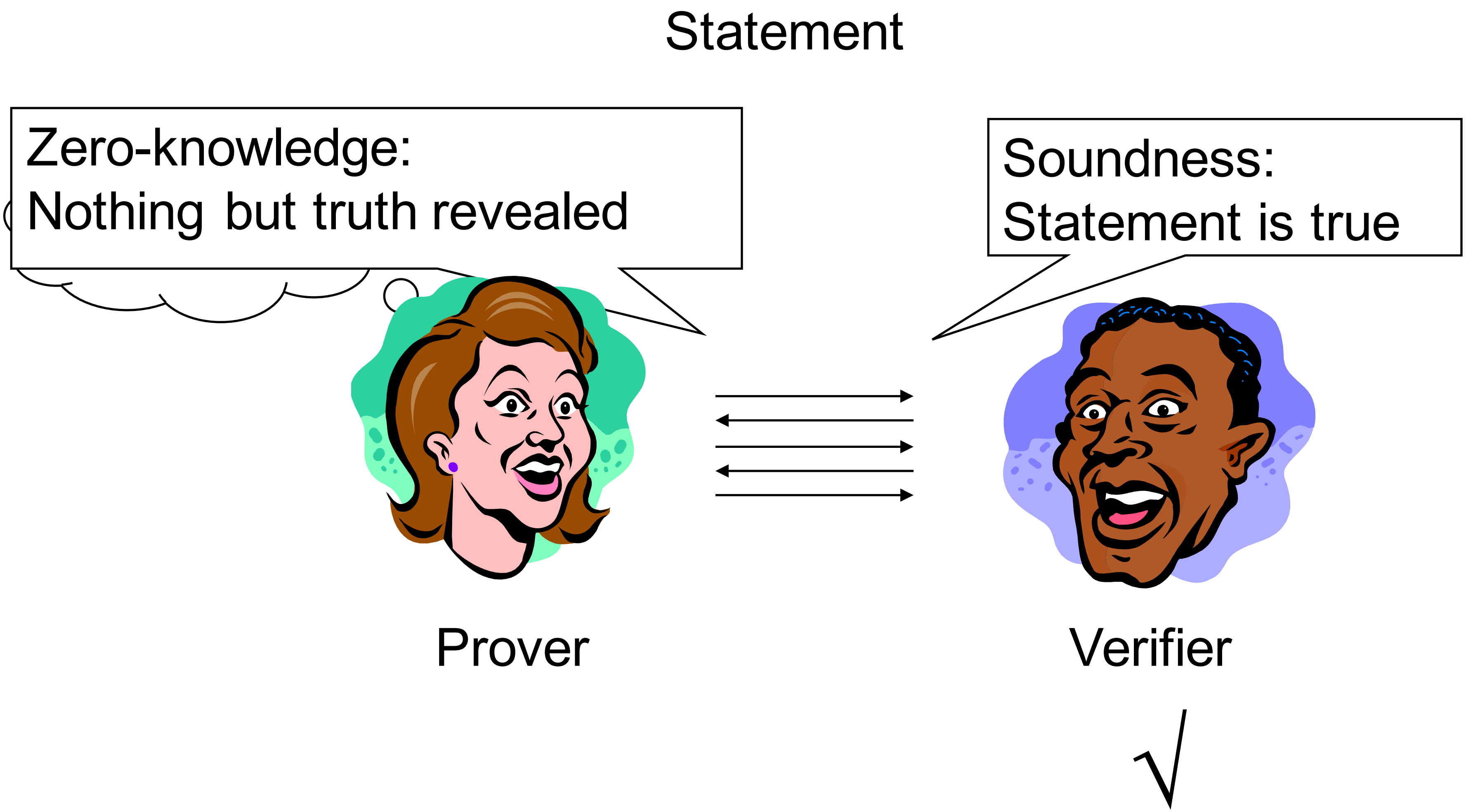
Efficient Zero-Knowledge Proofs

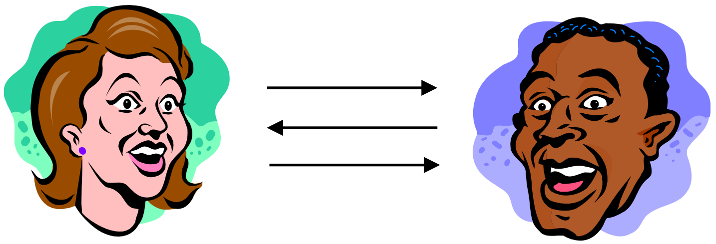
Jens Groth
University College London

Zero-knowledge proof

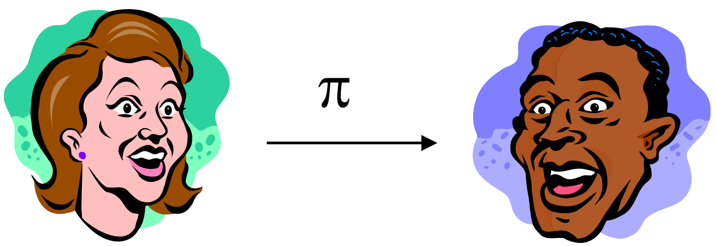


Round complexity

- Interactive zero-knowledge proof



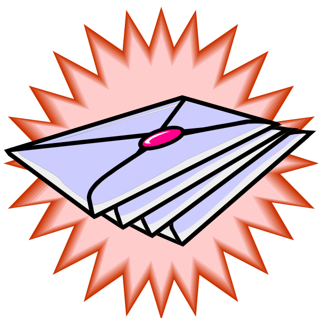
- Non-interactive zero-knowledge proof



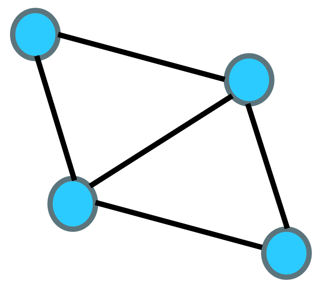
Statements

$$(x_1 \wedge x_2 \wedge \neg x_3) \vee (x_2 \wedge \mathrm{x}_4 \wedge x_5)$$

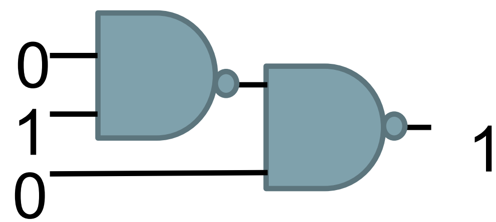
SAT



Plaintext is signature on…



Hamiltonian



Circuit SAT

- Statements are $\phi \in L$ for a given NP-language L
- Prover knows witness w such that $(\phi, w) \in R_L$
  - But wants to keep the witness secret!

Proof system (Setup,Prove,Verify)

- $\text{Setup}(1^\lambda) \rightarrow crs$:
  - Sometimes we assume a trusted setup. This is in particular required for non-interactive zero-knowledge.
- $\langle \text{Prove}(crs, \phi, w); \text{Verify}(crs, \phi) \rangle \rightarrow \text{accept/reject}$
