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 22, 2020

1 Graph isomorphism

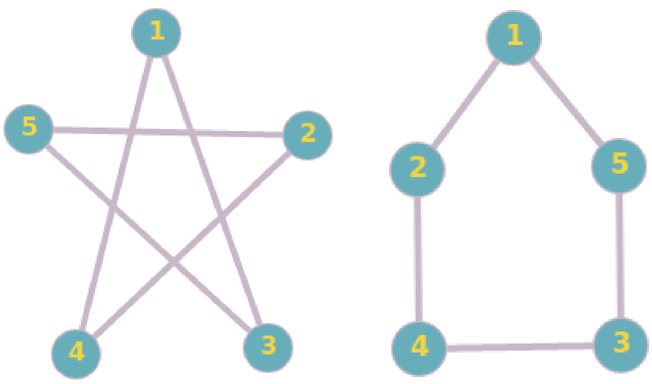
Definition 1 (Graph [1])**.** *An undirected graph consists of a set of vertices (nodes) V and a set of edges E.*
Two nodes u and v are said to be adjacent if there is an edge $(u, v) \in E$.

We can describe graph using its 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 a graph G respectively. Then, a pair of graphs (G_0, G_1) are **isomorphic** (denoted $G_0 \cong G_1$) if there exists a bijective map $\Pi : V(G_0) \longmapsto V(G_1)$such that $\forall x, y \in V(G_0), (x, y) \in E(G_0)$ 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 relabel vertices in one graph we get the other graph (with 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.

	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: adjacency matrix of G_0

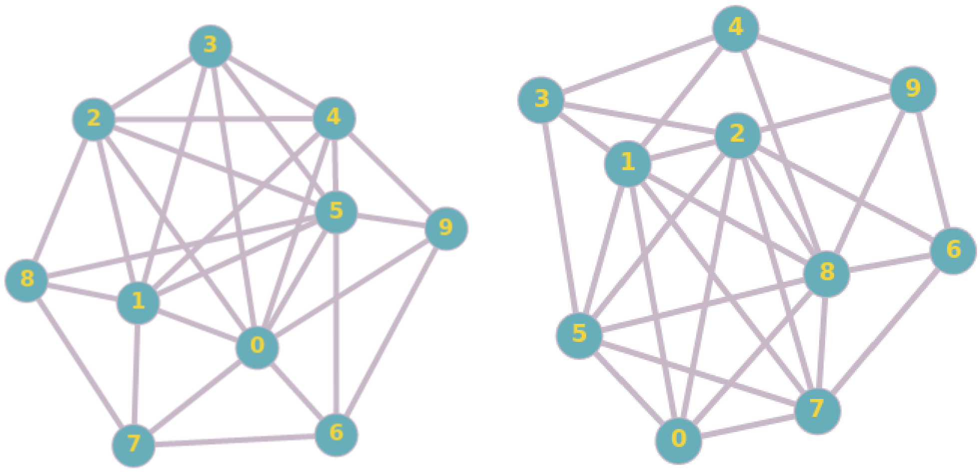
	1	**2**	**3**	**4**	**5**
1	0	1	0	0	1
2	1	0	0	1	0
3	0	0	0	1	1
4	0	1	1	0	0
5	1	0	1	0	0

Table 2: adjacency matrix of G_1

In Figure 1 G_1 is obtained, by relabeling the vertices of G_0 according to the following permutation: (1, 4, 5, 2, 3). This means that Node 3 in G_0 becomes Node 5 in G_1, Node 4 becomes Node 2.

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 (P) and verifier(V), P wants to prove to V he knows the secret Π without showing him what is Π exactly. In the previous example, we can see that it is easy to show if two graphs are isomorphic or not but this process isn't always simple; suppose we have two graphs each with 10 vertices and 28 edges, such as the graphs in Figure 2:



(a) G_0. (b) G_1.

Figure 2: Two isomorphic graphs.

Then ZKP can provide a protocol that P can prove to V he knows the secret Π without revealing Π itself.
The protocol is done by applying a random permutation (φ) on G_0 by P, 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 $H \cong G_0$ or $H \cong G_1$)

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 of n elements.
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 sends $H = \sigma(G_0)$.
2. **verifier** chooses ch randomly from $\{0, 1\}$ and sends it to the prover.
3. **prover** if $ch = 0$: then sends $\varphi = \sigma$ else sends $\varphi = \sigma \circ \Pi^{-1}$.
4. **verifier** output ACCEPT if $H = \varphi(G_{ch})$ else output is REJECT.

1.3 graph isomorphism (GI) decision problem

Definition 4 (graph isomorphism (GI) decision problem [3])**.** *Input: pair of two finite graphs (G_0, G_1).*
Output: ACCEPT if and only if they are isomorphic.
Complexity can be measured in the number of vertices.

