Introduction to Zero-Knowledge Proofs and NIZK Arguments

Jens Groth

DFINITY

- contributor to the Internet Computer blockchain
- www.internetcomputer.org, which runs
- decentralized apps on-chain at web speed

Agenda

- Why zero-knowledge proofs are useful
- Definitions: setup, statements, security
- Sigma-protocols: a common type of zk-proofs
- Efficiency: effortless verification of complex statements!

Privacy and verifiability

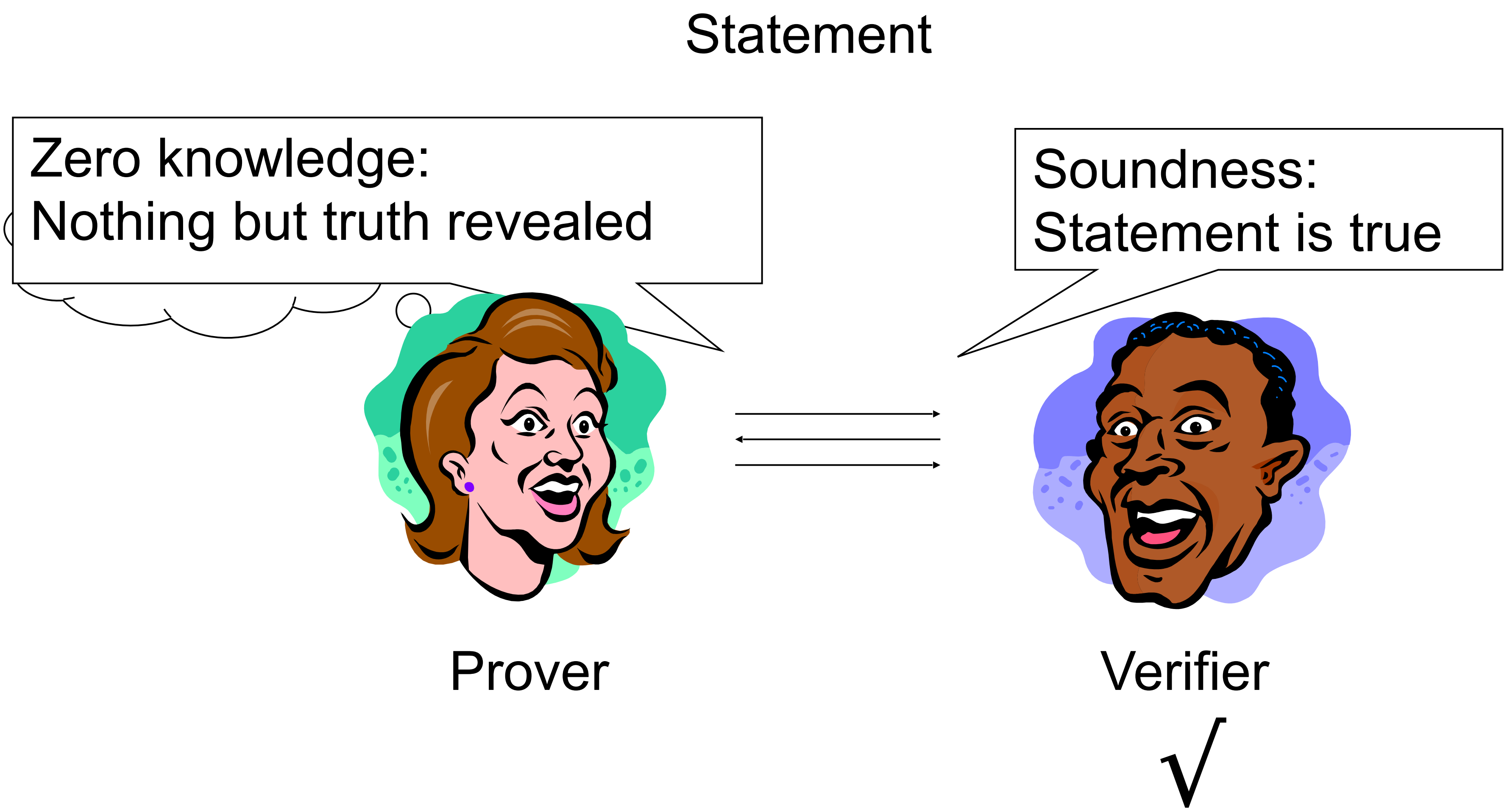
No!
It is a trade secret.

Did I lose all my money? Show me the current portfolio!

Hedge fund

Investor

Zero-knowledge proof system



Zero-knowledge proofs

- Completeness
  - Prover can convince verifier when statement is true
- Soundness
  - Cannot convince verifier when statement is false
- Zero knowledge
  - No leakage of information (except truth of statement) even if interacting with a cheating verifier

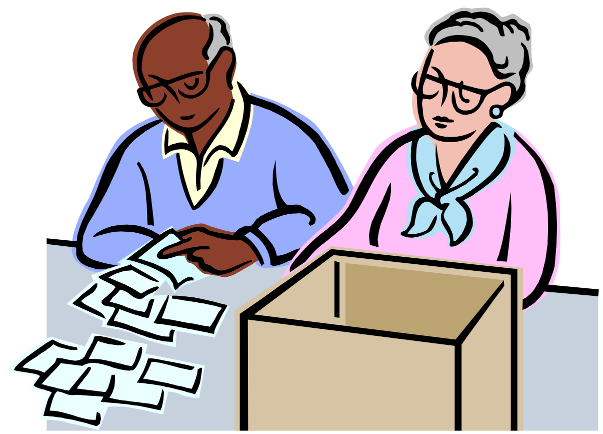
Internet voting

Vote

Encrypts vote to keep it private

Tally without decrypting individual votes

Ciphertext



Voter

Election authorities

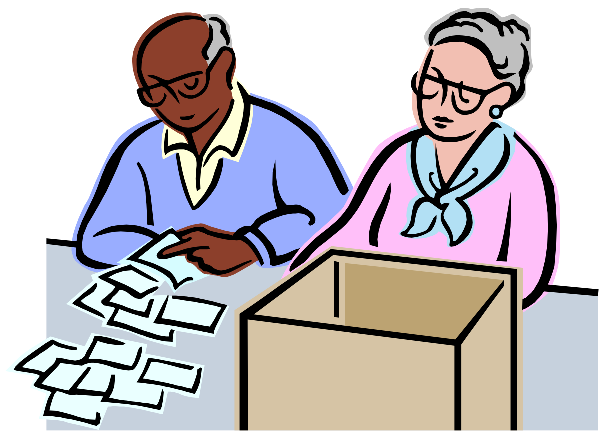
Election fraud

Not Bob

Encrypts -100 votes for Bob

Is the encrypted vote valid?

