What Makes Fiat-Shamir zkSNARKs (Updatable SRS) Simulation Extractable?

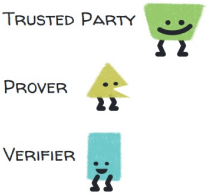
Chaya Ganesh [1] Hamidreza Khoshakhlagh [2] Markulf Kohlweiss [3,4] Anca Nitulescu [5] Michal Zajac [6]

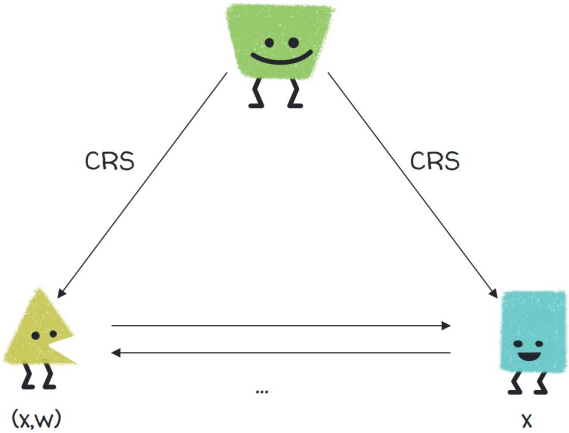
[1]Indian Institute of Science
[2]Aarhus University
[3]University of Edinburgh
[4]IOHK
[5]Protocol Labs
[6]Clearmatics

Outline

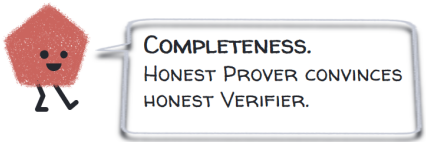
1. Motivation – what the paper does and why it matters?
2. A few words about Σ-protocols, Fiat–Shamir transformation, and simulation extractability of Σ-protocol-based NIZKs
3. Generalizing previous results to SNARKs
4. Glimpse into the main proof for simulation extractability

