Lecture 9

Lecture date: March 7-9, 2005 *Scribe: S. Bhattacharyya, R. Deak, P. Mirzadeh*

1 Interactive Proof Systems/Protocols

1.1 Introduction

The traditional mathematical notion of a proof is a simple *passive* protocol in which a *prover* P outputs a complete proof to a *verifier* V who decides on its validity. The interaction in this traditional sense is minimal and one-way, prover $\rightarrow$ verifier. The observation has been made that allowing the verifier to interact with the prover can have advantages, for example proving the assertion faster or proving more expressive languages. This extension allows for the idea of *interactive proof systems (protocols)*.

The general framework of the interactive proof system (protocol) involves a prover P with an exponential amount of time (computationally unbounded) and a verifier V with a polynomial amount of time. Both P and V exchange multiple messages (challenges and responses), usually dependent upon outcomes of fair coin tosses which they may or may not share. It is easy to see that since V is a poly-time machine (PPT), only a polynomial number of messages may be exchanged between the two. P's objective is to convince (prove to) the verifier the truth of an assertion, e.g., claimed knowledge of a proof that $x \in L$. V either accepts or rejects the interaction with the P.

1.2 Definition of Interactive Proof Systems

An *interactive proof system* for a language L is a protocol PV for communication between a computationally unbounded (exponential time) machine P and a probabilistic poly-time (PPT) machine V such that the protocol satisfies the properties of *completeness* and *soundness.*

Definition 1 (Goldwasser, Micali, Rackoff) *An Interactive Proof (IP) for a language L is a protocol PV for a prover P and a verifier V for which the following two properties hold:*

- **Completeness:** Completeness is the property that the protocol work correctly given any prover can always make V "accept" if $x \in L$. That is, an interactive proof (protocol) is *complete* if, given an honest P and an honest V, the protocol PV succeeds with overwhelming probability (i.e., V accepts P's claim). More formally, if $x \in L$ then P has a overwhelming chance of convincing V that $x \in L$:

$$\forall c > 0 \quad \exists N \quad \text{s}.t. \quad \forall x \in L \quad \text{where} \quad |x| > N$$
$$\underset{coins\ flips\ of\ P,\ V}{Prob}[P \longleftrightarrow V[x] \ \textit{makes V[x] accept}] > 1 - \frac{1}{|x|^c}$$

- **Soundness:** Soundness is the property that a dishonest prover (claiming $x \in L$ when $x \notin L$) cannot convince an honest verifier of the false assertion with more than a negligible probability. If $x \notin L$ then every dishonest prover P' has negligible chance of convincing V:

$$\forall P' \quad \forall c > 0 \quad \exists N \quad \text{s}.t. \quad \forall x \notin L \quad \text{where} \quad |\text{x}| > N$$
$$\underset{coins\ flips\ of\ P',\ V}{Prob}[P' \longleftrightarrow V[x] \ \textit{makes V[x] accept}] < \frac{1}{|x|^c}$$

Definition 2 *Class IP.*

With the definition of an interactive-proof system (protocol) PV, we can define the class of languages that have interactive proofs as IP.

$$IP = \{L \mid L \ \textit{has interactive proof.}\}$$

We note that IP is equal to another complexity class *PSPACE*, which includes for example *co-NP*. This is however outside the scope of this course.

1.3 Mathematical Background: Graph Isomorphism

To illustrate examples for interactive-proof systems we first introduce our two case example languages of *Graph Isomorphism (GI)* and *Graph Non-Isomorphism (GNI)*. A graph G_0 is *isomorphic* to another graph G_1 is the two graphs have the same form, that is if there exists a 1-1 edge-invariant mapping π of the vertices of the first graph to the vertices of the second graph, and similarly *non-isomorphic* when no such mapping exists. These languages are our case examples because they are easy to describe and their protocols can be abstracted to certain hard number theory problems, such as the hardness of *quadratic residue* and *quadratic non-residue* assumptions. Many of the interactive proof protocols we will discuss will depend upon the creation of random permutations of graphs, created simply by creating a random permutation of vertex labels and applying it to the graph. Formally we define the GI and GNI languages as:

Definition 3 *Graph Isomorphism (GI)*

$x \in GI$ *iff* x is a pair of graphs (G_0, G_1) and $G_0 \sim G_1$. We will abbreviate this as $x = \{G_0 \sim G_1\}$

Definition 4 *Graph Non-Isomorphism (GNI)*

$x \in GNI$ *iff* x is a pair of graphs (G_0, G_1) and $G_0 \not\sim G_1$. We will abbreviate this as $x = \{G_0 \not\sim G_1\}$

1.4 Examples of Interactive Proofs

A Not-So-Interactive Proof

Here is an example of a proof where the P proves x ∈ L.

