# Towards a White-Box Secure Fiat-Shamir Transformation

Gal Arnon

WEIZMANN INSTITUTE OF SCIENCE

Eylon Yogev

# Sigma Protocol

- 3-message protocol
- Public-coin
- Pre-processing

I know $w$ s.t.
$C(w) = y$

$\psi \leftarrow Digest(C)$

Prover

$m_1$ →

← $\rho$

$m_2$ →

Verifier

$m_1 \leftarrow P_{SP}$

$m_2 \leftarrow P_{SP}(\rho)$

$V_{SP}(\psi, y, m_1, \rho, m_2) \stackrel{?}{=} 1$

Soundness:

- **Statistical:** against unbounded provers (a.k.a. **proof** )
- **Computational:** against bounded provers (a.k.a. **argument**)

# Fiat-Shamir (FS)

I know $w$ s.t.
$C(w) = y$

$\psi \leftarrow Digest(C)$

Fiat-Shamir transformation:

interactive → non-interactive

Idea: replace verifier randomness with hash function $h$

Prover

$m_1 \leftarrow P_{SP}$

$\rho = h(\psi, y, m_1)$

$m_2 \leftarrow P_{SP}(\rho)$

$\xrightarrow{m_1, m_2}$

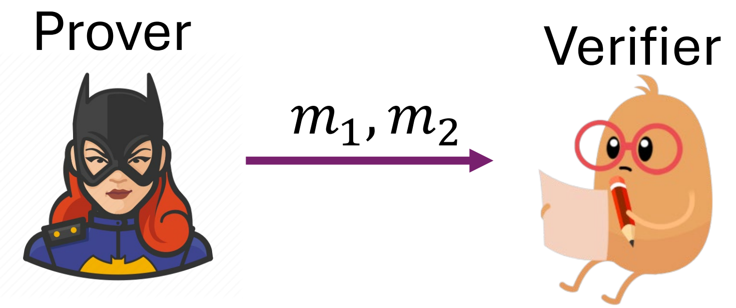
Verifier

$\rho = h(\psi, y, m_1)$

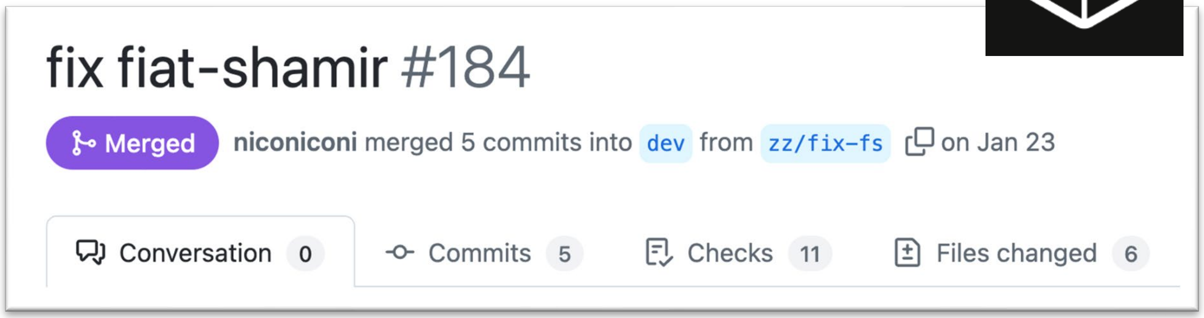
$V_{SP}(\psi, y, m_1, \rho, m_2) \stackrel{?}{=} 1$

FS is widely deployed real-world crypto systems, protecting billions of dollars: signature schemes, blockchains, ...

# Security of FS




- Pointcheval and Stern [PS96]: secure in the ROM
  - For both proofs and arguments
- **Proofs:** exist hash functions for which FS is secure
  [CCR16, KRR17, CCRR18, HL18, CCHLRRW19, PS19, BKM20, JJ21, HLR21, CJJ21, HJKS22, KLV23,...]
- **Arguments:** line of attacks using “white-box” techniques (“diagonalization”)
  - Interactive protocols that become insecure with FS for any concrete hash
- Examples of attacks:
  - [Bar01, GK03]: contrived identification schemes
  - [BBHMR19]: contrived CRH for Kilian’s protocol
  - **[KRS25]: direct attack on natural variant of the [GKR15] protocol**
- No attacks on Schnorr’s protocol