Ciphertext



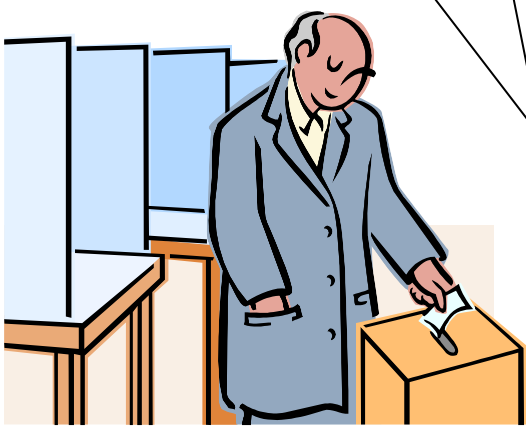
Voter

Election authorities

Zero-knowledge proof to prevent cheating

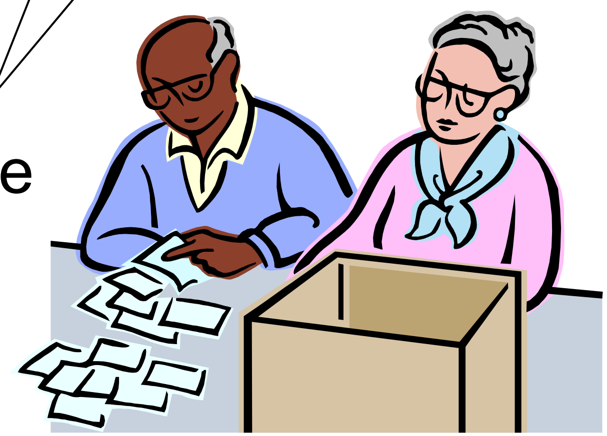
Zero knowledge:
Vote is secret

Soundness:
Vote is valid



Ciphertext

Zero-knowledge
proof for valid
vote encrypted



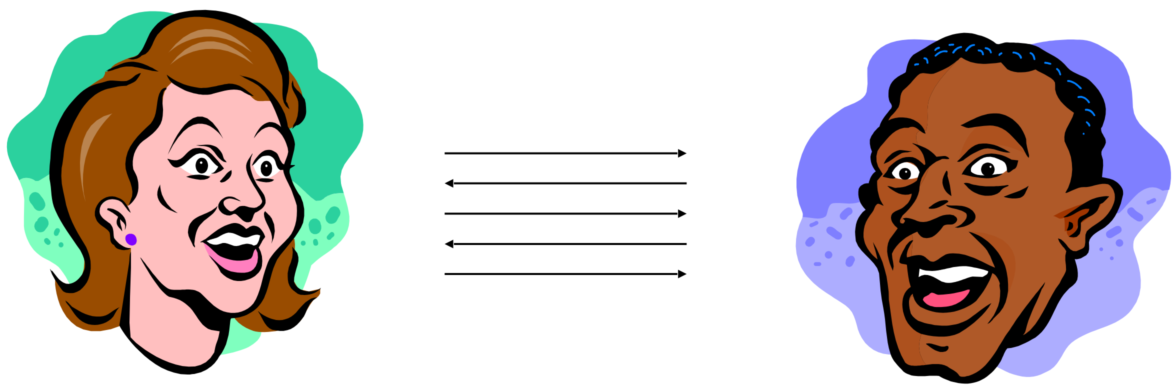
Voter

Election authorities

Preventing deviation (active attacks) by keeping participants honest

Yes, here is a zero-knowledge proof that everything is correct

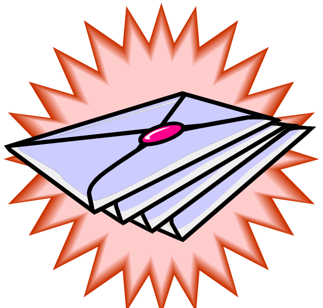
Did you follow the protocol honestly without deviation?

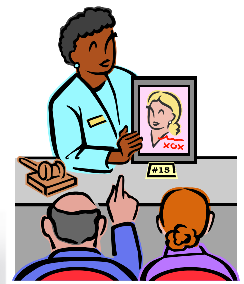


Alice

Bob

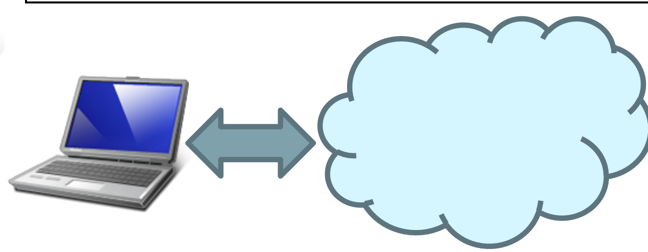
Zero-knowledge proofs ensure compliance





Problems typically arise when attackers deviate from the protocol (active attack)

Zero-knowledge proofs prevent deviation and give security against active attacks



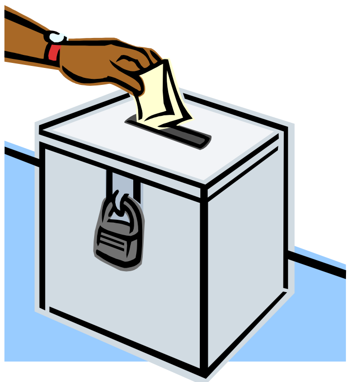
P and NP

- A language L is a set of **instances**
- Languages in P can be efficiently decided
  - Definition: L is in NP if there is a polynomial time decision procedure to tell for any instance ϕ whether the **statement** $\phi \in L$ is true or false
  - Examples
    - Instance: n, p, q
      Statement: $n = pq$
    - Instance: fast program P, input, output
      Statement: P(input) = output
- Languages in NP can be efficiently decided *with advice (a witness)*
  - Definition: L is in NP if there is an polynomial time decidable relation $R_L = \{(\phi, w)\}$ such that $\phi \in L$ if and only if there exists a **witness** w such that $(\phi, w) \in R_L$
  - Examples
    - Instance: n
      Statement: there are primes p, q such that $n = pq$
      Witness: p, q
    - Instance: fast program P, output
      Statement: these exists input such that P(input) = output
      Witness: input

P vs NP

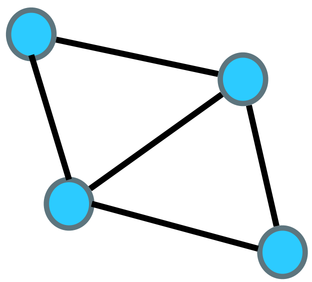
- Every language in P is also in NP
    - With relation $R_L = \{(\phi, w) \,|\, \phi \in L, w = \text{empty}\}$
- Many believe P $\neq$ NP but we do not know
    - Millenium prize problem
- There are *NP-complete* languages L'
    - For any P-language L the statement $\phi \in L$ can be reduced to an equivalent statement $\phi' \in L'$ in polynomial time
        - Efficient reduction $\phi \mapsto \phi'$
          such that $\phi' \in L'$ if and only if $\phi \in L$
    - For most well-known examples of NP-complete languages we have efficient translations to corresponding witnesses
        - Efficient reduction $(\phi, w) \mapsto w'$
          such that $(\phi, w) \in R_L$ if and only if $(\phi', w') \in R_{L'}$
- If you have a zero-knowledge proof system for an NP-complete language, then you have zero-knowledge proofs for all languages in NP

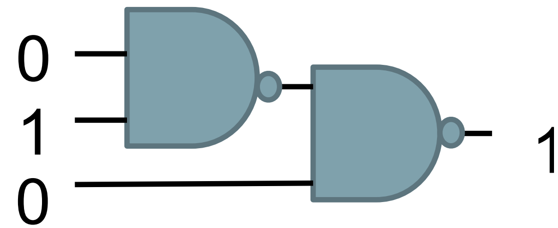
Statements


Encrypted valid vote

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

SAT


Hamiltonian path exists



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 prover wants to keep the witness secret!

Setup

- The prover and verifier operate in a context; they may have a predefined **setup**
  - Examples
    - A prime order group $\mathbb{G}$ where it is hard to compute discrete logarithms and one or more generators g, h for the group
    - A description of a hash function Hash : $\{0,1\}^* \rightarrow \{0,1\}^k$
    - A **common reference string** (CRS) from a trusted source

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**

      - Uniformly random reference string
      - Structured reference string
    - Nothing or just a security parameter λ to indicate desired security level

