Chapter 2

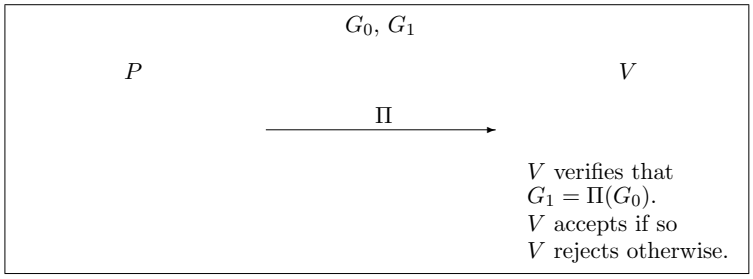
Interactive Proofs

An **Interactive Proof** is a protocol between two parties [1] a prover P and a veryfier V where P proves to V some assertion: that the string x belongs to some language L.

Take the problem of Graph-Isomorphism (G-ISO). Let $G_0 = \langle V, E_0 \rangle$ and $G_1 = \langle V, E_1 \rangle$ be two undirected graphs, where V is a set of vertices and E_i is a set of edges between the vertices of V. Furthermore $|E_0| = |E_1|$.

We say two graphs are isomorphic if and only if there exists a permutation Π of V such that for all vertices u and v in V, the edge (u, v) belongs to E_0 if and only if the edge $(\Pi(u), \Pi(v))$ belongs to E_1. We shall then write $G_0 \approx G_1$ or $G_1 = \Pi(G_0)$.

Let P and V be two Turing machines, then figure 2.1 is a protocol that lets the prover P prove to the verifier V that G_0 and G_1 are isomorphic.



Protocol 2.1: An IP protocol for Graph Isomorphism

Definition 1 (Interactive Proof) *A pair of Turing machines P and V, where machine P has no time or space limitation and machine V is probabilistic-polynomial time (PPT), constitute an IP system for the language $\mathcal{L}$ if the interaction between the two machines has the two following properties:*

***Completeness**: for all $x \in \mathcal{L}$*

$$\Pr[V \textit{ accepts } x \textit{ after interacting with } P] \geqslant \frac{2}{3} \tag{2.1}$$

***Soundness**: for all $x \notin \mathcal{L}$ and all P'*

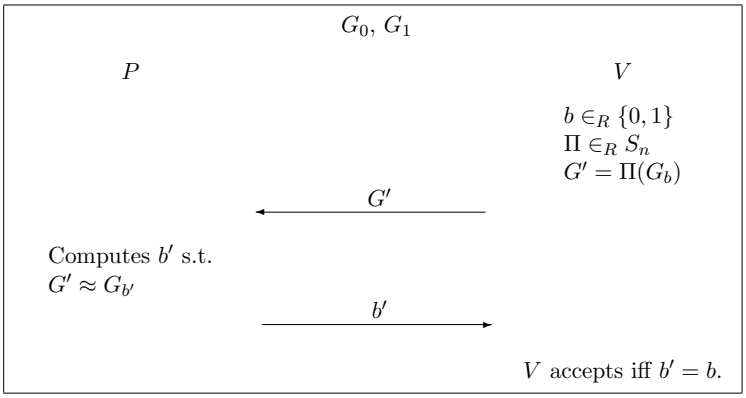
$$\Pr[V \textit{ accepts } x \textit{ after interacting with } P'] \leqslant \frac{1}{3}. \tag{2.2}$$

[1] Normally considered to be Turing Machines

A protocol is *complete* when the prover P can convince with high probability a verifier V of the veracity of the assertion $x \in \mathcal{L}$ if in fact $x \in \mathcal{L}$. In other words, the protocol is actually useful at proving to V that $x \in \mathcal{L}$. On the other hand, a protocol is *sound* when it ensures that the prover P cannot abuse V's gullibility. That is, if x is not in $\mathcal{L}$, then with high probability, if V follows the protocol, V will not believe the prover, whatever the prover does.

The set of languages for which there exists an Interactive Proof will be called **IP**. Obviously, G-ISO is in IP. The language G-ISO being in NP, there exists a witness (the permutation Π) that P can give to V and that V can verify all by himself. The same reasoning holds for all languages in NP, hence NP $\subseteq$ IP.

The problem of Graph-NonIsomorphism (GNI) is a more interesting example of an IP system as it is not known to be in NP (it is believed that no witness can be provided to the verifier by the prover). Two graphs, G_0 and G_1 are said to be non-isomorphic if no permutation exists such that $G_1 = \Pi(G_0)$. Figure 2.2 shows an IP protocol for Graph-nonisomophism. In that Protocol, if the two graphs G_0 and G_1 are isomorphic, then the prover P will not be able to guess b every time, hence, if V really chooses b randomly, then the probability that $b' = b$ is only one half.



Protocol 2.2: An IP protocol for Graph non-Isomorphism

On the other hand if the two graphs are nonisomorphic, it is always possible for a powerful prover P to find a unique b' for which there exists a permutation such that $G' = \Pi'(G_{b'})$, hence P can always win the game.

Hence

— Completeness: if $G_0 \not\approx G_1$ then $\Pr[(P,V)(G_0,G_1) = 1] = 1$

— Soundness: if $G_0 \approx G_1$ then $\forall P', \Pr[(P',V)(G_0,G_1) = 1] \leqslant \frac{1}{2}$.

Where $(P,V)(x) = 1$ means that after interacting with P on common input x, V accepts. As it is, this protocol does not satisfy definition 1, but this is only a technical issue. If P and V repeat the protocol twice (and V accept if and only if he accepts in both runs), then this is an IP protocol. By repeating the protocol, we mean that V's random bit and random permutation are chosen anew in this second round and independently from the first round. Hence, the two rounds being independent, soundness drops to one quarter (below one third) and therefore satisfies the definition.

By repeating several times this two-step protocol, the verifier can be convinced up to an exponentially low error probability of the validity of the proof. Note that GNI is not known to belong to NP but is in Co-NP and yet GNI belongs to IP. In fact, it turns out as proven by Adi Shamir that IP=PSPACE.

In general, we can always amplify the probability that V accepts or rejects by repeating a protocol. In the case where the completeness is smaller than 1, the verifier will run a given protocol which belongs to IP k times recording each time whether he accepted or rejected the prover's claim. At the end of the k rounds, the verifier takes a majority vote : he accepts the prover's claim, if at least $\lfloor k/2 \rfloor + 1$ time, he accepted the IP proof and rejects otherwise. Now let's consider the case where x does not belong to the language but the prover is trying to prove otherwise. What is the probability of error p_e, that is the probability that V accepts x as belonging to $\mathcal{L}$?