# Our Results

# Our Results

A new transformation (XFS) aimed to
mitigate white-box attacks

- Focus on practicality: negligible overhead to prover and verifier
- Circumvents recent attack on GKR
- Evidence for security: prove secure in a relativized model where FS is insecure

Attacks we don't defend against:
- [Bar01, GK03] for arbitrary poly-time verifiers
- [BBHMR19] CRH attack where to compute CRH need to verify a SNARK

But these attacks are contrived

# A Toy Protocol

# Toy Protocol

Let $Com$ be a succinct commitment scheme

I know $w$ s.t. $C(w) = y$

$\psi \leftarrow Com(C)$

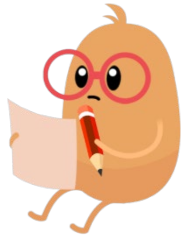
Prover

$m_1 = Com(w)$

$\rho$

decommit $C, w$

Verifier



$C(w) \stackrel{?}{=} y$

or

$C(w) \stackrel{?}{=} 1||\rho$

**Completeness:** if $C(w) = y$ verifier accepts

**Soundness (computational):** follows from the commitment scheme

- the circuit $C$ along with $w$ cannot predict $\rho$

# FS for Toy Protocol

I know $w$ s.t.
$C(w) = y$

Fiat-Shamir hash function $h$

$C(w)$:

1. Parse $w = \psi || y$
2. $m_1 \leftarrow Com(w)$
3. Output $1||h(\psi, y, m_1)$

Output starts with 1

Set $y = 0^m$, $\psi = Com(C)$, and $w = \psi || y$

Prover

$m_1 = Com(w)$

decommit $C, w$

$\psi \leftarrow Com(C)$

Verifier

$\rho = h(\psi, y, m_1)$

$C(w) \stackrel{?}{=} y$

or

$1||\rho = 1||h(\psi, y, m_1) = C(w) \stackrel{?}{=} 1||\rho$

**Insecure!**

For any $h$:

a prover strategy such that for all $w$, $C(w) \neq y$ but verifier accepts

# Attack on FS

**Main problem:** $C$ computes "verifier next message" $\rho = h(\psi, y, m_1)$

**Naïve solution:** make $h$ "more complex" than $C$

**Drawbacks:**

- Slow verifier (computes $h$, more complex than $C$)
- Non compatible with recursion
- Security unclear

We propose an alternative solution using strong proof of work

Intuitively: make next message function more complex than $C$, but easy to verify

# The XFS Transformation

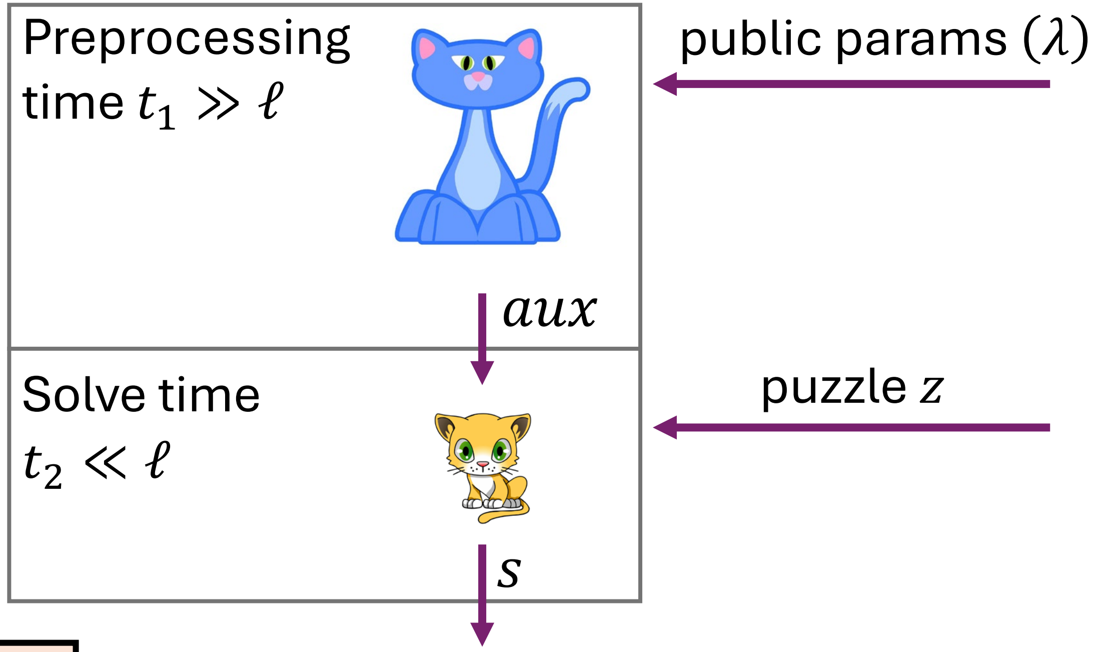
But first, a strong proof of work

# Strong Proof of Work

Proof of work with hardness $\ell$ ($\ll 2^{\lambda}$):