  - Stateful algorithms Prove and Verify interact. In the end Verify accepts or rejects the proof.

In non-interactive proofs the prover generates a proof using $\text{Prove}(crs, \phi, w) \rightarrow \pi$ and the verifier runs $\text{Verify}(crs, \phi, \pi)$ to decide whether to accept or reject

Zero-knowledge proofs



- Completeness
  - Prover can convince the verifier when statement is true
- Soundness
  - Cheating prover cannot convince the verifier when statement is false
- Zero-knowledge
  - No leakage of information (except truth of statement) even if interacting with a cheating verifier
  - Defined as there being a simulator that can produce a transcript without knowing the witness (and therefore not leaking anything about the witness)

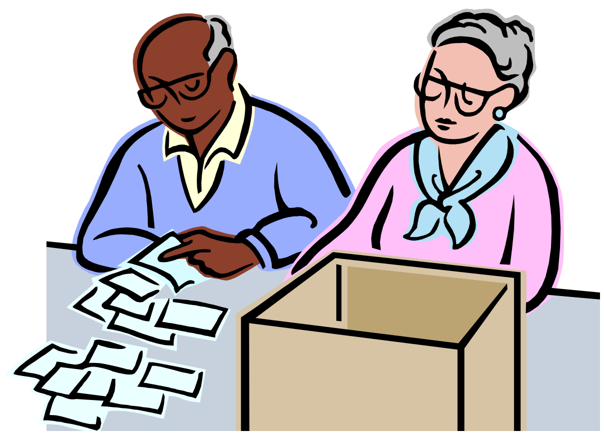
Internet voting

Vote

Encrypts vote to keep it private

Tally without decrypting individual votes

Ciphertext →



Voters

Election authorities

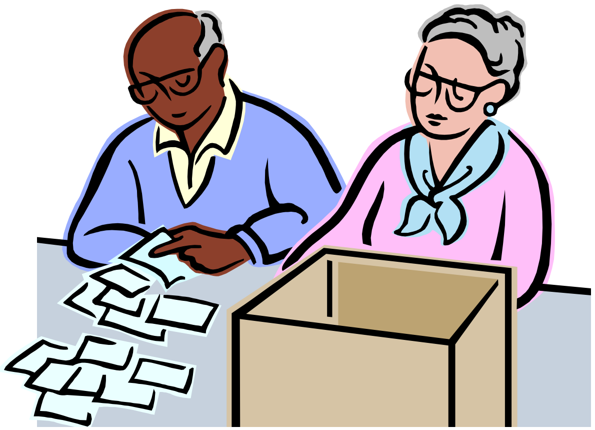
Election fraud

Not Bob

Encrypts -100 votes for Bob

Is the encrypted vote valid?