Statements

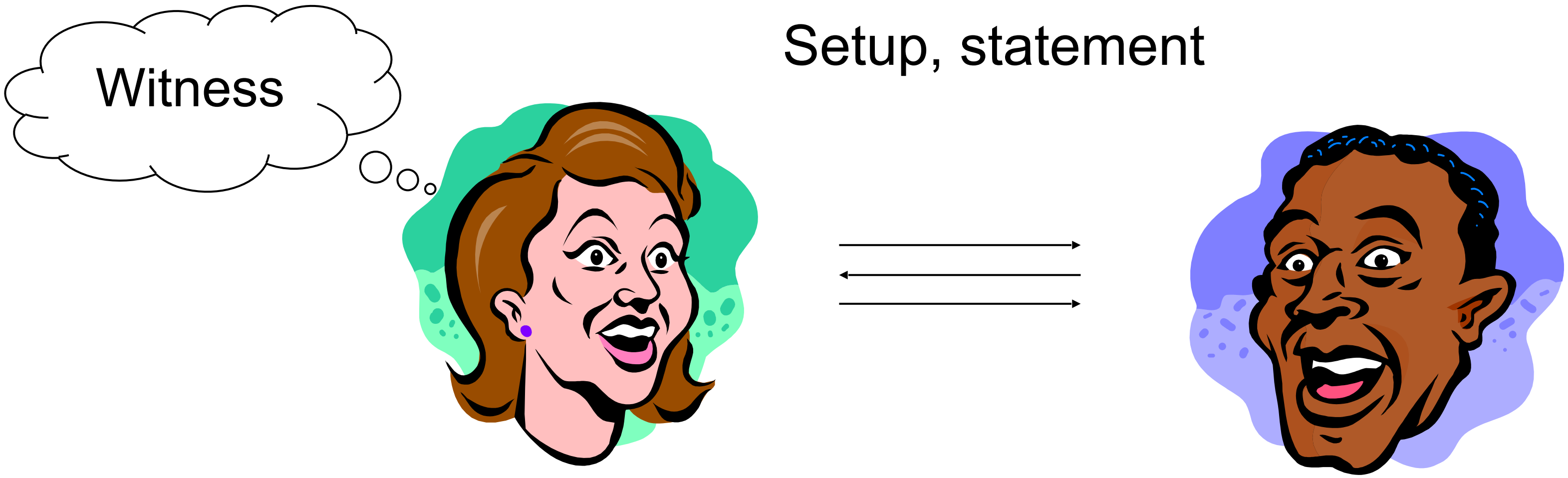
Standard binary NP relation is just the special case where R ignores σ

- We consider efficiently decidable ternary relations R containing triples (σ, ϕ, w)
  - Setup σ
  - Instance ϕ
  - Witness w
- A statement is specified by a relation R, a setup σ and an instance ϕ, and claims there exists a witness w such that $(\sigma, \phi, w) \in R$
  - We write such a statement as $\phi \in L_\sigma$

Syntax

- A proof system for a relation R consists of three probabilistic, stateful algorithms
- $\mathrm{Setup}(\lambda) \to \sigma$
  - Given security parameter λ generate setup σ
- $\left\langle \mathrm{Prove}\left(\sigma, \phi, w\right); \mathrm{Verify}\left(\sigma, \phi\right) \right\rangle \to \mathrm{accept/reject}$
  - Prover and verifier interact and after a number of rounds stop with the verifier outputting a decision

Completeness



- **Perfect completeness**: an honest prover always convinces an honest verifier if the setup is honestly generated and the statement is true
  - $\Pr[\text{Setup}(\lambda) \to \sigma : \text{for all } (\phi, w) \in R_\sigma \ \langle \text{Prove}(\sigma, \phi, w); \text{Verify}(\sigma, \phi) \rangle \to \text{accept}] = 1$
- **Statistical completeness**: there is overwhelming probability (even for a worst-case true statement made by an unbounded adversary) an honest prover will convince an honest verifier
  - $\Pr[\text{Setup}(\lambda) \to \sigma; \text{Adversary}(\sigma) \to (\phi, w) \in R_\sigma : \langle \text{Prove}(\sigma, \phi, w); \text{Verify}(\sigma, \phi) \rangle \to \text{accept}] \approx 1$
- **Computational completeness**: there may exist worst case true statements where an honest prover fails to reliably convince an honest verifier but they're hard to find (bounded adversary)
  - $\Pr[\text{Setup}(\lambda) \to \sigma; \text{Adversary}(\sigma) \to (\phi, w) \in R_\sigma : \langle \text{Prove}(\sigma, \phi, w); \text{Verify}(\sigma, \phi) \rangle \to \text{accept}] \approx 1$

Soundness

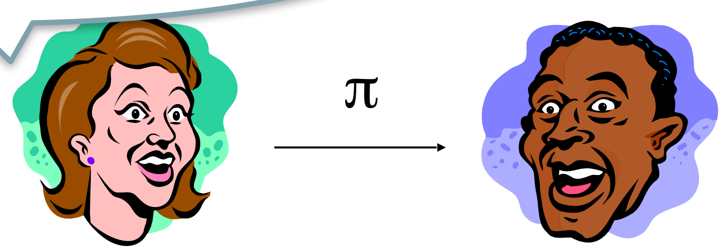
Sometimes people distinguish
Proof = perfect or statistical soundness
Argument = computational soundness

Setup, statement

- **Perfect soundness**: a cheating prover never convinces an honest verifier of a false statement (if the setup is honestly generated)
  - $\Pr[\text{Setup}(\lambda) \rightarrow \sigma : \text{for all } \phi \notin L_\sigma \ \langle \text{Adversary}(\sigma, \phi); \text{Verify}(\sigma, \phi) \rangle \rightarrow \text{accept}] = 0$
- **Statistical completeness**: a cheating prover is unlikely to convince an honest verifier (even with infinite computing power)
  - $\Pr[\text{Setup}(\lambda) \rightarrow \sigma : \text{for all } \phi \notin L_\sigma \ \langle \text{Adversary}(\sigma, \phi); \text{Verify}(\sigma, \phi) \rangle \rightarrow \text{accept}] \approx 0$
- **Computational completeness**: a computationally bounded cheating prover is unlikely to fool an honest verifier
  - $\Pr[\text{Setup}(\lambda) \rightarrow \sigma; \text{Adversary}(\sigma) \rightarrow \phi \notin L_\sigma : \langle \text{Adversary}; \text{Verify}(\sigma, \phi) \rangle \rightarrow \text{accept}] \approx 0$

Proofs of knowledge

- Informal definition: the prover "knows" a witness
  - Recall the earlier example
    - Instance: n
    - Statement: there exists primes p, q such that $n = pq$
  - A proof of membership just demonstrates n has two prime factors
    - Maybe nobody knows them, maybe easy to determine number of factors
  - **Knowledge soundness**: a proof of knowledge demonstrates we could extract p and q from the prover and write them down



Zero knowledge

- Zero knowledge:
  - The proof only reveals the statement is true, it does not reveal anything else
- Defined by simulation:
  - The verifier could have simulated the proof without knowing the prover’s witness