The verifier will accept if and only if he accpeted in at least $\lfloor k/2 \rfloor + 1$ rounds. Hence, since the soundness probability is no larger than one third, and that all k rounds are independent, the expected value of the number of times V accepted x over k runs is at most $k/3$. To err, it must be that the number of times where V accepts is deviating by at least $k/6 + 1$ from its expected value $k/3$. But by the Chernoff bound (Formula 3), the probability that this happens is exponentially small for sufficiently large k.

If $x \in \mathcal{L}$, then if $\Pr[(P,V)(x) = 1] \geqslant \frac{2}{3}$ we can amplify to $\Pr[(P,V)(x) = 1] \geqslant 1-\varepsilon$ where $\varepsilon = \frac{1}{poly(|x|)}$ or more precisely $\varepsilon = \frac{1}{exp(|x|)}$.

Definition 2 (Negligible function) *A function $\mu : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for every positive polynomial $p(\cdot)$ there exists an n_0 such that for all $n > n_0$ we have*

$$\mu(n) < \frac{1}{p(n)}.$$

Interactive Proofs where $\Pr[V \text{accepts } x \text{ after interacting with } P] = 1$ for all $x \in \mathcal{L}$ can be amplified faster by taking advantage of this special property. If we repeat k times and accept if and only if all executions accept then the completeness probability remains one while the soundness probability drops to p_s^k if p_s was the soundness probability of a single run. Notice that we have used a distinct accpeting criteria. This technique can be used whenever $p_s < 1$, not only for $p_s < \frac{1}{3}$.

2.1 Proof of Knowledge

A proof of knowledge is a variation on IP protocols. In this case, the goal is to convince a verifier that the prover *knows* something. For example, the prover might want to convince the verifier that he *knows* the two prime factors, p and q, of a given RSA number $n = pq$. The difference with IP protocols is that we need to formally define a notion of *knowledge* to exclude the possibility that the prover proves that n has exactly two prime factors without knowing them.

Consider a relation $\mathcal{R}$ constituted of pairs (x, w), where w is a witness. For example, the relation for RSA numbers is the set $\{(n, (p, q))|n = pq, p \text{ and } q \text{ are prime}\}$; or a pair which constitutes a witness to the RSA-Number language. Proving that n is composite does not require knowledge of p and q because that's exactly the outcome of Rabin's primality test. Proving that n has exactly two prime factors seems much harder to do without knowledge of the explicit factors.

Another good example is $((G_0, G_1), \pi)$ where (G_0, G_1) forms the language of pairs of isomorphic graphs, and π is the permutation such that $G_1 = \pi(G_0)$. In this last case, a proof of knowledge is kind of an IP protocol such that at the end, not only is the verifier convinced that $G_0 \approx G_1$, but the verifier is also convinced that indeed the prover knows the permutation π.[2]

The language $\mathcal{L}_{\mathcal{R}}$ associated with $\mathcal{R}$ is the set of all x such that there exists a witness for which $\mathcal{R}(x, w) = 1$: $\mathcal{L}_{\mathcal{R}} \triangleq \{x | \exists w \text{ s.t. } \mathcal{R}(x, w) = 1\}$. An *explicit* proof of knowledge, would be, for instance, the interaction where P provides V with the witness w (as in protocol 2.1). However, the notion of proof of knowledge is a lot more subtle. If w appears in the conversation between P and V in an implicit form, say $\bar{w}$ for instance, it is still a proof of knowledge. That's because there exists an efficient algorithm that computes w from $\bar{w}$ (by flipping all the bits), and more precisely w can efficiently be computed from the conversation between P and V. We shall call this algorithm the **Knowledge Extractor**. To give a very general notion of *implicit knowledge* we push the definition of Knowledge Extractor even further: we say that an efficient algorithm K_P is a knowledge extractor for a relation $\mathcal{R}$ if given $x \in \mathcal{L}_{\mathcal{R}}$, $\mathcal{R}(x, K_{P(w)}(x)) = 1$ with non-negligeable probablity. When no such witness exists, we don't care what $K_{P(w)}$ outputs. So here is the formal definition:

Definition 3 (Proof of Knowledge) *A pair of (Probabilistic Polynomial-time) Turing machines, P and V, constitute an interactive proof of knowledge for a relation $\mathcal{R}$ if the following two conditions hold*

***Completeness**: for all $(x, w) \in \mathcal{R}$*

$$\Pr[(P(w), V)(x) = 1] \geqslant 1 - \nu(|x|) \tag{2.3}$$

***Soundness**: for all prover P' there exists an efficient [3] $K_{P'}$ such that for all x and all w we have*

$$\Pr[\mathcal{R}(x, K_{P'(w)}(x)) = 1] \geqslant \Pr[(P'(w), V)(x) = 1] - \kappa(|x|), \tag{2.4}$$

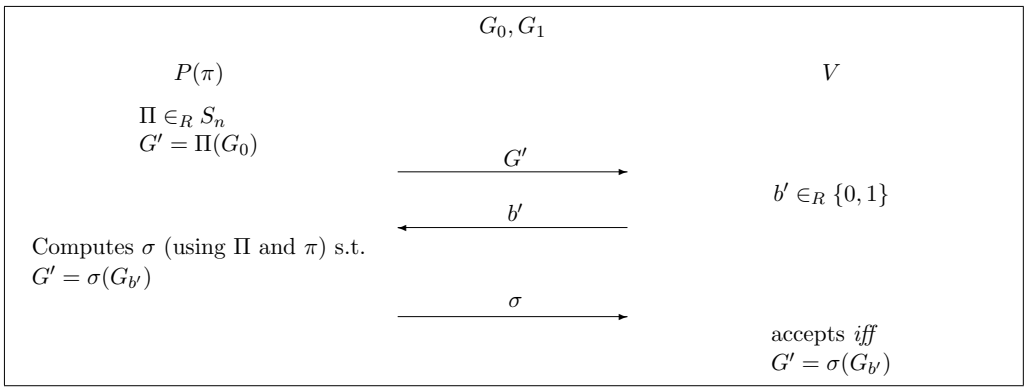
where κ and ν are negligeable error-functions.

Notice that when $x \notin \mathcal{L}_{\mathcal{R}}$ we have that $\Pr[\mathcal{R}(x, K_{P'(w)}(x)) = 1] = 0$ and therefore $\kappa(|x|) \geqslant \Pr[(P'(w), V)(x) = 1]$ which is a strong soundness condition of IP protocols. Therefore, all proofs of knowledge are IP protocols but not the other way around... Intuitively, this definition states that whenever a prover P' given a candidate-witness w manages to convince a verifier that $x \in \mathcal{L}_{\mathcal{R}}$ then we can obtain a witness (not necessarily w) by running $K_{P'(w)}(x)$ with similar probability in an amount of time not too much bigger...

As mentioned before, protocol 2.1 is an explicit proof of knowledge of the isomorphism between G_0 and G_1. The following protocol 2.3 will be the basis for an implicit proof of knowledge for the same witness.