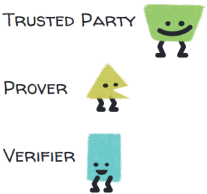
Argument systems in the trust model

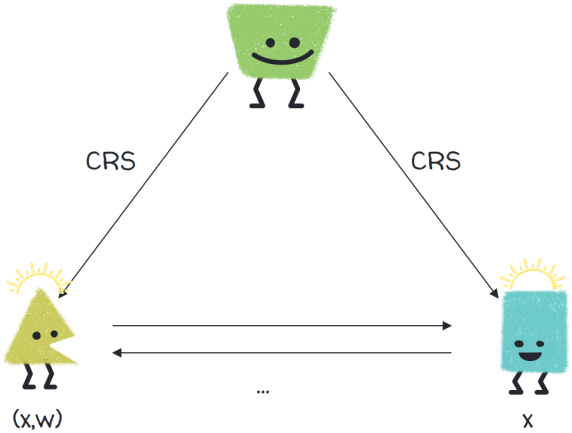




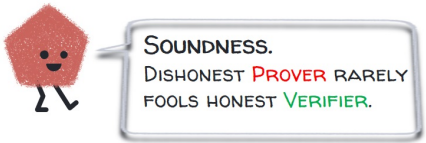
Argument systems in the trust model

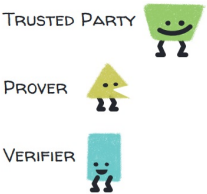


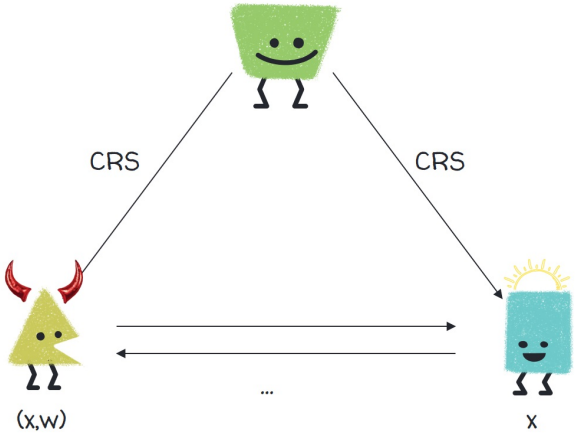




Argument systems in the trust model

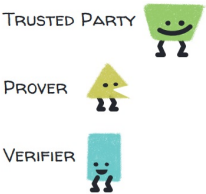


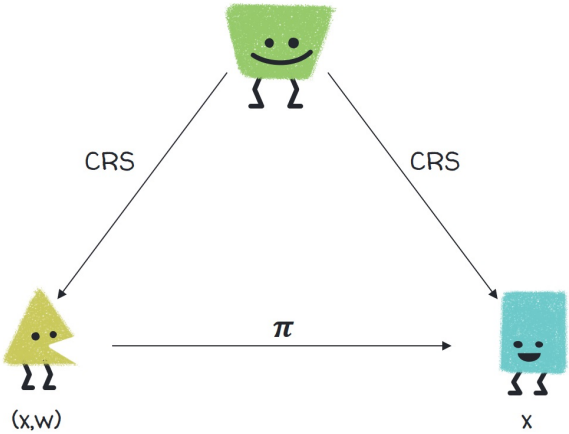




Argument systems in this work

Non-interactive
Zero-knowledge
Succinct
Of knowledge





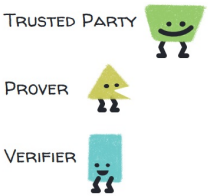
Argument systems in this work

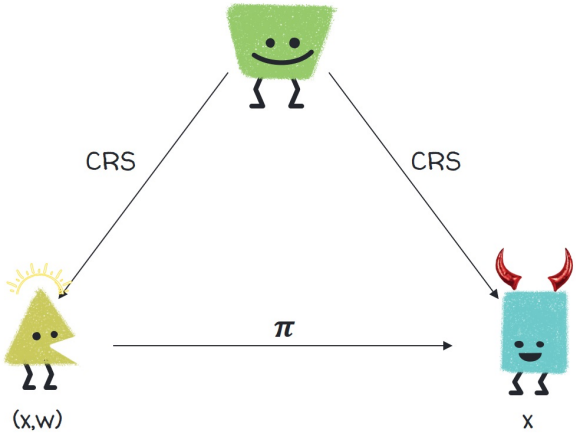
Non-interactive

Zero-knowledge

Succinct

Of knowledge

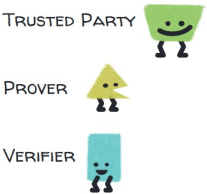


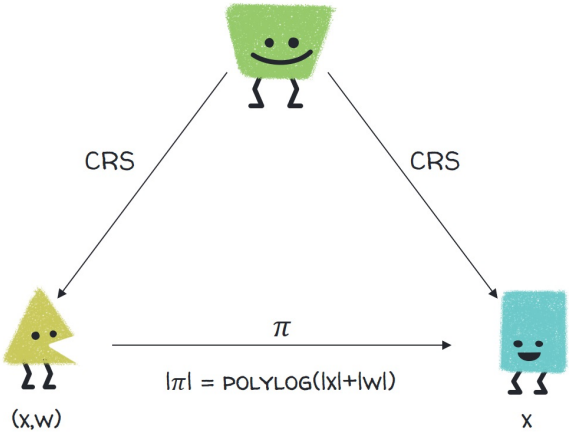


ZK: Dishonest Verifier only learns that $x \in L$

Argument systems in this work

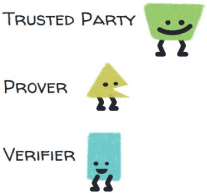
- Non-interactive
- Zero-knowledge
- Succinct: π is short.
- Of knowledge

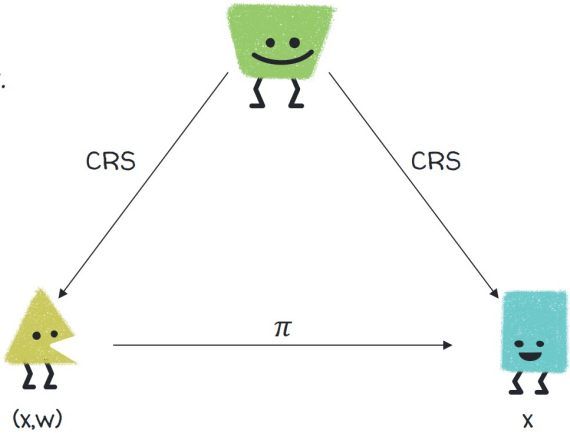




Argument systems in this work

- Non-interactive
- Zero-knowledge
- Succinct
- Of knowledge: Prover knows w.



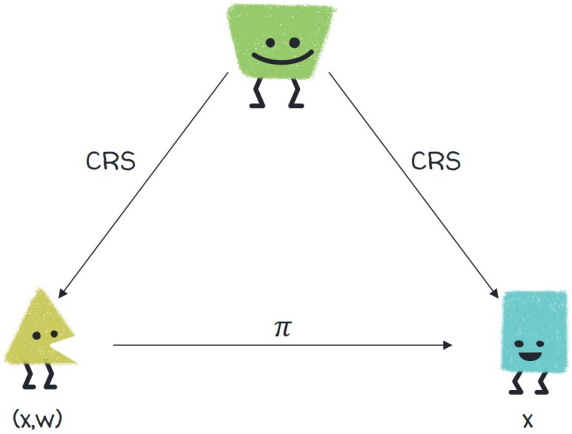


Argument systems in this work

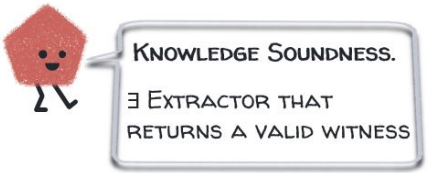
Trusted Party

Prover

Verifier

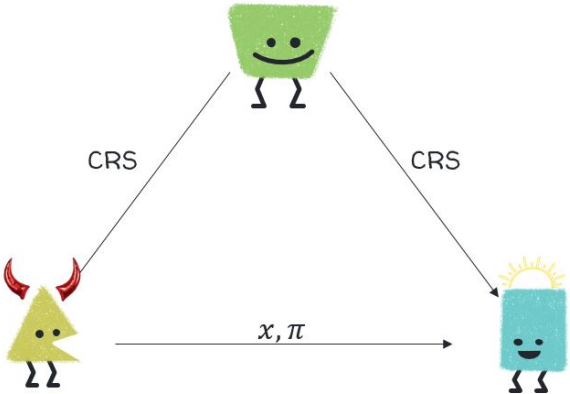


Is proof of knowledge enough for SNARKs?

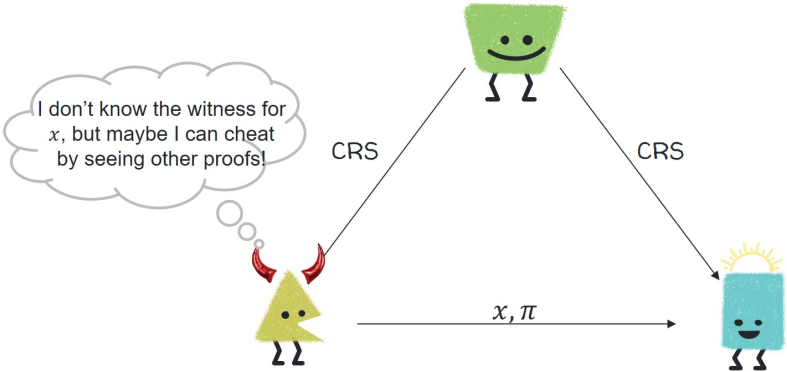


w

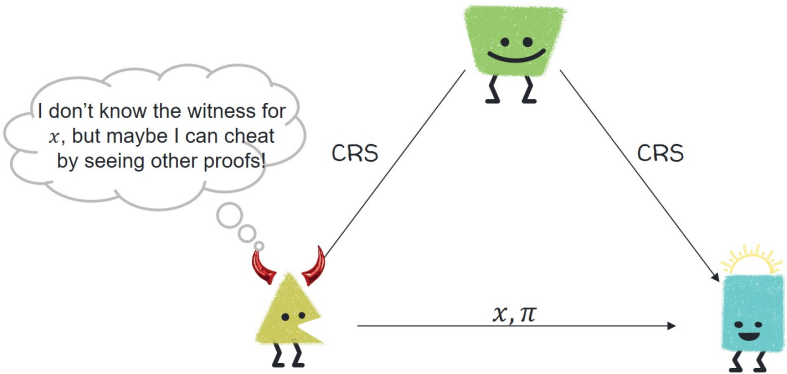
s.t $(x, w) \in R$



Is proof of knowledge enough for SNARKs?



