AFRICAN INSTITUTE FOR MATHEMATICAL SCIENCES

(AIMS RWANDA, KIGALI)

Name: Sittana Osman Afifi Mohamedelmubarak ch 2

Zero-Knowledge proofs: Implementation of the Graph Isomorphism Protocol Date: May 19, 2020

1 Graph isomorphism

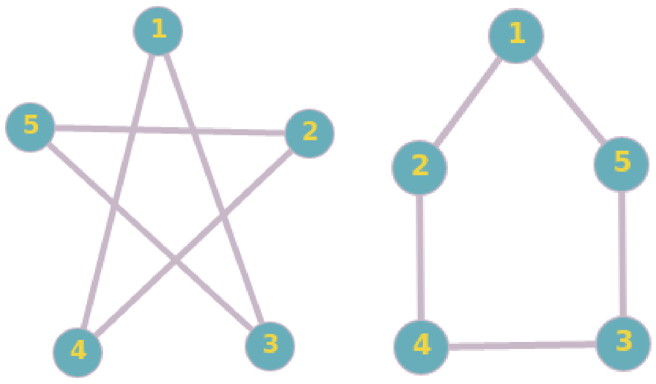
Definition 1. *(Graph)[1]:A graph consist of set of vertices(nodes) V and set of edges E. Two nodes u and v is said to be adjacent if there is an edge $(u, v) \in E$.*

we can describe graph using adjacency matrix which is a square matrix $M_{n\times n}$, with $[m_{ij}] = 1$ if $(i, j) \in E$ and 0 otherwise.

Definition 2. *Let $V(G)$, $E(G)$ denote the vertex set and edge set of G respectively. Then, a pair of of graphs$(G_0, G_1)$is* ***isomorphic*** *(denoted $G_0 \simeq G_1$) if there exists a permutation $\Pi : V(G0) \longmapsto V(G1)$such that $\forall x, y \in V(G_0), xy \in E(G_0)$ if and only if $\Pi(x)\Pi(y) \in E(G_1)$. The permutation Π is called an isomorphism.[2]*

In other words: two graphs are said to be isomorphic if after we relabeling vertices in one graph we get the other graph (the same adjacency matrix).

Example 3. *Two isomorphic graphs with their corresponding adjacency matrices.*



(a) G_0. (b) G_1.

Figure 1: Two isomorphic graphs.

Graph (b) is obtained, by relabeling the vertices of Graph (a) according to the following permutation: (1, 4, 5, 2, 3). This means that Node 3 in Graph (a) becomes Node 5 in Graph (b), Node 4 becomes Node 2.

	1	**2**	**3**	**4**	**5**
1	0	0	1	1	0
2	0	0	0	1	1
3	1	0	0	0	1
4	1	1	0	0	0
5	0	1	1	0	0

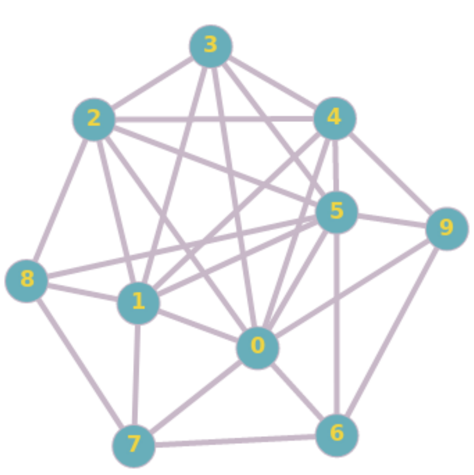
Table 1: ajacency matrix of G_0

	1	**2**	**3**	**4**	**5**
1	0	1	0	0	1
2	1	0	0	1	0
3	0	0	0	1	1
4	0	1	1	0	0
5	1	0	1	0	0