Zero knowledge

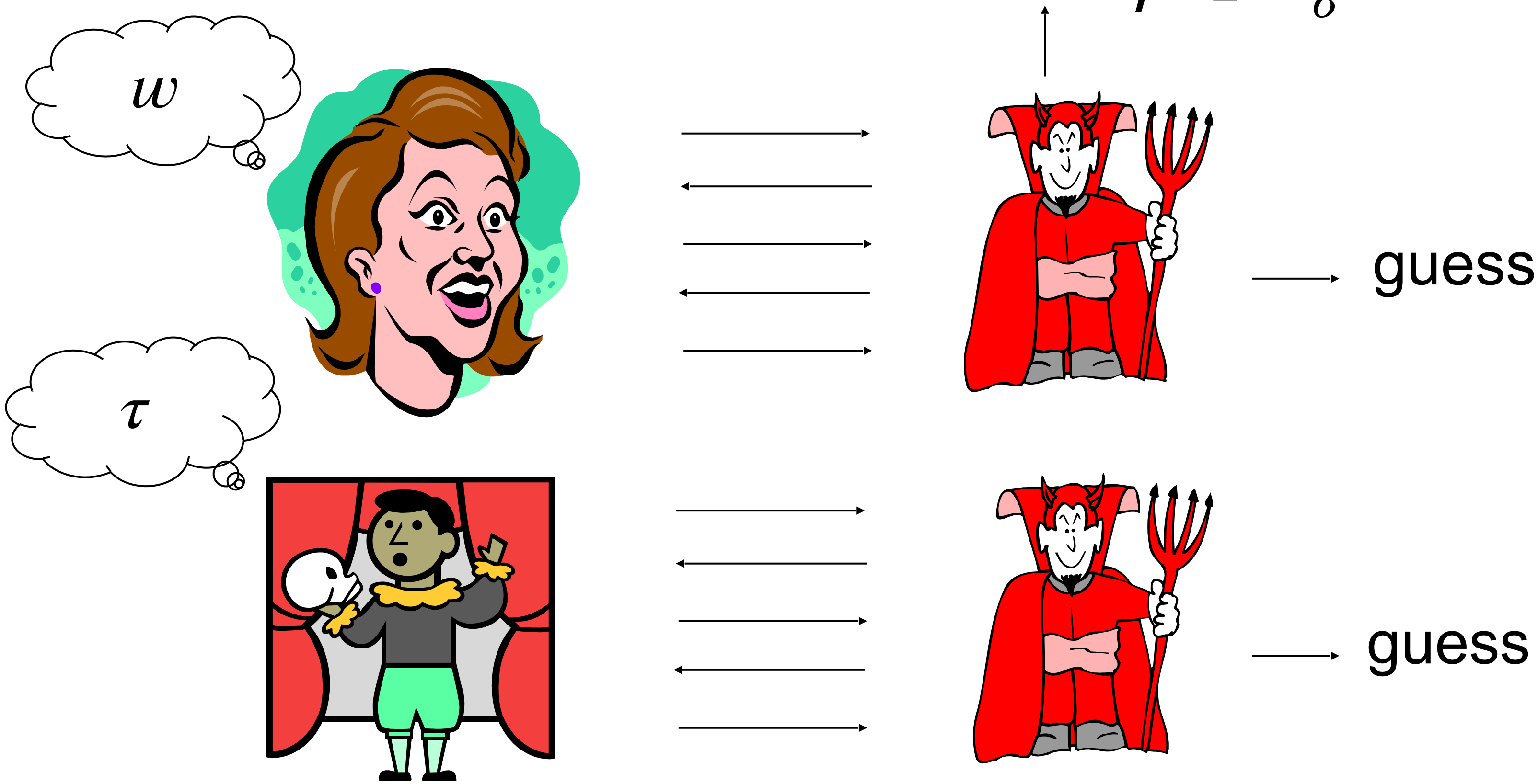
The simulator needs some extra power to simulate. If you could simulate without extra power, you could break soundness!

- Many variations of simulation, here's an example of how to define a simulator
- $\mathrm{SimSetup}(\lambda) \rightarrow (\sigma, \tau)$
  - Given security parameter, returns simulated setup together with a simulation trapdoor
- $\left\langle \mathrm{SimProve}(\sigma, \phi, \tau); \mathrm{Verify}(\sigma, \phi) \right\rangle \rightarrow \mathrm{accept/reject}$
  - Interactive algorithm SimProve does not know the witness, but instead uses knowledge of trapdoor to simulate the interaction
  - Cannot leak information about witness

Zero knowledge

Real/simulated setup σ

True statement $\phi \in L_\sigma$



Indistinguishability between real proofs and simulated proofs

$\Pr[\text{Real proof} : \text{guess} = \text{real}] \approx \Pr[\text{Simulation} : \text{guess} = \text{real}]$

Zero knowledge

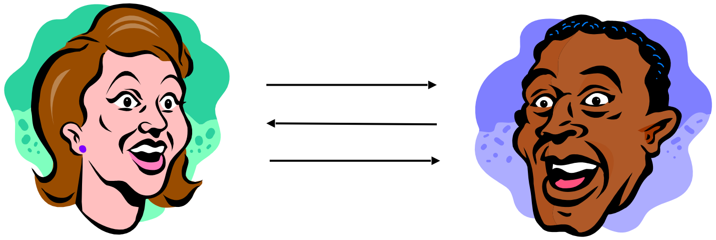
- Zero knowledge can be
  - **Perfect**: the simulation is identical to a real proof (even for a worst case true statement)
  - **Statistical**: the simulation is hard to tell apart from a real proof (even to an evil verifier with unlimited computational resources)
  - **Computational**: the simulation is hard to tell apart from a real proof for a computationally bounded adversary
- Other useful but weaker definitions
  - **Witness hiding**: the proof does not help you to compute the witness
  - **Witness indistinguishability**: the proof does not help you tell which out of several possible witnesses the prover has
  - **Honest verifier zero knowledge (HVZK)**: if the verifier creates honest challenges according to the proof system, the proof leaks nothing about the witness

Performance parameters

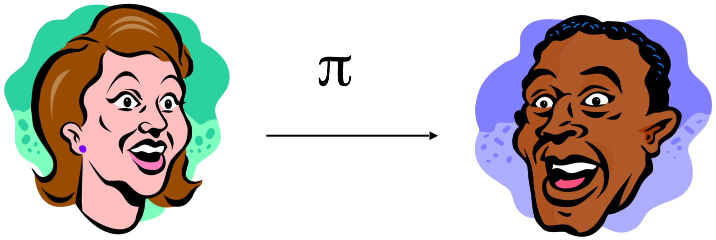
- Most common measures of efficiency
  - Communication (bits)
  - Prover's computation (seconds)
  - Verifier's computation (seconds)
  - Round complexity (number of messages)
- Depending on use case other measures may be important
  - Size of the setup if using a common reference string
  - Memory consumption
  - Parallelisation
  - Energy consumption

Round complexity

- Interactive zero-knowledge proof (ZK proof)

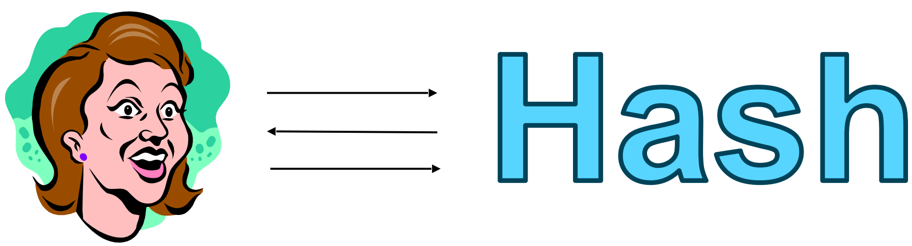


- Non-interactive zero-knowledge proof (NIZK proof)



Fiat-Shamir heuristic

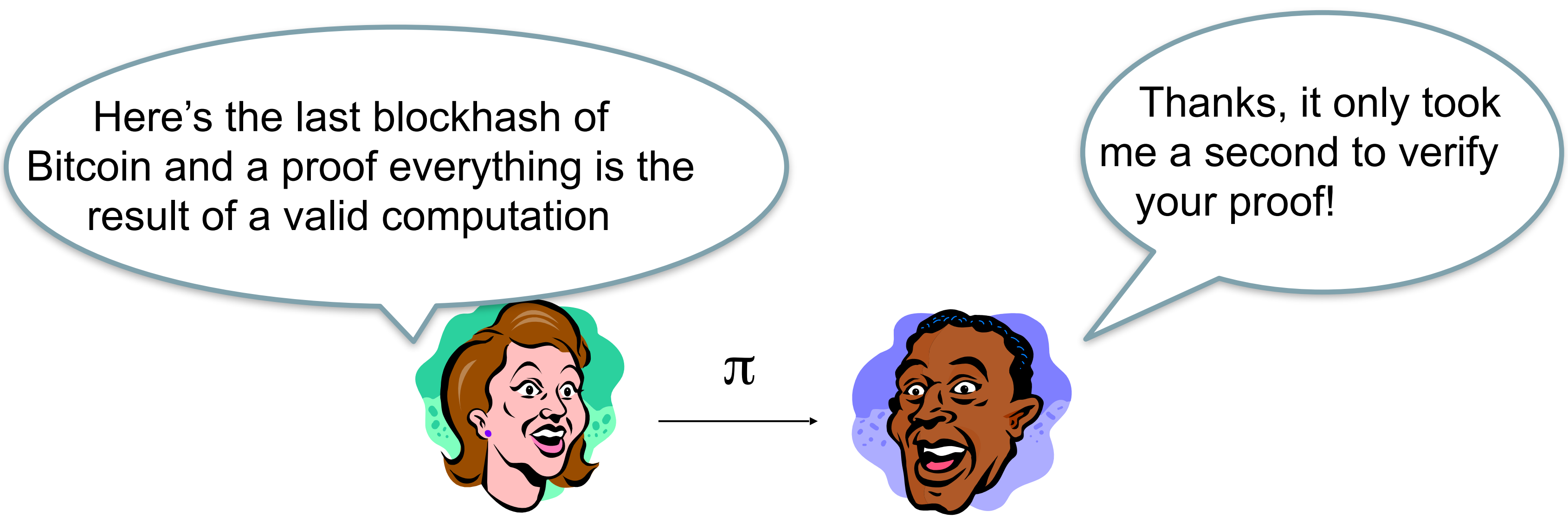
- An interactive proof system is **public coin** if the verifier only sends uniformly random challenges
- If these challenges are big enough (e.g. random 256-bit strings) then one can use the Fiat-Shamir heuristic to make the proof system non-interactive