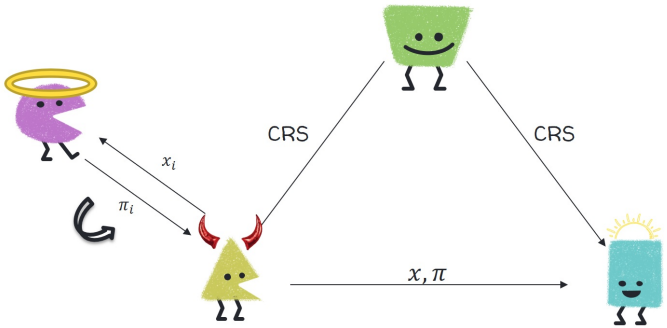
Is proof of knowledge enough for SNARKs?



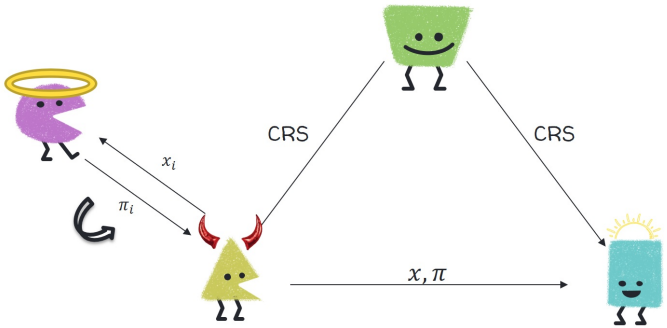
Example

Two users A and B.
A knows a secret w that allows to transfer money from his accounts (by showing a proof π of knowledge of w).
If π is randomizable, B seeing π can produce a valid PoK π' and transfer the funds too **(without knowing** w**)**.

Simulation Extractability (SE)



Simulation Extractability (SE)

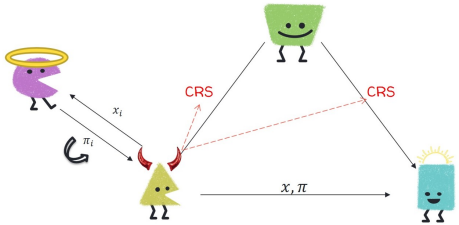


SE: for any PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathsf{Ext}_{\mathcal{A}}$ s.t. the following is negligible:

$$\Pr[(\mathsf{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{SimO}(\cdot)}(\mathsf{crs}); \mathsf{w} \leftarrow \mathsf{Ext}_{\mathcal{A}} : V(\mathsf{crs}, \mathsf{x}, \pi) = 1 \wedge (\mathsf{x}, \mathsf{w}) \notin \mathbf{R} \wedge (\mathsf{x}, \pi) \text{ is fresh }]$$

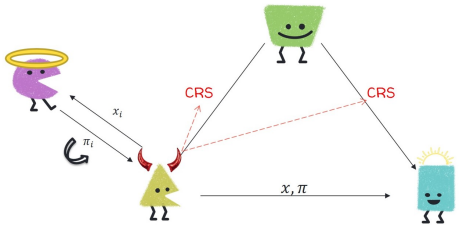
The result

Main result: **updatable simulation extractability** for a class of RO-based SNARKs (Plonk, Sonic, Marlin)



The result

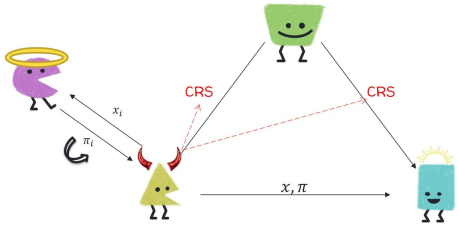
Main result: **updatable simulation extractability** for a class of RO-based SNARKs (Plonk, Sonic, Marlin)



How related to snarky signatures?

The result

Main result: **updatable simulation extractability** for a class of RO-based SNARKs (Plonk, Sonic, Marlin)

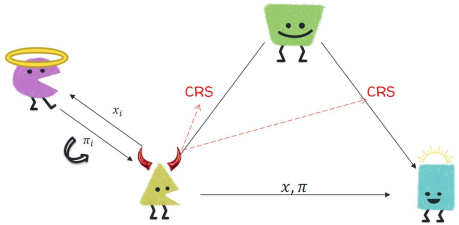


How related to snarky signatures?

- A signature of knowledge (SoK) uses an NP statement as the public verification key and signing requires knowledge of the NP witness.

The result

Main result: **updatable simulation extractability** for a class of RO-based SNARKs (Plonk, Sonic, Marlin)

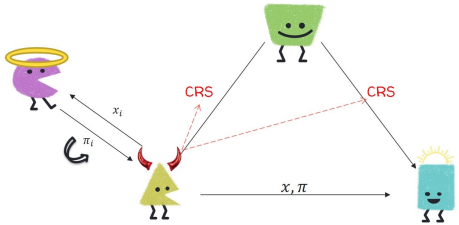


How related to snarky signatures?

- A signature of knowledge (SoK) uses an NP statement as the public verification key and signing requires knowledge of the NP witness.
- SE NIZK PoK implies SoK (and viceversa)

The result

Main result: **updatable simulation extractability** for a class of RO-based SNARKs (Plonk, Sonic, Marlin)



How related to snarky signatures?

- A signature of knowledge (SoK) uses an NP statement as the public verification key and signing requires knowledge of the NP witness.
- SE NIZK PoK implies SoK (and viceversa)
- A SoK from SE-SNARK is called Snarky signature.