P		V
If $G_0 \sim G_1$	$\xrightarrow{\pi:\, G_0 \to G_1}$	accept if $\pi(G_0) = G_1$

Figure 1: Non-Interactive Proof Diagram for Graph Isomorphism (GI)

In general, it is not known how to find a graph isomorphism between two arbitrary graphs. However for many classes of graphs with additional properties, it is known how to find a graph isomorphism between two graphs. Here note that the prover has infinite resources and V is only a poly-time machine. Since P provides V with π, the permutation that shows the isomorphism between G_0 and G_1, V can apply the permutation to G_0 and check to see if it indeed results in G_1 within poly-time.

Interactive Proof for Graph Isomorphism [Protocol P_GI] - $(G_0 \sim G_1)$

The following steps describe the interactive proof protocol for graph isomorphism GI. P generates a random bit b and a random permutation π of G_b (if $b = 0$ then $G_b = G_0$ else if $b = 1$ then $G_b = G_1$). P sends this new graph H to V who responds with a random bit b'. P then responds to V's request by sending the permutation which maps the new graph H to $G_{b'}$. V checks that this permutation is actually an isomorphism between the graphs. Repeat this k times one after the other (i.e. sequentially). V accepts if P was able to send a correct permutation every time.

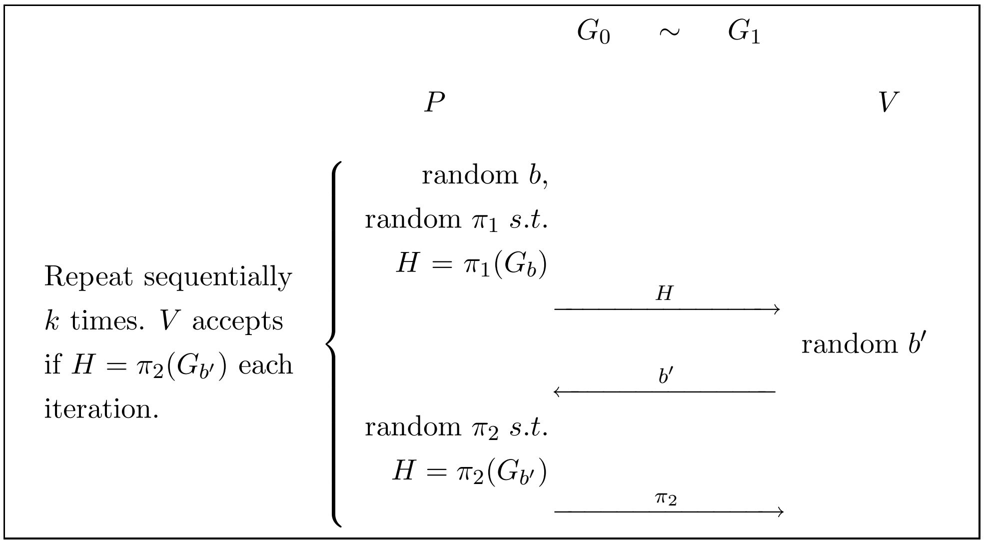

Figure 2: Interactive Proof Diagram for Graph Isomorphism (GI)

- **Completeness:** If $G_0 \sim G_1$, then it does not matter what b' is, since P shows an isomorphism to H.
- **Soundness:** If P is lying, and $G_0 \nsim G_1$, then H could be isomorphic to G_0 or G_1 but not both. Then at each iteration, V will catch P cheating with probability $\frac{1}{2}$. So with k iterations, the probability of successfully cheating becomes at most $\frac{1}{2^k}$.

Interactive Proof for Graph Non-Isomorphism [Protocol P_GNI] - ($G_0 \nsim G_1$)

Here again P has infinite resources and the V is a poly-time machine. V picks randomly which graph P must show an isomorphism to and permutes this graph with π. The resulting graph is H. Since P is all-powerful, it can determine which graph H is isomorphic to in order to recover b. If G_0 and G_1 are isomorphic, contrary to P's claim, then P, no matter how powerful, will not be able to differentiate whether H was supposed to be isomorphic to G_0 or G_1, and will be forced to guess with a success probability of $\frac{1}{2}$. V can require k rounds of such verifications to approve the P's claims, resulting in a success probability of $\frac{1}{2^k}$.

$$G_0 \not\sim G_1$$

	P		V
Repeat k times.			random b random π *s.t.* $H = \pi(G_b)$
		$\xleftarrow{\quad H \quad}$	
	determine b' *s.t.* $H \sim G_{b'}$		
		$\xrightarrow{\quad b' \quad}$	
			accept this iteration if $b = b'$, otherwise reject entire conversation.