- Prover computes transcript where the verifier's challenges are replaced by message digests

Ultimate performance: zk-SNARK

- Succinct non-interactive argument of knowledge (SNARK)
  - Proof system with succinct proofs (a few kbits)
  - Verification may be very efficient
  - Useful for languages with complex statements requiring a lot of computation to verify



Succinctness

- SNARKs are also interesting for languages in P
  - Even if the verifier could decide the statement in polynomial time it may be cheaper to verify a succinct proof
  - For such use cases you only need completeness and soundness
- People mix the terms SNARK and zk-SNARK
  - Use zk-SNARK if you really care about zero knowledge
  - Use SNARK if you do not care about zero knowledge
- Rule of thumb for research
  - A succinct proof is small and cannot leak much information
  - Usually it is hard to get soundness
  - Usually it is easy to to get or to add zero knowledge

Sigma-protocols

Statement $\phi \in L$

Witness w

$(\phi, w) \in R_L$

$\xrightarrow{\quad a \quad}$

$x \leftarrow \text{Random}$

$\xleftarrow{\qquad\quad}$

$\xrightarrow{\quad z \quad}$

$\text{Verify}(\phi, a, x, z) \rightarrow \text{accept/reject}$

Setup: prime order groups

- Let p be a prime and $\mathbb{Z}_p$ the integers modulo p
- Let $\mathbb{G}$ be a cyclic group of size p
- Let g be a group element in $\mathbb{G}$
  - For all $a, b \in \mathbb{Z}_p : g^a \cdot g^b = g^{a+b}$
  - For all $a, b \in \mathbb{Z}_p : (g^a)^b = g^{ab}$
- The discrete logarithm problem is given g, g^α to find α
- The DDH problem is given g, h, g^α, h^β to guess if $\alpha = \beta$

Sigma-protocol for DDH tuples

Instance $g, h, u, v \in \mathbb{G}, g \neq 1, h \neq 1$

Witness α

$u = g^{\alpha}, v = h^{\alpha}$

$r \leftarrow \mathbb{Z}_p$

$a = g^r$

$b = h^r$

$\xrightarrow{a, b}$

$\xleftarrow{x \leftarrow \mathbb{Z}_p}$

$\xrightarrow{z}$

$z = \alpha x + r \pmod{p}$

Accept if and only if

$u^x a = g^z$ and $v^x b = h^z$

Perfect completeness

Instance $g, h, u, v \in \mathbb{G}, g \neq 1, h \neq 1$

Witness α

$u = g^{\alpha}, v = h^{\alpha}$

$r \leftarrow \mathbb{Z}_p$

$a = g^r$

$b = h^r$

$\xrightarrow{a, b}$

$\xleftarrow{x \leftarrow \mathbb{Z}_p}$

$z = \alpha x + r \pmod{p}$

$\xrightarrow{z}$

Accept if and only if

$u^x a = g^z$ and $v^x b = h^z$

$$\boxed{\begin{aligned} u^x a &= (g^{\alpha})^x g^r = g^{\alpha x + r} = g^z \\ v^x b &= (h^{\alpha})^x h^r = h^{\alpha x + r} = h^z \end{aligned}}$$

Perfect honest verifier zero knowledge

Instance $g, h, u, v \in \mathbb{G}, g \neq 1, h \neq 1$

Simulation

$x \leftarrow \mathbb{Z}_p$

$z \leftarrow \mathbb{Z}_p$

$a \leftarrow g^z u^{-x}$

$b \leftarrow h^z v^{-x}$

$\xrightarrow{a, b}$

$\xleftarrow{x}$

$\xrightarrow{z}$

Accept if and only if

$u^x a = g^z$ and $v^x b = h^z$

Both in simulation and in real proof x, z are random and uniquely define a, b, so simulated proof looks like real proof

Statistical soundness

Instance $g, h, u, v \in \mathbb{G}, g \neq 1, h \neq 1$

$$\xrightarrow{a, b}$$

$$\xleftarrow{x \leftarrow \mathbb{Z}_p}$$

$$\xrightarrow{z}$$

$u^x a = g^z$ and $v^x b = h^z$

False statement with $u = g^\alpha, v = h^\beta, \alpha \neq \beta$

We have for some r, s that $a = g^r, b = h^s$

Now the prover gets a random challenge x

Since $u^x a = g^z$ the prover needs $z = \alpha x + r$

Since $v^x b = g^z$ the prover needs $z = \beta x + s$

Unlikely random x hits the intersection of two distinct lines

Sigma-protocol for discrete logarithm

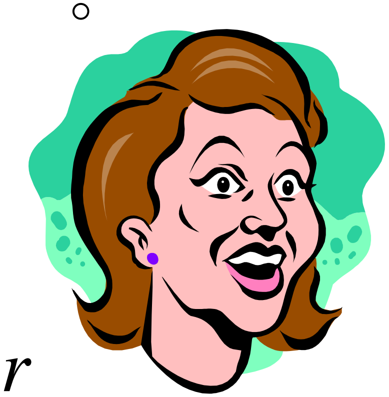
Instance $g, u \in \mathbb{G}, g \neq 1$

Witness α

$u = g^{\alpha}$

$r \leftarrow \mathbb{Z}_p$

$a = g^{r}$



$\xrightarrow{\quad a \quad}$

$\xleftarrow{\quad x \leftarrow \mathbb{Z}_p \quad}$

$\xrightarrow{\quad z \quad}$

$z = \alpha x + r \pmod{p}$

Accept if $u^{x} a = g^{z}$

Exercise

- Verify the Sigma protocol for discrete logarithm is (perfect) complete
- Show the Sigma protocol for discrete logarithm is (perfect) honest verifier zero-knowledge
- Show the Sigma protocol for discrete logarithm has (statistical) knowledge soundness
  - Hint: Imagine the prover after sending a is able to answer two random challenges x, x' with correct z, z'

Perfect completeness

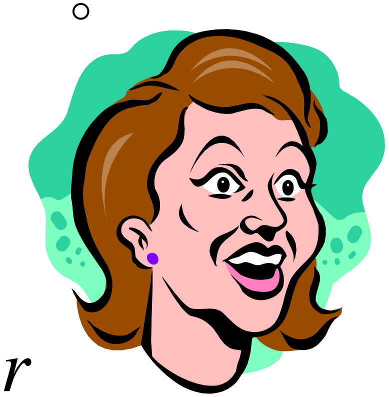
Instance $g, u \in \mathbb{G}, g \neq 1$

Witness α

$u = g^{\alpha}$

$r \leftarrow \mathbb{Z}_p$

$a = g^r$



$$\xrightarrow{\quad a \quad}$$

$$\xleftarrow{\quad x \leftarrow \mathbb{Z}_p \quad}$$

$$\xrightarrow{\quad z \quad}$$

Accept if $u^x a = g^z$

$z = \alpha x + r \pmod{p}$

$$\boxed{u^x a = (g^{\alpha})^x g^r = g^{\alpha x + r} = g^z}$$

Perfect honest verifier zero knowledge

Instance $g, u \in \mathbb{G}, g \neq 1$

Simulation

$x \leftarrow \mathbb{Z}_p$
$z \leftarrow \mathbb{Z}_p$
$a \leftarrow g^z u^{-x}$

$\xrightarrow{a}$

$\xleftarrow{x}$

$\xrightarrow{z}$

Accept if $u^x a = g^z$

Both in simulation and in real proof x, z are random and uniquely define a so simulated proof looks like real proof

Perfect soundness!?