Problem Statement

- Efficient updatable and universal zkSNARKs use random oracle and FS transformation.
- Knowledge soundness of Fiat-Shamir-based constructions rely on **forking lemma**.
- Forking lemma shows security only for a narrow class of protocols that **requires only a single rewinding** Not a case for any known zkSNARK.

Problem Statement

- Efficient updatable and universal zkSNARKs use random oracle and FS transformation.
- Knowledge soundness of Fiat-Shamir-based constructions rely on **forking lemma**.
- Forking lemma shows security only for a narrow class of protocols that **requires only a single rewinding** Not a case for any known zkSNARK.

Previous results

Faust et al. (Indocrypt 2012) showed that Σ-protocols that have unique-response property are simulation-extractable (after Fiat–Shamir transformation)

- Covers only 3-message protocols
- Witness has to be extractable from 2 transcripts
- Doesn't cover protocols with srs

Problem Statement

- Efficient updatable and universal zkSNARKs use random oracle and FS transformation.
- Knowledge soundness of Fiat-Shamir-based constructions rely on **forking lemma**.
- Forking lemma shows security only for a narrow class of protocols that **requires only a single rewinding** Not a case for any known zkSNARK.

Previous results

Faust et al. (Indocrypt 2012) showed that Σ-protocols that have unique-response property are simulation-extractable (after Fiat–Shamir transformation)

- Covers only 3-message protocols
- Witness has to be extractable from 2 transcripts
- Doesn't cover protocols with srs

Here

- Generalized forking lemma to work with wider class of protocols – multi-round, many transcripts to extract witness
- Covering protocols with srs

Problem Statement

- Efficient updatable and universal zkSNARKs use random oracle and FS transformation.
- Knowledge soundness of Fiat-Shamir-based constructions rely on **forking lemma**.
- Forking lemma shows security only for a narrow class of protocols that **requires only a single rewinding** Not a case for any known zkSNARK.

Previous results

Faust et al. (Indocrypt 2012) showed that Σ-protocols that have unique-response property are simulation-extractable (after Fiat–Shamir transformation)

- Covers only 3-message protocols
- Witness has to be extractable from 2 transcripts
- Doesn't cover protocols with srs

Here

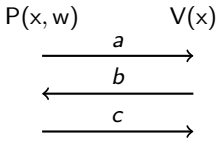
- Generalized forking lemma to work with wider class of protocols – multi-round, many transcripts to extract witness
- Covering protocols with srs

Required properties

- k-Unique response property
- Generalized forking lemma
- Trapdoor-less simulatability

Σ-protocols and Fiat–Shamir transformation

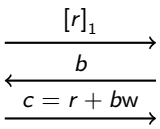
Σ-protocols



- **Completeness** Honest verifier accepts proof from an honest prover.
- **Special soundness** Given two transcripts for instance (x, a, b, c) and (x, a, b', c') one can compute witness w.
- **Honest verifier zero knowledge** The protocol is zero-knowledge if the verifier picks its challenges randomly.

Σ-protocols. Schnorr protocol – proof of knowledge of a discrete logarithm

$\mathsf{P}(\mathsf{x} = [\mathsf{w}]_1, \mathsf{w})$ $\quad$ $\mathsf{V}(\mathsf{x} = [\mathsf{w}]_1)$



V accepts iff $[c]_1 = [r + b\mathsf{w}]_1$

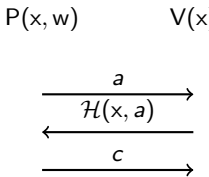
Special soundness.
From $([r]_1, b, r + b\mathsf{w})$ and $([r]_1, b', r + b'\mathsf{w})$, one computes

$$r + b\mathsf{w} - (r + b'\mathsf{w}) = (b - b')\mathsf{w}$$

$$\frac{r + b\mathsf{w} - (r + b'\mathsf{w})}{b - b'} = \mathsf{w}$$

Hence, for $b \neq b'$ one may reveal w.

Σ-protocols – Fiat–Shamir transformation



- **Completeness** Honest verifier accepts proof from an honest prover.
- **Special soundness** Given two transcripts for instance (x, a, b, c) and (x, a, b', c') one can compute witness w.
- **Honest verifier zero knowledge** The protocol is zero-knowledge if the verifier picks its challenges randomly.
- **Public coin** The verifier's challenges are public function of its randomness.

Special soundness of the Fiat–Shamir transformation

How to get two transcripts from $\mathcal{A}_{\mathsf{FS}}$

1. Get one transcript $(\mathsf{x}, a, b = \mathcal{H}(\mathsf{x}, a), c)$

Special soundness of the Fiat–Shamir transformation

How to get two transcripts from $\mathcal{A}_{\mathsf{FS}}$

1. Get one transcript $(\mathsf{x}, a, b = \mathcal{H}(\mathsf{x}, a), c)$
2. Rewind $\mathcal{A}_{\mathsf{FS}}$ after it sent a

Special soundness of the Fiat–Shamir transformation

How to get two transcripts from $\mathcal{A}_{\mathsf{FS}}$

1. Get one transcript $(\mathsf{x}, a, b = \mathcal{H}(\mathsf{x}, a), c)$
2. Rewind $\mathcal{A}_{\mathsf{FS}}$ after it sent a
3. Pick new $\mathcal{H}$ response b' for $\mathcal{H}(\mathsf{x}, a)$

Special soundness of the Fiat–Shamir transformation

How to get two transcripts from $\mathcal{A}_{\mathsf{FS}}$

1. Get one transcript $(\mathsf{x}, a, b = \mathcal{H}(\mathsf{x}, a), c)$
2. Rewind $\mathcal{A}_{\mathsf{FS}}$ after it sent a
3. Pick new $\mathcal{H}$ response b' for $\mathcal{H}(\mathsf{x}, a)$
4. Get another transcript (x, a, b', c')

Special soundness of the Fiat–Shamir transformation

How to get two transcripts from $\mathcal{A}_{\mathsf{FS}}$

1. Get one transcript $(\mathsf{x}, a, b = \mathcal{H}(\mathsf{x}, a), c)$
2. Rewind $\mathcal{A}_{\mathsf{FS}}$ after it sent a
3. Pick new $\mathcal{H}$ response b' for $\mathcal{H}(\mathsf{x}, a)$
4. Get another transcript (x, a, b', c')

Problem

$\mathcal{A}$ has **one shot** to convince the verifier V.
If $\mathcal{A}_{\mathsf{FS}}$ does not like V's challenge, it may pick **another** instance x or a and try again.

