Introduction to Zero Knowledge Proofs

There are a prover and verifier `String` = proof submitted by the prover Verifier rejects or accepts the proof (i.e. the `String`)

In CS we talk about polynomial proofs, that can be verified in **polynomial** time → **NP proofs**
- The `string` is short
- Polynomial time constraints

Given claim `x`, the length of the proof `w` should be of polynomial length. More formally: `|w|` = polynomial in `|x|`.

The verifier has a **polynomyial** time contraints for execution.

`V` *accepts* `w` if $V(x, w) = 1$, otherwise *rejects*.

Prover (Uncontrained runtime) $\xrightarrow{\text{Proof } \mathbf{w}}$ **Verifier** (Verifies in polynomial time of claim `x`)

Efficently verifiable proofs

We can formalise the the input as:
- Language L - a set of binary strings

More formally:

Definition:
L is an **NP** language (or NP decision problem), if there is a verifier V that runs in polynomial time of length of the claim x that where
- **Completeness [True claims have short proofs]**
  if $x \in L$, there is a polynomially (of $|x|$) long witness $w \in \{0,1\}^*$ st $V(x, w) = 1$
- **Soundness [False claims have no proofs]**
  if $x \notin L$, there is no witness. That is, for all $w \in \{0,1\}^*, V(x, w) = 0$

Proving Quadratic Residue[1]

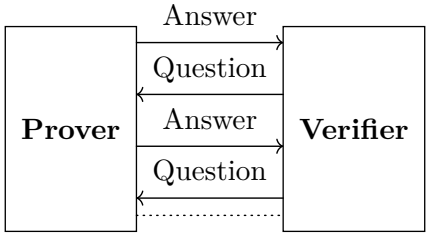
Goal: proof that y is a quadratic residue $\bmod N$.

Prover $\xrightarrow{\text{Proof} = \sqrt{y} \bmod N \in Z_N^*}$ **Verifier**

The gist: Prove (or persuade) the verifier that the prover could prove the statement.

Adding additional components - **Interactive** and **Probabilistic** proofs
- **Interaction** - verifier engages in the *non-trivial* interaction with the prover
- **Randomness** - verifier is randomised, and can accept am invalid proof with a small probability (quantified).

[1] https://mathworld.wolfram.com/QuadraticResidue.html



Examples: https://medium.com/@houzier.saurav/non-technical-101-on-zero-knowledge-proofs-cab77d671a40

Here is the interactive proof for the quadratic residue:

1. Prover chooses a random r s.t. $1 \leq r \leq N$ s.t. $\gcd(r, N) = 1$
2. Prover sends s, s.t. $s = r^2 \bmod N$
   - If the prover sends $\sqrt{s} \bmod N$ and $\sqrt{s \times y} \bmod N$ later, then the verifier coulde deduce $x = \sqrt{y} \bmod N$
   - Instead, the prover gives either one of the roots.
3. Verifier randomly chooses $b \in \{0, 1\}$
4. if $b = 1$: verifier sends $z = r$, otherwise $z = r \times \sqrt{y} \bmod N$
5. Verifier accepts the proofs only if $z^2 = sy^b \bmod n$
   - if $b = 0$ then $z^2 = s \bmod N$
   - otherwise, $z^2 = sy \bmod N$

During the first try, the probability of error is $\frac{1}{2}$, after k interactions => it is $\left(\frac{1}{2}\right)^k$.

Note: at the beginning of each interaction, the Prover generates a new r which is $\sqrt{s} \bmod N$

Assessing the properties:

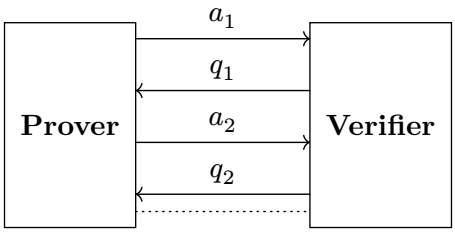
- **Completeness**: if the claim is true, the verifier will accept it.
- **Soundness**: if the claim is false, $\forall$provers, $P(\text{Verifier accepts}) \leq \frac{1}{2}$
  - After k interactions: $\forall$provers, $P(\text{Verifier accepts}) \leq \left(\frac{1}{2}\right)^k$

How does previous example worked?

