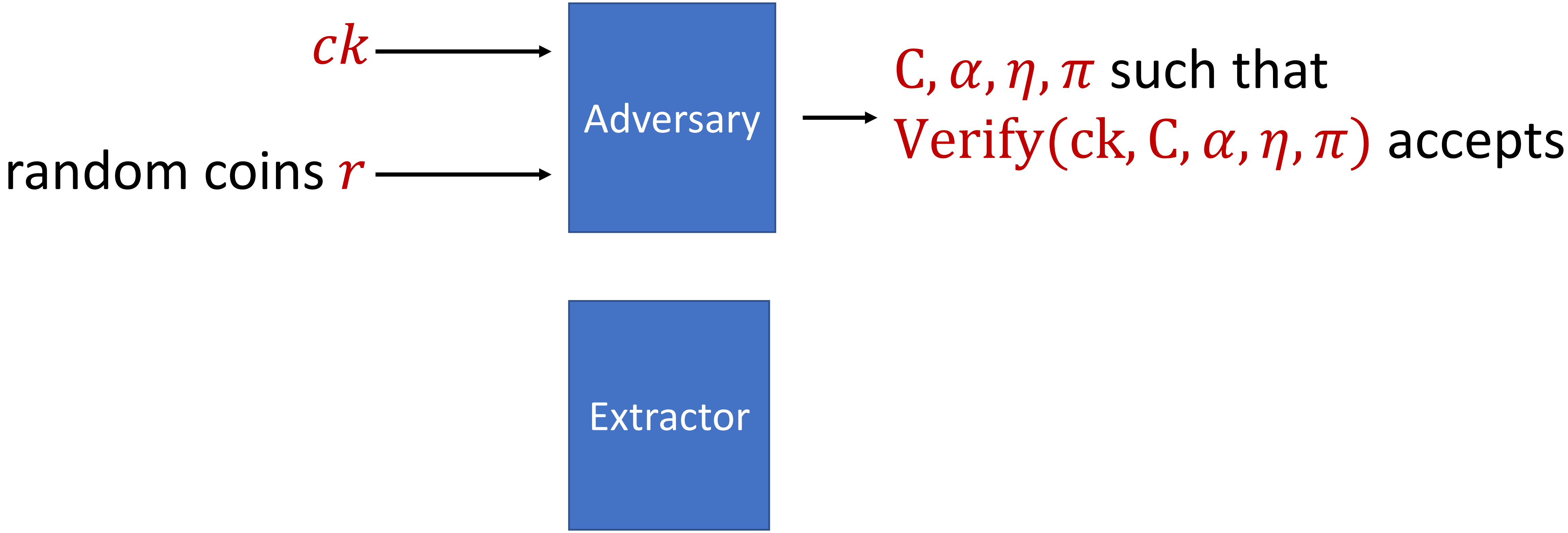
# Non-black-box Extractability



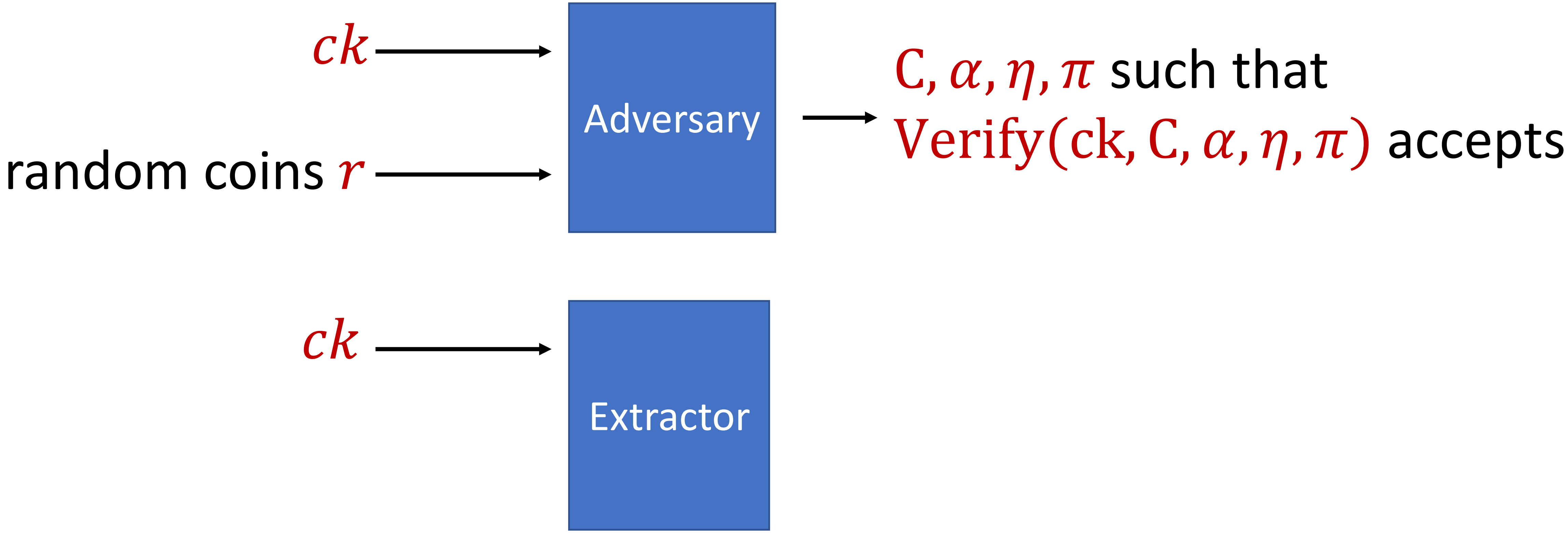
# Non-black-box Extractability



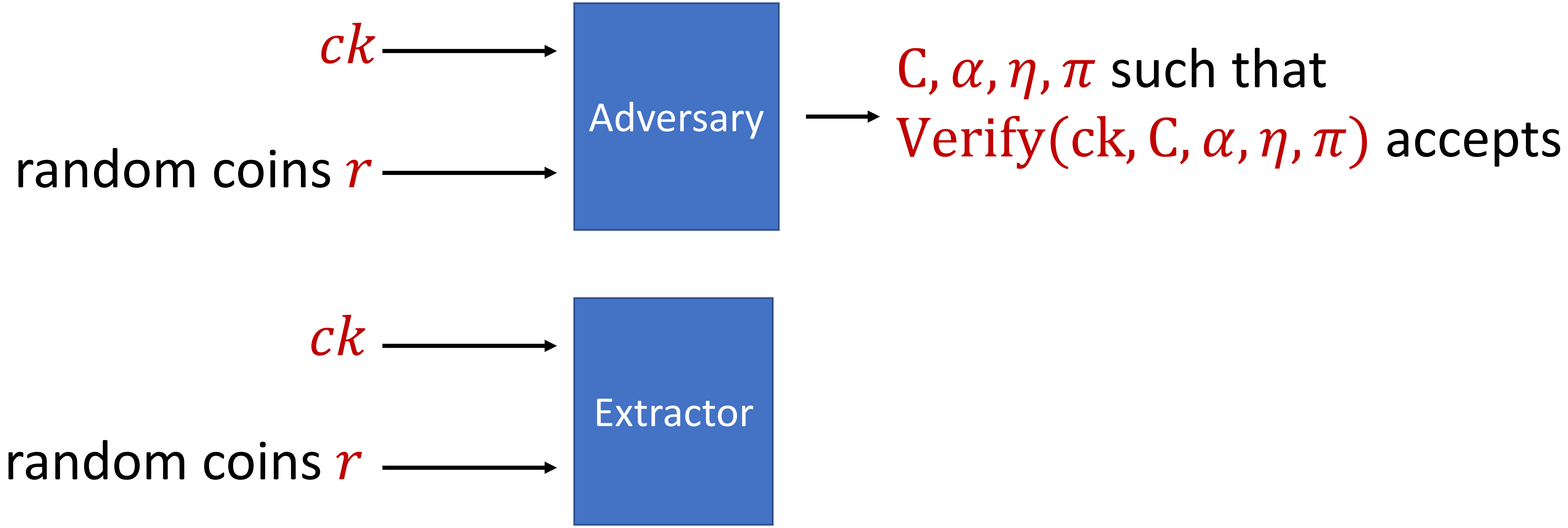
# Non-black-box Extractability



# Non-black-box Extractability



# Non-black-box Extractability



# Non-black-box Extractability

$ck \longrightarrow$ Adversary

random coins $r \longrightarrow$ Adversary

Adversary $\longrightarrow$ $\mathrm{C}, \alpha, \eta, \pi$ such that $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha, \eta, \pi)$ accepts