What if the adversary keeps changing the instance so we cannot get 2 transcripts?

Special soundness of the Fiat–Shamir transformation

How to get two transcripts from $\mathcal{A}_{\mathsf{FS}}$

1. Get one transcript $(\mathsf{x}, a, b = \mathcal{H}(\mathsf{x}, a), c)$
2. Rewind $\mathcal{A}_{\mathsf{FS}}$ after it sent a
3. Pick new $\mathcal{H}$ response b' for $\mathcal{H}(\mathsf{x}, a)$
4. Get another transcript (x, a, b', c')

Problem

$\mathcal{A}$ has **one shot** to convince the verifier V.
If $\mathcal{A}_{\mathsf{FS}}$ does not like V's challenge, it may pick **another** instance x or a and try again.

What if the adversary keeps changing the instance so we cannot get 2 transcripts?

What is the probability that we obtain two acceptable transcripts (x, a, b, c) and (x, a, b', c')?

Special soundness of the Fiat–Shamir transformation

How to get two transcripts from $\mathcal{A}_{\mathsf{FS}}$

1. Get one transcript $(\mathsf{x}, a, b = \mathcal{H}(\mathsf{x}, a), c)$
2. Rewind $\mathcal{A}_{\mathsf{FS}}$ after it sent a
3. Pick new $\mathcal{H}$ response b' for $\mathcal{H}(\mathsf{x}, a)$
4. Get another transcript (x, a, b', c')

Problem

$\mathcal{A}$ has **one shot** to convince the verifier V.
If $\mathcal{A}_{\mathsf{FS}}$ does not like V's challenge, it may pick **another** instance x or a and try again.

What if the adversary keeps changing the instance so we cannot get 2 transcripts?

What is the probability that we obtain two acceptable transcripts (x, a, b, c) and (x, a, b', c')?

Forking lemma

Let acc be a probability that $\mathcal{A}_{\mathsf{FS}}$ returns an acceptable proof.
q – upper bound for a number of random oracle queries $\mathcal{A}_{\mathsf{FS}}$ may make.
h – random oracle's range size.

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right).$$

- What if there is more than 3 messages?
- What if more than 2 transcripts are necessary?

Problem 1 – unique response property

Unique response property

We say that an NI protocol has k-**unique response property** if it is infeasible for a PPT adversary $\mathcal{A}_{\mathsf{FS}}$ to produce two **different** transcripts that have the same first k-messages.

Problem 1 – unique response property

Unique response property

We say that an NI protocol has k-**unique response property** if it is infeasible for a PPT adversary $\mathcal{A}_{\mathsf{FS}}$ to produce two **different** transcripts that have the same first k-messages.

- Ψ a $(2\mu+1)$-message protocol
- Non-interactive by Fiat–Shamir
- $(a_1, \ldots, a_{\mu+1})$ – prover's messages

$$\Pr\left[\begin{array}{l|l} \vec{a} = (a_1, \ldots, a_{\mu+1}), \vec{a'} = (a'_1, \ldots, a'_{\mu+1}), & \\ \vec{a} \neq \vec{a'}, a_1, \ldots, a_k = a'_1, \ldots, a'_k, & \mathsf{x}, \vec{a}, \vec{a'} \leftarrow \mathcal{A}^{\mathcal{H}, \mathsf{UpdO}}(1^\lambda) \\ \mathsf{V}_{\mathsf{FS}}(\mathsf{srs}, \mathsf{x}, \vec{a}) = \mathsf{V}_{\mathsf{FS}}(\mathsf{srs}, \mathsf{x}, \vec{a'}) = 1 & \end{array}\right] \leq \varepsilon_{\mathsf{ur}}(\lambda)$$

Example: 1-ur Schnorr protocol

Transcript for instance $[x]_1$: $([r]_1, b, [r + bx]_1)$
After challenge b is sent, $[r + bx]_1$ is **determined**.

Updatable SRS Schemes and Oracle UpdO

We always deal with updatable SRS.

Updatable SRS schemes

- $(\mathsf{srs}, \rho) \leftarrow \mathsf{KGen}(\mathbf{R})$ outputs a SRS srs with correctness proof ρ.
- $(\mathsf{srs}', \rho') \leftarrow \mathsf{Upd}(\mathsf{srs}, \{\rho_j\}_{j=1}^{n})$ outputs an updated SRS with a proof of correct update.
- $\{0,1\} \leftarrow \mathsf{VerifySRS}(\mathsf{srs}, \{\rho_j\}_{j=1}^{n})$ accepts or rejects srs.

Updatable SRS Schemes and Oracle UpdO

We always deal with updatable SRS.

Updatable SRS schemes

- $(\mathsf{srs}, \rho) \leftarrow \mathsf{KGen}(\mathbf{R})$ outputs a SRS srs with correctness proof ρ.
- $(\mathsf{srs}', \rho') \leftarrow \mathsf{Upd}(\mathsf{srs}, \{\rho_j\}_{j=1}^n)$ outputs an updated SRS with a proof of correct update.
- $\{0,1\} \leftarrow \mathsf{VerifySRS}(\mathsf{srs}, \{\rho_j\}_{j=1}^n)$ accepts or rejects srs.

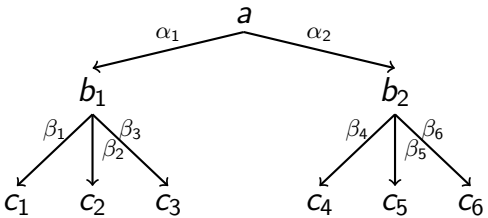
We define an SRS update oracle UpdO by which $\mathcal{A}$ updates the SRS.

