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 20, 2020

1 Graph isomorphism

[Jan: Better way to write this in latex: \begin{defn}[Graph \cite{bla}]]
[Jan: Clarify if your graphs are undirected.]

Definition 1. *(Graph)[1]: A graph consists of a set of vertices(nodes)* [Jan: Space before parenthesis] V *and a set of edges* E.
Two nodes u *and* v *is* [Jan: 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* $G_0 \simeq G_1$*) if there exists a map* $\Pi : V(G_0) \longmapsto V(G_1)$ [Jan: Say that map is bijective.] *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.*

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: 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

[Jan: adjacency]

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.

[Jan: Please define graph isomorphism (GI) decision problem formally. Input: pair of graphs (G_0, G_1). Accept if and only if they are isomorphic. Complexity can be measured in the number of vertices.]

[Jan: I think it would be a good idea to discuss complexity of GI before you jump to ZK proof. First, GI is not known to be in P (if it was in P, then there is no point of ZK proof: verifier can just check if graphs are isomorphic). On the other hand, it is believed not to be NP-complete, so it seems it is somehow a hard, but not very hard problem.]

[Jan: You might also mention that GI was recently proved to have "quasipolynomial"-time algorithm. That is, algorithm that runs in $O(\exp(\log(n)^c))$ for some. This is slower than polynomial time, because $n^c = \exp(c \log n)$, but is considered "closer" to polynomial than exponential, which is $\exp(cn)$. Reference is L. Babai, "Graph Isomorphism in Quasipolynomial Time". You can also read a newspaper article for more background: https://www.quantamagazine.org/algorithm-solves-graph-isomorphism-in-record-time-20151214/d]

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) [Jan: Prover is p or P?] and verifier(v), P wants to prove to V he knows the secret Π without showing him what is Π exactly. [Jan: Previous sentence should be divided into two parts.] From [Jan: from –> in] the previous example, we can see that it is easy to show if two graphs are isomorphic or not but this process isnt always simple; suppose we have two graphs each with 10 vertices and 28 edges, such as the graphs in Figure 2:

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, 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 $H \simeq G_1$)

[Jan: It is unclear who applies random permutation.]

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 [Jan: Usual notation is 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.

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

Proof

[Jan: From this point you are basically copying the lecture notes. Please delete and rewrite everything in your own words.] **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. $(ch = 0)$: 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 $\sigma(G_0) \simeq \sigma(G_0)$ [Jan: This should be $=$, not $\simeq$.]
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
   $$H = \varphi(G_0)$$
   then:
   $$H = \varphi(\Pi^{-1}(G_1)$$
   $$H = (\varphi \circ \Pi^{-1})(G_1)$$
   So the [Jan: Delete "the". Also the next equation is wrong.]
   $$\varphi = \varphi \circ \Pi^{-1}$$
   because
   $$\varphi(G_1) = \sigma \circ \Pi^{-1}(G_1) = \sigma(G_0) = H$$

Soundness $\frac{1}{2}$. if $G_0 \nsim G_1$ [Jan: This is wrong symbol, please look up the symbol that corresponds to the correct one.] then for every probabilistic polynomial time algorithm algorithm $\hat{P}$, there exist a negligible function $negl(\cdot)$ such that:

$$Pr[\hat{P} \text{ convinces } V \text{ that } G_0 \simeq G_1] \leq negl(\cdot)$$

but $negl(\cdot) = \frac{1}{2}$ because: [Jan: You shouldn't write it like that. 1/2 is not negligible. You are just proving soundness 1/2 instead of negligible function.]
suppose $G_0 \nsim G_1$, since $\simeq$ is transitive so for any graph G'either $G' \simeq G_0$ or $G' \simeq G_1$ not both; this indicates that the prover could pass only one of the tests not both.
In other words, the prover will pass the test when the verifier chooses $ch = 0$ 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 $G_0 \nsim G_1$.
since the verifier has only two possible choices $\{0,1\}$ then:

$$Pr[\hat{P} \text{ convinces } V \text{ that } G_0 \simeq G_1] \leq \frac{1}{2}$$

[Jan: This is not entirely correct. The argument should work for any prover, not just honest prover. So you cannot say things like "The prover will pass the test" when verifier chooses $ch = 0$, because maybe it's a different prover that wins only when $ch = 1$? Or maybe prover chooses b at random?] !

[Jan: A correct argument is: 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.] !

zero-knowledge. [Jan: Please delete and write again in your own words. Do not copy "incorrect attempt", "correct attempt", write as best as you can in your own words.] ! 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 a simulator S as follows,

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 it 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 of the 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 $\sigma(G_0) = \sigma(G_1)$

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,

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

Where σ_S is a permutation chosen 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 completes the proof.

For 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], e.g 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.