- $\mathrm{Solve}(z)$ solves puzzle $z$ in time $\ell$
- $\mathrm{Check}(z, s)$ verifies a solution $s$ to puzzle $z$

Security:
$neg(\lambda)$ probability with preprocessing




$\Pr[\mathrm{Check}(z, s) = 1] = neg(\lambda)$

Bitcoin PoW < Strong PoW < Proof of Sequential Work

guessing a solution works with probability $1/\ell$

We give: two constructs of strong proof of work

# XFS Transformation

I know $w$ s.t.
$C(w) = y$

$\psi \leftarrow Digest(C)$

Prover

$m_1, s, m_2$

Verifier

**Prover:**

$m_1 = P_{SP}$

$z = h(\psi, y, m_1)$

$s = \text{Solve}(z)$

$\rho = h(\psi, y, m_1, s)$

$m_2 = P_{SP}(\rho)$

**Verifier:**

$z = h(\psi, y, m_1)$

$\rho = h(\psi, y, m_1, s)$

$\text{V}_{\text{SP}}(\psi, y, m_1, \rho, m_2) \stackrel{?}{=} 1$

$\text{Check}(z, s) \stackrel{?}{=} 1$

# XFS for the Toy Protocol

# XFS for Toy Protocol

I know $w$ s.t.
$C(w) = y$

$\psi \leftarrow Com(C)$

Prover

$m_1 = Com(w)$ →

$\rho$ ←

decommit $C, w$ →

Verifier

$C(w) \stackrel{?}{=} y$

or

$C(w) \stackrel{?}{=} 1||\rho$

Does the attack work on XFS?

Need to modify $C$ to compute $\rho$

# XFS for Toy Protocol

Fiat-Shamir hash function $h$

Attack: circuit computes $\rho$

$C(w)$:

1. Parse $w = \psi || y$
2. $m_1 = Com(w)$
3. $z = h(\psi, y, m_1)$
4. $s = \text{Solve}(z)$
5. Output $1 || h(\psi, y, m_1, s)$

Set $y = 0^m$, $\psi = Com(C)$, and $w = \psi || y$

I know $w$ s.t.
$C(w) = y$

Prover

$m_1 = Com(w)$, $s$
decommit $C, w$

$\psi \leftarrow Com(C)$

Verifier

$z = h(\psi, y, m_1)$
$\rho = h(\psi, y, m_1, s)$
$\text{Check}(z, s) \stackrel{?}{=} 1$

$C(w) \stackrel{?}{=} y$
or
$C(w) \stackrel{?}{=} 1 || \rho$

Set PoW Hardness $\ell > |C|$ so $C$ can't compute $Solve(z)$

Can $w$ help solve the puzzle?