Figure 3: Interactive Proof Diagram for Graph Non-Isomorphism (GNI)

- **Completeness:** As long as $G_0 \not\sim G_1$, then either G_0 or G_1 will be isomorphic to H but not both. The infinitely powerful P will always be able to find H.
- **Soundness:** If $G_0 \sim G_1$, then P will not be able to decipher whether $H \sim G_0$ or $H \sim G_1$. The P will be forced to guess b with success probability $\frac{1}{2}$.

2 Zero-Knowledge Interactive Proofs

2.1 Introduction

In the previous discussion we mentioned the advantages of interactive-proof systems over the traditional passive verifier proof, that is, the interaction may allow P to convince V of more complicated statements such as those in *co-NP*. Here we also want P not to reveal any information about why the theorem is true. For example, the simplest proof protocol for graph isomorphism (GI) is for P to simply send the isomorphic mapping π between G_0 and G_1 to V. Corresponding to the traditional proof method, this protocol reveals the secret knowledge the provider claimed to V, namely the isomorphism mapping between the two input graphs. In most cases, we do not want the P to reveal *any* knowledge to V outside the truthfulness of the assertion itself, that the graphs are isomorphic.

In this section we introduce *Zero-Knowledge (ZK) Proofs* or protocols, instances of interactive-proof systems, which are designed to allow P to demonstrate knowledge of a secret while revealing no information whatsoever (beyond what V was able to deduce by himself in poly-time) of use to V in conveying this demonstration of knowledge to others. It should be noted that the interactive proof for GI (P_GI) in the previous section is actually a zero-knowledge interactive proof.

Motivating Story

This is a story due to Goldwasser, Micali and Rackoff. One night in a small town there is a murder. Verry Fire, the local reporter, hears about the murder and wants to write a story for her newspaper. She goes to a pay phone and calls the detective to get the facts of the murder case, but the detective simply tells her, "there was a murder" and hangs up. She calls back several times, but every time she just hears the same phrase, "there was a murder." Verry already knew there was a murder, so she certainly didn't need to call the detective to obtain this information. She could have just saved her money and generated this phrase herself. Feeling a little frustrated, she decides to call the police chief. The chief, who enjoys playing games with reporters, flips a coin. If the coin lands heads, the chief says, "there was a murder." If the coin lands tails, the chief says, "no comment." Verry calls back several times, only to hear one of the two phrases each time. Like the conversation with the detective, her conversation with the chief provides no new information with which to write her column. She could have just flipped a coin herself and generated the chief's phrases with the same probability distribution. Verry proceeds to write her column. Only now does she realize that she could have written her column without having talked to either the detective or the chief, as the conversations with them were *zero-knowledge*.

2.2 Definition of Zero-Knowledge Proofs

We can informally define *zero-knowledge proofs*, or interactive proof systems with the *zero-knowledge property*, as a protocol in which the message conversation V has with P, the same distribution of messages between the two parties could have been generated by V himself. Before we can formally define a zero-knowledge proof, we must give the definition of the notion of a verifier's *view*. Simply put, the *view* is the sequence of messages (challenges and responses) exchanged between the verifier V and prover P as well as the string of random bits (coin tosses) that V used. Formally:

Definition 5 *View.*

For an interactive protocol PV, let $PV(x)$ denote the verifiers *view* during the protocol on input x. Specifically, $PV(x)$ is composed of:

- Sequence of messages between P and V

- Random bit string (coin tosses) of V

Let $[PV(x)]$ represent the random variable which represents the distribution of *view* points $PV(x)$ taken over the coin tosses of P and V.

With the definition of a view, we can describe a *zero-knowledge proof* or say an interactive proof system for language L is *zero-knowledge* if $\forall x \in L$ and $\forall V \in PPT$ there exists a probabilistic poly-time machine $S_v(x)$ whose output distribution *$[S_v(x)]$* taken over input x and coins of S is indistinguishable from the distribution of the *view $[PV(x)]$*. We call this PPT machine $S_v(x)$ a *simulator.* We note that the definition of zero-knowledge should hold only for honest provers.

Definition 6 *Zero-Knowledge Interactive Proof*

An Interactive Proof PV* for a language L is said to be *Zero-Knowledge* if

$$\forall V^* \in PPT \quad \exists S_{V^*} \quad \in PPT \quad \forall x \quad \in L \quad s.t.$$
$$[P \leftrightarrow V^*[x]] \cong [S_{V^*}(x)]$$

That is, for every verifier V^* there exists an expected poly-time algorithm S_{V^*} *(simulator)* which can produce, upon input of the assertion x to be proven, but without interacting with the real prover, transcripts indistinguishable from those resulting from interaction with the real prover or more specifically points from [PV^*(x)]. Intuitively, the existence of S_{V^*} means the verifier V^* does not need to interact with P, in relation to our motivating story Verry Fire does not need to talk to the detective or the chief, since it can generate or simulate the *view* itself.