Ciphertext →



Voters

Election authorities

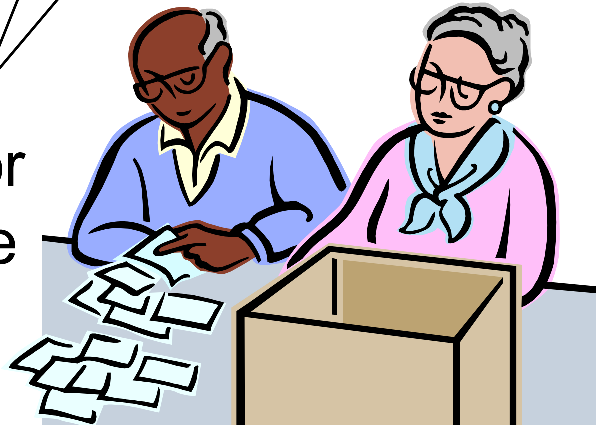
Zero-knowledge proof as solution

Zero-knowledge:
Vote is secret

Soundness:
Vote is valid

Ciphertext

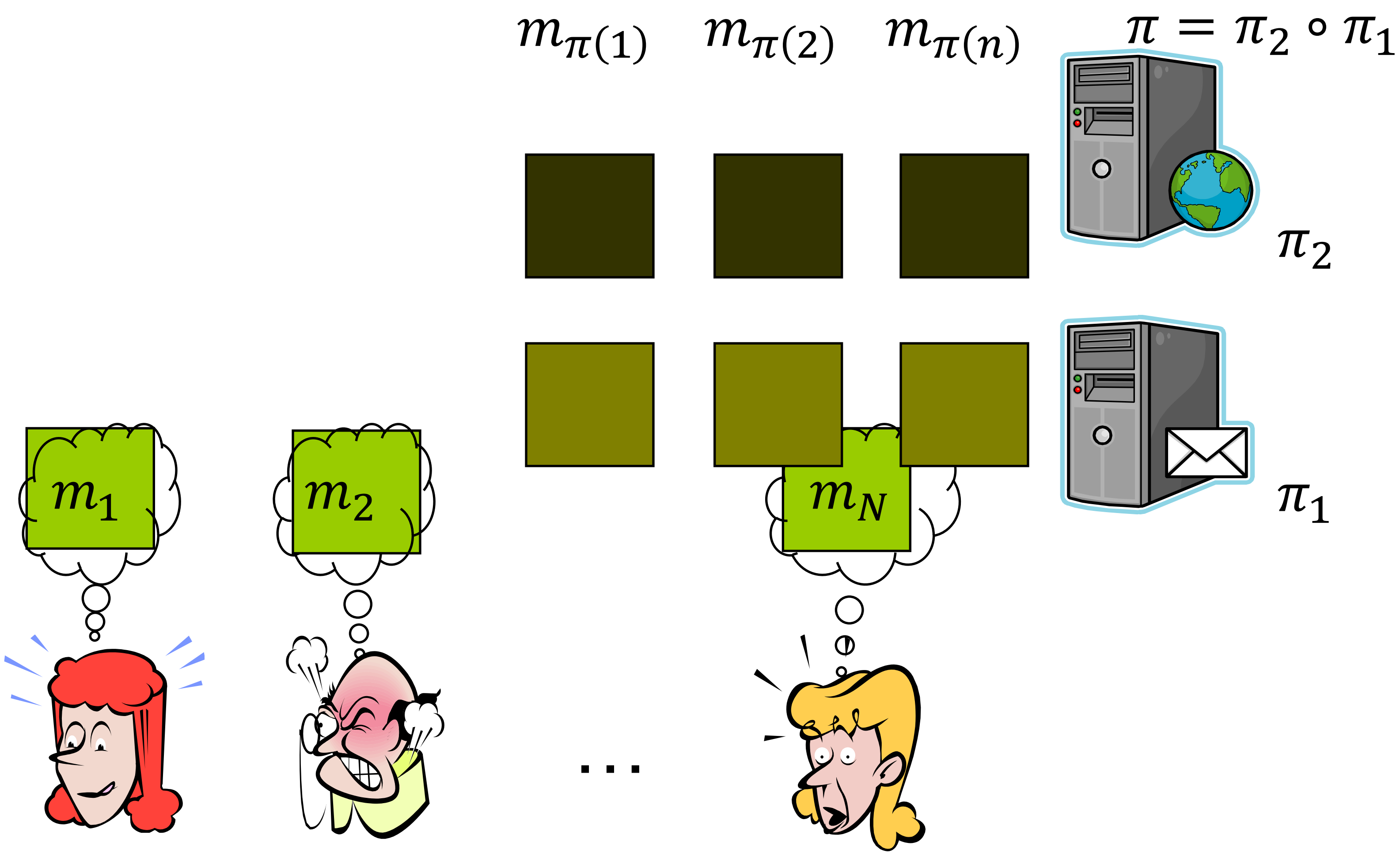
NIZK proof π for
valid vote inside



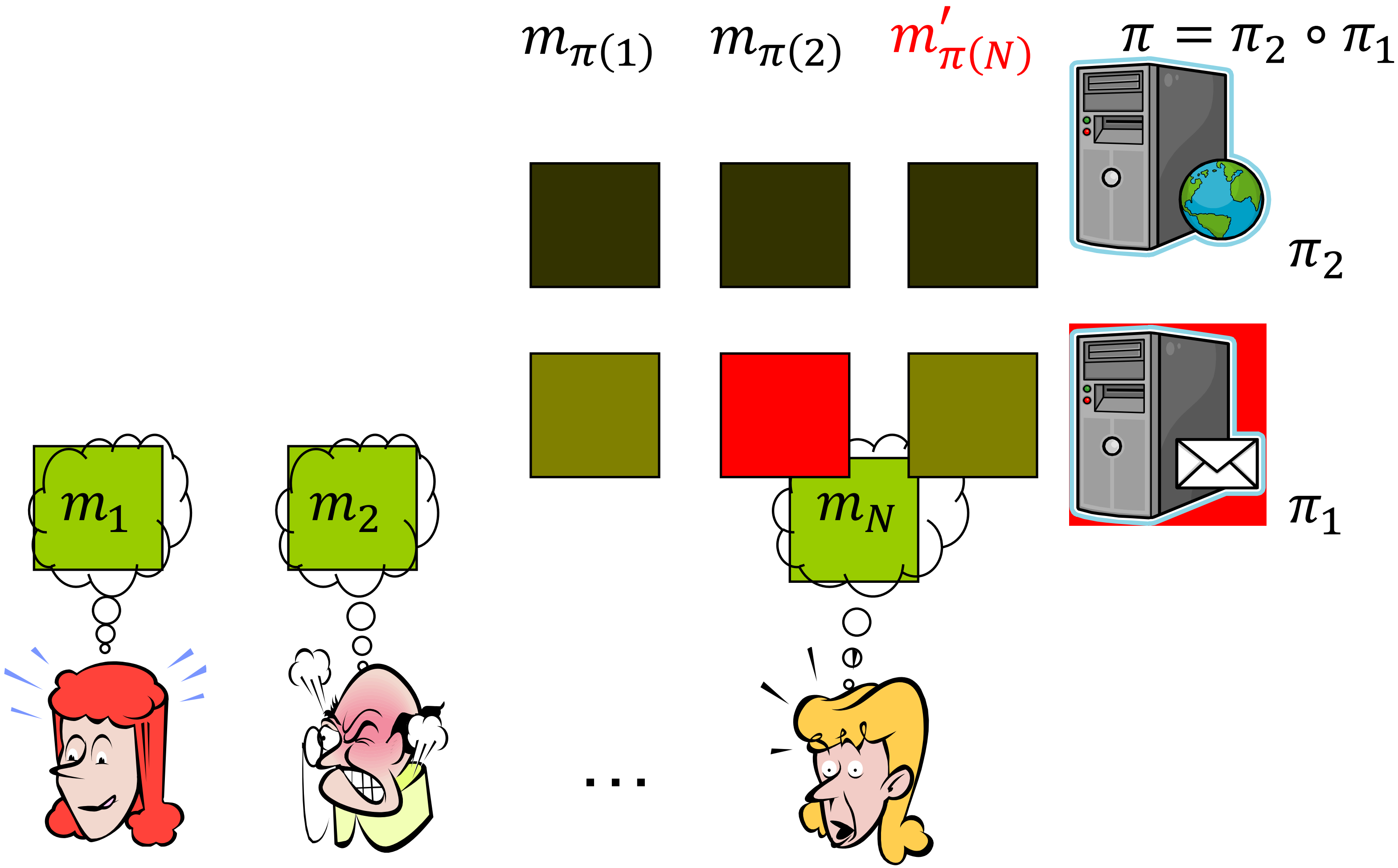
Voters