Instance $g, u \in \mathbb{G}, g \neq 1$

a $\longrightarrow$

$x \leftarrow \mathbb{Z}_p$ $\longleftarrow$

z $\longrightarrow$

Accept if $u^x a = g^z$

Since g is a generator for $\mathbb{G}$ it is clear that $u = g^\alpha$ for some $\alpha \in \mathbb{Z}_p$. So the prover cannot cheat!

Actually, the Sigma protocol for membership in the language is silly, you did not need a Sigma protocol proof to tell the verifier that a discrete logarithm exists

Special soundness

If the prover can answer two distinct challenges then possible to efficiently compute witness

$$\xrightarrow{\quad a \quad}$$

$$\xleftarrow{\quad x \leftarrow S \quad} \qquad \xleftarrow{\quad x' \leftarrow S \quad}$$

$$\xrightarrow{\quad z \quad} \qquad \xrightarrow{\quad z' \quad}$$

$$\text{Extract}(\phi, x, x, x', z') \rightarrow w$$

Strategy: clone the prover's state after having sent a and try many challenges.

The prover "knows" w because we can extract the witness from this state.

This is also known as **rewinding** - run the proof, rewind back to a previous state, run again…

Statistical knowledge soundness

Instance $g, u \in \mathbb{G}, g \neq 1$

$$a \longrightarrow$$

$$x \leftarrow \mathbb{Z}_p \;\longleftarrow$$

$$z \longrightarrow$$

Accept if $u^x a = g^z$

Suppose the prover has probability ε of convincing the verifier after having sent a. If ε is negligibly small, the verifier is unlikely to accept. If ε is significant, we can rewind and try many $x \leftarrow \mathbb{Z}_p$ until we have answers z, z' to two challenges $x \neq x'$

This gives us $u^x a = g^z$ and $u^{x'} a = g^{z'}$

Division of the two equations gives us $u^x a/(u^{x'} a) = u^{x-x'} = g^{z-z'}$

So $u = g^{(z-z')/(x-x')}$ and $\alpha = (z-z')/(x-x')$

Fiat-Shamir heuristic

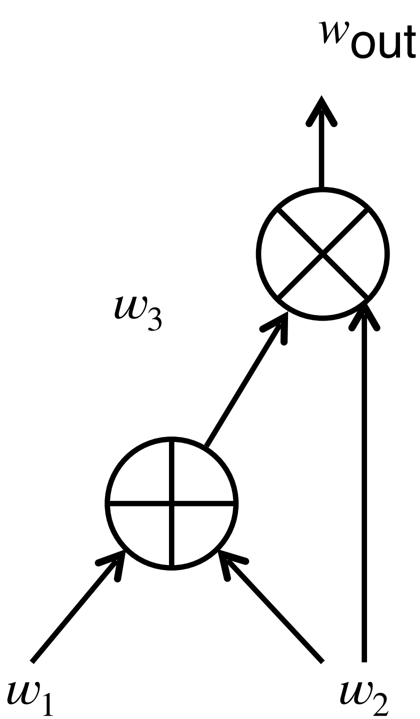
Statement $\phi \in L$

$\xrightarrow{\quad a \quad}$	$\Rightarrow$	$\xrightarrow{\quad a \quad}$
$\xleftarrow{\quad x \leftarrow S \quad}$		$\xleftarrow{\quad x = \mathrm{Hash}(\phi, a) \quad}$
$\xrightarrow{\quad z \quad}$		$\xrightarrow{\quad z \quad}$

Non-interactive zero-knowledge (NIZK) argument in the random oracle model, where Hash is modelled as random function to S

This justifies the honest verifier zero-knowledge notion, the verifier is "honest" because the verifier is a random oracle!

Arithmetic circuit satisfiability



Arithmetic circuit satisfiability for a circuit consisting of addition and multiplication gates over $\mathbb{Z}_p$.

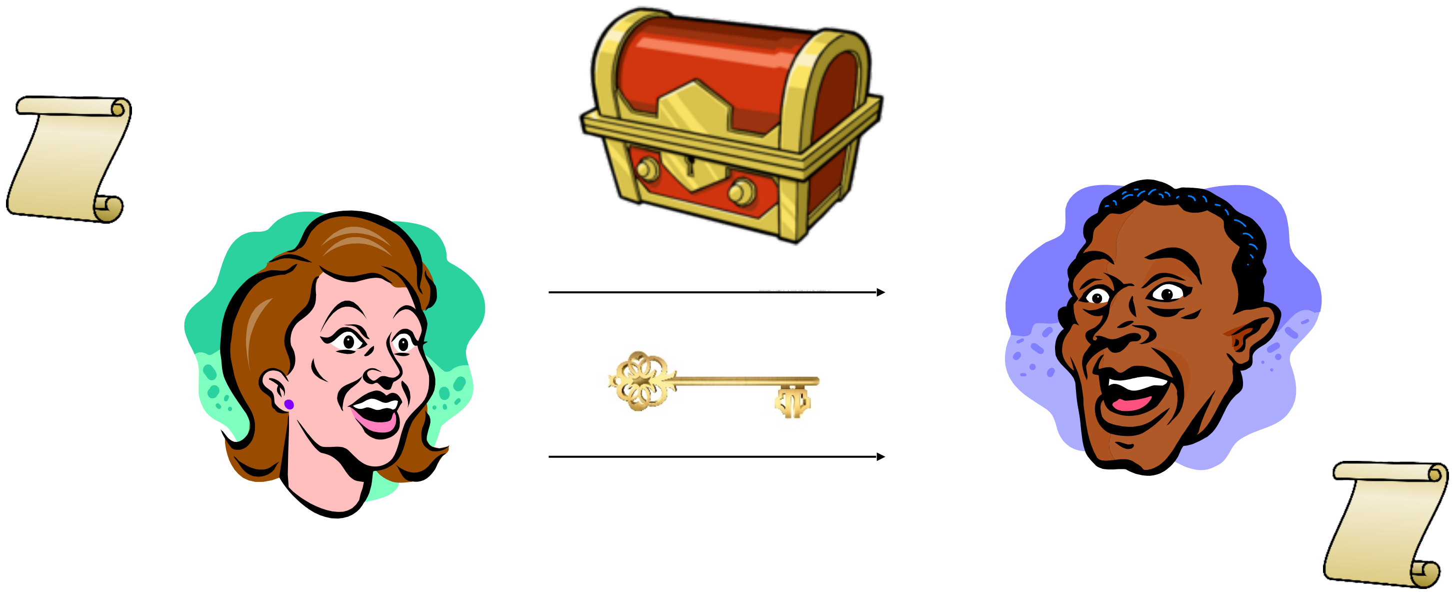
- Instance: prime p, circuit C
- Witness: inputs to make the circuit output 0

Variation: some of the wires are fixed to constants specified in the instance, e.g., $w_2 = 4$

Popular type of statement

- NP-complete
- Natural, lots of cryptography use finite field arithmetic

Commitment scheme



- Hiding
    - The commitment does not reveal information about the message
- Binding
    - It is infeasible to open a commitment to two different messages

Pedersen commitments

Broken by quantum computers; but other homomorphic commitment schemes exist

- Key generation (setup)
  - Pick a group $\mathbb{G}$ of prime order p with random generators g and h. Commitment key $ck = (\mathbb{G}, p, g, h)$.
- Commitment
  - Given $m \in \mathbb{Z}_p$ pick $r \leftarrow \mathbb{Z}_p$ and compute $c = g^m h^r$
- The opening of the commitment is $(m; r)$
  - The receiver can recompute to see the opening is valid
- Exercise
  - Verify the commitment scheme is homomorphic, i.e.,
    $\mathrm{com}_{ck}(m; r) \cdot \mathrm{com}_{ck}(m'; r') = \mathrm{com}_{ck}(m + m'; r + r')$
  - Argue the commitment scheme is perfectly hiding