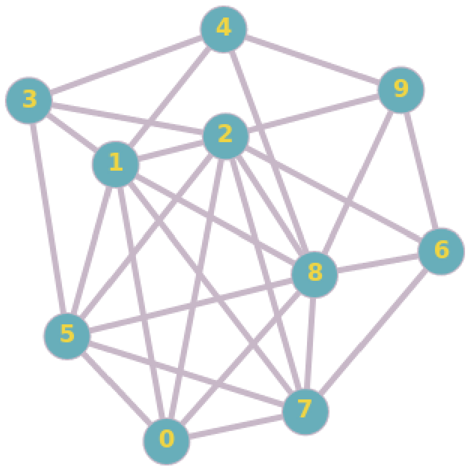
Table 2: ajacency matrix of G_1

1.1 Graph Isomorphism based Zero-Knowledge Proofs

Suppose we have two isomorphic graphs G_0 and G_1 and $G_1 = \Pi(G_0)$, with limited messages between the prover and verifier, P wants to prove to V he knows the secret Π without showing him what is Π exactly. From the previous example, we can see that it is easy to show that two graphs are isomorphic or not but this process isn't always simple; suppose we have two graphs have 10 vertices and 28 edges, like these graphs:



(a) G_0.



(b) G_1.

Figure 2: Two isomorphic graphs.

So ZKP can provide a protocol that the P can prove to V he knows the secret Π without revealing Π itself.
The protocol is done by apply random permutation (φ) on G_0, with:

$$H = \varphi(G_0)$$

and the honest prover has to be able to find a permutation such that he could transform H to either G_0 or G_1.(i.e to prove $H \simeq G_0$ or $H \simeq G_1$)

1.2 Zero-Knowledge Protocol for Graph Isomorphism

We have two graphs known by both parties G_0 and G_1 such that they have n vertices, define s_n as a set of permutations has n element.

The protocol proceeds by the following:[2]
Input: pair of graphs (G_0, G_1)

1. **prover** chooses random permutation σ from s_n, and apply $H = \sigma(G_0)$.
2. **verifier** chooses ch randomly from $\{0, 1\}$ and send it to the prover.
3. **prover** if $ch = 0$: $\varphi = \sigma$ else $\varphi = \sigma \circ \Pi^{-1}$.
4. **verifier** output <u>ACCEPT</u> iff $H = \varphi(G_{ch})$ else output is <u>REJECT</u>.

Theorem 4. *[2] The above protocol satisfies completeness, soundness $\frac{1}{2}$, and zero-knowledge.*

Proof
Completeness. In order to show this protocol is complete, we have to show that if the prover knows the correct permutation Π and interacts with an honest verifier, the output will be **ACCEPT**.
Assume we have two isomorphic graphs with a witness Π such that $G_1 = \Pi(G_0)$, we will check when $ch = 0$ an $ch = 1$:

1. case 1 ($ch = 0$): P has to find a map from H to G_0, or to show that $H \simeq \varphi(G_0)$: Since $H = \sigma(G_0)$ then the prover will return $\varphi = \sigma$.Certainly $\sigma(G_0) \simeq \sigma(G_0)$
2. case 2 ($ch = 1$): P has to find a map from H to G_1 or to show that $H \simeq \varphi(G_1)$: We know that:
$$G_1 = \Pi(G_0)$$
and
$$H = \varphi(G0)$$
then:
$$H = \varphi(\Pi^{-1}(G_1)$$
$$H = (\varphi \circ \Pi^{-1})(G_1)$$
So the
$$\varphi = \varphi \circ \Pi^{-1}$$
because
$$\varphi(G_1) = \sigma \circ \Pi^{-1}(G_1) = \sigma(G0) = H$$

Soundness $\frac{1}{2}$. if $G_0 \not\simeq G_1$ then for every probabilistic polynomial time algorithm algorithm $\hat{P}$, there exist a negligible function $negl(\cdot)$ such that:

$$Pr[\hat{P} \text{ convinces} V \text{that } G0 \simeq G1] \leq negl(\cdot)$$

but $negl(\cdot) = \frac{1}{2}$ because:
suppose $G_0 \not\simeq G_1$, since $\simeq$ is transitive so for any graph $G^{'}$either $G^{'} \simeq G_0$ or $G^{'} \simeq G_1$ not both; this indicate that the prover could pass only one of the tests not both.
In other words, the prover will pass the test when the verifier choose $ch = 0$ because he will return σ, but he could not pass the test when the verifier choose $ch = 1$ because the prover can

not find φ with $\sigma(G_0) \simeq \varphi(G_1)$ since $G_0 \not\simeq G_1$.
since the verifier has only two possible choices $\{0,1\}$ then:

$$Pr[\hat{P} \text{ convinces} V \text{that } G0 \simeq G1] \leq \frac{1}{2}$$

zero-knowledge. Our goal is to construct a simulator S which produces a transcript that is computationally indistinguishable from the execution of the above protocol between an honest prove and an honest verifier. *suggested protocol*::Define S as follow,