$ck \longrightarrow$ Extractor

random coins $r \longrightarrow$ Extractor

Extractor $\longrightarrow$ $f(X)$ such that $\mathrm{Com}\big(\mathrm{ck}, f(\mathrm{X})\big) = \mathrm{C}$ and $\deg(f) \leq n$

# Mix PIOP and Poly-Com

$R$

**Prover** $(x, w)$

**Verifier** $(x)$

# Mix PIOP and Poly-Com

$R$

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \dots, \mathrm{Com}\big(\mathrm{ck}, i_\ell(X)\big)$

**Prover** $(x, w)$

**Verifier** $(x)$

# Mix PIOP and Poly-Com

$R$

ck

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \ldots, \mathrm{Com}\big(\mathrm{ck}, i_\ell(X)\big)$

**Prover** $(x, w)$

**Verifier** $(x)$

# Mix PIOP and Poly-Com


$R$

$\mathrm{ck}$

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \ldots, \mathrm{Com}(\mathrm{ck}, i_\ell(X))$

**Prover** $(x, w)$

$\mathrm{C}_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$

**Verifier** $(x)$

# Mix PIOP and Poly-Com


$R$

$\mathrm{ck}$

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \dots, \mathrm{Com}(\mathrm{ck}, i_\ell(X))$

**Prover** $(x, w)$

$\mathrm{C}_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$

$z_1$

**Verifier** $(x)$

# Mix PIOP and Poly-Com

$R$

$\mathrm{ck}$

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \dots, \mathrm{Com}(\mathrm{ck}, i_\ell(X))$

**Prover** $(x, w)$

$\mathrm{C}_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$

$z_1$

$\mathrm{C}_2 = \mathrm{Com}(\mathrm{ck}, f_2(X))$

**Verifier** $(x)$

# Mix PIOP and Poly-Com

$R$

ck

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \ldots, \mathrm{Com}(\mathrm{ck}, i_\ell(X))$

**Prover** $(x, w)$

$C_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$

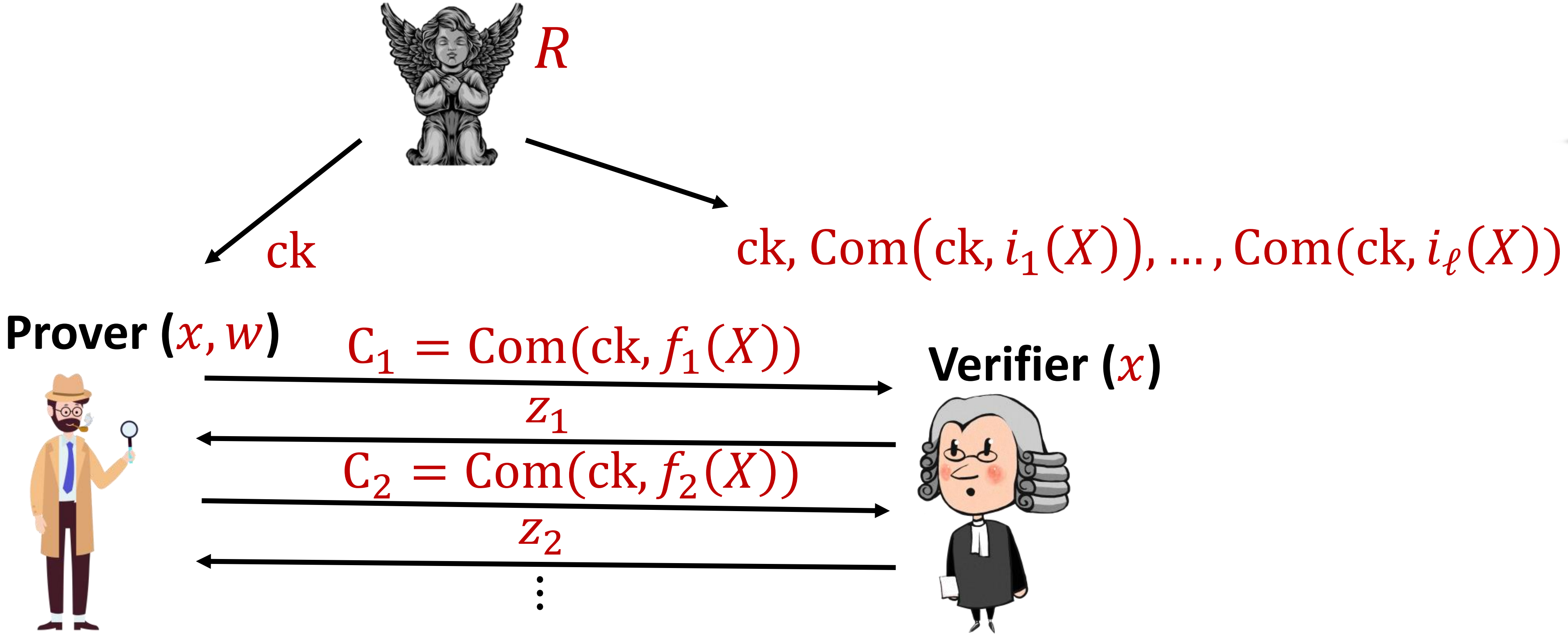
$z_1$

$C_2 = \mathrm{Com}(\mathrm{ck}, f_2(X))$

$z_2$

**Verifier** $(x)$

# Mix PIOP and Poly-Com



# Mix PIOP and Poly-Com

$R$

ck

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \ldots, \mathrm{Com}\big(\mathrm{ck}, i_\ell(X)\big)$

**Prover** $(x, w)$

$\mathrm{C}_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$

$z_1$

$\mathrm{C}_2 = \mathrm{Com}(\mathrm{ck}, f_2(X))$

$z_2$

$\vdots$

$\alpha_1$

**Verifier** $(x)$

# Mix PIOP and Poly-Com

$R$

ck

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \ldots, \mathrm{Com}\big(\mathrm{ck}, i_\ell(X)\big)$

**Prover** $(x, w)$

$\mathrm{C}_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$

$z_1$

$\mathrm{C}_2 = \mathrm{Com}(\mathrm{ck}, f_2(X))$

$z_2$

$\vdots$

$\alpha_1$

$\eta_1 = i_1(\alpha_i), \pi_1$

**Verifier** $(x)$

# Mix PIOP and Poly-Com


$R$

ck

ck, $\mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \ldots, \mathrm{Com}(\mathrm{ck}, i_\ell(X))$

**Prover** $(x, w)$

**Verifier** $(x)$

$C_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$ →

$z_1$ ←

$C_2 = \mathrm{Com}(\mathrm{ck}, f_2(X))$ →

$z_2$ ←

$\vdots$

$\alpha_1$ ←

$\eta_1 = i_1(\alpha_i), \pi_1$ →

$\vdots$

# Mix PIOP and Poly-Com

$R$

ck

ck, $\mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \dots, \mathrm{Com}(\mathrm{ck}, i_\ell(X))$

**Prover** $(x, w)$

$C_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$ →

← $z_1$

$C_2 = \mathrm{Com}(\mathrm{ck}, f_2(X))$ →

← $z_2$

⋮

← $\alpha_1$

$\eta_1 = i_1(\alpha_i), \pi_1$ →

⋮

**Verifier** $(x)$

❑ Verify opening proofs

# Mix PIOP and Poly-Com

$R$

$\mathrm{ck}$

$\mathrm{ck}, \mathrm{Com}\big(\mathrm{ck}, i_1(X)\big), \dots, \mathrm{Com}\big(\mathrm{ck}, i_\ell(X)\big)$

**Prover** $(x, w)$

$C_1 = \mathrm{Com}(\mathrm{ck}, f_1(X))$

$z_1$

$C_2 = \mathrm{Com}(\mathrm{ck}, f_2(X))$

$z_2$

⋮

$\alpha_1$

$\eta_1 = i_1(\alpha_i), \pi_1$

⋮

**Verifier** $(x)$

- ❑ Verify opening proofs
- ❑ Run PIOP's testing algorithm

# Recipe for SNARKs

- We now have an interactive argument of knowledge

# Recipe for SNARKs

- We now have an interactive argument of knowledge
  - Non-black-box extractability to extract polynomials from commitments

# Recipe for SNARKs

- We now have an interactive argument of knowledge
  - Non-black-box extractability to extract polynomials from commitments
  - Polynomials reveal $w$

# Recipe for SNARKs

- We now have an interactive argument of knowledge
  - Non-black-box extractability to extract polynomials from commitments
  - Polynomials reveal $w$
- Succinctness:

# Recipe for SNARKs

- We now have an interactive argument of knowledge
  - Non-black-box extractability to extract polynomials from commitments
  - Polynomials reveal $w$
- Succinctness:
  - If PIOP has a small number of rounds

# Recipe for SNARKs

- We now have an interactive argument of knowledge
  - Non-black-box extractability to extract polynomials from commitments
  - Polynomials reveal $w$
- Succinctness:
  - If PIOP has a small number of rounds
  - Polynomial commitment is succinct

# Recipe for SNARKs

- ❑ We now have an interactive argument of knowledge
  - Non-black-box extractability to extract polynomials from commitments
  - Polynomials reveal $w$
- ❑ Succinctness:
  - If PIOP has a small number of rounds
  - Polynomial commitment is succinct

**Ingredient 3**: Fiat-Shamir for non-interactivity

# Where to get ingredients?

# Where to get ingredients?

- ❑ Efficient PIOPs exist

# Where to get ingredients?

- ❑ Efficient PIOPs exist
- ❑ What about poly-com?