Election authorities

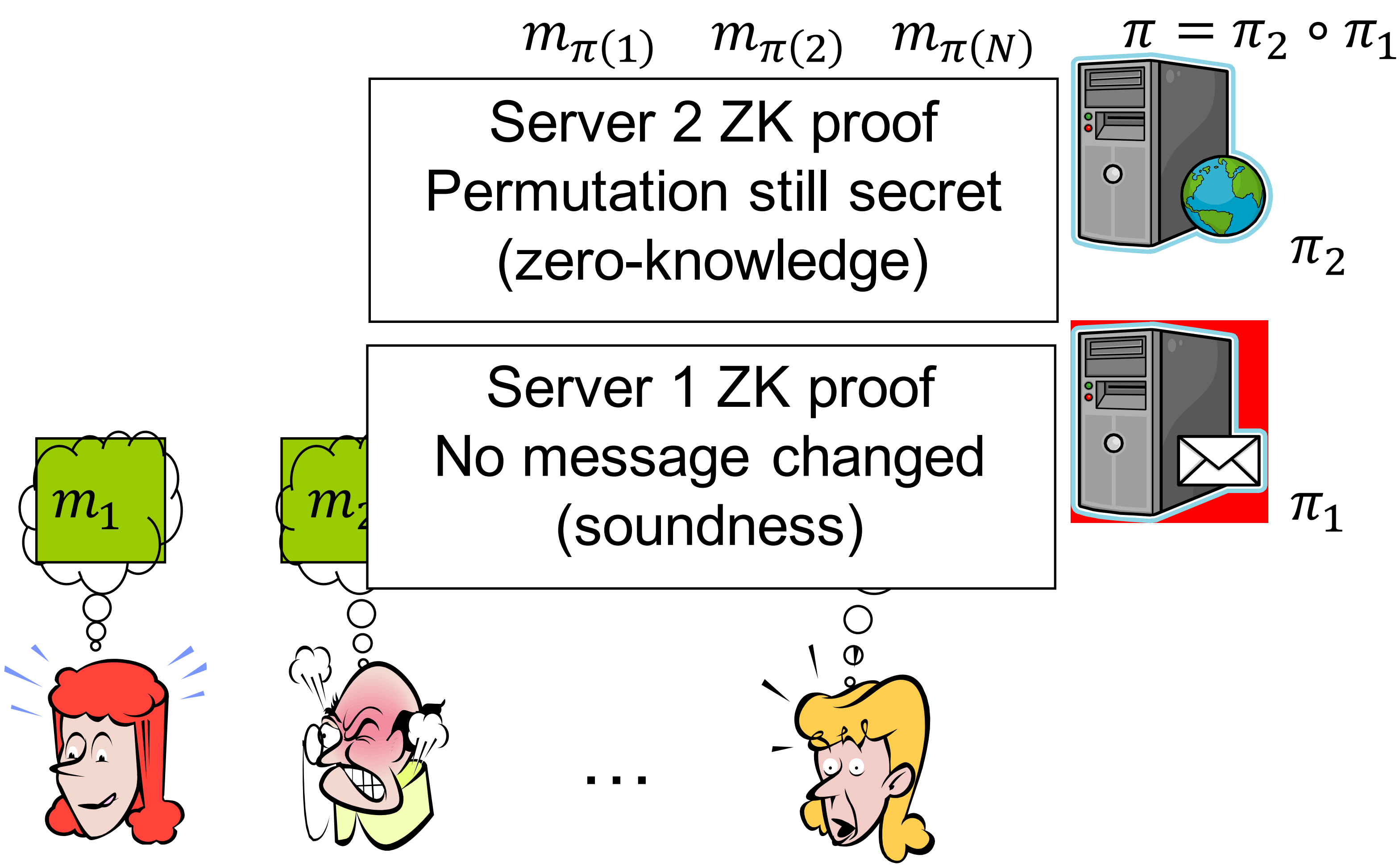
Mix-net: Anonymous message broadcast



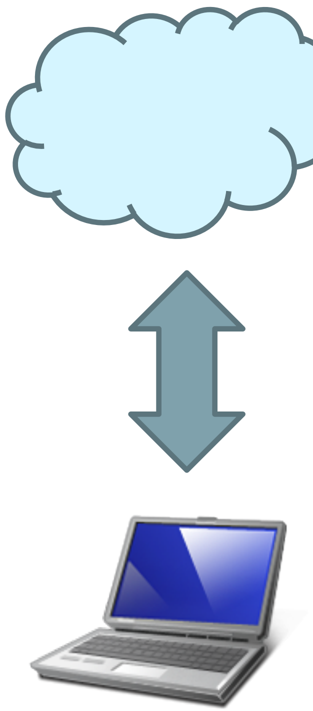
Problem: Corrupt mix-server



Solution: Zero-knowledge proof



Verifiable outsourced computation



- Client outsources computation to the cloud
- Gets back result based on its own data and cloud data