The zero-knowledge property implies that a prover executing the protocol, against all verifiers, even dishonest verifiers, does not release any information (other than the truth of the assertion). The existence of a PPT simulator S_{V^*} for a verifier V^* means *interaction* between P and V^* is zero-knowledge. If all interactions with P for every V^* are zero-knowledge then the protocol is called zero-knowledge.

Definition 7 *Three Variants of Zero-Knowledge:*

- **Perfect ZK:** *"="*

  The distributions of the Simulator and Protocol are exactly equal. This is the strictest definition of zero-knowledge.

- ***Statistical ZK:*** "$\stackrel{s}{=}$"

  The distributions of the Simulator and Protocol are statistically close.
  From the definition, two distributions $\{X_n\}$ and $\{Y_n\}$ are statistically close iff

$$\forall c \; \exists N \; s.t. \; \forall n > N$$

$$\sum_{\alpha \in \{0,1\}^n} \left| Pr_{\{X_n\}} \left[X_n = \alpha \right] - Pr_{\{Y_n\}} \left[Y_n = \alpha \right] \right| < \frac{1}{n^c}$$

  Which is to say at least n^c samples are required to differentiate $\{X_n\}$ and $\{Y_n\}$, or in other words more than a polynomial number of samples are required to distinguish the distributions.

- ***Computational ZK:*** "$\stackrel{c}{=}$"

  The distributions of the Simulator and Protocol are computationally indistinguishable to a poly-time machine.

  Recall the definition, $\{X_n\}$ and $\{Y_n\}$ are poly-time indistinguishable iff:

$$\forall c \;\; \forall A \in PPT \;\; \exists N \;\; s.t. \;\; \forall n > N$$

$$\left| Pr_{\{\{X_n\},\, A's\, coins\}} \left[A \left(X_n \right) = 1 \right] - Pr_{\{\{Y_n\},\, A's\, coins\}} \left[A \left(Y_n \right) = 1 \right] \right| < \frac{1}{n^c}$$

Note that distributions are equal $\Rightarrow$ statistically close $\Rightarrow$ computationally indistinguishable.

The converses may not be true however. If 1-way functions exist, then
computational indistinguishability $\not\Rightarrow$ *statistical closeness.*

Let $\{X_n\}$ be the output distribution of pseudo-random generator G: $\{0,1\}^n \rightarrow \{0,1\}^{2n}$.
Let $\{Y_n\}$ be the output distribution of a truly random generator for 2n-bit strings.

$\{X_{2n}\}$ contains at most 2^n different strings.
$\{Y_{2n}\}$ contains 2^{2n} different strings.

So by definition these distributions are statistically distinguishable, however since G is a pseudo-random generator, the distributions are computationally indistinguishable.

2.3 Constructing a Simulator: Two Analogies

In order to construct a simulator, S for V that simulates the conversation with P, we need to ensure that the probability distribution induced by S on $x \in GI$ is the same as the probability distribution induced by the view $PV(x)$. In order to do this, we will depend on the fact that S is a Turing Machine whose state can be saved and restarted.

The verifier we are interested in is a multi-tape Turing Machine with the following structure. It is a finite state machine with tapes for input and output, a work tape, a random tape, and tapes for communication input and output. Since the the state of these tapes completely defines the state of the verifier, we can save the state of the verifier at any time. We can place an input on the communication input tape and let the tape continue running. At any time, we can restore the verifier to a previous state.

This abstraction clearly applies to other computational devices, i.e. we can restart any computer to a previous state provided that we take a "snapshot" of its entire configuration, including, memory, CPU, registers, etc...

Film analogy

Imagine the shooting of a movie. Although movies typically appear as a continuous stream of scenes which tend to play for a total of one and a half to two and a half hours, movies generally take much longer to produce. Certain scenes have to be shot multiple times before the scene is acceptable for the screen. Although a director may shoot a scene a number of times, in the final production the bad scenes are typically edited out, leaving only the good scenes. This analogy shows us what resetting the verifier allows us to do; it allows us to "reshoot" the scene.

Coin-flipping extravaganza analogy

Jay Lentil, host of a popular late-night television show, convinces the great, world-famous magician Flippo to be on his show. To the amazement of all viewers, Flippo proceeds to flip a normal coin that lands heads 100 times in a row. Not to be outdone, Dave Numberman, host of a competing show, tries to find a magician to match this incredible feat. Dave however does not find a suitable magician, so he disguises his assistant Paul in a magician's costume. Dave's show is pre-taped, so in the studio, he has Paul flip a coin again and again until 100 heads have landed. Then the tape is edited to remove all the coin flips that landed tails. When the tape is shown that night, the viewers are amazed to see a man flipping a coin and having it land heads 100 times in a row!