# Where to get ingredients?

- ❑Efficient PIOPs exist
- ❑What about poly-com?
- ❑There are many

# Where to get ingredients?

- ❑Efficient PIOPs exist
- ❑What about poly-com?
- ❑There are many
- ❑Popular option: KZG commitment

# Where to get ingredients?

- Efficient PIOPs exist
- What about poly-com?
- There are many
- Popular option: KZG commitment
- Proposed By **K**ate, **Z**averucha, and **G**oldberg in [Asiacrypt 2010]

# Where to get ingredients?


- Efficient PIOPs exist
- What about poly-com?
- There are many
- Popular option: KZG commitment
- Proposed By **K**ate, **Z**averucha, and **G**oldberg in [Asiacrypt 2010]
  - good efficiency

# Where to get ingredients?


- ❑Efficient PIOPs exist
- ❑What about poly-com?
- ❑There are many
- ❑Popular option: KZG commitment
- ❑Proposed By **K**ate, **Z**averucha, and **G**oldberg in [Asiacrypt 2010]
  - good efficiency
  - constant size $\mathrm{C}$ and $\pi$

# KZG Polynomial Commitment

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions
  - Generic group model/algebraic group model, or

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions
  - Generic group model/algebraic group model, or
  - Knowledge assumption (almost tautological)

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions
  - Generic group model/algebraic group model, or
  - Knowledge assumption (almost tautological)
- **Falsifiable assumption:** interaction between efficient adversary and efficient challenger

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions
  - Generic group model/algebraic group model, or
  - Knowledge assumption (almost tautological)
- **Falsifiable assumption:** interaction between efficient adversary and efficient challenger
  - above assumptions are non-falsifiable

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions
  - Generic group model/algebraic group model, or
  - Knowledge assumption (almost tautological)
- **Falsifiable assumption:** interaction between efficient adversary and efficient challenger
  - above assumptions are non-falsifiable
  - **Folklore:** no way to avoid in KZG extractability

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions
  - Generic group model/algebraic group model, or
  - Knowledge assumption (almost tautological)
- **Falsifiable assumption:** interaction between efficient adversary and efficient challenger
  - above assumptions are non-falsifiable
  - <u>**Folklore:**</u> no way to avoid in KZG extractability
- **Not true!**

# KZG Polynomial Commitment

- Non-black-box extractable under strong assumptions
  - Generic group model/algebraic group model, or
  - Knowledge assumption (almost tautological)
- **Falsifiable assumption:** interaction between efficient adversary and efficient challenger
  - above assumptions are non-falsifiable
  - **<u>Folklore:</u>** no way to avoid in KZG extractability
- **Not true!**
  - We show extractability with <u>rewinding</u> under a relatively standard <u>falsifiable assumption</u>

# Pairings

# Pairings

- Bilinear groups: $\mathbb{G}_1$, $\mathbb{G}_2$,$\mathbb{G}_T$ of prime size $p$ with generators $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_T$

# Pairings

- Bilinear groups: $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime size $p$ with generators $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_T$
- Additive notation & bracket notation:

# Pairings

- ❑Bilinear groups: $\mathbb{G}_1$, $\mathbb{G}_2$,$\mathbb{G}_T$ of prime size $p$ with generators $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_T$
- ❑Additive notation & bracket notation:
  - $a \cdot \mathcal{P}_1 \coloneqq [a]_1$ for $a \in \mathbb{Z}_p$

# Pairings

- Bilinear groups: $\mathbb{G}_1$, $\mathbb{G}_2$,$\mathbb{G}_T$ of prime size $p$ with generators $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_T$
- Additive notation & bracket notation:
  - $a \cdot \mathcal{P}_1 \coloneqq [a]_1$ for $a \in \mathbb{Z}_p$
  - $a \cdot \mathcal{P}_2 \coloneqq [a]_2$ for $a \in \mathbb{Z}_p$

# Pairings

- Bilinear groups: $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime size $p$ with generators $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_T$
- Additive notation & bracket notation:
  - $a \cdot \mathcal{P}_1 \coloneqq [a]_1$ for $a \in \mathbb{Z}_p$
  - $a \cdot \mathcal{P}_2 \coloneqq [a]_2$ for $a \in \mathbb{Z}_p$
  - $a \cdot \mathcal{P}_T \coloneqq [a]_T$ for $a \in \mathbb{Z}_p$

# Pairings

- Bilinear groups: $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime size $p$ with generators $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_T$
- Additive notation & bracket notation:
  - $a \cdot \mathcal{P}_1 \coloneqq [a]_1$ for $a \in \mathbb{Z}_p$
  - $a \cdot \mathcal{P}_2 \coloneqq [a]_2$ for $a \in \mathbb{Z}_p$
  - $a \cdot \mathcal{P}_T \coloneqq [a]_T$ for $a \in \mathbb{Z}_p$
- Bilinear map: $[a]_1 \cdot [b]_2 = [ab]_T$.

# KZG Polynomial Commitment

- ❑ $\mathrm{KGen}(n)$:

- ❑ $\mathrm{Com}(\mathrm{ck}, f)$:

- ❑ $\mathrm{Open}(\mathrm{ck}, \mathrm{C}, \alpha, f)$:

- ❑ $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha, \eta, \pi)$:

# KZG Polynomial Commitment

- KGen($n$):

$$\sigma \leftarrow_r \mathbb{Z}_p, ck = ([\,1, \sigma, \sigma^2, \dots, \sigma^n]_1, [1, \sigma]_2)$$

- Com(ck, $f$):

- Open(ck, C, $\alpha, f$):

- Verify(ck, C, $\alpha, \eta, \pi$):

# KZG Polynomial Commitment

- $\mathrm{KGen}(n)$:

$$\sigma \leftarrow_r \mathbb{Z}_p,\ ck = ([\,1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2)$$

- $\mathrm{Com}(\mathrm{ck}, f)$:

$$C = [f(\sigma)]_1 = \sum_{i=0}^{n} f_i [\sigma^i]$$

- $\mathrm{Open}(\mathrm{ck}, \mathrm{C}, \alpha, f)$:

- $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha, \eta, \pi)$:

# KZG Polynomial Commitment

- KGen($n$):

$$\sigma \leftarrow_r \mathbb{Z}_p, ck = ([\,1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2)$$

- Com(ck, $f$):

$$C = [f(\sigma)]_1 = \sum_{i=0}^{n} f_i [\sigma^i]$$

- Open(ck, C, $\alpha$, $f$):

$$\eta = f(\alpha), h(X) = \frac{f(X) - \eta}{X - \alpha}, \; \pi = [h(\sigma)]_1,$$

- Verify(ck, C, $\alpha$, $\eta$, $\pi$):

# KZG Polynomial Commitment

- ❑ $\mathrm{KGen}(n)$:

$$\sigma \leftarrow_r \mathbb{Z}_p,\ ck = ([\,1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2)$$

- ❑ $\mathrm{Com}(\mathrm{ck}, f)$:

$$C = [f(\sigma)]_1 = \sum_{i=0}^{n} f_i [\sigma^i]$$

- ❑ $\mathrm{Open}(\mathrm{ck}, \mathrm{C}, \alpha, f)$:

$$\eta = f(\alpha), h(X) = \frac{f(X) - \eta}{X - \alpha},\ \pi = [h(\sigma)]_1,$$

- ❑ $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha, \eta, \pi)$:

$$([f(\sigma)]_1 - \eta[1]_1) \cdot [1]_2 = [h(\sigma)]_1 \cdot ([\sigma]_2 - \alpha[1]_2)$$