1. Sample $\sigma \longleftarrow S_n$ and choose $b \longleftarrow 0, 1$, put $H = \sigma(G_b)$.
2. Choose $ch \longleftarrow \{0,1\}$.
3. If $ch = b$ output σ,otherwise repeat from (1).
4. Output **ACCEPT**.

The simulator should protect the honest prover from a cheating verifier who wants to learn more about the secret from the verifier, but according to the suggested protocol above a cheating verifier can be unfair on how will choose ch, V may decide to always send $ch = 1$ then V will always send $ch = 1$ whereas the simulator S will set $ch = 1$ always with probability $\frac{1}{2}$. since S will be always fair and the original protocol is unfair the transcript $\tau^{‘} \simeq \tau$ such that $\tau^{‘}$ from S and τ from the original protocol.
In order to fix this issue, we can give S access to an arbitrary black-box verifier V^* that provides S with the random bit for ch, using V^* we correct the suggested protocol.

Correct protocol:Define S as follow,

1. Sample $\sigma \longleftarrow S_n$ and choose $b \longleftarrow 0, 1$, put $H = \sigma(G_b)$.
2. Feed H into V^* to get ch.
3. If $ch = b$ output σ,otherwise repeat from (1).
4. Output **ACCEPT**.

Now, there is no bias with choosing ch, the next step is to prove that $\tau \simeq \tau^{'}$, it suffices to show that the distribution of the output from step (1) is indistinguishable from the output of step (1) in the original protocol, since the rest steps are similar when $G_0 \simeq G_1$.

Fact 1: *If $G_0 \simeq G_1$ then for $\sigma \longleftarrow S_n$, the distributions$\sigma(G_0)$ and $\sigma(G_1)$ are equal.*

Using fact 1 and by the assumption $G_0 \simeq G_1$ then we can conclude that $\sigma(G0) = \sigma(G1)$ for any σ is choosen from S_n. On the other hand, when $\sigma \longleftarrow S_n$, a fixed $\sigma^{‘} \in S_n$ so $\sigma \circ \sigma^{‘}$ still behaves as a uniformly random permutation, Thereafter,

$$\sigma(G_0) = (\sigma \circ \Pi^{-1}(G_1) = \sigma_s(G_b)$$

When σ_S is a permutation choose in step (1) of S, and $b \longleftarrow \{0,1\}$. Thus, the distribution of the output from step (1) and the output of step (1) in the original protocol are the same, so certainly $\tau \simeq_c \tau^{'}$.this complete the proof.

The verifier to be convinced that the prover is honest($G_0 \simeq G_1$and he knows Π) he needs to apply this procedure several times. if we just apply the procedure once the prover may be lucky when the prover chooses $ch = 0$ so the probability is 0.5.
but after repeating the procedure k times the probability for the cheating prover to fail is at least $1 - \frac{1}{2^k}$ according to **lemma 1** in [3], if we put $k = 10$ the probability would be at least 99.90%.

References

[1] J. L. Gross and J. Yellen, *Handbook of graph theory.* CRC press, 2003.

[2] V. Goyal and J. Ackerman, "Introduction to Cryptography,Lecture 19: Zero-Knowledge Proofs I," 2018. URL: `https://www.cs.cmu.edu/~goyal/s18/15503/scribe_notes/lecture21.pdf`. Last visited on 2020/05/13.

[3] V. Goyal and J. Ackerman, "Introduction to Cryptography,Lecture 20: Zero-Knowledge Proofs II," 2018. URL: `https://www.cs.cmu.edu/~goyal/s18/15503/scribe_notes/lecture22.pdf`. Last visited on 2020/05/13.