$\mathsf{UpdO}(\mathtt{intent}, \mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$

if $\mathsf{srs} \neq \perp$: **return** $\perp$
if $(\mathtt{intent} = \mathtt{setup})$:
  $(\mathsf{srs}', \rho') \leftarrow \mathsf{KGen}(\mathbf{R})$; $Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$
  return (srs', ρ')
if $(\mathtt{intent} = \mathtt{update})$:
  if $\mathsf{VerifySRS}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n) = 0$: **return** $\perp$
  $(\mathsf{srs}', \rho') \leftarrow \mathsf{Upd}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$; $Q_{\mathsf{srs}} \leftarrow Q_{\mathsf{srs}} \cup \{(\mathsf{srs}', \rho')\}$
  return (srs', ρ')
if $(\mathtt{intent} = \mathtt{final})$:
  $b \leftarrow \mathsf{VerifySRS}(\mathsf{srs}_n, \{\rho_j\}_{j=1}^n)$
  if $(b = 0) \vee Q_{\mathsf{srs}} \cap \{\rho_j\}_i = \emptyset$: **return** $\perp$
  $\mathsf{srs} \leftarrow \mathsf{srs}_n$; **return** srs
else return $\perp$

Problem 2 – generalizing forking lemma

Tree of accepting transcript



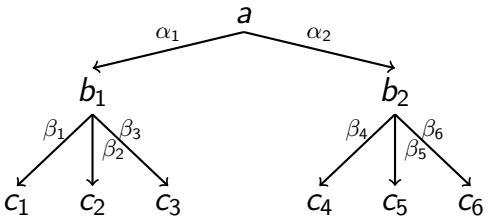
Prover's messages: a, b_i, c_j
Verifier's challenges α_k, β_l

We call such a tree a $(2, 3)$-**tree of acceptable transcripts**

Used to generalize **special soundness** to $(2, 3)$-special sound protocol – i.e. we can get a witness from a tree of **acceptable** transcripts as above

Problem 2 – generalizing forking lemma

Tree of accepting transcript

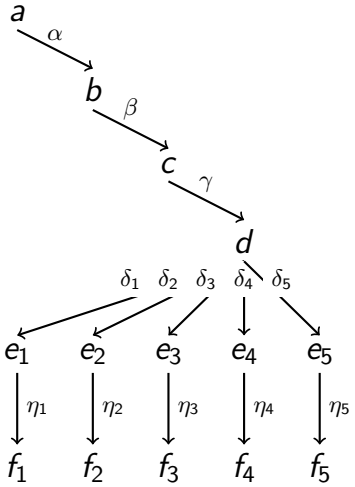


Prover's messages: a, b_i, c_j
Verifier's challenges α_k, β_l

We call such a tree a $(2, 3)$-**tree of acceptable transcripts**

Used to generalize **special soundness** to $(2, 3)$-special sound protocol – i.e. we can get a witness from a tree of **acceptable** transcripts as above

Ex: $(1, 1, 1, 5, 1)$-tree of acceptable transcripts



Problem 2 – generalizing forking lemma

Recall

Forking lemma states that probability of getting 2 acceptable transcripts (x, a, b, c), (x, a, b', c') is at least

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right).$$

acc: the probability $\mathcal{A}_{\mathsf{FS}}$ returns an acceptable proof.
q: upper bound on the number of RO queries $\mathcal{A}_{\mathsf{FS}}$ makes.
h: random oracle's range size.

Problem 2 – generalizing forking lemma

Recall

Forking lemma states that probability of getting 2 acceptable transcripts (x, a, b, c), (x, a, b', c') is at least

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right).$$

acc: the probability $\mathcal{A}_{\mathsf{FS}}$ returns an acceptable proof.
q: upper bound on the number of RO queries $\mathcal{A}_{\mathsf{FS}}$ makes.
h: random oracle's range size.

Generalized forking lemma

Let Ψ be a $(2\mu + 1)$-message (interactive) protocol.
Assume that the witness can be extracted from a $(1, ..., n_k = m, ..., 1)$-tree of acceptable transcript. Then,

$$\mathsf{frk} \geq \frac{\mathsf{acc}^m}{q^{m-1}} - \mathsf{acc} \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right).$$

Problem 2 – generalizing forking lemma

Recall

Forking lemma states that probability of getting 2 acceptable transcripts (x, a, b, c), (x, a, b', c') is at least

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right).$$

acc: the probability $\mathcal{A}_{\mathsf{FS}}$ returns an acceptable proof.
q: upper bound on the number of RO queries $\mathcal{A}_{\mathsf{FS}}$ makes.
h: random oracle's range size.

Generalized forking lemma

Let Ψ be a $(2\mu + 1)$-message (interactive) protocol.
Assume that the witness can be extracted from a $(1, ..., n_k = m, ..., 1)$-tree of acceptable transcript. Then,

$$\mathsf{frk} \geq \frac{\mathsf{acc}^m}{q^{m-1}} - \mathsf{acc} \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right).$$

What is m?

- e.g., $3n$ for Plonk, where n is the number of circuit constraints.
- $\rightarrow$ exponential loss when the number of constraints is non-constant : (

Problem 2 – generalizing forking lemma

Recall

Forking lemma states that probability of getting 2 acceptable transcripts (x, a, b, c), (x, a, b', c') is at least

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right).$$

acc: the probability $\mathcal{A}_{\mathsf{FS}}$ returns an acceptable proof.
q: upper bound on the number of RO queries $\mathcal{A}_{\mathsf{FS}}$ makes.
h: random oracle's range size.

Alternative approach

Use a tighter forking lemma e.g., recent result by Attema et al 2021/1377 (work in progress).

Generalized forking lemma

Let Ψ be a $(2\mu + 1)$-message (interactive) protocol.
Assume that the witness can be extracted from a $(1, ..., n_k = m, ..., 1)$-tree of acceptable transcript. Then,

$$\mathsf{frk} \geq \frac{\mathsf{acc}^m}{q^{m-1}} - \mathsf{acc} \cdot \left(1 - \frac{h!}{(h-m)! \cdot h^m}\right).$$

What is m?