2.4 Examples of Zero-Knowledge Proofs

In the following section we discuss examples of zero-knowledge proofs for our case examples of graph isomorphism and non-isomorphism as well as a zero-knowledge interactive proof for graph 3-coloring which can be abstracted for all NP problems (since graph 3-coloring G3C is NP-Complete).

Zero-Knowledge for Graph Isomorphism (GI)

In the previous section of Interactive Proofs, we introduced a protocol for graph isomorphism (GI) called protocol P_GI (see Figure 2) which was shown to have the properties of completeness and soundness, thus an interactive proof. We also stated that it was also zero-knowledge. In this section we show that the protocol is zero-knowledge by constructing the simulator S_V for any verifier V, whose output can be shown to be indistinguishable with the distribution of the *view* in protocol P_GI.

Statistical Simulator for Graph Isomorphism In review, to prove that protocol P_GI is zero-knowledge we must create a simulator S_V whose output distribution is statistically close with the distribution of the P_GI protocol messages exchanged and random bit string of V. The simulator story demonstrates that it is possible to run an experiment many times and pick only successful outcomes, the *rewinding* trick the simulator will employ. Since V is simply a turing machine, we can record the state of the verifier and rewind to a previous state (rewind) if necessary. The following is the simulator pseudo-code:

1. Choose a random bit b.
2. Choose a random isomorphic permutation of input graph G_b; $H \leftarrow \pi(G_b)$.
3. Record the state information of V.
4. Give H to verifier V.
5. Get random bit b' from V.
6. (a) IF ($b = b$') we can simulate so output the prover's output.
   (b) ELSE reset V to previous state using step 3 information.
7. GOTO Step 1 (REPEAT k times sequentially).
8. Then output the resulting conversation and V's random string.

Each iteration of the simulator code is successful in matching V's random bit b' with probability $\frac{1}{2}$, therefore S_V is expected to run $2k$ iterations to output the k *view* points identical to *[PV(x)]* (ignoring the issue of "aborts", which may make the proof more delicate). The distributions are statistically close because S_V chooses H exactly the same way as P, and V interacts with the simulator with a random bit b' as it would with the real P since it cannot distinguish who it is talking to.

A Parallel Version of a Zero Knowledge Proof for Graph Isomorphism? Since the three round protocol of the zero knowledge proof for graph isomorphism runs k times, $3k$ communications are sent between the prover and the verifier over the course of the proof. In an attempt to minimize the number of communications (and the associated overhead) during the proof, we now suggest an alternative zero knowledge proof in which the communications occur in parallel. Such a proof could be constructed as follows.

P	$G_0 \sim G_1$	V
for $i \leftarrow 1$ **to** k random π_i $s.t.$ $H_i = \pi_i(G_0)$		
	$\xrightarrow{H_1,\ldots,H_k}$	
		Generate random $\$b_1, \ldots, \$b_k$
	$\xleftarrow{b_1,\ldots,b_k}$	
for $i \leftarrow 1$ **to** k determine ϕ_i $s.t.$ $H_i = \phi_i(G_{b_i})$		
	$\xrightarrow{\phi_i,\ldots,\phi_k}$	
		Accept if $\forall i$ $H_i = \phi_i(G_{b_i})$

Figure 4: Parallel Interactive Proof for GI (not zero-knowledge)

To show this is an interactive proof we need to show:

- **Completeness:** If $G_0 \sim G_1$, then $\forall i$, P will be able to show an isomorphism between H_i and G_{b_i}.
- **Soundness:** If $G_0 \not\sim G_1$, then V only accepts when P can guess b_i k times in a row. The probability that this happens is $\frac{1}{2^k}$.

It is not actually known whether this protocol is zero knowledge. The simulator that we constructed before does not work in this case. This is because for a parallel simulator to produce a successful experiment, it must simultaneously guess correctly each of the k b_i's. The probability of this occurring is $\frac{1}{2^k}$. To illustrate, consider the case where we have a dishonest verifier which acts deterministically instead of probabilistically in choosing $b_1, \ldots, b_k$, for example some V^* using a collision-resistant hash function h to output

$h(b_1, ..., b_k)$ instead of $b_1, ..., b_k$, clearly rewinding will not help. In fact, it was shown by Goldreich and Krawzyk that if this protocol is zero knowledge, then $GI \in BPP$.