[2] The fact that a prover can convince a verifier that two graphs are isomorphic does not imply that the prover knows the permuatation π, even if it is hard to imagine otherwise.

[3] We allow $K_{P'(w)}$ to run for polynomially more time than $P'(w)$. One simple way of enforcing this condition is to define $K_{P'}$ as a poynomial-time algorithm that runs P' as a black-box.

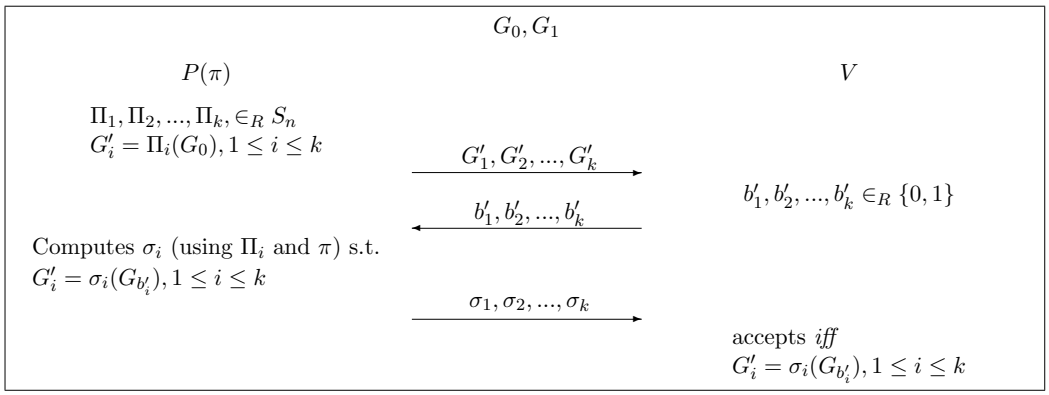


Protocol 2.3: Basic protocol for an implicit proof of knowledge of Isomorphism between two Graphs

First, check that protocol 2.3 is an IP protocol for GI. To convince yourself of this fact, check that the completeness condition is perfect: $\Pr[(P,V)(x)=1]=1$. Whereas the IP soundness condition is also satisfied, when G_0 and G_1 are not isomorphic, $\Pr[(P'(\pi),V)(x)=1] \leq \frac{1}{2}$ whatever P' does (the last step of the protocol can be acomplished properly by P' for only one value of b' because G' cannot be isomorphic to two graphs that are not isomorphic to each other).

As is, the above protocol does not constitute a proof of knowledge according to our above definition. That's because the soundness success probability of a prover P' that knows no witness can be as much as $\frac{1}{2}$. This would imply that the knowledge extractor K_P should have $\Pr[\mathcal{R}(x, K_{P'}(x))=1] \geqslant \frac{1}{2} - \kappa(|x|) > 0$ which is impossible when $x \notin \mathcal{L}$.

Consider instead, executing k independant copies of protocol 2.3 for the same input graphs G_0, G_1 as follows:



Protocol 2.4: Implicit proof of knowledge of Isomorphism between two Graphs

The resulting protocol still has completeness probability $\Pr[(P(\pi),V)(x)=1]=1$. Whereas the strong IP soundness condition is also satisfied, when G_0 and G_1 are not isomorphic, $\Pr[(P'(\pi),V)(x)=1] \leq \frac{1}{2^k}$ whatever P' does. Moreover, the stronger proof of knowledge soundness condition with $\kappa(|x|) = \frac{1}{2^k}$ is also satisfied:

$$\Pr[\mathcal{R}(x, K_{P'}(x))=1] \geqslant \Pr[(P'(\pi),V)(x)=1] - \kappa(|x|).$$

When $x \notin \mathcal{L}$, we get $\Pr[\mathcal{R}(x, K_{P'}(x))=1] = 0$ which yields $\Pr[(P'(\pi),V)(x)=1] \leq \frac{1}{2^k}$. When $x \in \mathcal{L}$, we get $\Pr[\mathcal{R}(x, K_{P'}(x))=1] \geqslant \epsilon(x) - \kappa(|x|)$ where $\epsilon(x) = \Pr[(P'(\pi),V)(x)=1]$. If P' is

so dum that $\epsilon(x)$ is negligeable, then $\epsilon(x) - \kappa(|x|)$ is negligeable (or even negative!!) and therefore $\Pr[(P'(\pi), V)(x) = 1]$ may also be negligeable. In this case, there is no requirements on $K_{P'}$ to succeed at all. If however, P' is such that $\epsilon(x)$ is not negligeable then we expect that $K_{P'}$ will succeed with probability at least $\epsilon(x)$. We now present a polynomial time $K_{P'}$ that succeeds in producing a witness with probability nearly 1 under the condition that $\epsilon(x)$ is not negligeable.

The purpose of the knowledge extractor $K_{P'}$ is to obtain two executions of P' such that the graph G' submitted in both runs are identical but the choice bit b' of the extractor are distinct. Here is the knowledge extractor for the Protocol 2.4

1. Initialize P': copy fresh random bits to the prover's random tape and fill up the Auxiliary-Input tape with a witness π if any.
2. Run P' until it sends $G'_1, G'_2, ..., G'_k$
3. Send random $b'_1, b'_2, ..., b'_k$ to P' and wait for $\sigma_1, \sigma_2, ..., \sigma_k$
4. Store $d'_1, d'_2, ..., d'_k \leftarrow b'_1, b'_2, ..., b'_k$ and $\gamma_1, \gamma_2, ..., \gamma_k \leftarrow \sigma_1, \sigma_2, ..., \sigma_k$
5. Restart P' as in step 2 and run it until it sends $G'_1, G'_2, ..., G'_k$ again
6. Send random $b'_1, b'_2, ..., b'_k$ to P' and wait for $\sigma_1, \sigma_2, ..., \sigma_k$
7. Let i be such that $b'_i \neq d'_i$. Compute and output $\pi = \sigma_i^{-1^{b'_i}} \gamma_i^{-1^{d'_i}}$. STOP
8. Go to step 5.