- e.g., $3n$ for Plonk, where n is the number of circuit constraints.
- $\rightarrow$ exponential loss when the number of constraints is non-constant : (

Problem 3 – Trapdoor-less simulatability (TLS)

Zero-Knowledge in Programmable Random Oracle

$\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ is stateful and runs in two modes:

Mode 1. $(h, st') \leftarrow \mathsf{Sim}(1, st, \mathsf{srs}, q)$ that answers random oracle calls to $\mathcal{H}$ on q (notation: $h \leftarrow \mathsf{Sim}_1(\mathsf{srs}, q)$).

Mode 2. $(\pi, st') \leftarrow \mathsf{Sim}(2, st, \mathsf{srs}, \mathsf{x})$ that simulates the argument for x (notation: $\pi \leftarrow \mathsf{Sim}_2(\mathsf{srs}, \mathsf{x})$).

Problem 3 – Trapdoor-less simulatability (TLS)

Zero-Knowledge in Programmable Random Oracle

$\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ is stateful and runs in two modes:

Mode 1. $(h, st') \leftarrow \mathsf{Sim}(1, st, \mathsf{srs}, q)$ that answers random oracle calls to $\mathcal{H}$ on q (notation: $h \leftarrow \mathsf{Sim}_1(\mathsf{srs}, q)$).

Mode 2. $(\pi, st') \leftarrow \mathsf{Sim}(2, st, \mathsf{srs}, \mathsf{x})$ that simulates the argument for x (notation: $\pi \leftarrow \mathsf{Sim}_2(\mathsf{srs}, \mathsf{x})$).

Trapdoor-Less Simulatable Proof System

Let $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be the Fiat–Shamir variant of a $(2\mu + 1)$-message proof system, and $\mathcal{H}$ be the random oracle.
Ψ is **trapdoor-less simulatable** if for any adversary $\mathcal{A}$,

$$\Pr\left[\mathcal{A}^{\mathsf{UpdO},\mathcal{H},\mathsf{P}}(1^\lambda)\right] \approx \Pr\left[\mathcal{A}^{\mathsf{UpdO},\mathsf{Sim}_1,\mathsf{Sim}_2}(1^\lambda)\right]$$

Problem 3 – Trapdoor-less simulatability (TLS)

Zero-Knowledge in Programmable Random Oracle

$\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ is stateful and runs in two modes:

Mode 1. $(h, st') \leftarrow \mathsf{Sim}(1, st, \mathsf{srs}, q)$ that answers random oracle calls to $\mathcal{H}$ on q (notation: $h \leftarrow \mathsf{Sim}_1(\mathsf{srs}, q)$).

Mode 2. $(\pi, st') \leftarrow \mathsf{Sim}(2, st, \mathsf{srs}, \mathsf{x})$ that simulates the argument for x (notation: $\pi \leftarrow \mathsf{Sim}_2(\mathsf{srs}, \mathsf{x})$).

Trapdoor-Less Simulatable Proof System

Let $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ be the Fiat–Shamir variant of a $(2\mu + 1)$-message proof system, and $\mathcal{H}$ be the random oracle.
Ψ is **trapdoor-less simulatable** if for any adversary $\mathcal{A}$,

$$\Pr\left[\mathcal{A}^{\mathsf{UpdO},\mathcal{H},\mathsf{P}}(1^\lambda)\right] \approx \Pr\left[\mathcal{A}^{\mathsf{UpdO},\mathsf{Sim}_1,\mathsf{Sim}_2}(1^\lambda)\right]$$

We show Plonk, Sonic and Marlin are TLS.

Definition (Simulation-extractable NIZK)

A NIZK proof system $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ is **updatable simulation-extractable** with respect to $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ with **extraction error** ν if for any PPT adversary $\mathcal{A}$ that is given oracle access to an SRS update oracle UpdO and Sim and that produces an accepting proof for Ψ with probability acc, where

$$\mathsf{acc} = \Pr\left[\begin{array}{c|c} \mathsf{V}(\mathsf{srs}, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) = 1 & r \leftarrow\$ \mathsf{R}(\mathcal{A}) \\ \wedge (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \notin Q & (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{UpdO},\mathsf{Sim}}(1^{\lambda}; r) \end{array}\right]$$

there exists an extractor $\mathsf{Ext}_{\mathsf{se}}$ such that

$$\mathsf{ext} = \Pr\left[\begin{array}{c|c} \mathsf{V}(\mathsf{srs}, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) = 1 \\ \wedge\ (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \notin Q \\ \wedge\ \mathbf{R}(\mathsf{x}_{\mathcal{A}}, \mathsf{w}_{\mathcal{A}}) = 1 \end{array} \middle| \begin{array}{l} r \leftarrow\$ \mathsf{R}(\mathcal{A}), (\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathsf{UpdO},\mathsf{Sim}}(1^{\lambda}; r) \\ \mathsf{w}_{\mathcal{A}} \leftarrow \mathsf{Ext}_{\mathsf{se}}(\mathsf{srs}, \mathcal{A}, r, \mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}}, Q, Q_{\mathcal{H}}) \end{array}\right] \geq \frac{1}{\mathsf{poly}(\lambda)}(\mathsf{acc} - \nu)^d - \varepsilon(\lambda)$$

for some polynomial $\mathsf{poly}(\lambda)$, constant d and negligible $\varepsilon(\lambda)$ whenever $\mathsf{acc} \geq \nu$.

- srs is the finalized SRS.
- Q contains all (x, π) pairs where x is an instance queried to Sim_2 by $\mathcal{A}$ and π is the simulator's answer.
- $Q_{\mathcal{H}}$ contains all $\mathcal{A}$'s queries to Sim_1 and the (simulated) random oracle's answers.