Zero-Knowledge for Graph Non-Isomorphism (GNI)

It was shown previously (in § 1.4) that graph non-isomorphism has an interactive proof. We will show here that the previous interactive proof is not a zero-knowledge proof for graph non-isomorphism. So why is P_GNI in § 1.4 not zero-knowledge?

$$G_0 \not\sim G_1$$

	P		V
Repeat sequentially k times. V accepts if $b = b'$ each iteration.			$\$b$ random π $s.t.$ $H = \pi(G_b)$
		$\xleftarrow{\quad H \quad}$	
	determine b' $s.t.$ $H \sim G_{b'}$		
		$\xrightarrow{\quad b' \quad}$	

Figure 5: Interactive Proof Diagram for GNI (not zero-knowledge)

We saw before that this protocol is an interactive proof. It is not however a zero knowledge proof. Suppose that V has a graph H that he knows is isomorphic to exactly one of G_0 or G_1 but he does not know to which one it is isomorphic. If V sends H to P and P answers honestly, then V has gained knowledge about which graph is isomorphic to H.

There is indeed a zero knowledge proof for graph non-isomorphism, but before it is presented, we introduce a new protocol and show where it fails as a zero knowledge proof. Consider the following protocol:

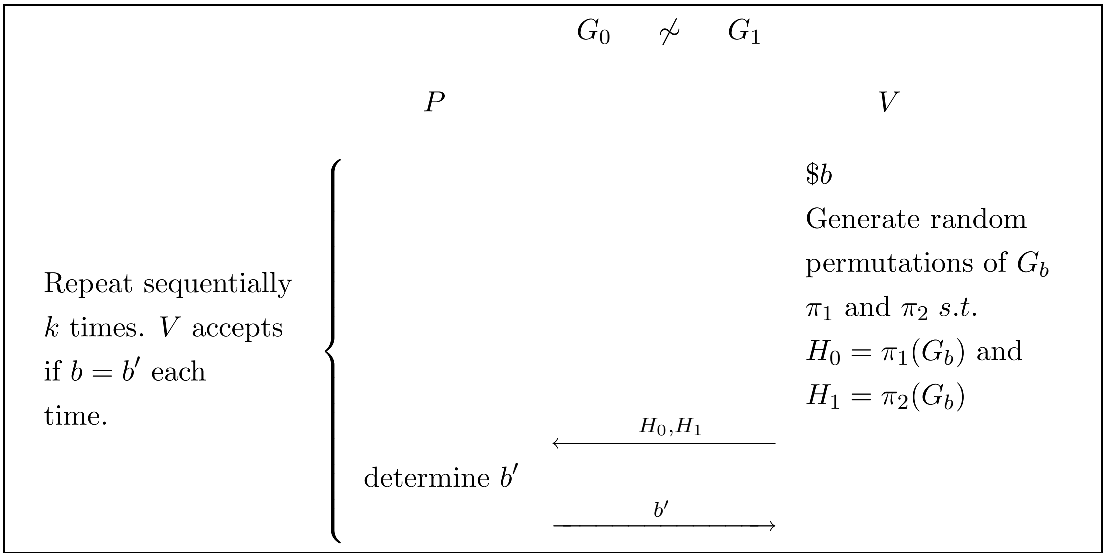

Figure 6: Alternative Interactive Proof Diagram for GNI (not zero-knowledge)

Notice that this is an interactive proof because it satisfies:

- **Completeness:** If $G_0 \not\sim G_1$ then $H_0 \not\sim H_1$, so P can distinguish between them and ensure that $b' = b$.
- **Soundness:** If $G_0 \sim G_1$ then $H_0 \sim H_1$, so no prover can distinguish between them. Thus P must guess between the pairs with probability $\frac{1}{2}$. P has only a $\frac{1}{2^k}$ probability of fooling V k times.

Notice that this is not a zero knowledge proof however because V may not know that value of the coin flip b before P reveals it to him. Thus V may gain some knowledge through communication with P. To, counteract this, we need a protocol in which V can show P that he knows the value of the random b. The revised protocol utilizes the following notation and follows as seen in Figure 7:

Notation:

$\{A, B\} \sim \{C, D\}$: $(\{A \sim C \textit{ and } B \sim D)$ or $(A \sim D \textit{ and } B \sim C)$
$\pi : \{A, B\} \rightarrow \{C, D\}$: $\{A, B\} \sim \{C, D\}$ and π is the pair of mappings.