- Cloud gives zero-knowledge proof that result is correct

Ring and group signatures

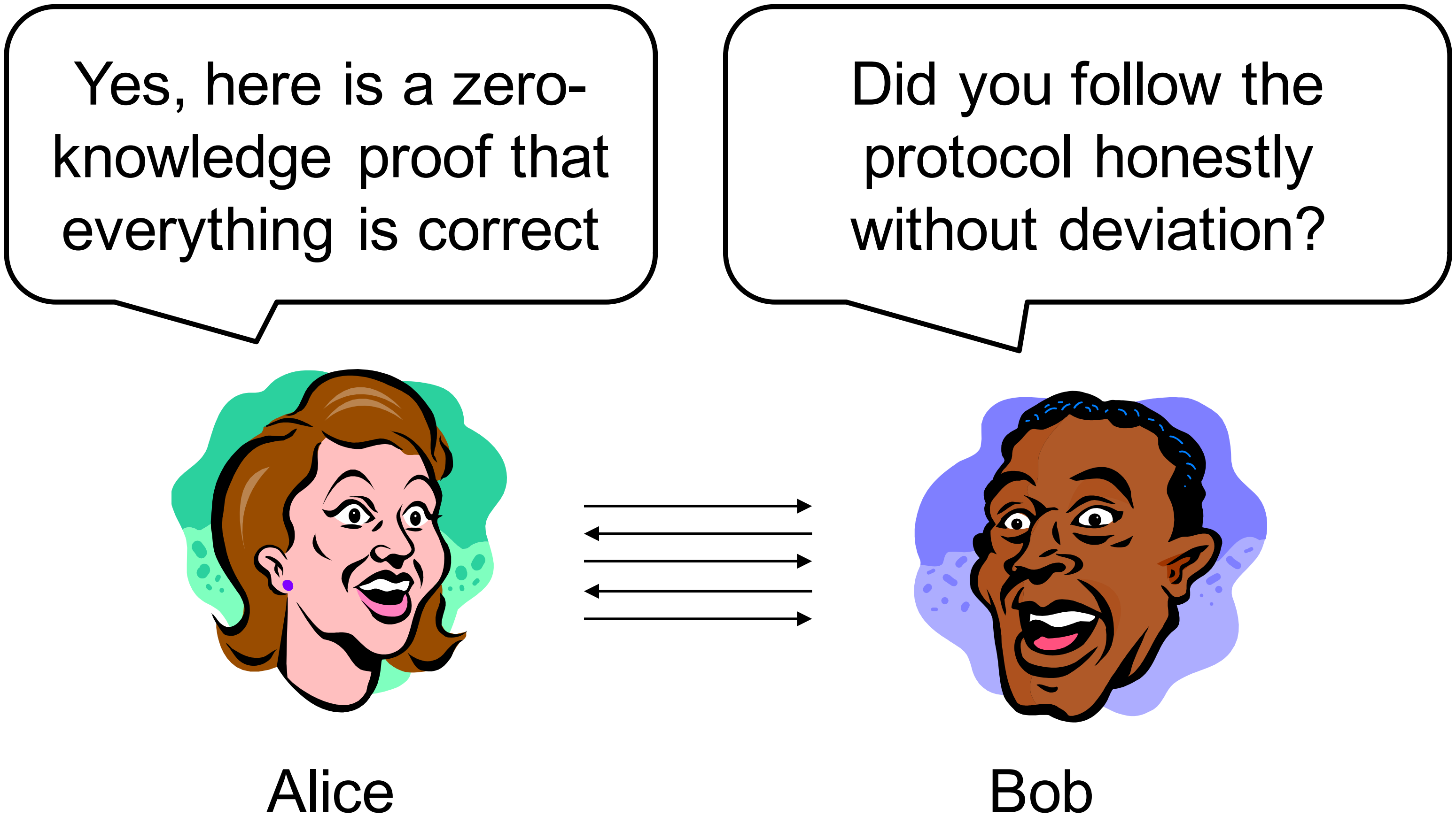
- Want to sign as member of group
- Anonymous within group
- Core techniques
  - NIZK proof that signer is member of group
  - Or NIZK proof that signer has signature certifying membership