Extractor 2.1: Sketch of the Knowledge Extractor for Graph Isomorphism

The Extractor above is just a sketch because many subtleties have to be considered. As written, we assume that P' is always answering valid $\sigma_1, \sigma_2, ..., \sigma_k$ for arbitrary $G'_1, G'_2, ..., G'_k$ and $b'_1, b'_2, ..., b'_k$. Let p be the probability that P' actually answers valid $\sigma_1, \sigma_2, ..., \sigma_k$ at step 3. If the first time the extractor tries this Step, P' answers with invalid $\sigma_1, \sigma_2, ..., \sigma_k$ then the extractor aborts. The running time of this possibility is independent of p. If the first time the extractor tries this Step, P' answers with valid $\sigma_1, \sigma_2, ..., \sigma_k$ then the extractor should extract a withness. It will do that by finding another set of $b'_1, b'_2, ..., b'_k$ for the same $G'_1, G'_2, ..., G'_k$ that produces valid $\sigma_1, \sigma_2, ..., \sigma_k$. If p was the probability of hitting a situation that lead P' to issue valid $\sigma_1, \sigma_2, ..., \sigma_k$ then the probability that this happens again is still p. Therefore, the expected number of tries until this situation happens again is $1/p$. Therefore the expected running time to produce a witness is $p \times 1/p \times t$ where t is the time to run one test at Steps 5–6. This expected running time is again independent of p.

Nevertheless, this argument is slightly wrong: the probability p is computed as an expected value over the choices of $b'_1, b'_2, ..., b'_k$. The Extractor fails if the Prover completes the protocol consistently on the same sequence $b'_1, b'_2, ..., b'_k$. The extreme example is if the Prover solely completes on a single sequence $b'_1, b'_2, ..., b'_k$. In this case, the Extractor would not succeed because he never finds a second sequence of $b'_1, b'_2, ..., b'_k$'s. To escape this undesired situation, the extractor always runs in parallel to Step 5 a complete enumeration of all possible permutations until he finds one that maps G_0 to G_1. If by any chance, it runs long enough that this enumeration completes then the Extractor output the computer permutation. This case will arise with probability no greater than 2^{-k}, when no other sequence succeeds. All in all, this will double the expected running time of any situation where ultimately a second sequence is found, while it will force termination of any situation where no other sequence is actually found. This will only increase the expected running time of the Extractor by a factor of two as long as 2^k is larger than the number of permutations.

Problems

2.1 Formalize the argument for majority amplification using the Chernoff bound. Do this for both Soundness and completeness.

2.2 Let n be a composite number chosen by the Prover. Let y be a quadratic non-residu modulo n (with Jacobi symbol $+1$) also chosen by the Prover. Give an interactive proof for the language

$$N\text{-}QNR = \{(n, y) | n \text{ is composite and } y \text{ is a quadratic non-residu mod } n \text{ (with Jacobi symbol } +1)\}.$$

2.3 Let n be a composite number and y be a quadratic non-residu modulo n (with Jacobi symbol $+1$) chosen by the Prover. Let $z \in QR_n \bigcup yQR_n$ where we define $yQR_n = \{z \in QNR_n | z/y \in QR_n\}$. Give a proof of knowledge for the residuosity of z where the prover convinces the verifier that he knows whether z is a quadratic residue or non-residue but does not tell the verifier which it is...

2.4 Let p be a public prime number with publicly known factorization of $p-1$ and g be a public primitive element. Let (a, b, c) be a triple of elements from $\mathbb{Z}_p^*$ chosen by the Prover as $a = g^x \bmod p$, $b = g^y \bmod p$ and $c = g^{xy+z} \bmod p$ for some x, y, z. Give a proof of knowledge for the Decisional Diffie-Hellman set such that $(a, b, c) \in DDH$ if and only if there exists x, y such that $a = g^x \bmod p$, $b = g^y \bmod p$ and $c = g^{xy} \bmod p$. The prover convinces the verifier that he knows whether (a, b, c) is a valid DDH triple or not but does not tell the verifier which it is...

Hint: Give a proof of knowledge of z without disclosing it.

2.5 Let p be a public prime number with publicly known factorization of $p-1$ and g be a public primitive element. Suppose the Prover has free access to an oracle to decide the DDH. Show how to use the protocol of 2.4 in order to convince the verifier that he is able to decide DDH. Make sure that the verifier never learns whether a triple (a, b, c) is a valid DDH-triple or not, unless he already knows...

Chapter 3

Zero-Knowledge

The proofs seen in the last chapter are interesting, but not so much cryptographically as the verifier may learn everything. Once the verifier learns the witness to an NP assertion, then he can himself prove the assertion to anyone else. In this chapter we shall develop another property for protocols. We shall require that not only do they convince the verifier of the validity of the assertion but that the verifier learns *nothing else* from the interaction with the prover. After the proof, the verifier is really convinced of the validity of the assertion that the prover was in fact proving. Ideally, the verifier should learn nothing; so little that the verifier cannot even convince his verifier-friends that he indeed talked to the Prover (or anyone who knew a witness). We shall do that via a very profound technique which is known as *simulation*. The result of this definition will be a powerful new tool.

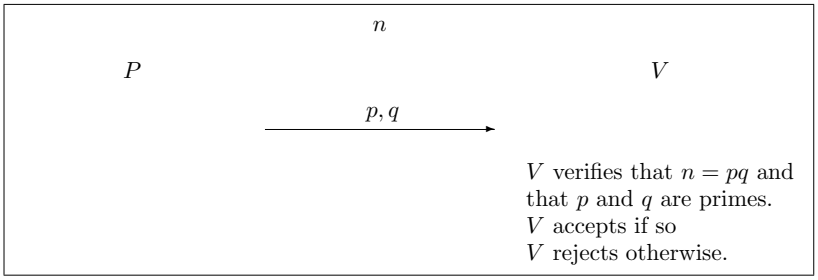
We shall simulate the interaction of V', for any V', even a potentially dishonest V', with P without having access to P or his knowledge. Let $VIEW[(P,V')(x)]$ be the random variable associated with all the possible transcripts of all the messages exchanged between P and V' and all of V''s random coins and x (basically, anything that V' knows and sees). We shall call the *transcript* the list of all messages exchanged between P and V. Clearly, the View contains more information than the transcript.

Definition 4 *An IP protocol is Zero-Knowledge (ZK) if for all verifiers V' there exits a probabilistic polynomial-time machine $S_{V'}$ such that for all $x \in \mathcal{L}$*

$$VIEW[(P,V')(x)] =_0 S_{V'}(x). \tag{3.1}$$

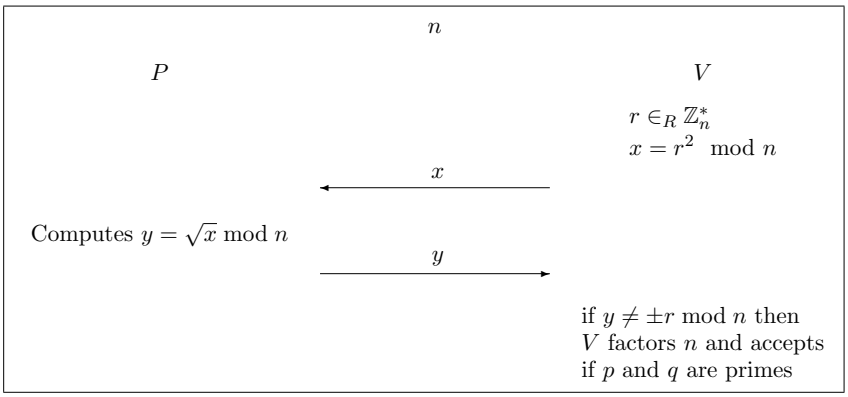
In this definition, $S_{V'}(x)$ is the random variable that represents the output of the simulator S that has access to V' (as a black-box or an oracle) and the notation $X =_0 Y$ means that the two random variables X and Y have the exact same distribution. In layman words, an IP protocol is Zero-Knowledge if there exists a PPT machine that can simulate the interaction between P and V' without knowing anything specific about x or $\mathcal{L}$ (a witness for example).