# Computational Special-Soundness

# Computational Special-Soundness

- New notion for polynomial commitments

# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems

# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems

# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems

Adversary

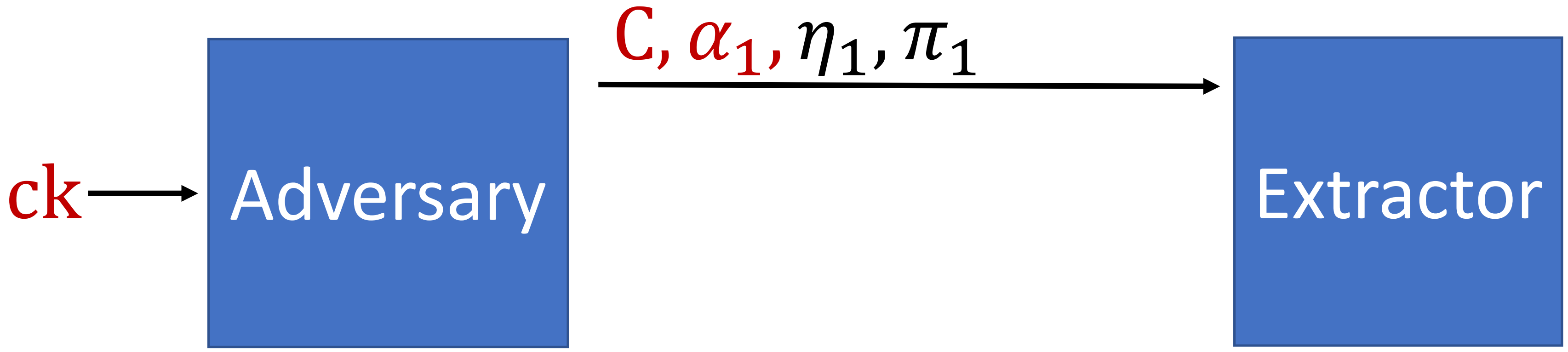
Extractor

# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems

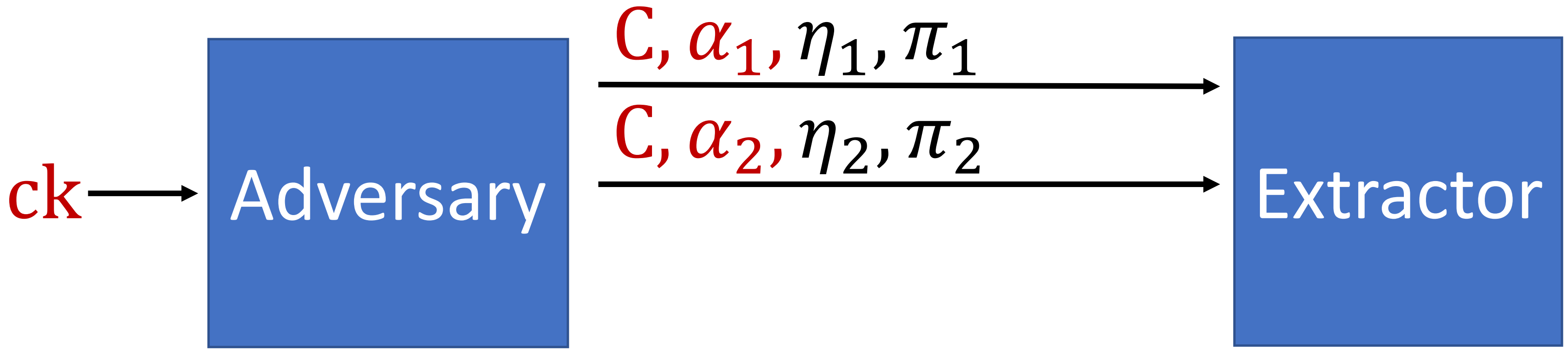
# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems



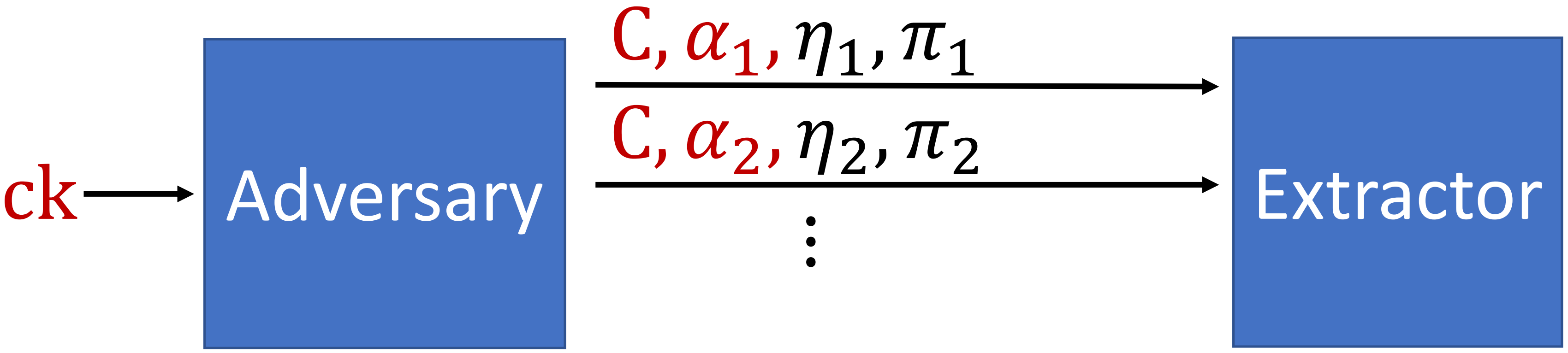
# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems



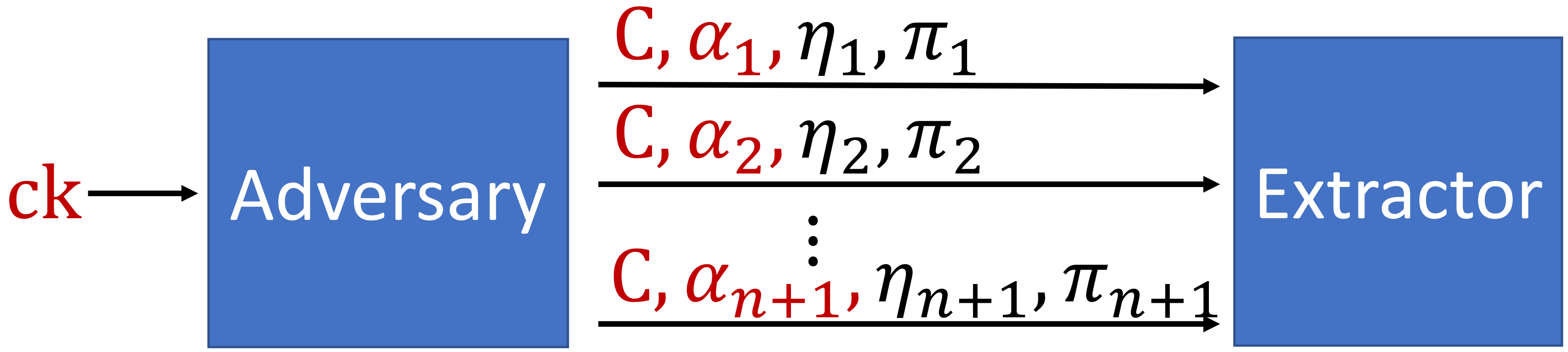
# Computational Special-Soundness

- ❑ New notion for polynomial commitments
  - well-known for proof systems



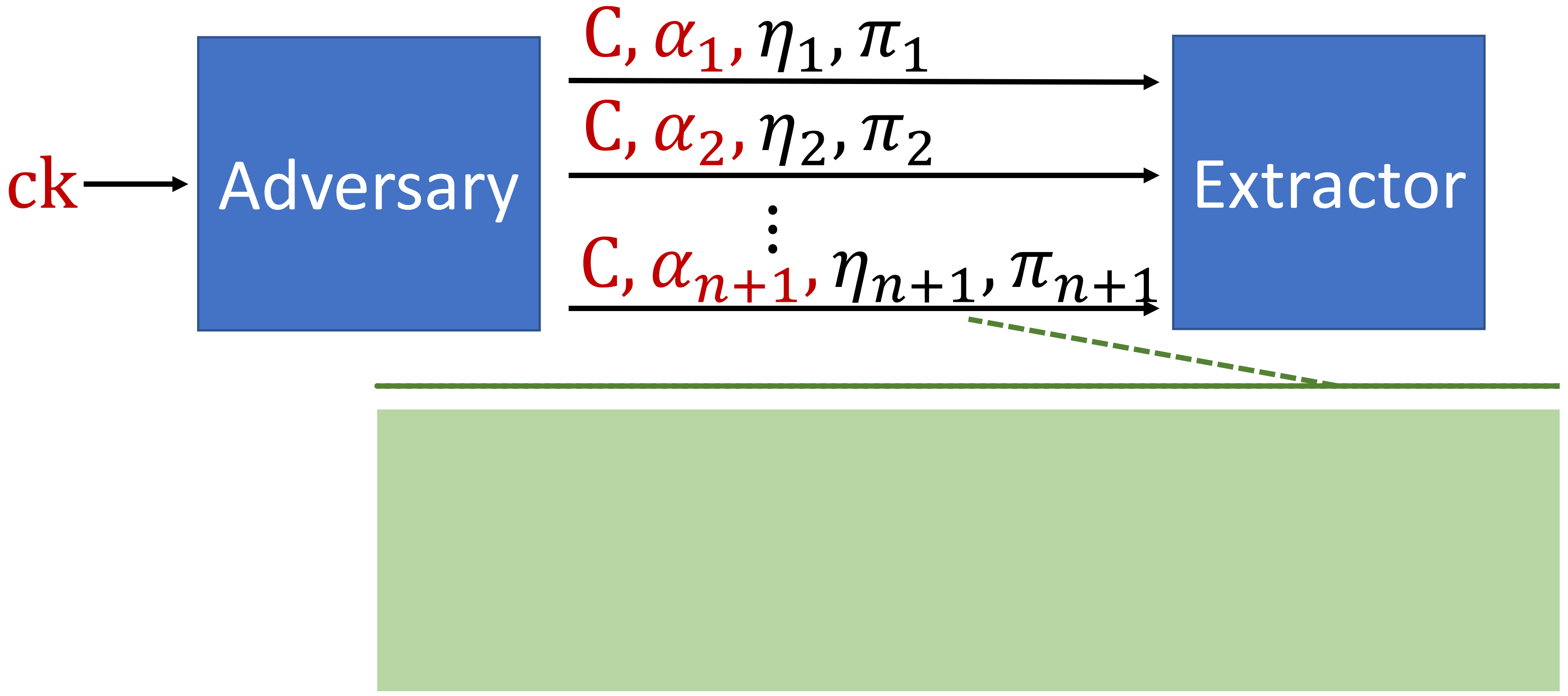
# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems



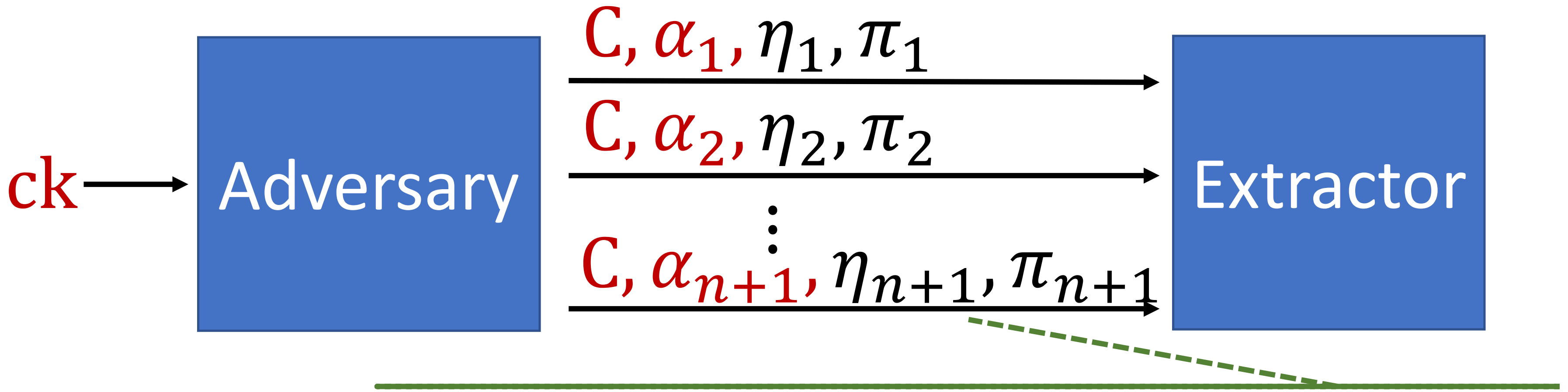
# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems



# Computational Special-Soundness

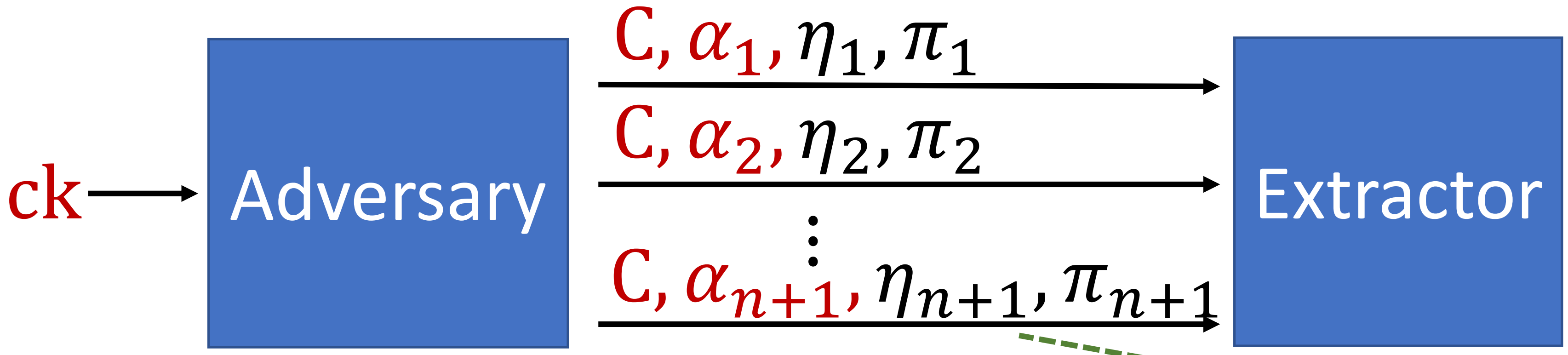
- New notion for polynomial commitments
  - well-known for proof systems




1. $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha_i, \eta_i, \pi_i)$ accepts $\forall i$

# Computational Special-Soundness

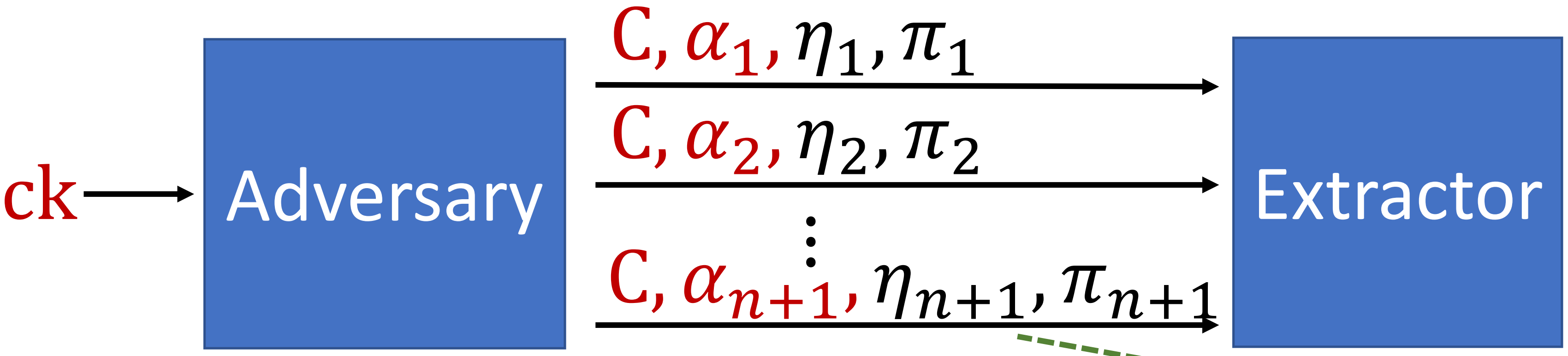
- New notion for polynomial commitments
  - well-known for proof systems




1. $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha_i, \eta_i, \pi_i)$ accepts $\forall i$
2. Same commitment $\mathrm{C}$