Zerocoin

Coin spending
Reveal serial number

Anonymity
Each coin has unique secret serial number known only to owner
Use zero-knowledge proof to demonstrate one of the coins has revealed serial number

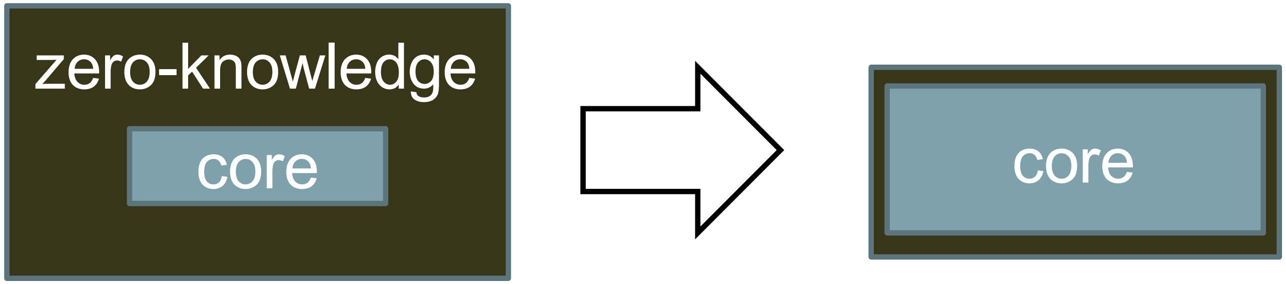
Preventing deviation (active attacks) by keeping people honest



From malicious adversary to honest but curious adversary

Vision

- Main goal
  - Efficient and versatile zero-knowledge proofs
- Vision
  - Negligible overhead from using zero-knowledge proofs