Binding

$\longleftarrow \qquad ck \leftarrow \mathrm{KeyGen}(\lambda)$

$\longrightarrow \qquad (m, r),\ (m', r')$

$$\Pr\left[\begin{array}{l}\mathrm{com}_{ck}(m; r) = \mathrm{com}_{ck}(m'; r') \\ m \neq m'\end{array}\right] \approx 0$$

- Exercise
  - Show if the adversary can find two different openings of a Pedersen commitment such that $c = g^m h^r = g^{m'} h^{r'}$, then the adversary can break the discrete logarithm problem and find τ such that $h = g^\tau$

Σ-protocol for knowledge of an opening

- Setup: $ck = (\mathbb{G}, p, g, h)$
- Instance: $c \in \mathbb{G}$
- Witness: $(m; r)$ such that $c = \text{com}_{ck}(m; r)$

$b, s \leftarrow \mathbb{Z}_p$

$a = \text{com}_{ck}(b; s)$

$f = mx + b$

$z = rx + s$

$\xrightarrow{\quad a \quad}$

$\xleftarrow{\quad x \leftarrow \mathbb{Z}_p \quad}$

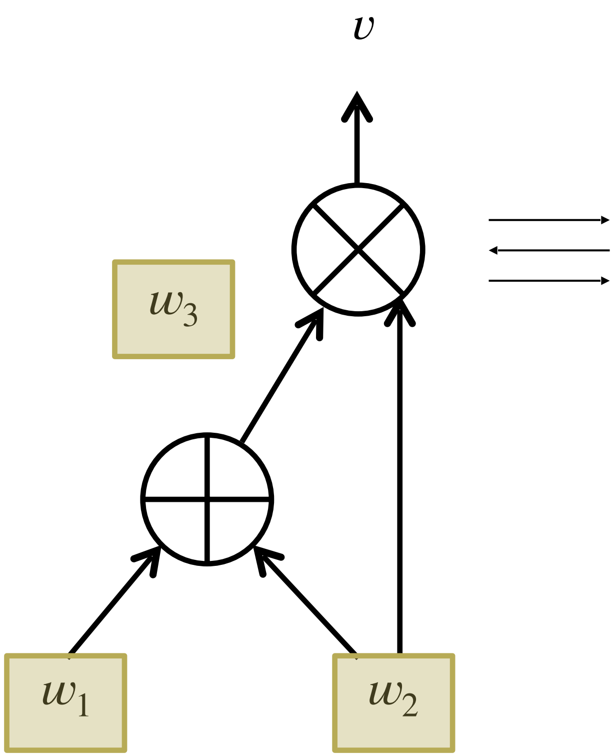
$\xrightarrow{\quad f, z \quad}$

Accept if

$c^x a = \text{com}_{ck}(f; z)$

- Exercise
  - Show it is complete, special sound and honest verifier zero knowledge
  - Modify the protocol to prove the committed m is 0

Sigma-protocol for arithmetic circuit over $\mathbb{Z}_p$



Strategy

- Commit to the wires
- For public values, commit in a directly verifiable manner, e.g., com$(v;0)$
- Use homomorphism to handle addition gates com(w_1) · com(w_2) $\rightarrow$ com(w_3)
- Use Sigma-protocols to prove the committed values satisfy multiplication gates, e.g., $v = w_2 \cdot w_3$

Addition gates

- Consider a gate saying $w_3 = w_1 + w_2$
- Given commitments

  $$c_1 = \mathrm{com}_{ck}(w_1; r_1) \text{ and } c_2 = \mathrm{com}_{ck}\left(w_2; r_2\right)$$

  compute the commitment to w_3 as

  $$c_3 = c_1 \cdot c_2$$

  which by the homomorphic property of the commitment scheme automatically gives a verifiable commitment to

  $$w_3 = w_1 + w_2$$

Sigma-protocol for multiplication gates

- Instance: c_1, c_2, c_3
- Witness: $w_1, r_1, w_2, r_2, w_3, r_3$ satisfying

$$w_3 = w_1 w_2 \qquad c_1 = \text{com}(w_1)$$
$$c_2 = \text{com}(w_2) \qquad c_3 = \text{com}(w_3)$$

$b, s, t \leftarrow \mathbb{Z}_p$
$a = \text{com}(r; s)$
$b = \text{com}(-w_2 r; t)$

$\xrightarrow{a, b}$

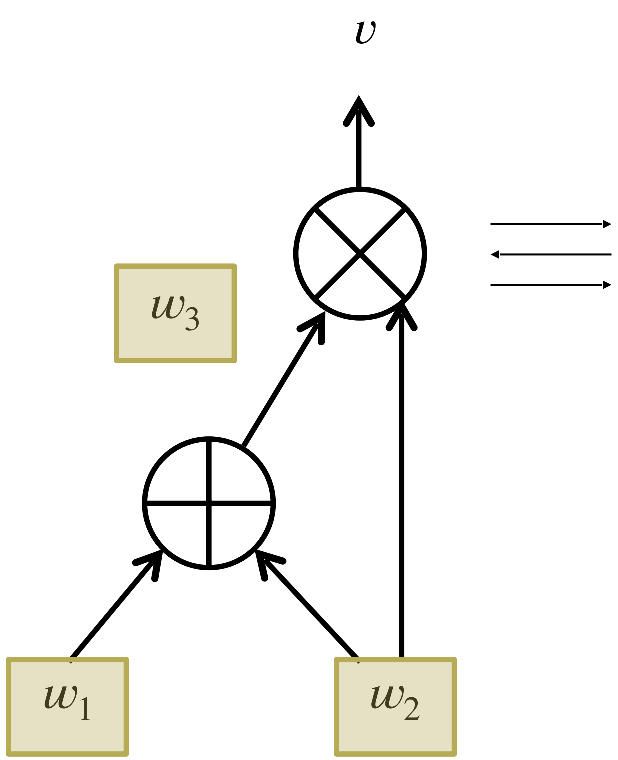
$\xleftarrow{x \leftarrow \mathbb{Z}_p}$

$\xrightarrow{f, z_1, z_2}$

Sketch of soundness
$f = w_1 x + r$
$f w_2 - x w_3 + \beta = (w_1 w_2 - w_3)x + (r w_2 + \beta) = 0$

Accept if
$c_1^x a = \text{com}(f; z_1)$
$c_2^f c_3^{-x} b = \text{com}(0; z_2)$

Cost to prove arithmetic circuit satisfiability



- Commit to the inputs to the circuit and inputs to multiplication gates
  - A commitment per wire
- For public values, commit in a directly verifiable manner, e.g., com$(v; 0)$
  - Free
- Use homomorphism to handle addition gates
  com(w_1) $\cdot$ com(w_2) $\rightarrow$ com(w_3)
  - Free
- Use Sigma-protocols to prove the committed values satisfy multiplication gates, e.g., $v = w_2 \cdot w_3$
  - A few group and field elements per multiplication gate
- **In total for an N-gate circuit**
  - $O(N)$ group and field elements

Σ-protocol for knowledge of many openings

- Setup: $ck = (\mathbb{G}, p, g, h)$
- Instance: $c_1, \ldots, c_t \in \mathbb{G}$
- Witness: openings $(m_i; r_i)$ such that $c_i = \text{com}(m_i; r_i)$

Communication
$1\ \mathbb{G} + 2\ \mathbb{Z}_p$
for instance size t $\mathbb{G}$

$b, s \leftarrow \mathbb{Z}_p$
$a = \text{com}(b; s)$

$\xrightarrow{a}$

$\xleftarrow{x \leftarrow \mathbb{Z}_p}$

$\xrightarrow{f, z}$

$f = b + \sum x^i m_i$
$z = s + \sum x^i r_i$

Accept if
$a \prod c_i^{x^i} = \text{com}(f; z)$

- Sketch of knowledge soundness
  - For $t = 1$ this is exactly as before, if the prover can answer two distinct challenges x, x' we can combine the two resulting verification equations to open c_1
  - For larger t if the prover can answer $t + 1$ distinct challenges $x, x', x'', \ldots$ we can for each c_i combine the $t + 1$ verification equations to find an opening