1.3.1 The complexity of GI [3]

If we have two graphs and we want to check whether they are isomorphic or not; the brute force algorithm is to apply all possible permutation in S_n but this run in $n!$ and it's slower than any exponential algorithm.
There is a petter algorithm but not in polynomial-time, because no polynomial-time algorithm yet is known to solve it (if it was in P, then there is no need to apply ZKP; the verifier can check if graphs are isomorphic).
Moreover, GI also isn't considered to be NP-complete, This put GI in intermediate complexity problems (neither P nor NP-complete). In László Babai's paper [4] according to **Corollary, 1.1.2** GI can be solved in quasipolynomial time and it's run in $O(exp(log(n)^c))$

Theorem 5. *[3]*
The above protocol satisfies completeness, soundness $\frac{1}{2}$, and zero-knowledge.

Proof
Completeness. Completeness is quite straight forward, here we are dealing with honest prover $G_0 \cong G_1$ who knows Π, the verifier will check φ for both possible cases:

1. $(ch = 0)$: from step (1) P calculates $H = \sigma(G_0)$ and he should find a bijective map from H to G_0:
   it's clear that $\varphi = \sigma$ because $\sigma(G_0) = \sigma(G_0$
2. $(ch = 1)$: P has to find a map from H to G_1 or to show that $H \cong \varphi(G_1)$:
   We know that:
$$G_1 = \Pi(G_0) \quad (1)$$
$$H = \sigma(G_0) \quad (2)$$

After we comibne equation 1 and equation 2 :

$$H = \sigma(\Pi^{-1}(G_1))$$
$$H = (\sigma \circ \Pi^{-1})(G_1)$$

So

$$\varphi = \sigma \circ \Pi^{-1}$$

The V has to check that $H \cong \varphi(G_1)$:

$$\varphi(G_1) = \sigma \circ \Pi^{-1}(G_1) = (\sigma \circ \Pi^{-1} \circ \Pi)(G_0) = H$$

As we see if the prover knows Π he can always return the correct permutation.
Soundness $\frac{1}{2}$. For any prover P^* such that $G_0 \ncong G_1$ interacting with an honest verifier V^*, then P^* must submit some φ to the verifier, P^* can do something like:

1. choose random φ.
2. pick random H and send it to V.

Since $G_0 \ncong G_1$ then H such is isomorphic to G_0 or G_1, so whatever prover does, after it sends H, H cannot be isomorphic to both G_0 and G_1.
So whatever happens, honest verifier always has at least 1/2 chance to choose b that fails the prover, so:

$$Pr[P^* \text{ convinces } V \text{ that } G_0 \cong G_1] \leq 1/2$$

If we want to decrease the soundness to be negligible we have to repeat the verification procedure sufficiently many times, if we repeate it n times it will be soundness with 2^{-n} which is smaller than any polynomial-time algorithm.

zero-knowledge. We want to show that there exists a polynomial-time simulator S such that if G_0 and G_1 are isomorphic then for an honest prover and for every verifier V^*:

$$\text{Distribution of transcript } (P, V^*) = \text{Distribution of transcript } S(G_0, G_1, V^*) \tag{3}$$

How S works:

1. $S_n \longleftarrow$ random permutation from S_n.
2. $b \longleftarrow$ random bit from $\{0, 1\}$..
3. $H = \sigma(G_b)$.
4. Send H to V^* to get ch.
5. If $ch = b$, output σ and return the transcript, else loop.

Now we want to show equation 3 is satisfied:
First: the transcript is $\{H, ch(H, V^*), \sigma\}$ where H is a result of applying a random permutation on G_0 (same as a result of applying a random permutation on G_1), ch was generated by V^* from H and σ is a random permutation from S_n.
In the simulator, the distribution is generated in the same way, only there is always probability $\frac{1}{2}$ the simulator loop. But this probability ($\Pr[b \neq ch]$) is independent of H so it doesn't change the distribution of transcripts.
Then we want to show is a polynomial-time:

Let k be the number of loops, we want to show that S is polynomial-time by calculating the expected value of k:
When $k = 1 : E[k] = 1/2$
When $k = 2 : E[k] = 1/4$
When $k = 3 : E[k] = 1/8$ and so on
But this is a geometric distribution:

$$P(X = x) = q^{x-1}p$$

where $q = 1 - p$ With $E[k] = \sum_{i=1} i * pr(k = i) = \frac{1}{p} = 2$
Since the expected value of k while k is the number of loops is 2, then S is polynomial-time.

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] O. Goldreich, *Foundations of cryptography: volume 1, basic tools.* Cambridge university press, 2007.

[4] L. Babai, "Graph isomorphism in quasipolynomial time," in *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 684–697, 2016.