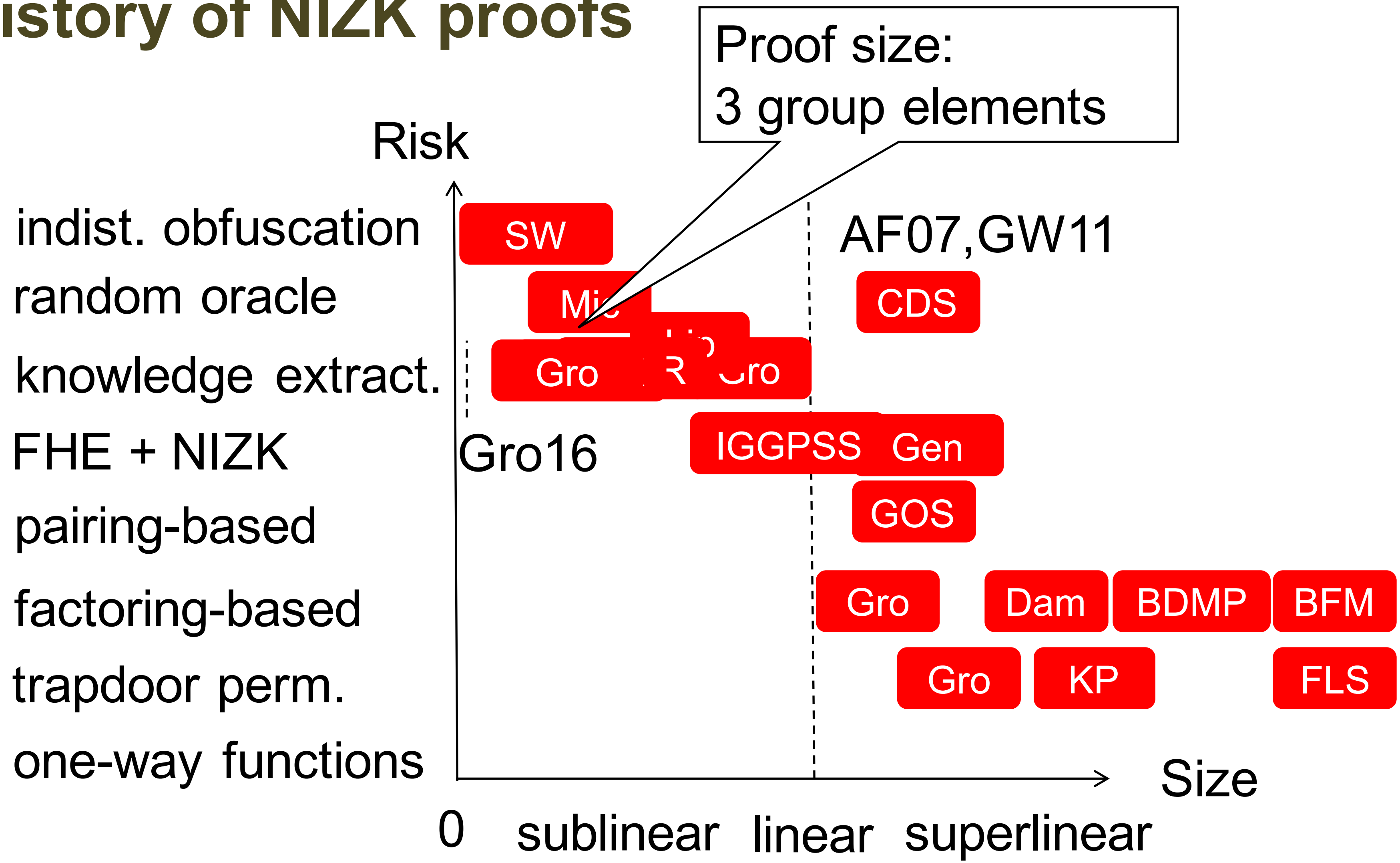


  - Security against active attacks standard feature

Performance parameters

- Prover’s computation
  - Time and memory
- Verifier’s computation
  - Time and memory
- Communication
  - Bits transmitted
  - Number of messages exchanged

History of NIZK proofs



Groth
EUROCRYPT 2016

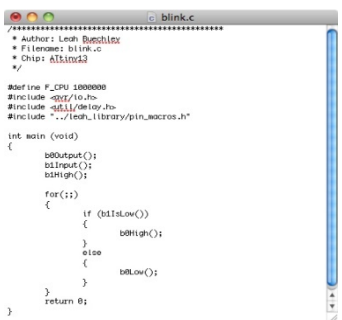
Rounds	Prover	Verifier	Communication
Non-interactive	N exponentiations	$\|\phi\|$ exponentiations	3 group elements

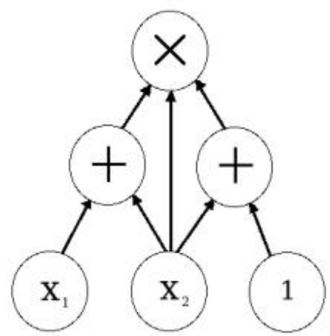
- Arithmetic circuit
  - N multiplication gates
  - $|\phi|$ public input wires
- NIZK argument
  - Perfect completeness
  - Perfect zero-knowledge
  - Computational soundness
    - Generic group model

zk-SNARK
Succinct Non-interactive
Argument of Knowledge

Verifiable computation zk-SNARKs

- Pinnocchio, Libsnark, Pantry, Buffet,…
- Prove program P with input x outputs y
  - Zero-knowledge useful if part of x is secret





$$\sum a_i u_i(x) \cdot \sum a_i v_i(x) \equiv \sum a_i w_i(x) + h(x)t(x)$$

π

Libsnark implementation

- 4x faster prover, 200B proofs

Prime order bilinear groups

- $\mathrm{Gen}(1^k)$ generates $(p, G_1, G_2, G_T, e, g, h)$
- G_1, G_2, G_T finite cyclic groups of prime order p generated by g, h and $e(g, h)$
- Bilinear map
  - $e(g^a, h^b) = e(g, h)^{ab}$
- Generic group operations efficiently computable

  Deciding group membership, group multiplications, pairing

Asymmetric bilinear groups (Type III): No efficiently computable isomorphism between G_1 and G_2

Additive notation

- Given bilinear group $(p, G_1, G_2, G_T, e, g, h)$ define
$$[a]_1 = g^a \qquad [b]_2 = h^b \qquad [c]_T = e(g, h)^c$$
and use additive notation for elements in brackets
- The generators can now be written $[1]_1, [1]_2, [1]_T$
- Define dot products using linear algebra notation
$$[\vec{a}]_* \cdot \vec{b} = \left[\vec{a} \cdot \vec{b}\right]_* \qquad [\vec{a}]_1 \cdot \left[\vec{b}\right]_2 = \left[\vec{a} \cdot \vec{b}\right]_T$$
- And for matrix multiplication
$$M[\vec{a}]_* = [M\vec{a}]_*$$

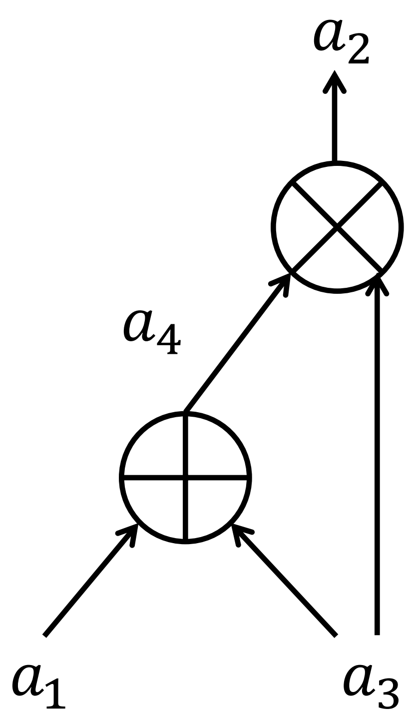
Pairing-based SNARK

- NP-relation R with statements ϕ and witnesses w
- Common reference string
  - Generate $(\vec{\sigma}_1, \vec{\sigma}_2, \tau) \leftarrow \mathrm{Setup}(R)$
  - Let common reference be $(R, [\vec{\sigma}_1]_1, [\vec{\sigma}_2]_2)$
- Proof
  - $(\Pi_1, \Pi_2) \leftarrow \mathrm{ProofMatrix}(R, \phi, w)$
  - $\pi = ([\vec{\pi}_1]_1, [\vec{\pi}_2]_2) = (\Pi_1[\vec{\sigma}_1]_1, \Pi_2[\vec{\sigma}_2]_2)$
- Verification
  - $(T_1, \ldots, T_\eta) \leftarrow \mathrm{Test}(R, \phi)$
  - Accept the proof π if and only if for all $T_1, \ldots, T_\eta$