Let us look at factorization as a first (*bad*) example. Here, in Protocol 3.1, $n = pq$, or the language $\mathcal{L}$ is the set of all numbers with exactly two prime factors.



Protocol 3.1: A first attemp at Zero-knowledge

Of course, Protocol 3.1 cannot be Zero-Knowledge as no simulator that does not know p and q, which for large n could be hard to obtain, can simulate the interaction between any V' and P efficiently. But even worse, V learns the factorization of n. As we shall see, a protocol is Zero-Knowledge if the verifier learns nothing but the validity of the assertion even if the verifier does not behave honestly. This is proven by exhibiting a simulator whose outputs is distributed as the View of V'. But how can this be simulated without knowing p and q? In fact, Protocol 3.1 has the intent of giving p and q to V. In this case obviously V learns more than just the fact that n has two factors. We can try a second approach where the factors of n do not appear explicitely on the transcript, see protocol 3.2.



Protocol 3.2: A second attemp at Zero-knowledge

But alas, even if p and q are per say not on the transcript, V still learns the factorization and no polynomial-time S should be able to simulate this protocol as extracting square roots is as hard as factoring (if P can reliably accomplish the expected task then V can use him to find p and q efficiently).

Take a look at the representation of the Verifier of Figure 3.3.

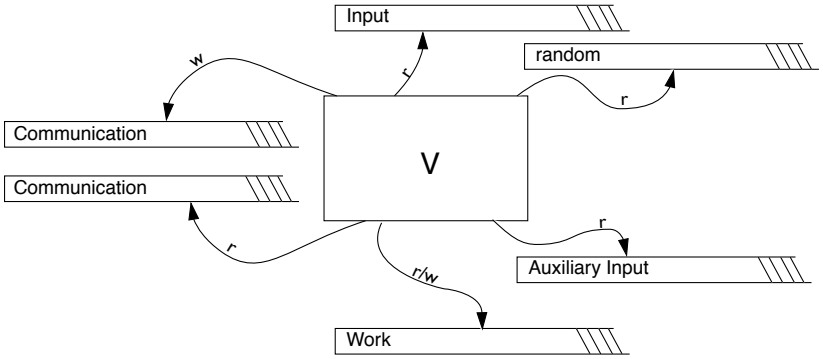


Figure 3.3: Schematization of the Verifier Turing Machine

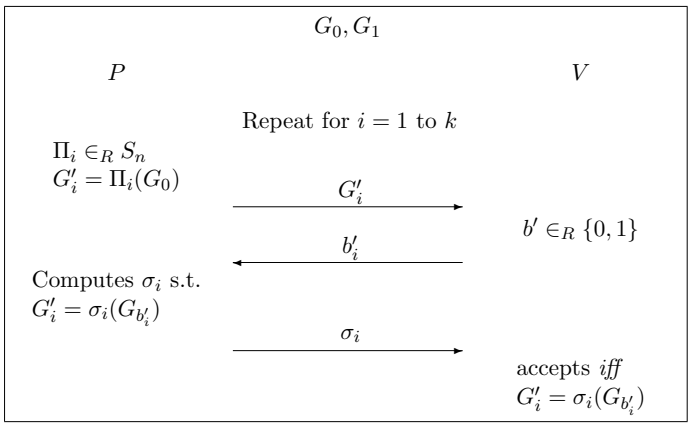
This Turing machine has 6 tapes: two of which are used to communicate with the prover, one contains the input, one is prefllled with unbiased random bits, one is a work tape and the last one contains the auxiliary input. The auxiliary input represents some prior knowledge that the verifier might posess: For example, the verifier might know that p in Protocol 3.1 is congruent to 3 mod 4 with probability one quarter. The definition of Zero-Knowledge should hold even if V' already knows something, that is V' should not know more after the protocol than before the protocol and whatever (computational) uncertainty he had about the world has not changed. Look at protocol 2.2 for Graph-Non-Isomorphism. Imagine that V' in this protocol knows a third graph G_2 and is trying to know to which graph, G_0 or G_1, G_2 is isomorphic. Then, by cheating and sending graph G_2 instead of choosing between G_0 and G_1 randomly and permuting it, then V' could learn something it is not suppose to learn that is $G_2 \approx G_b$ for $b \in \{0, 1\}$. Note that we never claimed that Protocol 2.2 was safe in that respect.

We can revisit the definition of Zero-Knowledge to protect the prover against such bad behaviors from V'.

Definition 5 *An IP protocol is Zero-Knowledge (ZK) if for all verifiers V' there exits a probabilistic polynomial-time machine $S_{V'}$ such that for every $x \in \mathcal{L}$ and for all auxiliary input $\aleph$ we have*

$$VIEW[(P, V'(\aleph))(x)] =_0 S_{V'(\aleph)}(x). \tag{3.2}$$

Let us revisit Protocol 2.3 that we used in the context of Proof of Knowledge previously:



Protocol 3.4: A successful attempt at Zero-Knowledge: Graph-Isomorphism

Note that in this protocol, σ is easy to compute for the prover: if $b' = 0$, then $\sigma = \Pi$ and if $b' = 1$ then $\sigma = \Pi \circ \rho$ where ρ is a permutation such that $G_1 = \rho(G_0)$.

This protocol obiously belongs to IP if it is repeated k times:

— Completeness: If $G_0 \approx G_1$, then $\Pr[(P, V)(x) = 1] = 1$,

— Soundness: If $G_0 \not\approx G_1$, then $\forall P' \Pr[(P', V)(x) = 1] \leqslant \frac{1}{2^k}$.

But contrary to what we did in the proof of knowledge of Protocol 2.4, to obtain Zero-Knowledge the repetitions in Protocol 3.4 are *sequential* instead of in parallel.

To prove that this protocol is Zero-Knowledge, we only have to present a simulator, see Figure 3.5.