Sending both parts of the QR, proves that the verifier **could** solve the original equation. Additionally, sending just the s reveal no knowledge about the original solution.

The ability of the prover, to provide solution to either part persuades the verifier that it can solve the original equation.

Interactive Proofs



Claim / Theorem X

Definition:
(P, V) is an interactive proof for L, if V is probabilistic poly($|x|$) and
- **Completeness**: if $x \in L$, $P[(P, V)(x) = \text{accept}] \geq c$
- **Soundness**: if $x \notin L$ for every P^*, $P[(P^*, V)(x) = \text{accept}] \leq s$

We want to c to be as close to **b** as possible, and s to be as negligible as possible. Equivalent as long as $c - s \geq \frac{1}{\text{poly}(|x|)}$

Definition:
Class of interaction proof languages = $\{L$ for which there is an interactive proof$\}$

Zero-Knowledge views

The intuition behind zero knowledge interactions:

For any true statements, what the verifier can compute **after** the interaction is the same to what it could have computer **before** the interaction. In other words, the verifier **does not gain any more information** (i.e. knowledge) after the verification interaction. This should hold true **even for malicious verifiers.**

View is essentially a set of traces containing submitted questions q^* to the verifier and received answers a^* from it.

After the interaction V learns:
- Theorem (T) is true, $x \in L$
- A **view** of interactions

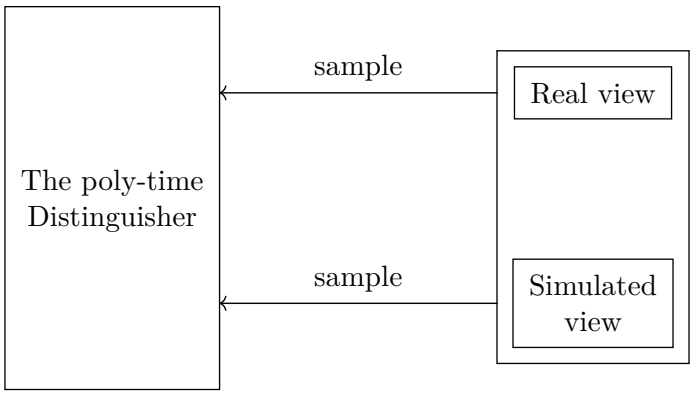
Definition:
$\text{view}_{v(P,V)}[x] = \{(q_1, a_1), (q_2, a_2), ...\}$, which is a probibility distributions over random values of V and P.

In the case of P, the random value it selects for QR is r, in the case of P it is the choice whether the reveal r or $r\sqrt{y} \bmod N$

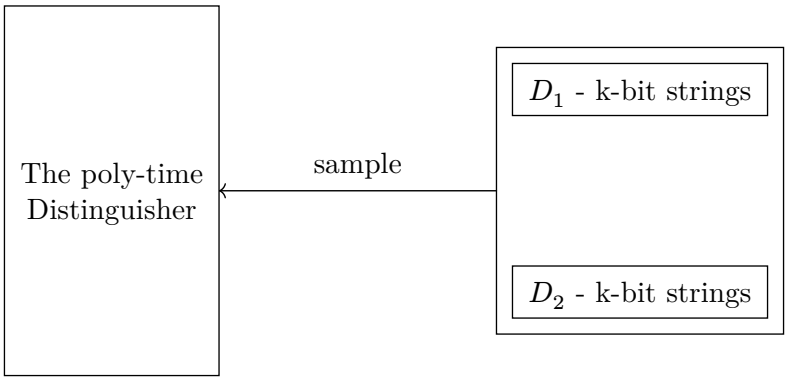
Simulation Paradigm

We can say that V's view gives no further knowledge to it, if it could have simulated it on its own s.t. the simulated view and the real view are *computationally indistinguishable*.