$$\begin{bmatrix} \vec{\sigma}_1 \\ \vec{\pi}_1 \end{bmatrix}_1 \cdot T_i \begin{bmatrix} \vec{\sigma}_2 \\ \vec{\pi}_2 \end{bmatrix}_2 = [0]_T$$

Generic group operations

Arithmetic circuit



- Write as quadratic equation
$$(a_1 + a_3) \cdot a_3 = a_2$$
- In general arithmetic circuit can be written as a set of equations of the form
$$\sum a_i u_i \cdot \sum a_i v_i = \sum a_i w_i$$
over variables $a_1, \ldots, a_m$ and by convention $a_0 = 1$
- Arithmetic circuit defines an NP-language with statements $(a_1, \ldots, a_\ell)$ and witnesses $(a_{\ell+1}, \ldots, a_m)$

Rewriting the circuit as polynomial equations

- Consider an equation $\sum a_i u_i \cdot \sum a_i v_i = \sum a_i w_i$
- Let $u_i(x), v_i(x), w_i(x)$ be polynomials such that
$$u_i(r) = u_i \quad v_i(r) = v_i \quad w_i(r) = w_i$$
- Then equation satisfied if
$$\sum a_i u_i(x) \cdot \sum a_i v_i(x) \equiv \sum a_i w_i(x) \bmod (x - r)$$
- Pick degree $n - 1$ polynomials $u_i(x), v_i(x), w_i(x)$ such that this holds for all equations, using distinct $r_1, \ldots, r_n$ for the n equations in the circuit
- Values $a_0, \ldots, a_m$ satisfy all equations if
$$\sum a_i u_i(x) \cdot \sum a_i v_i(x) \equiv \sum a_i w_i(x) \bmod \prod (x - r_j)$$

Quadratic arithmetic program

- A quadratic arithmetic program over $\boldsymbol{Z}_p$ consists of polynomials $u_i(x), v_i(x), w_i(x), t(x) \in \boldsymbol{Z}_p[x]$
- It defines an NP-relation with
  - Statements $(a_1, \ldots, a_\ell)$
  - Witnesses $(a_{\ell+1}, \ldots, a_m)$
  - Satisfying (using $a_0 = 1$ to handle constants)

    $\sum a_i u_i(x) \cdot \sum a_i v_i(x) \equiv \sum a_i w_i(x) \mod t(x)$

Knowledge soundness

Generic group adversary

- Random encodings $[\cdot]_i : \boldsymbol{Z}_p \to G_i$
- Gets encodings $[\vec{\sigma}_1]_1, [\vec{\sigma}_2]_2$
- Oracle access to polynomially many group additions and pairings

Outline of proof we have soundness

- Generic group adversary must pick $(\phi, [A]_1, [C]_1, [B]_2)$ where $[A]_1, [C]_1$ are computed linearly from $[\vec{\sigma}_1]_1$ and $[B]_2$ from $[\vec{\sigma}_2]_2$
- We argue that generic adversary cannot learn non-trivial information about common reference string using generic group operations, so linear combinations chosen obliviously of $\vec{\sigma}_1, \vec{\sigma}_2$
- Careful analysis shows this choice is unlikely to satisfy verification equation

satisfies verification

$$[A]_1 \cdot [B]_2 = [\alpha]_1 \cdot [\beta]_2 + \sum_{i=0} a_i \left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right]_1 \cdot [\gamma]_2 + [C]_1 \cdot [\delta]_2$$

Efficiency

Efficiency gain
1. Generic group model
2. Carefully crafted verification equations

Arithmetic circuits	Proof size	Prover	Verifier	Equations
[PGHR13] (symmetric)	$8\ G$	$7m + n\ E$	$\ell\ E, 11\ P$	5
This work (symmetric)	$3\ G$	$m + 3n\ E$	$\ell\ E, 3\ P$	1
[BCTV14]	$7\ G_1, 1\ G_2$	$6m + n\ E_1, m\ E_2$	$\ell\ E_1\ , 12\ P$	5
This work	$2\ G_1, 1\ G_2$	$m + 3n\ E_1, n\ E_2$	$\ell\ E_1, 3\ P$	1
Boolean circuits				
[DFGK14]	$3\ G_1, 1\ G_2$	$m + n\ E_1$	$\ell\ M_1\ , 6\ P$	3
This work	$2\ G_1, 1\ G_2$	$n\ E_1$	$\ell\ M_1\ , 3\ P$	1

Circuits with m wires, n gates, statement size ℓ $(\ell \ll n < m)$
Group element G, exponentiation E, pairing P, multiplication M

Thanks

- Questions?