1. Copy fresh random bits to the verifier's random tape and fill up the Auxiliary-Input tape.
2. Pick a bit $c \in_R \{0,1\}$ and permutation $\rho \in_R S_n$.
3. Compute $G' = \rho(G_c)$
4. Run the verifier V on input (G_0, G_1) and G' and wait for his reply b'.
5. (a) If $b' = c$, then send ρ and finish simulation
   (b) if $b' \neq c$, then rewind V as if nothing had happened and go back to step 2

Simulator 3.5: Simulator for Graph-Isomorphism

Here are a few comments on the simulator for Graph-Isomorphism:

1. As long as V runs in polynomial time, the simulator runs in expected polynomial time.
2. The simulator has access to all the tapes of V in read-write mode but does not have access to the internal logic of V.
3. The verifier has a reset switch to which S has access, using this button the simulator can rewind V to its initial state (right after step 1 in Simulator 3.5).

4. The distribution of G' does not depend on c.
5. Since $G_0 \approx G_1$, V' cannot know if G' was generated applying a permutation to G_0 or G_1.
6. The two distributions $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are identical.
7. The bit c never appears on the transcript.

To enlighten some of these comments, let us prove that $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are identically distributed. A View is the following tuple $(G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1), \ldots, (G'_k, b'_k, \sigma_k))$, where r is the content of V''s random tape which is constituted of uniform random bits and $\aleph$, the auxiliary input, is the same in both cases:

We shall write, for $j \leqslant k$,

$$(G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1), \ldots, (G'_j, b'_j, \sigma_j))^S$$

for a partial View generated by the simulator and

$$(G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1), \ldots, (G'_j, b'_j, \sigma_j))^P$$

for a partial View generated by a real interaction between P and V'. Note that these Views are really random variables. What we shall prove is that for any j the distribution of the real and of the simulated Views are the same.

The first thing to notice is that at any given step j, V' is really just a deterministic function of the prefix of the partial View up to that step. So before that first step we have

$$(G_0, G_1, r, \aleph)^S = (G_0, G_1, r, \aleph)^P \tag{3.3}$$

The next item on the View is G'_1. But the honest prover and the simulator create that graph according to the same distribution: that is they use the graph G_0 (or G_1 at random for the simulator) and compute a random isomorphism of it (by choosing uniformly a random permutation). But since G_0 and G_1 are isomorphic, a random permutation of either is the same. Hence

$$(G_0, G_1, r, \aleph, (G'_1))^S =_0 (G_0, G_1, r, \aleph, (G'_1))^P. \tag{3.4}$$

The next element, which is V''s challenge to G'_1 is a deterministic function of the View up to that point, as V' uses the bits in r as randomness. Hence Equation (3.4) means that

$$V'((G_0, G_1, r, \aleph, (G'_1))^P) =_0 V'((G_0, G_1, r, \aleph, (G'_1))^S),$$

hence

$$(G_0, G_1, r, \aleph, (G'_1, b'_1))^S =_0 (G_0, G_1, r, \aleph, (G'_1, b'_1))^P. \tag{3.5}$$

And finally as both the prover and the simulator chose their permutation to create G'_1 at random uniformly, and that for a fixed permutation γ, $\{\Pi \circ \gamma\}_\Pi = S_n$ the probability to see a given permutation is the same in both Views

$$(G_0, G_1, r, \aleph, (G'_1, b'_1, \rho_1))^S =_0 (G_0, G_1, r, \aleph, (G'_1, b'_1, \sigma_1))^P. \tag{3.6}$$

However, the Simulator is only able to complete this part of the View when $b'_1 = c$. On the contrary, the Prover can always complete it because he knows an isomorphism between G_0 and G_1. Nevertheless, the choice of b'_1 by V' is completely independent of c since the distribution of G'_1 is

the same for both values of c. This observation is absolutely crucial. This is the reason why the Simulator may fail and that doing so will not skew the distribution of the simulated View. If the Simulator "forgets" that he tried and fails (rewinds) and try again, the distribution will be exactly the same again next time... Notice that if we ran the proof in parallel, the success probability of the simulator at Step 5a would be 2^{-k} which would lead its running time to be exponential. That is why we favor runing the protocol sequentially.

This reasoning is for the first triple of the View. But the same argument will hold for all steps j larger than 1 as well (as long as the rewinding used by the simulator loops back to the most recent execution of Step 2. Once an iteration is successfully simulated, the simulator never tries to undo it.). Hence

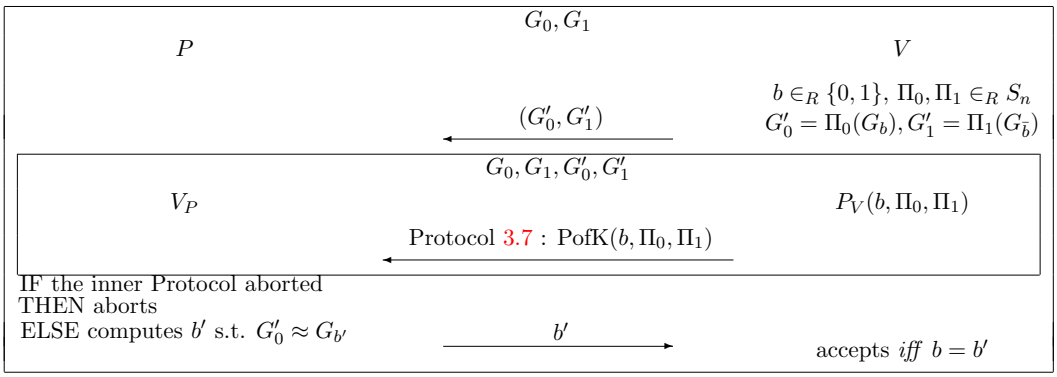
$$VIEW[(P, V'(\aleph))(x)] = S_{V'(\aleph)(x)},$$

that the simulator outputs can be the View of the verifier and this happens with the exact same probability.