What that means is that we can have a polynomial time "distinguisher" that extracts the trace from either of the views, and if it can not differentiate it with the probability less than 50/50, we can say that they views are *computationally indistinguishable*.



More formally:



Definition:
For all distinguisher algorithms, even after receiving a polynomial number of samples for D_b, if $P(\text{Distinguisher guesses } b) < \frac{1}{2} + c$ where c negligible constant,
then $D_1..D_k$ are **computationally indistinguishable**

Zero-Knowledge Definition

Definition:
An Interactive Protocol (P,V) is zero-knowledge for a language L if there exists a **PPT** (Probabilistic Polynomial Time) algorithm **Sim** (a simulator) such that for every $x \in L$, the following two probability distributions are **poly-time** indistinguishable:

1. $\text{view}_{v(P,V)}[x] = \text{traces}$
2. $\text{Sim}(x, 1^{\lambda})$

(P, V) is a zero-knowledge interactive protocol if it is **complete**, **sound** and **zero-knowledge**.

Flavours of Zero-Knowledge:
- Computationally indistinguishable distributions = CZK
- Perfectly identical distributions = PZK
- Statistically close distributions = SZK

Simulator Example

For QR, the view presented as $\text{view}_{v(P,V)} : (s, b, z)$

Simulator workflow:

1. Pick a random b
2. Pick a random $z \in Z_N^*$
3. Compute $s = z^2/y^b$
4. Output (s, z, b)

We would see that (s, z, b) is identically distributed as in real view.

For adversarial verifier V^*:

1. Pick a random b
2. Pick a random $z \in Z_N^*$
3. Compute $s = z^2/y^b$
4. If $V * (N, y, s) = b$, then output (s, z, b), otherwise go to step 1.

Within 2 iteration, we would still reach identical distribution.

Proof of knowledge

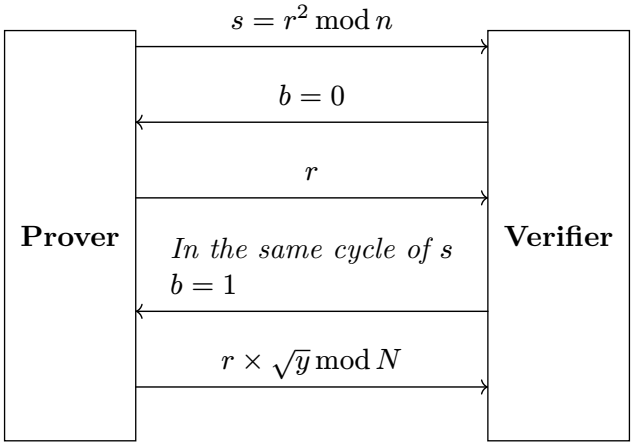
The prover convinces the verifier not only about the theorem but that it also knows the x.

Using that information, we can construct the knowledge extractor using the rewinding technique.

More formally:

Definition:
Consider $L_R = \{x : \exists w \text{ s.t. } R(x, w) = \text{accept}\}$ for poly-time relation R.

(P, V) is a proof of knowledge (POK) for L_R if $\exists$ PPT (knowledge) extractor algorithm E s.t. $\forall x \in L$ in expected poly-time $E^P(x)$ outputs w s.t. $V(x, w) = \text{accept}$



And the verifier can determine $r \times \sqrt{y} \bmod N$, hence, extract the knowledge.

All of NP is in Zero-Knowledge

Definition:
Theorem[GMW86,Naor]: If one-way functions exist, then every $L \in$ NP has computational zero knowledge interactive proofs

Shown that an NP-Complete Problem has a ZK interactive Proof.

[GMW87] Showed ZK interactive proof for G3-COLOR using bit commitments.

Reading

How to Prove All NP Statements in Zero-Knowledge and a Methodology of Cryptographic Protocol Design (Extended Abstract)