$G_0 \not\sim G_1$

P		V
		$\$b$ Generate random π_1 and π_2 *s.t.* $H_0 = \pi_1(G_b), H_1 = \pi_2(G_b)$
	$\xleftarrow{H_0, H_1}$	
		V proves he knows b
		Generate $D_1, ..., D_k$ *s.t.* $\{D_{2i-1}, D_{2k}\} \sim \{H_0, H_1\}$
	$\xleftarrow{D_1, ..., D_k}$	
coin flips $b'_1, ..., b'_k$		
	$\xrightarrow{b'_1, ..., b'_k}$	
		Find k pairs of isomorphisms $\phi_1, ..., \phi_k$ *s.t.* **if** $b'_i = 0$ $\phi_i : \{D_{2i-1}, D_{2i}\} \to \{G_0, G_1\}$ **else** $\phi_i : \{D_{2i-1}, D_{2i}\} \to \{H_0, H_1\}$
	$\xleftarrow{\phi_1, ..., \phi_k}$	
$\forall i$ check ϕ_i		
Determine bit b''		
	$\xrightarrow{b''}$	

Figure 7: Final Protocol for Zero Knowledge Proof of Graph Non-Isomorphism (GNI)

Here we will explain the zero-knowledge protocol for GNI given in Figure 7. To make sense of the middle part of the protocol dealing with V's proof of knowing b, consider a pair of graphs (D_{2i-1}, D_{2i}). P will ask V to show either the mapping $\{D_{2i-1}, D_{2i}\} \rightarrow \{G_0, G_1\}$ or the mapping $\{D_{2i-1}, D_{2i}\} \rightarrow \{C_0, C_1\}$. Since V does not know which mapping P will ask for, V must know both. But notice that if V knows both mappings, then V knows a mapping $\{G_0, G_1\} \rightarrow \{C_0, C_1\}$. Thus, V knows b. If P does not know b but guesses correctly on all of the b'_i, then he will fool V with probability $\frac{1}{2^k}$.

Now let us create a simulator $S_{V'}$ that is statistical ZK.

Simulator:

1. Get (H_0, H_1) and (D_1, D_2), ..., (D_{2k-1}, D_{2k}) from V'.
2. Record V''s state utilizing the Turing machine notion developed previously.
3. Send random bits b'_1, ..., b'_k.
4. Get ϕ_1, ..., ϕ_k from V'.
5. Reset V''s state to the one recorded in step 2.
6. Send another set of random bits, b''_1, ..., b''_k. Get the corresponding ϕ'_1, ..., ϕ'_k from V'. Then with probability $(1 - \frac{1}{2^k})$ $\exists i$ for which $b'_i \neq b''_k$. For this i, $\phi_i : \{D_{2k-1}, D_{2k}\} \rightarrow \{G_0, G_1\}$ and $\phi'_i : \{D_{2k-1}, D_{2k}\} \rightarrow \{H_0, H_1\}$. Thus, we can determine the isomorphism between $\{H_0, H_1\}$ and $\{G_0, G_1\}$, and hence b.

Since the distribution created by the output of the success runs of this simulator are statistically identical to distribution induced by the actual conversation between the prover and the verifier, this simulator is statistically zero knowledge.

2.5 Computational Zero-Knowledge for all of *NP*

Theorem $[GMW]$ Assuming computational commitment protocols exist implies all of NP is in computational zero-knowledge.

The first step to proving this amazing result is to reduce the zero-knowledge protocol to graph 3-colorability. Since graph 3-colorability is *NP-complete*, using *NP* reductions we can prove any other *NP* statement.

So how does one show that a graph is 3-colorable?

First permute the colors. Note the colors here have meaning that will be important when the reduction takes place. Commit the colors using one's favorite bit-commitment protocol. V picks an edge and P must then show the two colors of vertices adjacent to that edge. P shows V the colors of two nodes and V checks that the colors are different.