The main theorem

Theorem (Simulation-extractable multi-message protocols)

Let $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an interactive* $(2\mu + 1)$*-message proof system for* $\mathcal{R}(1^\lambda)$ *that is*

- *zero-knowledge with trapdoor-less simulatability,*
- *has* k*-*ur *property with security loss* $\varepsilon_{\mathrm{ur}}(\lambda)$*, and*
- *is* $(1, \ldots, n_k = m, \ldots, 1)$*-special sound, all in the updatable setting.*

The main theorem

Theorem (Simulation-extractable multi-message protocols)

Let $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an interactive* $(2\mu + 1)$*-message proof system for* $\mathcal{R}(1^\lambda)$ *that is*

- *zero-knowledge with trapdoor-less simulatability,*
- *has* k-ur *property with security loss* $\varepsilon_{\mathrm{ur}}(\lambda)$*, and*
- *is* $(1, \ldots, n_k = m, \ldots, 1)$*-special sound, all in the updatable setting.*

Let $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^\lambda$ *be a random oracle.*

The main theorem

Theorem (Simulation-extractable multi-message protocols)

Let $\Psi = (\mathsf{KGen}, \mathsf{P}, \mathsf{V}, \mathsf{Sim})$ *be an interactive* $(2\mu + 1)$*-message proof system for* $\mathcal{R}(1^\lambda)$ *that is*

- *zero-knowledge with trapdoor-less simulatability,*
- *has* k-ur *property with security loss* $\varepsilon_{\mathsf{ur}}(\lambda)$*, and*
- *is* $(1, \ldots, n_k = m, \ldots, 1)$*-special sound, all in the updatable setting.*

Let $\mathcal{H} \colon \{0,1\}^* \to \{0,1\}^\lambda$ *be a random oracle.*

Then Ψ_{FS} *is simulation-extractable with extraction error* $\varepsilon_{\mathsf{ur}}(\lambda)$ *against* PPT *adversaries that makes up to* q *random oracle queries and returns an acceptable proof with probability at least* acc*. The extraction probability* ext *is at least* $\mathsf{ext} \geq \frac{1}{q^{m-1}}(\mathsf{acc} - \varepsilon_{\mathsf{ur}}(\lambda))^m - \varepsilon(\lambda)$*, for some negligible* $\varepsilon(\lambda)$*.*

Proof idea

Game hops – starting from simulation-extractability game.
Define games and show that probability they abort is negligible

Proof idea

Game hops – starting from simulation-extractability game.
Define games and show that probability they abort is negligible

Game 0

Simulation extractability game: $\mathcal{A}$ has access to oracles $(\mathsf{UpdO}, \mathsf{Sim})$ and eventually outputs $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$
Game aborts if extractor $\mathsf{Ext}_{\mathcal{A}}$ fails to extract the corresponding witness.

Proof idea

Game hops – starting from simulation-extractability game.
Define games and show that probability they abort is negligible

Game 0

Simulation extractability game: $\mathcal{A}$ has access to oracles $(\mathsf{UpdO}, \mathsf{Sim})$ and eventually outputs $(\mathsf{x}_{\mathcal{A}}, \pi_{\mathcal{A}})$
Game aborts if extractor $\mathsf{Ext}_{\mathcal{A}}$ fails to extract the corresponding witness.

Game 1

The game is identical to Game 0, except it additionally aborts if the adversary outputs proof π that matches at first k places with some simulated proof, i.e.

$$(\mathsf{x}_{\mathcal{A}}, \pi[1..k]) = (\mathsf{x}_{\mathcal{A}}, \pi_{\mathsf{Sim}}[1..k])$$

If Game 1 aborts with non-negligible probability then $\mathcal{A}$ may be used to break k-ur property of Ψ.

Proof idea

Game 2

This game is identical to Game 1, except it additionally aborts if the extractor Ext fails to build a tree of accepting transcripts T.
Probability of that event is bounded by **generalized forking lemma.**

Proof idea

Game 2

This game is identical to Game 1, except it additionally aborts if the extractor Ext fails to build a tree of accepting transcripts T.
Probability of that event is bounded by **generalized forking lemma.**

Game 3

This game is identical to Game 2, except it additionally aborts if extractor $\mathsf{Ext}_{\mathsf{T}}$ fails to extract the witness from a tree of acceptable transcripts.
From the $(1, \ldots, n_k = m, \ldots, 1)$-special soundness definition – it is impossible for the adversary to make this game abort.

Thank you!

Question?

Lemma (General forking lemma)

Fix $q \in \mathbb{Z}$ and a set H of size $h > 2$. Let $\mathcal{Z}$ be a PPT *algorithm that on input $y, h_1, \ldots, h_q$ returns (i, s), where $i \in [0\,..\,q]$ and s is called a* **side output**. *Denote by* IG *a randomised instance generator. We denote by* acc *the probability*

$$\Pr[i > 0 \mid y \leftarrow \mathsf{IG}; h_1, \ldots, h_q \leftarrow\!\$\, H; (i, s) \leftarrow \mathcal{Z}(y, h_1, \ldots, h_q)]\,.$$

Let $\mathsf{F}_{\mathcal{Z}}(y)$ denote the algorithm described in Fig. 1, then the probability frk *defined as* $\mathsf{frk} := \Pr[b = 1 \mid y \leftarrow \mathsf{IG}; (b, s, s') \leftarrow \mathsf{F}_{\mathcal{Z}}(y)]$ *holds*

$$\mathsf{frk} \geq \mathsf{acc}\left(\frac{\mathsf{acc}}{q} - \frac{1}{h}\right).$$

$\mathsf{F}_{\mathcal{Z}}(y)$

$\rho \leftarrow\!\$\, \mathsf{R}(\mathcal{Z})$
$h_1, \ldots, h_q \leftarrow\!\$\, H$
$(i, s) \leftarrow \mathcal{Z}(y, h_1, \ldots, h_q; \rho)$
if $i = 0$ **return** $(0, \bot, \bot)$
$h'_i, \ldots, h'_q \leftarrow\!\$\, H$
$(i', s') \leftarrow \mathcal{Z}(y, h_1, \ldots, h_{i-1}, h'_i, \ldots, h'_q; \rho)$
if $(i = i') \wedge (h_i \neq h'_i)$ **return** $(1, s, s')$
else return $(0, \bot, \bot)$

Figure: *Forking algorithm* $\mathsf{F}_{\mathcal{Z}}$

Simulation-extractability of Σ-protocols

Let Ψ_{FS} be a Fiat–Shamir transformed Σ-protocol Ψ. Then Ψ_{FS} is simulation-extractable

Caveats

The protocol Ψ has to have **unique response property**
Simulation extractability depends on the probability acc

Unique response property

Ψ has unique response property if no PPT adversary $\mathcal{A}$ can come up with two acceptable transcripts
(x, a, b, c) and (x, a, b, c').