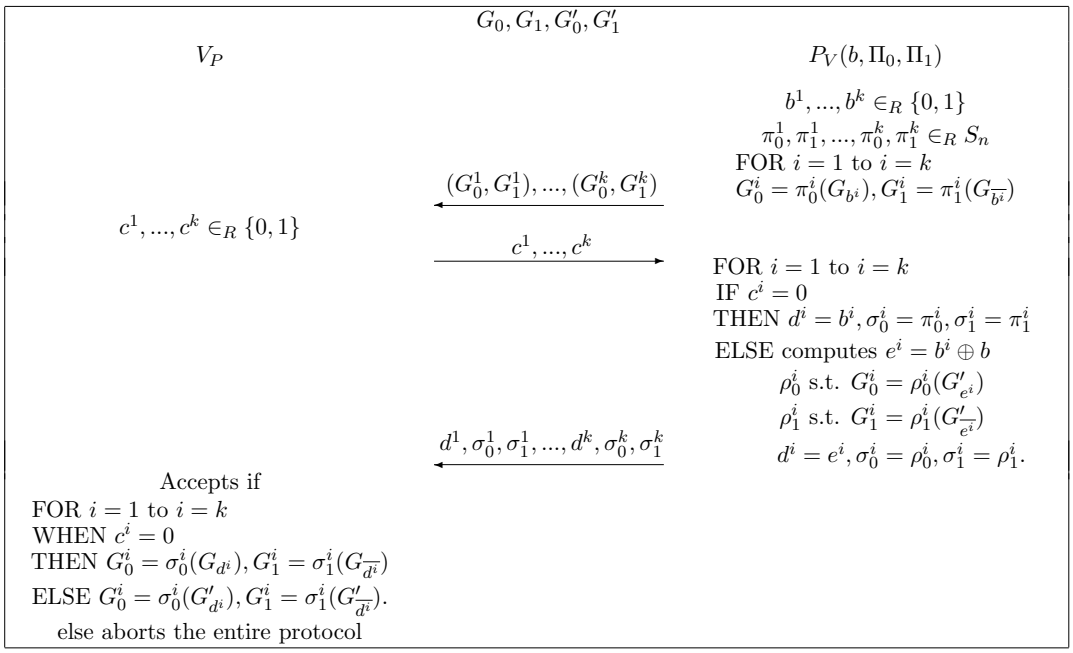
3.1 Graph Non-Isomorphism

Let us discuss a more elaborate example : a Zero-Knowledge Interactive Proof for Graph-Non-Isormophism.

Take a look back at Protocol 2.2. As we have already discussed, this protocol cannot be Zero-Knowledge as V' might not compute G' honestly. In fact, a clever V' could feed a third graph G_2 and learn whether G_2 is isomorphic to G_0 or G_1. One solution would be to force V' to compute G' honestly. Fortunately we can use a variation on the Proof of Knowledge for Graph-Isomorphism that we already know (Protocol 2.4) in order to achieve this. The following Protocol is an improvement of Protocol 2.2 by introducing an extra Step represented by the internal box.



Protocol 3.6: Zero-Knowledge proof for Graph-Non-Isomorphism



Protocol 3.7: Proof of Knowledge of (b, Π_0, Π_1)

The Internal box is really a Proof of Knowledge going backwards : the Verifier $P_V(b, \Pi_0, \Pi_1)$ proves to the prover V_P that he knows b such that $(G'_b, G'_{\overline{b}}) = (\Pi_0(G_0), \Pi_1(G_1))$. First, let's check that the modified Protocol is still an Interactive Proof for Graph Non-Isomorphism. It should be obvious that the completeness of the outer Protocol still holds : the sub-Protocol does not change that. We have to work harder to prove that the soundness of the outer Protocol still holds. Soundness could be lost if somehow b was communicated to V_P during the sub-protocol. If we prove that everything that happens inside the internal box is statistically independent from the bit b, we demonstrate that this issue is not a problem. If that is the case, then the internal protocol carries no information about b and hence, cannot help P' cheat soundness if the two graphs are in fact isomorphic.

It is interesting to notice that we do not need to guarantee that the internal protocol carries no information about b if the two graphs are in fact non-isomorphic. Moreover, in the case where the graphs are isomorphic, we do not require the sub-protocol to be zero-knowledge, but simply carries no information about b. If P is infinitely powerful, as we often assume in Interactive Proofs, the zero-knowledge property (efficiency of a simulator) of the sub-Protocol would be totally irrelevant.

For the sake of argument, let us assume that G_0 and G_1 are isomorphic to start with. Then, for honest P_V, all graphs in the sub-Protocol will be isomorphic to each other. Hence, even for an infinitely powerful prover $V_{P'}$ it is impossible to compute b nor any of the b_i's from (G_0, G_1), (G'_0, G'_1) and all the (G_0^i, G_1^i) because they are all isomorphic to G_0 (and to G_1). Then answer to c_i is also totally independent from b as b^i, π_0^i and π_1^i are all chosen at random independently from b. The same holds for e^i, ρ_0^i and ρ_1^i, as e^i is really a one-time-pad encryption of b using b^i as a key. But since nothing was learned about the b_i's from (G_0, G_1), (G'_0, G'_1) and all the (G_0^i, G_1^i)'s, b is perfectly encrypted. Hence we conclude that when the time comes to compute b' even an infinitely P' cannot have better probability of guessing b than one half (that is if $G_0 \approx G_1$).

After checking that the modified Protocol is still an Interactive Proof, let's consider its Zero-Knowledge aspect. Here is the simulator for the Protocol 3.6

1. Initialize V: copy fresh random bits to the verifier's random tape and fill up the Auxiliary-Input tape.
2. Run V until it sends (G_0^1, G_1^1)
3. Run the Knowledge-Extractor on $P_V(b, \Pi_0, \Pi_1)$ to obtain (b, Π_0, Π_1).
4. Run the full protocol as honest P would until V waits for b' and give b' as extracted in Step 3.

Simulator 3.8: Simulator for Graph Non-Isomorphism

This simulator is written with honest V in mind, but something could go wrong with this simulator if V' is cheating. What if there is no bit b such that $(G_0, G_1) \approx (G'_b, G'_{\overline{b}})$, or $P_{V'}$ does not know it. The simulator will most likely discover this at step 3 because the knowledge extractor will fail to obtain such values. There is, however, a small probability that V_P accepts and then tells V' what he wants to know. Although that probability is low, it is not zero. Hence, with very low probability, V could abuse P and learn something which he is not supposed to.

How is the simulator suppose to deal with this? Well, after discovering that V' is trying to cheat, S should simply try to finish the protocol (step 4) and hope to catch V' in the cheating. If V' is lucky, that is, if V' successfully cheats every round, S will be forced to output $\perp$ (that is S does not output a View) or S should go ahead and compute b' such that $(G_0, G_1) \approx (G_{b'}, G_{\overline{b'}})$.

In the latter case, the simulator will run in expected polynomial time since for all runs where V' gets caught cheating the simulator runs in polynomial time (as long as V' runs in polynomial time), but for the case in which V' can cheat without being caught by S, which happens with negligible probability (that is lower than 2^{-k}), then S will run in exponential time. If $k \in \omega(\log t(n))$, where n is the number of vertices of G_0 and G_1 and computing the isomorphism between G_0 and G'_b belongs to $\mathcal{O}(t(n))$, then on average, S would still run in expected polynomial time because $2^{-k}t(n)$ can be made less than 1 for an appropriate choice of k (see the analysis of Protocol 2.1).

In the former case, were S outputs $\perp$, the transcript where V' can cheat and P answers anyway will never be produced, hence $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ will not be distributed identically. Although the distance between the two distributions is negligible, this protocol would not qualify as Zero-Knowledge according to Definition 5. But it is very close to satisfying it. In fact we say that if the statistical distance between the two random variables is negligible, then the protocol is called statistical Zero-Knowledge. Let us define this new concept more formally.