# Computational Special-Soundness

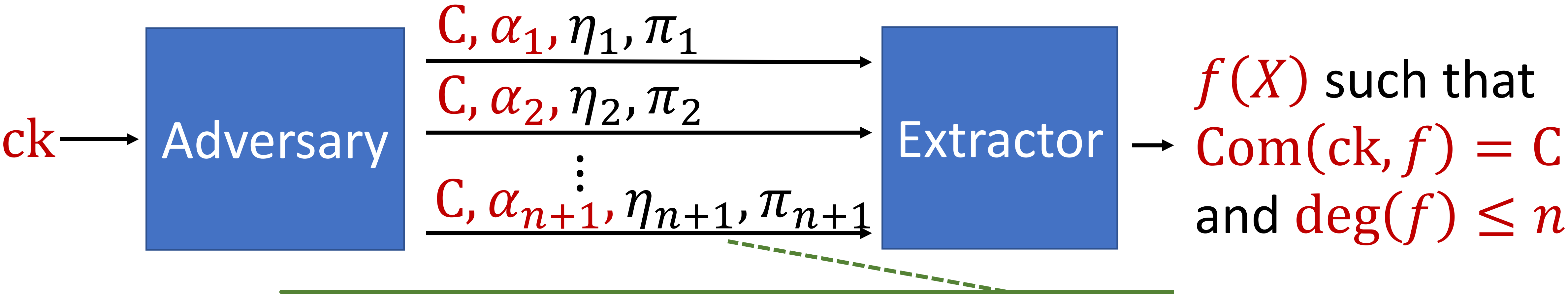
- New notion for polynomial commitments
  - well-known for proof systems




1. $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha_i, \eta_i, \pi_i)$ accepts $\forall i$
2. Same commitment $\mathrm{C}$
3. $\alpha_i$ are distinct

# Computational Special-Soundness

- New notion for polynomial commitments
  - well-known for proof systems




1. $\mathrm{Verify}(\mathrm{ck}, \mathrm{C}, \alpha_i, \eta_i, \pi_i)$ accepts $\forall i$
2. Same commitment $\mathrm{C}$
3. $\alpha_i$ are distinct

# KZG Special Soundness

# KZG Special Soundness

- Adversary provides: $\{(\alpha_0, \eta_0), \ldots, (\alpha_n, \eta_n)\}$

# KZG Special Soundness

- Adversary provides: $\{(\alpha_0, \eta_0), \dots, (\alpha_n, \eta_n)\}$
- **Basic math:** $n + 1$ poly evaluations define unique $f(X)$ of degree $\leq n$

# KZG Special Soundness

- Adversary provides: $\{(\alpha_0, \eta_0), \dots, (\alpha_n, \eta_n)\}$
- **Basic math:** $n + 1$ poly evaluations define unique $f(X)$ of degree $\leq n$

$$\mathrm{Ext}_{SS}([1, \sigma, \sigma^2, \dots, \sigma^n]_1, [1, \sigma]_2, [c]_1, \{\alpha_i, \eta_i, [\pi_i]_1\}_{i=0}^{n})$$

# KZG Special Soundness

- Adversary provides: $\{(\alpha_0, \eta_0), \ldots, (\alpha_n, \eta_n)\}$
- **Basic math:** $n + 1$ poly evaluations define unique $f(X)$ of degree $\leq n$

$\text{Ext}_{\text{SS}}([1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2, [c]_1, \{\alpha_i, \eta_i, [\pi_i]_1\}_{i=0}^{n})$

1. Interpolate $f(X)$;

# KZG Special Soundness

- Adversary provides: $\{(\alpha_0, \eta_0), \ldots, (\alpha_n, \eta_n)\}$
- <u>**Basic math:**</u> $n + 1$ poly evaluations define unique $f(X)$ of degree $\leq n$

$\mathrm{Ext}_{\mathrm{SS}}([1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2, [c]_1, \{\alpha_i, \eta_i, [\pi_i]_1\}_{i=0}^{n})$

1. Interpolate $f(X)$;
2. $[f(\sigma)]_1 =? [c]_1$;

# KZG Special Soundness

- Adversary provides: $\{(\alpha_0, \eta_0), \ldots, (\alpha_n, \eta_n)\}$
- **<u>Basic math:</u>** $n + 1$ poly evaluations define unique $f(X)$ of degree $\leq n$

$\mathrm{Ext}_{SS}([1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2, [c]_1, \{\alpha_i, \eta_i, [\pi_i]_1\}_{i=0}^{n})$

1. Interpolate $f(X)$;
2. $[f(\sigma)]_1 =? [c]_1$;

   **If** yes **then** return $f(X)$

# KZG Special Soundness

- Adversary provides: $\{(\alpha_0, \eta_0), \ldots, (\alpha_n, \eta_n)\}$
- **Basic math:** $n + 1$ poly evaluations define unique $f(X)$ of degree $\leq n$

$\mathrm{Ext}_{SS}([1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2, [c]_1, \{\alpha_i, \eta_i, [\pi_i]_1\}_{i=0}^{n})$

1. Interpolate $f(X)$;
2. $[f(\sigma)]_1 =? [c]_1$;

   **If** yes **then** return $f(X)$

   **else** break a new assumption

# Rational Strong Diffie-Hellman Assumption

- RSDH proposed by González and Ràfols [Asiacrypt, 2019]

# Rational Strong Diffie-Hellman Assumption

- RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- Falsifiable assumption

# Rational Strong Diffie-Hellman Assumption

- RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- Falsifiable assumption
- Fixed $S = \{\alpha_0, \ldots \alpha_n\}$

# Rational Strong Diffie-Hellman Assumption

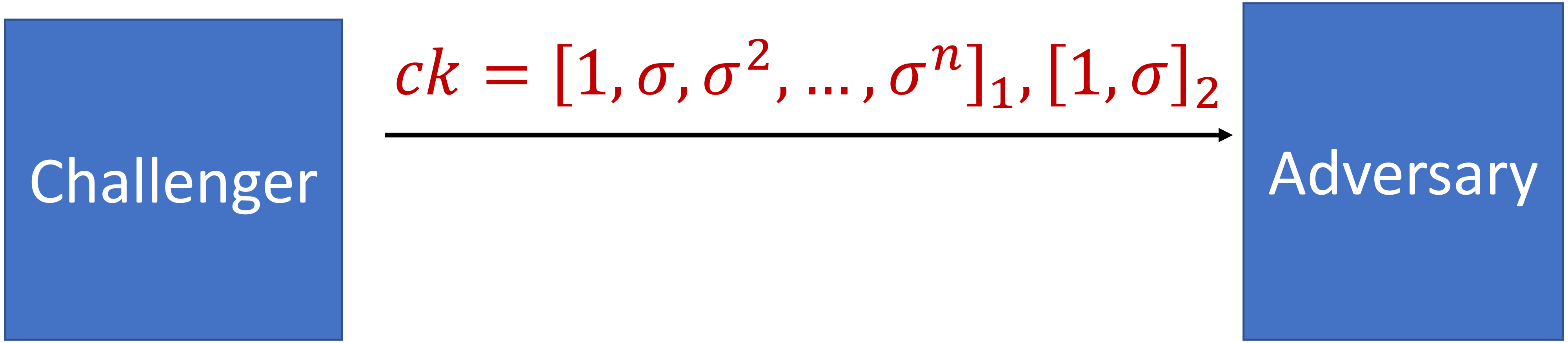
- RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- Falsifiable assumption
- Fixed $S = \{\alpha_0, \ldots \alpha_n\}$

Challenger

Adversary

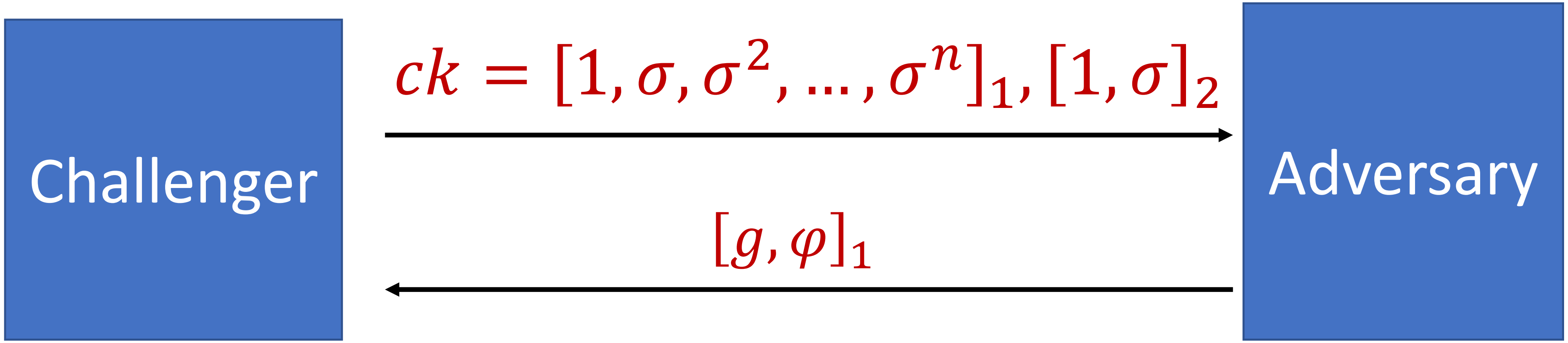
# Rational Strong Diffie-Hellman Assumption

- RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- Falsifiable assumption
- Fixed $S = \{\alpha_0, \dots \alpha_n\}$



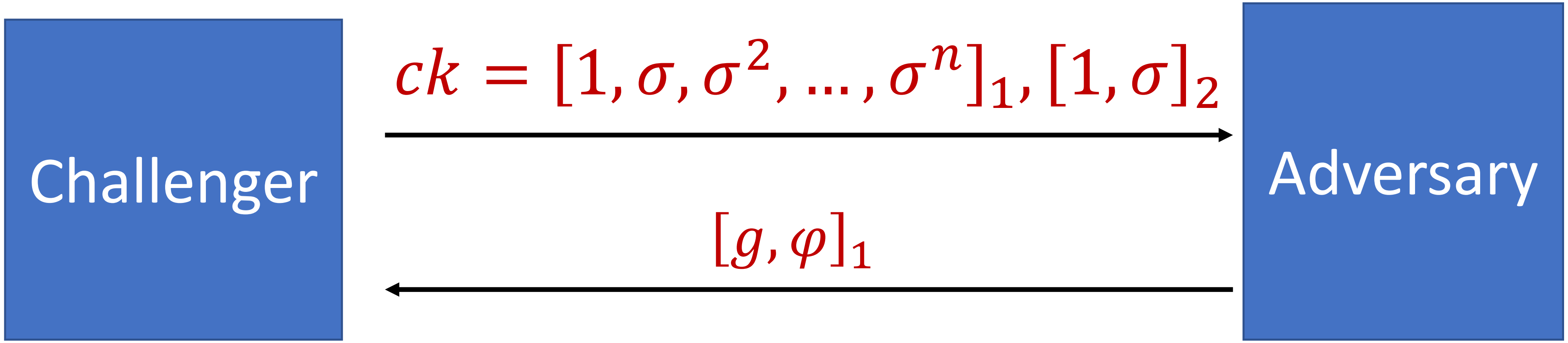
# Rational Strong Diffie-Hellman Assumption

- RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- Falsifiable assumption
- Fixed $S = \{\alpha_0, \dots \alpha_n\}$



# Rational Strong Diffie-Hellman Assumption

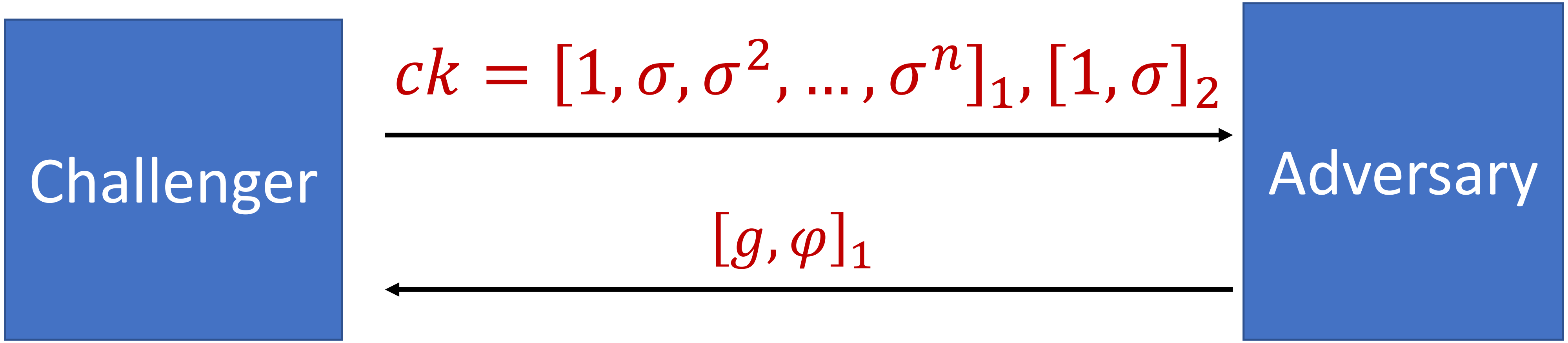
- ❑ RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- ❑ Falsifiable assumption
- ❑ Fixed $S = \{\alpha_0, \ldots \alpha_n\}$




**Win if:**

# Rational Strong Diffie-Hellman Assumption

- ❑ RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- ❑ Falsifiable assumption
- ❑ Fixed $S = \{\alpha_0, \ldots \alpha_n\}$

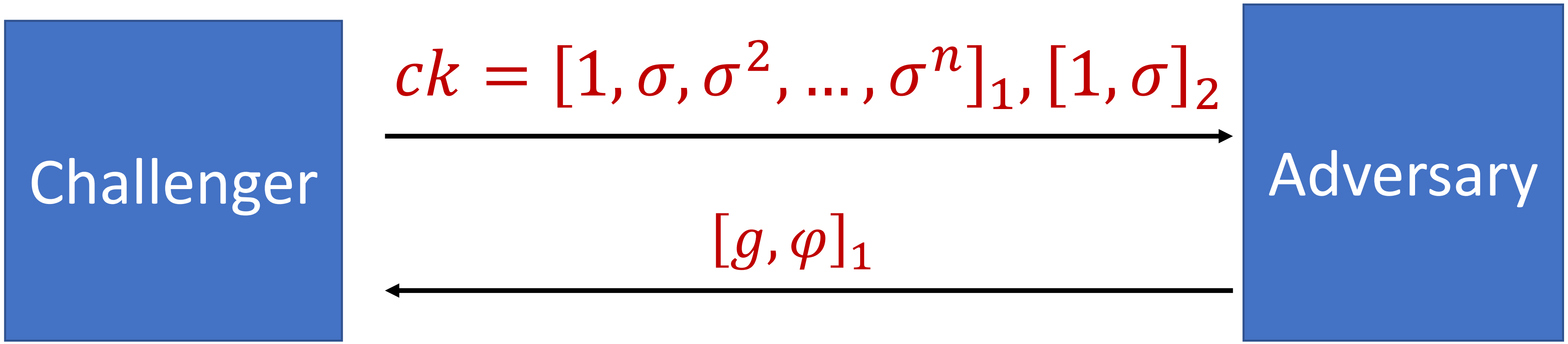



**Win if:**

- $[g]_1 \neq [0]_1$

# Rational Strong Diffie-Hellman Assumption

- RSDH proposed by González and Ràfols [Asiacrypt, 2019]
- Falsifiable assumption
- Fixed $S = \{\alpha_0, \ldots \alpha_n\}$




**Win if:**

- $[g]_1 \neq [0]_1$
- $[g]_1 \cdot [1]_2 = [\varphi]_1 \cdot [Z_S(\sigma)]_2$, where $Z_S(X) := \prod_{\alpha \in S}(X - \alpha)$