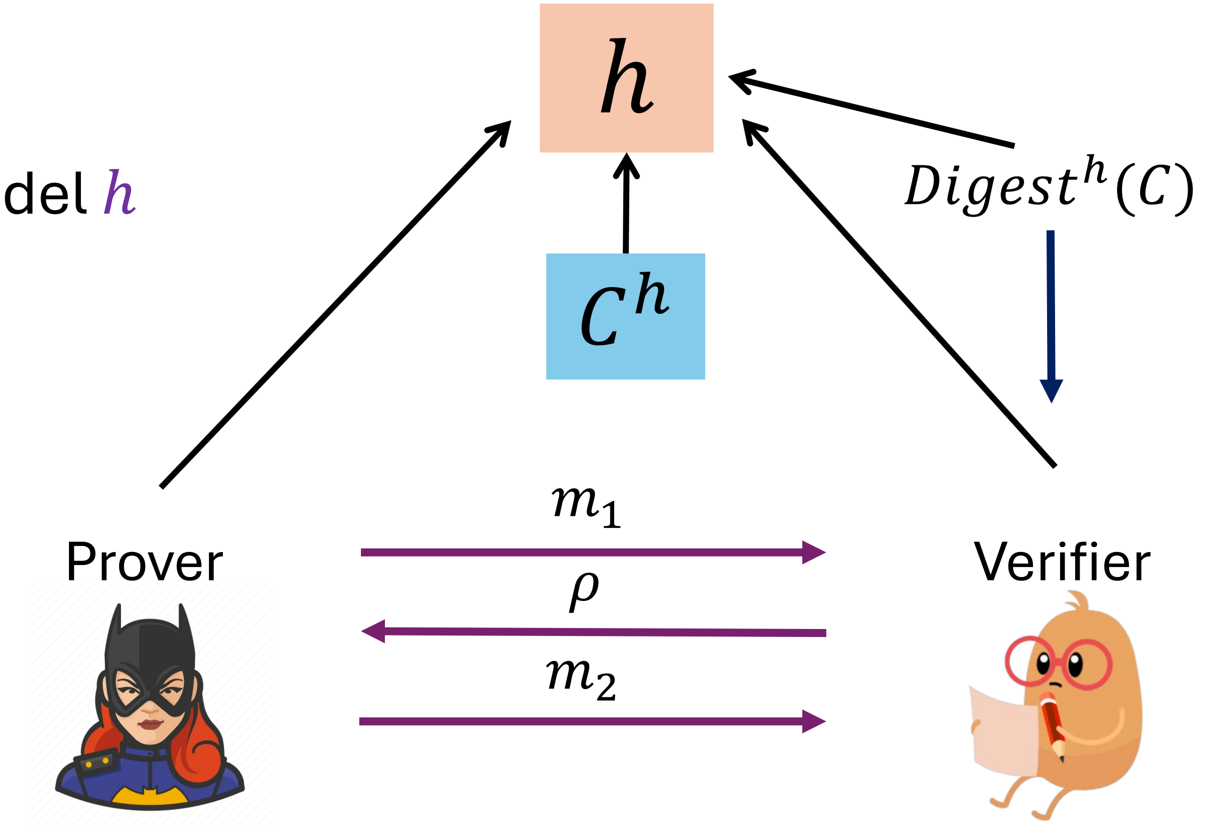
**Observe**: $z$ is computed after $C, w$ are committed

$$z = h(\psi, y, m_1) = h(Com(C), y, Com(w))$$

# On the Security of XFS

# The Relativized World

- An ideal model with a random oracle model $h$
- All parties have oracle access to $h$
- This model captures the toy protocol