3.2 Flavours of Zero-Knowledge

Definition 6 (Ensemble) *Let $\mathcal{I}$ be a countable set of indices. An Ensemble indexed by $\mathcal{I}$ is a sequence of random variables indexed by elements of $\mathcal{I}$.*

Namely, if $X = \{X_i\}_{i \in \mathcal{I}}$, where each X_i is a random variable, then X is an ensemble.

Definition 7 (Absolute (Perfect) Indistinguishability) *Let X and Y be two ensembles, then X and Y are perfectly indistinguishable if and only if*

$$\forall_h, \forall_n, \Pr[h(X_n) = 1] = \Pr[h(X_n) = 1], \tag{3.7}$$

where h is a predicate.

Definition 8 (Statistical Indistinguishability) *Let X and Y be two ensembles, then X and Y are statistically indistinguishable if and only if for all predicate h we have that*

$$|\Pr[h(X_n) = 1] - \Pr[h(Y_n) = 1]| \leqslant \mu(n) \tag{3.8}$$

where $\mu(n)$ is a negligible function.

Definition 9 (Computational Indistinguishability) *Let X and Y be two ensembles, then X and Y are computationally indistinguishable if and only if for all probabilistic polynomial time algorithm A we have that*

$$|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]| \leqslant \mu(n) \tag{3.9}$$

where $\mu(n)$ is a negligible function.

To each of these notions of indistinguishability corresponds a notion of Zero-knowledge: Perfect Zero-Knowledge, Statistical Zero-Knowledge and Computational Zero-Knowledge.

Going back to the Simulator for GNI, Simulator 3.8, we can conclude that Protocol 3.6 is a Statistical Zero-Knowledge protocol as the two random variables $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are identical but on points were S outputs $\perp$ which happens with negligible probability (see Problem 3.6).

If we consider n to be the number of vertices in G_0 or G_1, then if we fix k in protocol 3.6 to be in $\Theta(n)$, then the probability of failure of S is no more than $2^{-k} \leqslant 1/poly(n)$ for all $poly(n)$. And as was already argued, if S computes the isomorphism, then the protocol is a Perfect Zero-Knowledge protocol. The next Chapter will introduce Computational Zero-Knowledge protocols.

Problems

3.1 Prove formally that the simulator for Graph-Isomorphism is only expected-poly-time. Then fix the simulator so that it always runs in polynomial time. What can you conclude from your new simulator?

3.2 Prove formally that protocol 3.6 is a statistical-Zero-Knowledge protocol.

3.3
1. Prove that the proof of knowledge of Protocol 3.6, which we revisit at the end of this chapter, can be done in parallel. That is, if all pairs (G_0^i, G_1^i), all the challenges and all replies are each sent in one message so that the full proof of knowledge consists of three messages.
2. Is the Proof of knowledge still Zero-knowledge if done in parallel?
3. Prove that Protocol 3.6 is still zero-knowledge if the proof of knowledge is done in parallel.

3.4 Prove that for two ensemble X_n and Y_n that have negligible distance for all n, Theorem **??** implies that X_n and y_n are statistically indistinguishable (Definition 8).

3.5 quadratic residu.

1.

3.6 Prove that $D(VIEW[(P, V'(\aleph))(x)], S_{V'(\aleph)(x)})$ negligeable in the case were the Simulator outputs $\perp$. That is use convexity and the fact that the two random variables $VIEW[(P, V'(\aleph))(x)]$ and $S_{V'(\aleph)(x)}$ are the same with very high probability and very different with very low probability.

3.7 Let n be a product of two primes, and y be a quadratic non-residu $\bmod n$. Provide a protocol which is a Zero-Knowledge proof of Knowledge that a number x is a quadratic residu or a quadratic non-residu $\bmod n$. As your protocol is a proof of knowledge, it should not disclose whether x is or is not a residu, but only prove that the prover knows a witness to one or the other (it should also be simulatable).

Note that this is proving something *trivial* and is yet a very powerful tool. Can you find a situation where this could be used ?

3.8 We define **COCKS** as all the numbers n with two distinct prime factors p, q such that $\gcd(n, \phi(n)) = 1$, where $\phi(\cdot)$ is the euler totient function (so that n is invertible modulo $\phi(n)$ as required in Cocks' variation of the RSA cryptosystem using the public exponent $e = n$).

$$\textbf{COCKS} = \{n | \gcd(n, \phi(n)) = 1, n = pq, p \neq q, p \text{ and } q \text{ are primes}\}.$$

Provide a Zero-Knowledge proof, with its simulator, to prove that a given integer n is a COCKS number, under the assumption that the verifier already knows an arbitrary $y \in QNR_n[+1]$[1].

Hint:

Let **WRSA** (Weak-RSA) be the following language:

$$\textbf{WRSA} = \{n | n = p^\alpha q^\beta, p \neq q, p \text{ and } q \text{ are primes and } \alpha, \beta > 0\}.$$

A)** Construct a zero-knowledge proof for the language **WRSA**, under the assumption that the verifier already knows an arbitrary $y \in QNR_n[+1]$.

Sub-Hint: use Protocol 5.1. You may use the following theorem without proof:

Theorem 1 *If $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ (which is not a square) has exactly k distinct prime factors then exactly 2^{-k+1} of the elements in $\mathbb{Z}_n^*$ with Jacobi symbol $+1$ are quadratic residues $\bmod n$.*

Let **NphiN** be the following language:

$$\textbf{NphiN} = \{n | \gcd(n, \phi(n)) = 1\}.$$

B) Construct a zero-knowledge proof for the language **NphiN**.

Sub-Hint: You may use the following theorem without proof:

Theorem 2 *Let n be a composite odd number. If n belongs to* **NphiN***, then every element $x \in \mathbb{Z}_n^*$ has an n^{th} root $\bmod n$, i.e. there exists an element y such that $x \equiv y^n \bmod n$. On the contrary, if n is not in* **NphiN***, then at most half the elements in $\mathbb{Z}_n^*$ have an n^{th} root $\bmod n$.*

C) Prove that we have

$$\textbf{COCKS} = \textbf{WRSA} \cap \textbf{NphiN}.$$

D) Combine your two zero-knowledge proofs for languages **WRSA**, and **NphiN** to prove membership to **COCKS** in zero-knowledge, under the assumption that the verifier already knows an arbitrary $y \in QNR_n[+1]$.

3.9 Following on the previous question, define the following set of numbers

$$\textbf{BLUM} = \{n | n = pq, p \equiv q \equiv 3 \bmod 4, p \neq q \text{ are primes}\}.$$

Prove that the language **BLUM** has a statistical zero-knowledge proof.

Hint:

[1] This technicality is necessary because we currently do not know an efficient algorithm to find such a y from n only.