1 Minimum Bit Communication

Suppose we have two parties, Alice and Bob, who have private n-length bitstrings, A and B respectively. Bob wants to test to see if Alice has the same string as himself. Naively, Alice can simply send their string to Bob, who then tests if $a_i = b_i$ for all $i = 1...n$. However, this takes n bits of communication between the parties. Consider the following protocol:

1. Bob selects a random field element $r \leftarrow \mathbb{F}_q$ where q is prime and $q > n^2$ and sends q to Alice.
2. Alice calculates $f_A(r) = a_1x^1 + a_2x^2 + ... + a_nx^n$ and returns the answer to Bob.
3. Bob calculates the same for $f_B(r)$ and checks if $f_A(r) = f_B(r)$. He accepts if they are.

Note that the total number of bits sent are the size of elements in $\mathbb{F}_q$ which takes $\approx \log(n^2) = 2\log(n)$ bits twice, meaning the entire protocol takes $4\log(n)$ bits. Furthemore, the chance that the protocol incorrectly asserts that the two strings are equal is $n/n^2 = 1/n$.

2 Basic Definitions

Argument System: An **Argument System** is a protocol that assumes the prover is computationally bounded. For example, if a problem is assumed to be hard, an Argument system can be considered secure under the *assumption* that the prover cannot solve the problem.

Hash Function: A function H from $\{0,1\}^m \rightarrow \{0,1\}^n$ is considered a hash function if $n < m$.
Collision Resistant Hash Function: A hash function is collision resistent if not poly-time algorithm can find a pair (x_1, x_2) where $x_1 \neq x_2$ and $h(x_1) = h(x_2)$.

3 ZK Graph Isomorphism

Assume an infinitely powerful prover (P) wants to demonstrate two graphs are isomorphic to verifier (V). The trivial method of sending permutation π reveals this isomorphism directly, so the verifier learns something from

the prover. In zero-knowledge protocols, the prover can demonstrate $x \in L$ without giving the verifier any information about the isomorphism itself.

Consider the following protocol:

1. P selects a random permutation π_1, and sends $H = \pi_1(G_0)$ to V.
2. V selects a random $b \leftarrow \{0, 1\}$ and returns b to P.
3. P comes up with a permutation π_b that maps G_b to H.
4. Steps 1-3 are repeated k rounds until the verifier is convinced or rejects if $\pi_b(G_b) \neq H$

The completeness of this protocol is clear. The soundness comes from the fact that if the two graphs are not isomorphic, it will not be able to come up with π_b if b is not 0, so with probability 1/2 the verifier will reject each round if $w \notin L$.

3.1 Definition of Zero-Knowledge (Incomplete)

Intuitively, a protocol is zero-knowledge if a transcript produced between the verifier and prover is indistinguishable from a transcript that can produced by some PPT simulator S. This means that V cannot *learn anything* since the conversation they have with P is indistinguishable from something they could produce themselves. The formal definition is as follows:

A protocol is **Zero-Knowledge** if:

1. The protocol is an interactive proof system **(IP)**
2. $\forall x \in L$ there exists a PPT simulator S such that for any transcript between P and V, the probability it was produced by either $S(x)$ or $PV(x)$ is the same.

3.2 Simulator for GI

An example of a simulator that can produce transcripts in a way identical to the graph isomorphism protocol is as follows.

1. S selects a random $b \leftarrow \{0, 1\}$ and a random permutation π to produce $H = \pi(G_b)$
2. S outputs simulated transcript $(H, b, \pi_b^{-1}(H) = G_b)$

Firstly, since H is a random permutation of G_0, the probability that any H produced from G_0 or G_1 is the same as the probability that H is produced from just G_0. Secondly, b is chosen randomly by both S and V, meaning the distribution is also the same. Lastly, π is fixed from the first two messages in the protocol so it will always follow H and b, meaning that the probability distribution of all possible messages are the same between V and S.

3.3 Strengthing the Definition of Zero Knowledge

The prior simulator is able to create a transcript indisitinguishable from an actual interaction under the *assumption* that the verifier is acting honestly. However, consider a verifier that produces a bit based on an arbitrary function, such as one that hashes H and outputs the first bit as b. This will clearly not be perfectly simulated by S. Therefore, to account for this an extra condition is needed for the definition of a ZK protocol.

- $\forall x \in L, \forall V^*, \exists S^{V^*} \in$ PPT s.t. $S^{V^*}(X) = PV^*(X)$ or the transcript distribution produced by any verifier V^* is the same regardless of the actions of the verifier.

To achieve this, we assume that the simulator can interact with the verifier and rewind the protocol to the beginning, and also that the verifier has fixed randomness.

S will therefore simulate PV^* in the same manner, but will now send H_i permuted from $G_{b'}$ to the verifier, and if b sent from the verifier is ever not equal to b', it rewinds the tape, simulates all successful rounds, and chooses a new random b' and permutation H_i. Now S can successfully simulate any interaction between P and V^* for any V^*, honest or dishonest.

3.4 Fixing the Strengthed Definition

Consider the following protocol:

1. V selects a random b' and permutation π' and sends $H = \pi'(G_{b'})$ to P
2. P selects a random b and returns π that maps $G_b \to H$
3. This is continued until V learns the isomorphism in which they return true or π is incorrect and they reject.

Clearly, this is not zero knowledge since V learns the isomorphism (if P's b is ever not the same as b', then by composition of functions V can learn isomorphism between G_0 and G_1), however, consider the following simulator S that can simulate a conversation between P and V:

1. S picks a random b' and π' and creates $H = \pi'(G_{b'})$.
2. It returns $\pi = \pi'^{-1}$.

The output distribution here would be identical to that of PV, however, the protocol is very clearly not zero knowledge. In order to rectify this, we add another condition to the transcript that S has to generate.

- The simulator must be able to produce a **view** output distribution identical to that of a real interaction between P and V. A view is a combination of (x, transcript, V's randomness) that is produced in an interaction.

We can see why this fixes the previous issue, as outputting the b' generated by V in the simulator would cause b' and b to be identical in each case. However, in real interactions between P and V, b' would match the provers b only half of the time.

4 Full Defintion and Types of Zero Knowledge

The full definition of Zero Knowledge is as follows:
Language L has a **Zero-Knowledge Interactive Proof** (ZK IP) if $\forall x \in L, \forall V^*, \exists$ PPT Simulator S^{V^*} such that View$(PV^*(x))$ = View$(S^{V^*}(X))$.

Types of ZK

- **Perfect ZK (=)**: Simulator exists that can output distributions identically to those of real interactions
- **Statistical ZK ($\cong$)**: Simulator exists that can produce views that require an exponential number of samples for the verifier to distinguish the distributions.
- **Computational ZK ($\stackrel{c}{=}$)**: Simulator can produce views that are indisitinguishable to a PPT observer.