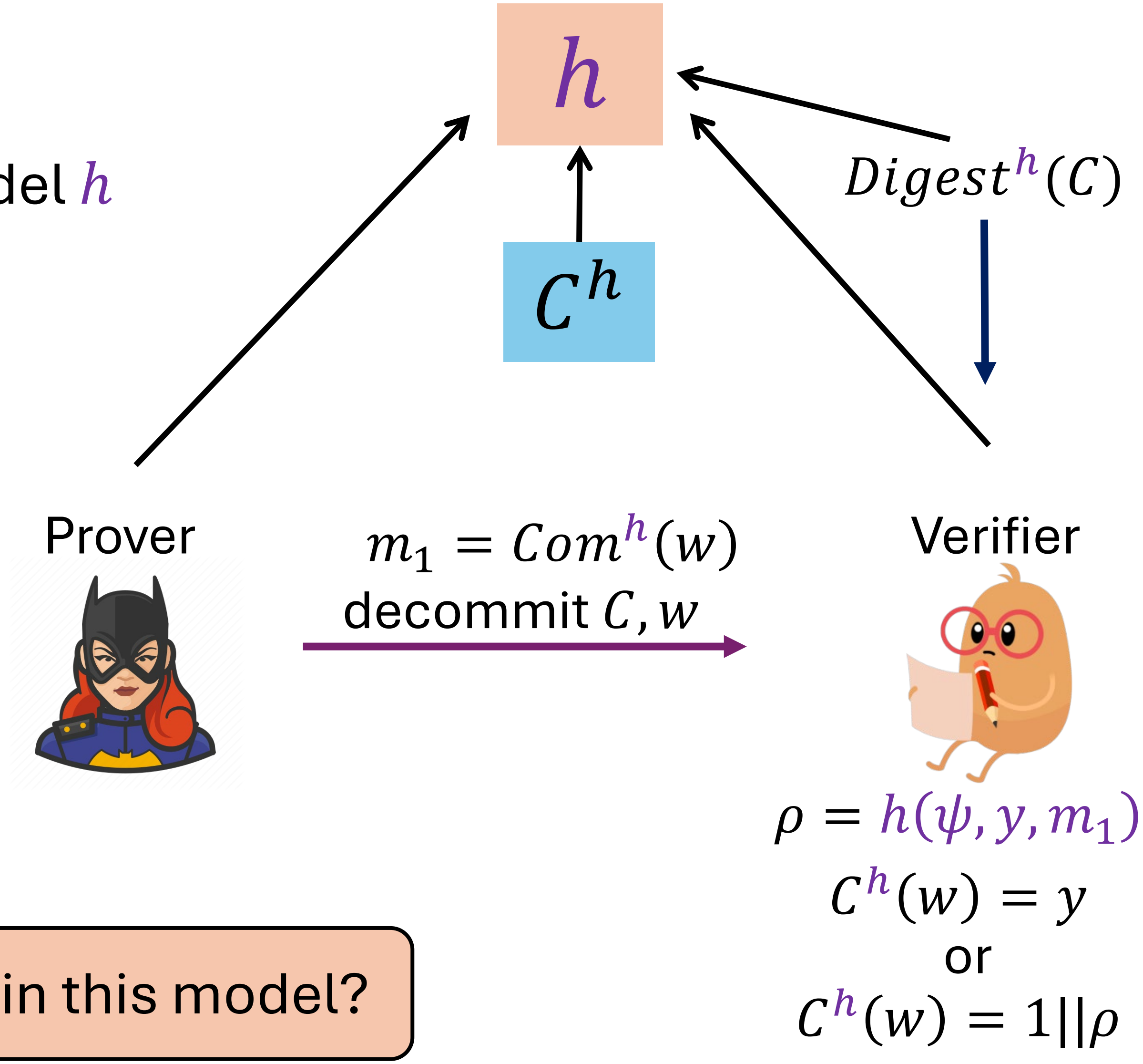


# The Relativized World

- An ideal model with a random oracle model $h$
- All parties have oracle access to $h$
- This model captures the toy protocol

$C^h(w)$:

1. Parse $w = \psi || y$
2. $m_1 \leftarrow Com(w)$
3. Output $1 || h(\psi, y, m_1)$




What about XFS in this model?

# Security in the Relativized World

**Theorem:**

In the relativized model, the XFS* transformation satisfies:

Input:

1. sigma protocol with round-by-round knowledge error $\kappa_{SP}$
2. strong PoW with error $\epsilon_{PoW}$

Output: non-interactive protocol with knowledge error

$$\kappa_{ARG} \leq O(t \cdot \kappa_{SP} + t \cdot \epsilon_{PoW})$$

where:

- $t$ is the query complexity of the malicious prover
- XFS* is a slight modification of the transformation we saw