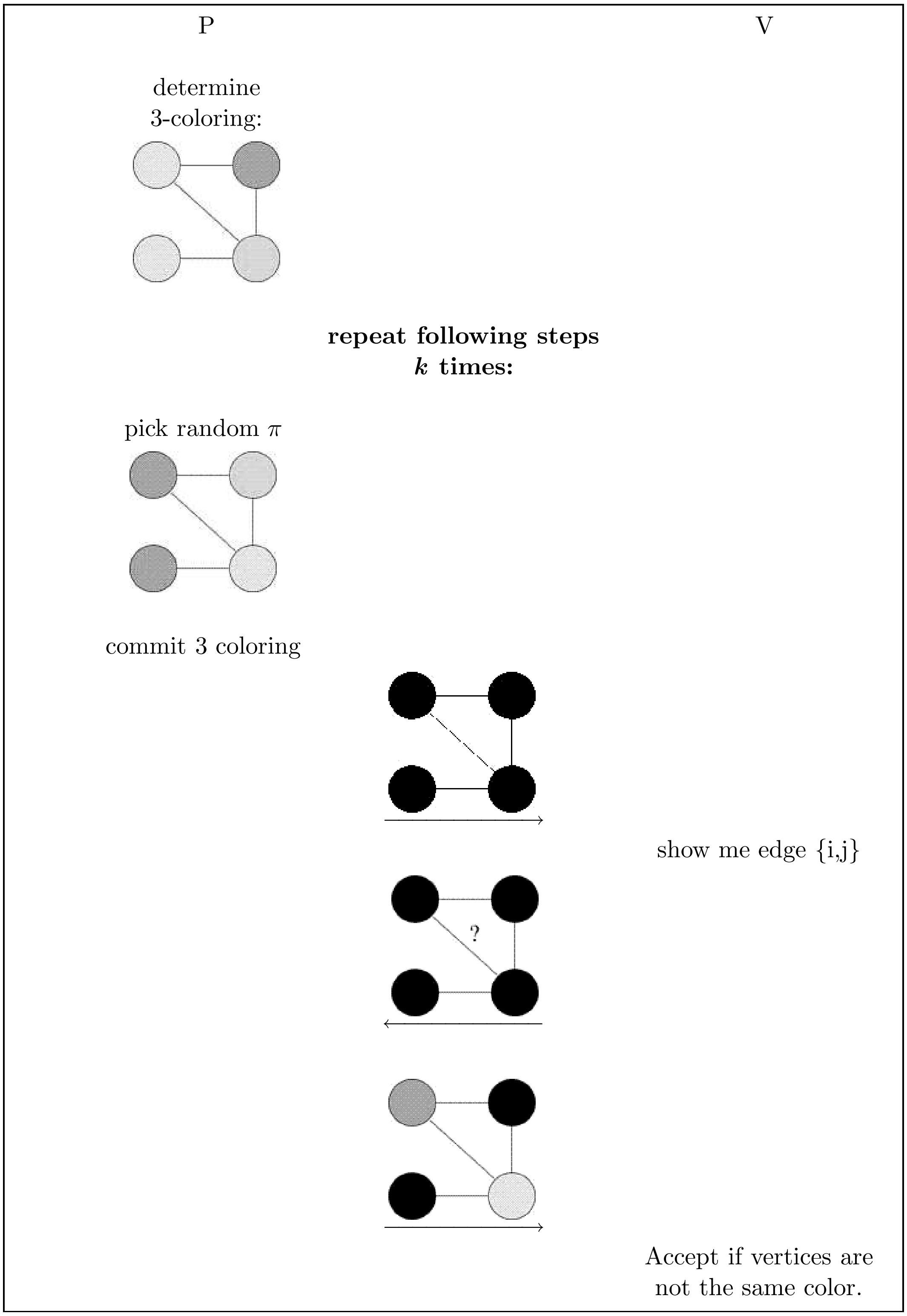

Figure 8: Example of Zero Knowledge Proof for Graph 3-Coloring (G3C)

This is an interactive proof since:

- **Completeness**: If this graph is 3-colorable and P does not lie, no two vertices that share an edge will have the same color.
- **Soundness**: If P is cheating then there is a $\frac{1}{e}$ chance of being caught with every iteration, where e is the number of edges. If this is repeated e^3 times then the error probability becomes *negligible*.

Why is this zero-knowledge?

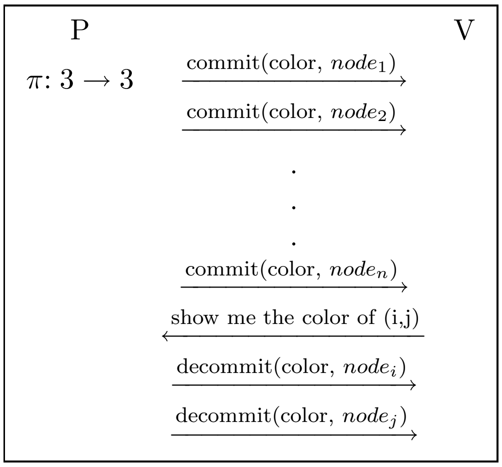

Figure 9: Zero-Knowledge Protocol for Graph 3-Coloring (G3C)

The simulator for this conversation would then be the following:

Simulator:

1. Record verifier V's state.
2. Guess edge (i,j).
3. Commit vertices i and j to two different colors (out of the three possible colors).
4. Commit remaining vertices to any color, say all red.
5. If V asks for the edge (i,j), show the commited vertices.
6. If V asks for another, rewind, and GOTO step 2.

This is computational zero-knowledge since no poly-time V can differentiate commited vertices from all red unopened vertices.

Since any NP problem can be reduced to graph 3-colorability in poly-time, you can reduce the NP problem in question to graph 3-colorability, commit the random permutation and convince V of the graph's coloring with zero-knowledge without revealing P's secret.