# Adaptive RSDH

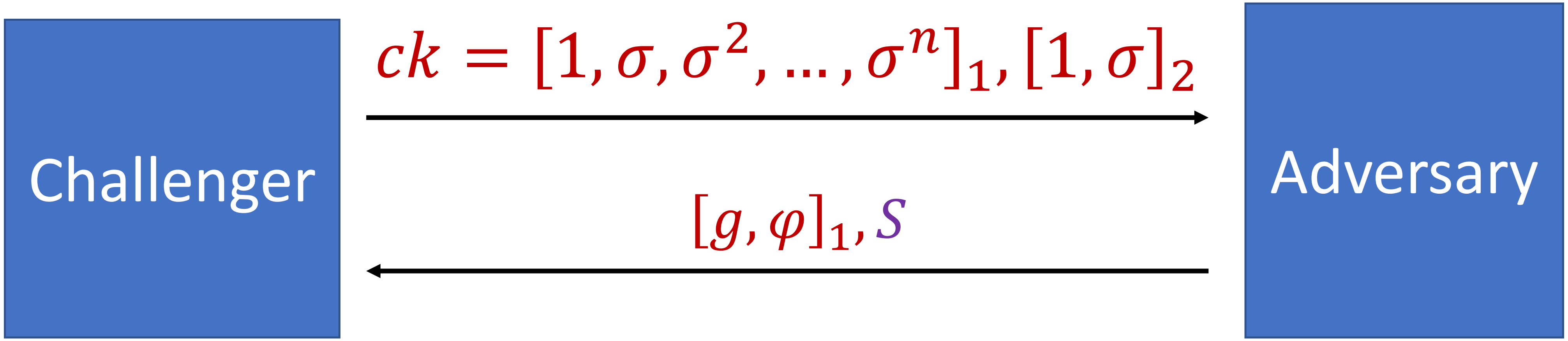
- A new assumption ARSDH

# Adaptive RSDH

- A new assumption ARSDH
- Falsifiable assumption

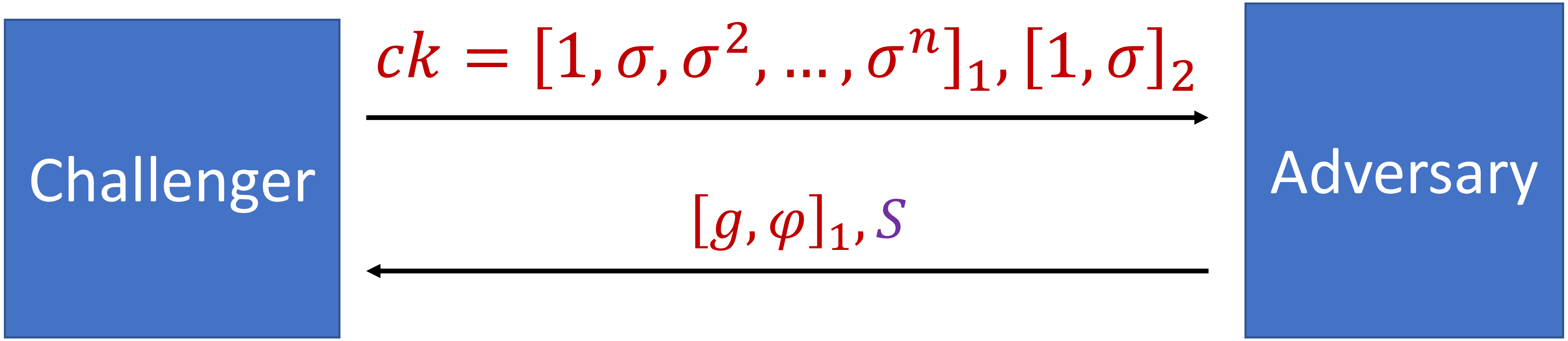
# Adaptive RSDH

- A new assumption ARSDH
- Falsifiable assumption



# Adaptive RSDH

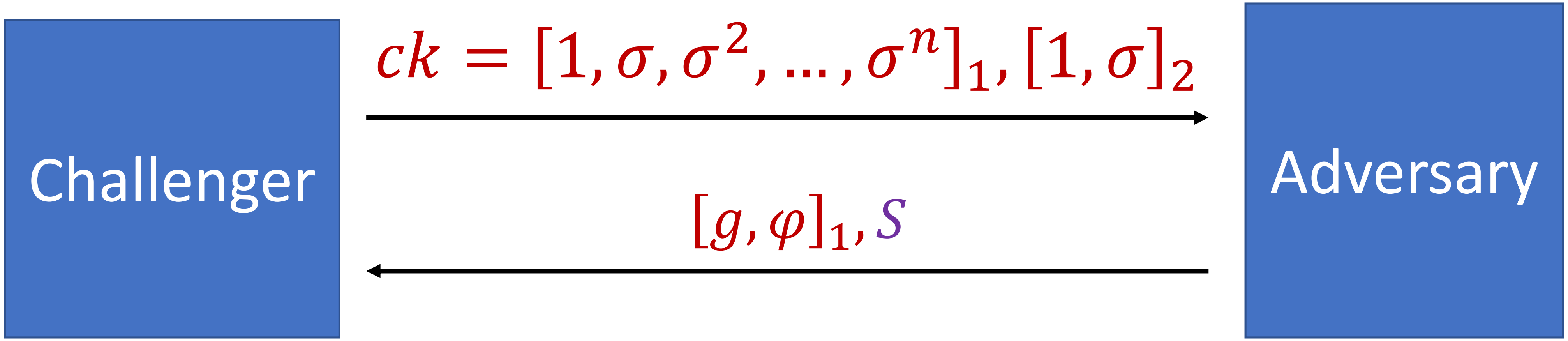
- ❑ A new assumption ARSDH
- ❑ Falsifiable assumption




**Win if:**

# Adaptive RSDH

- ❑ A new assumption ARSDH
- ❑ Falsifiable assumption




**Win if:**

- $S \subset \mathbb{Z}_p \wedge |S| = n + 1$

# Adaptive RSDH

- ❑ A new assumption ARSDH
- ❑ Falsifiable assumption

Challenger $\xrightarrow{ck = [1, \sigma, \sigma^2, \ldots, \sigma^n]_1, [1, \sigma]_2}$ Adversary

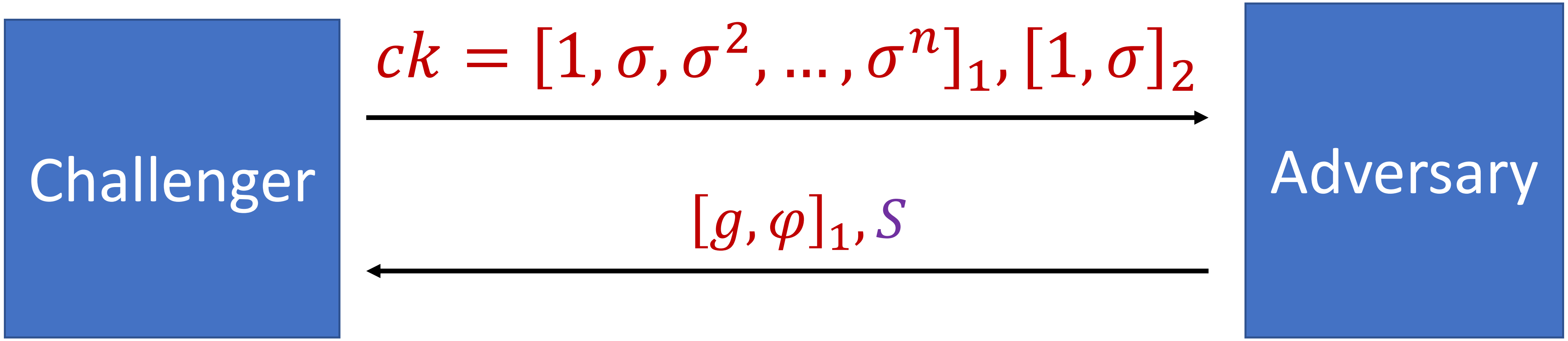
Challenger $\xleftarrow{[g, \varphi]_1, S}$ Adversary

**Win if:**

- $S \subset \mathbb{Z}_p \wedge |S| = n + 1$
- $[g]_1 \neq [0]_1$

# Adaptive RSDH

- ❑ A new assumption ARSDH
- ❑ Falsifiable assumption




**Win if:**

- $S \subset \mathbb{Z}_p \wedge |S| = n + 1$
- $[g]_1 \neq [0]_1$
- $[g]_1 \cdot [1]_2 = [\varphi]_1 \cdot [Z_S(\sigma)]_2$, where $Z_S(X) := \prod_{\alpha \in S}(X - \alpha)$

# More Results

- Special-soundness -> Black-box extractability

# More Results

- Special-soundness -> Black-box extractability
  - Rewind the adversary and run with distinct challenges

# More Results

- Special-soundness -> Black-box extractability
  - Rewind the adversary and run with distinct challenges
- Compiler for interactive arguments:

# More Results

- Special-soundness -> Black-box extractability
  - Rewind the adversary and run with distinct challenges
- Compiler for interactive arguments:
  - PIOP + black-box extractable polynomial commitment

# More Results

- Special-soundness -> Black-box extractability
  - Rewind the adversary and run with distinct challenges
- Compiler for interactive arguments:
  - PIOP + black-box extractable polynomial commitment
  - Similar to prior compilers

# Consequences

# Consequences

- KZG is black-box extractable under <u>falsifiable assumption</u>

# Consequences

- KZG is black-box extractable under falsifiable assumption
  - random evaluation point

# Consequences

- KZG is black-box extractable under falsifiable assumption
  - random evaluation point
- Constant-size interactive arguments under falsifiable assumption

# Consequences

- KZG is black-box extractable under falsifiable assumption
  - random evaluation point
- Constant-size interactive arguments under falsifiable assumption
- Constant-size SNARKs that are secure under falsifiable assumption and random oracle model

Thank you for attention

Questions?