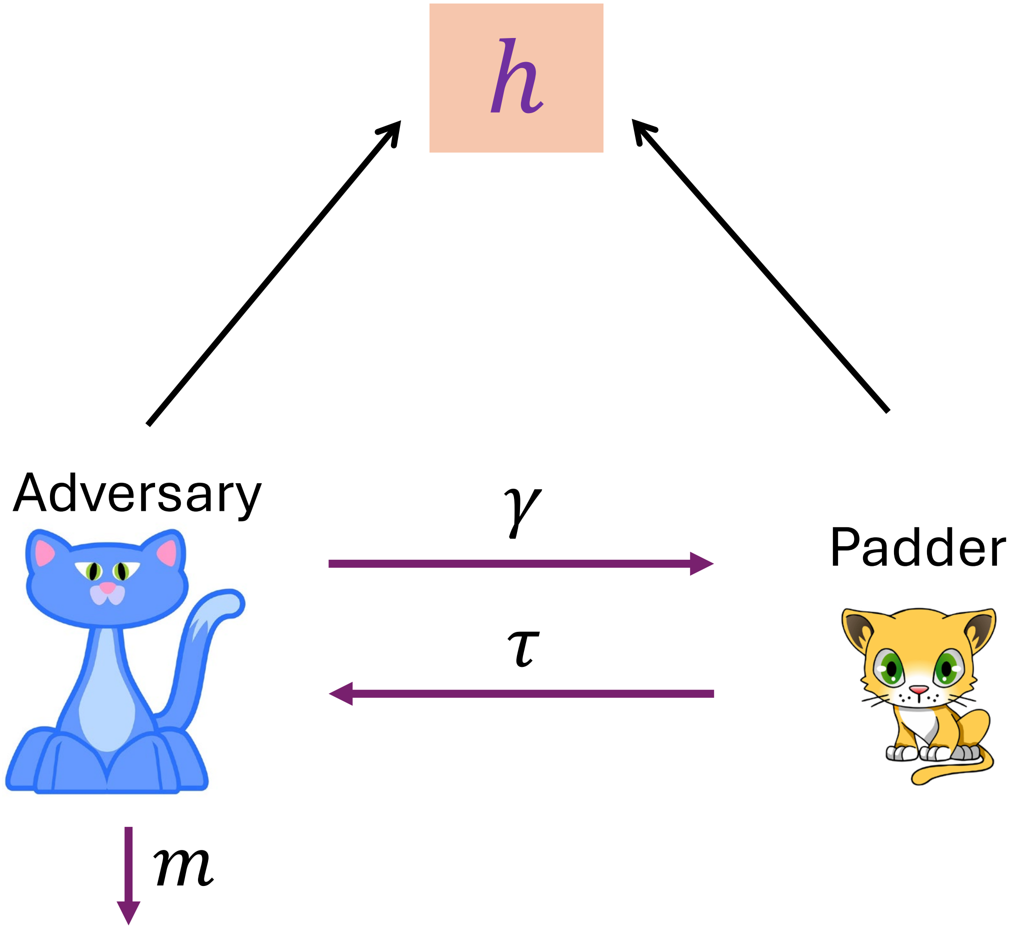
# Prefix Avoiding Padders

- Tool that facilitates our proof
  - Prevents the verifier from computing $\rho$
- Padder outputs a prefix $\tau$ (deterministically)

- For machine $M^h$ the security game is:

- Example of padders:
  - $Padder(\gamma) = 0^{|M|}$
  - $Padder(\gamma) = h(M, \gamma)$ (extended to be long)
  - In practice could be trivial

- Used for $M^h$ that simulates the verifier




$$\Pr\left[M^h(m) = (\tau, \gamma)\right] = neg(\lambda)$$

# XFS* Transformation

$h$

$\psi \leftarrow Digest^h(C)$

I know $w$ s.t.
$C(w) = y$

Prover

$m_1, s, m_2$

Verifier

$m_1 = P_{SP}$
$z = h(\psi, y, m_1)$
$s = \mathrm{Solve}^h(z)$
$\tau = \mathrm{Padder}^h(\psi, y, m_1, s)$
$\rho = h(\tau, \psi, y, m_1, s)$
$m_2 = P_{SP}(\rho)$

$z = h(\psi, y, m_1)$
$\tau = \mathrm{Padder}^h(\psi, y, m_1, s)$
$\rho = h(\tau, \psi, y, m_1, s)$
$\mathrm{V}_{\mathrm{SP}}^h(\psi, y, m_1, \rho, m_2) \stackrel{?}{=} 1$
$\mathrm{Check}^{\mathrm{h}}(\mathrm{z}, \mathrm{s}) \stackrel{?}{=} 1$

$M^h(m)$:
1. Parse $m = (\psi, y, m_1, \rho, m_2)$
2. Simulate $V_{SP}(\psi, y, m_1, \rho, m_2)$
3. Output a random query of $V_{SP}$

# Summary

We saw:

- FS insecure for arguments due to white-box attacks
- We propose XFS aimed to mitigate such attacks
- Uses strong PoW (ask me how to construct!)
- Heuristic proof of security in relativized model

Future work:

- Multi round version
- Security proofs in algebraic models
- Analyze with sponges

Thank You!

# Simple PoW construction

- Given a random puzzle $z$, and hardness $\ell$:
- Compute a Merkle tree of length $\ell$
  - The $i$-th leaf is $(\mathrm{z}, \mathrm{i})$
- Hash the root to get small subset $I \subseteq [\ell]$
- Open auth paths in $I$
- Any algorithm that performs at most $\ell/2$ hashes:
  - Can compute at most $\ell/2$ leaves
  - Probability of opening all leaves in $I$ is negligible