Generalized Pedersen commitment

- Key generation (setup)
  - Pick a group $\mathbb{G}$ of prime order p with random generators h and $g_1, \ldots, g_n$. Commitment key $ck = (\mathbb{G}, p, h, g_1, \ldots, g_n)$.
- Commitment
  - Given $m_1, \ldots, m_n \in \mathbb{Z}_p$ pick $r \leftarrow \mathbb{Z}_p$ and let $c = h^r \prod g_i^{m_i}$
  - The opening of the commitment is $(m_1, \ldots, m_n, r)$
- Properties
  - Perfectly hiding
  - Computationally binding under discrete log assumption
  - Homomorphic

$$\mathrm{com}\left(\vec{a}\,; r\right) \cdot \mathrm{com}\left(\vec{b}\,; s\right) = \mathrm{com}(\vec{a} + \vec{b}\,; r + s)$$

Σ-protocol for knowledge of many openings

- Setup: $ck = (\mathbb{G}, p, h, g_1, \ldots, g_n)$
- Instance: $c_1, \ldots, c_t \in \mathbb{G}$
- Witness: openings $(\vec{m}_i; r_i)$ such that $c_i = \text{com}(\vec{m}_i; r_i)$

Instance + proof size
$t + 1$ $\mathbb{G}$ + $n + 2$ $\mathbb{Z}_p$
for witness size tn

$\vec{b}, s \leftarrow \mathbb{Z}_p$

$a = \text{com}(\vec{b}; s)$

$\xrightarrow{a}$

$\xleftarrow{x \leftarrow \mathbb{Z}_p}$

$\xrightarrow{\vec{f}, z}$

$\vec{f} = \vec{b} + \sum x^i \vec{m}_i$

$z = s + \sum x^i r_i$

Accept if
$a \prod c_i^{x^i} = \text{com}(\vec{f}; z)$

- Sketch of knowledge soundness
  - For larger t if the prover can answer $t + 1$ distinct challenges $x, x', x'', \ldots$ we can for each c_i combine the $t + 1$ verification equations to find an opening
  - This time each opening is a size n vector and some randomness

Proofs of knowledge with sublinear communication

- This is a Sigma-protocol with sublinear communication
  - You can prove knowledge of tn field elements using only $O(t+n)$ elements to describe the instance and send messages in the proof
  - If we set $t \approx n$ we can prove knowledge of N field elements using only $O(\sqrt{N})$ communication!
- If you use the Fiat-Shamir heuristic it becomes a non-interactive proof system. I.e., we have a zk-SNARK for proving knowledge of many field elements at once, where the proof size is much smaller than the witness
  - And the computation to verify the proof is only $O(\sqrt{N})$ exponentiations, which is easier to compute than if you had the whole witness and needed to verify the openings directly

Proofs of arithmetic circuit satisfiability

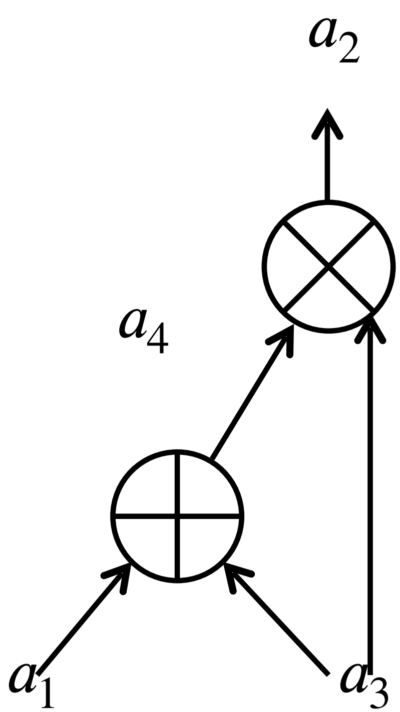
	Rounds	Prover	Verifier	Comm.
Cramer-Damgård 1997	3	6N expo	6N expo	11N elem
Groth 2009	7	6N/log N expo	O(N) mult	16√N elem
Bootle-Cerulli-Chaidos-Ghadafi-Groth 2016	2 log N + 1	12N expo	4N expo	6 log N elem

- Ideas behind the constructions
    - Commit to wires with Pedersen commitments, prove the wires respect the gates
    - Commit with √N-wide Pedersen commitments, prove the wires respect the gates
    - Start with N-wide generalised Pedersen commitments, don't show any N-wide openings but recursively prove the openings exist and are correct with less wide commitments
- **Bulletproofs** [Bünz-Bootle-Boneh-Poelstra-Wuille-Maxwell17] is a popular and widely used proof system that builds on [BCCGG16]

Can proof for arithmetic circuit satisfiability be even smaller?

- There are zk-SNARKs with O(1)-sized proofs and you can get as low as 3 group elements [Groth10,Groth16]
  - Which means you can prove an arithmetic circuit with billions of gates is satisfiable using only a few hundred bytes to convince the verifier!
- But the techniques are different; they rely on groups with pairings
  - Let p be a prime
  - Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of size p
  - Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an efficiently computable bilinear map
    - If g_1, g_2 are generators of the source groups $\mathbb{G}_1, \mathbb{G}_2$ then $e(g_1, g_2)$ generates $\mathbb{G}_T$
    - For all $a, b \in \mathbb{Z}_p$ we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$
- In groups with pairings, not only do you have 'additions and multiplication with a known constant' in the exponent; now you also have 'multiplication in the exponent'

Arithmetic circuit



- Write each multiplication gate as a quadratic equation, here
  $\left(a_1 + a_3\right) \cdot a_3 = a_2$
- In general arithmetic circuit can be written as a set of quadratic 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 $a_0 = 1$
- A fixed arithmetic circuit defines an NP-language with statements $(a_1, \ldots, a_\ell)$ and witnesses $(a_{\ell+1}, \ldots, a_m)$

zk-SNARK for the circuit being satisfiable for an instance $(a_1, \ldots, a_m)$

- Common reference string generation: the circuit defines what is called a quadratic arithmetic program (QAP) consisting of polynomials $\{u_i(x)\}, \{v_i(x)\}, \{w_i(x)\}, t(x)$. Pick secret $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{Z}_p$ and publish

$$\sigma = \left(g_1^{\alpha}, g_1^{\beta}, g_1^{\delta}, \left\{ g_1^{x^i} \right\}, \left\{ g_1^{\frac{x^i t(x)}{\delta}} \right\}, \left\{ g_1^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}} \right\}_{i \leq \ell}, \left\{ g_1^{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}} \right\}_{i \geq \ell}, g_2^{\beta}, g_2^{\gamma}, g_2^{\delta}, \left\{ g_2^{x^i} \right\} \right)$$

- Prove($\sigma, \phi = (a_1, \ldots, a_\ell), w = (a_{\ell+1}, \ldots, a_m)$): pick $r, s \leftarrow \mathbb{Z}_p$ and return $\pi = (g_1^A, g_2^B, g_1^C)$ where

$$A = \alpha + \sum a_i u_i(x) + r\delta \qquad B = \beta + \sum a_i v_i(x) + s\delta$$

$$C = \sum_{i > \ell} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \frac{h(x)t(x)}{\delta} + As + rB - rs\delta$$

- Verify($\sigma, \phi = (a_1, \ldots, a_\ell), \pi$): check $\phi \in \mathbb{Z}_p^\ell$ and $\pi \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$ and the pairing product equation

$$e\left(g_1^A, g_2^B\right) = e\left(g_1^{\alpha}, g_2^{\beta}\right) \cdot e\left(g_1^{\sum_{i \leq \ell} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}}, g_2^{\gamma}\right) \cdot e\left(g_1^C, g_2^{\delta}\right)$$