6.876/18.426: Advanced Cryptography 2/10/2003

Lecture 2: The Notion of Zero-Knowledgeness

Scribed by: Loizos Michael

1 Recap

Recall from the last lecture the language $ISO = \{(G, H) | G \cong H\}$, which contains all pairs of graphs which are isomorphic to each other. Also recall the Interactive Proof System for ISO defined as follows. On input $(G_0, G_1) \in ISO$:

1. The prover P randomly generates a graph C that is isomorphic to both G_0 and G_1 and sends C to the verifier V. This can be clearly done by choosing a random permutation π and setting $C = \pi(G_1)$.

2. The verifier V randomly chooses a bit b and sends b to the prover P.

3. The prover P computes the permutation σ for which $\sigma(G_b) = C$ and sends σ to the verifier V. This permutation always exists and is computable by the prover P. In fact, $\sigma = \pi$ if $b = 1$ and $\sigma = \pi \circ \phi$ if $b = 0$, where ϕ is the permutation for which $\phi(G_0) = G_1$. Recall from last lecture that permutation ϕ is computable by the computationally unbounded prover P.

4. The verifier V checks whether $\sigma(G_b) = C$ and accepts if and only if this is the case.

Notice the asymmetry in the definition of the protocol. The behavior of the two interacting parties is well defined in the case where $(G_0, G_1) \in ISO$, but nothing is mentioned about the case where $(G_0, G_1) \notin ISO$. This should be anticipated, since we are defining a protocol for proving that two graphs are isomorphic to each other; so it does not make sense to speak of the case where the two graphs are not isomorphic. However, we can imagine that in the latter case, an honest prover P would aboard the protocol, possibly sending a message of the type "These two graphs are not isomorphic to each other". This leads to the following question. Since the protocol is not defined when $(G_0, G_1) \notin ISO$, what is the behavior of a malicious (or dishonest) prover in that case? Well, in this case we can say that the malicious prover P' will actually try to deceive the verifier V into believing that the two graphs are isomorphic to each other. As we have seen in the last lecture, the verifier V is "protected" against such malicious provers by the soundness of an Interactive Proof System. Let us recall the two constituents parts of an IPS (for a more detailed discussion refer to the previous lecture).

Definition: A pair of interacting Turing Machines (P, V) is an *Interactive Proof System* for a language L, if the following properties are satisfied:

1. Completeness: $\forall x \in L, Pr[(\mathsf{P}, \mathsf{V})[x] = Yes] = 1$.

2. Soundness: $\forall x \notin L, \forall \mathsf{P}', Pr[(\mathsf{P}', \mathsf{V})[x] = Yes] \leq 1/2$.

2 Preliminaries

We have seen last time that the presented ISO protocol is an IPS. In fact, we finished by saying that this protocol has an additional property: "it leaks no information". Intuitively, this is what we will call the *zero-knowledgeness* property of an IPS. An IPS (P, V) for a language L has the *zero-knowledgeness* property, if the only knowledge that a verifier gets out of its interaction with a prover that tries to prove that some x is in L is that the statement is indeed true and nothing more. We defer a more formal definition of this property, until some background notions are first defined.

2.1 Asymmetry of the Zero-Knowledgeness Property

Notice again the asymmetry in the definition of the ZK property. Nothing is mentioned about the conveyed knowledge in the case where a string x is not in L. In a similar way as earlier, we can claim that we do not care about that case, since the protocols we examine are concerned about proving that a string belongs in a language and not that a string is outside a language. If this is not convincing, then we simply have to accept this treatment of "half of the world" as part of the definition of the ZK property. After all, this bias towards the positive instances is not foreign to Computer Science. The well-known complexity class NP is biased towards the languages whose elements have an easily verifiable witness, i.e. if an element is in the language, then we have an easy way to verify that. The definition says nothing about the case where an element is not in the language. In fact, it is believed that for some languages in NP (the NP-complete languages), there does not exist an easy (efficient) way to verify that an element not in the language is indeed not in the language. This follows from the belief that the complexity classes NP (all the languages for which the elements in a language can be easily verified to belong in the language) and $coNP$ (all the languages for which the elements not in a language can be easily verified not to belong in the language) do not coincide.[1]

As a last comment to the asymmetry in the definition of the ZK property, observe later in the lecture where a formal definition is given, that we are only concerned with cases where a (possibly dishonest) verifier interacts with an honest prover. According to our comments from the previous section, if an element is not in the language then the honest prover can simply aboard the protocol. This gives, trivially, no information to the verifier, other than the fact that the element does not belong in the language. This argument should be sufficient to convince that even if we insist in considering the case where an element does not belong in the language, the definition of the ZK property can be trivially extended to encompass that. After all, what we want is a proof that reveals nothing except that the theorem is true.

2.2 Knowledge Act

We now proceed to define the necessary background that will enable us to give a formal definition for the ZK property. There are some simple rules that we will follow when we

[1]The intersection $NP \cap coNP$ of the two complexity classes is not considered, since we do not know of any complete problem for this class, i.e. we cannot fully characterize the "strength"/"power" of this class. In other words, it may be that $P \neq NP$ but $P = NP \cap coNP$.

wish to prove that an IPS satisfies the ZK property. These rules will be collectively called the "Knowledge Act" and as the name colorfully suggests, they will "regulate" what we consider knowledge[2] to be.

Definition: We will refer to the following rules as the *Knowledge Act*:

1. Randomness comes for free.

2. Polynomial time computation comes for free.

What the Knowledge Act implies is that by learning the result of a random process or the results of a polynomial time computation we gain no knowledge. Why? Well, consider the following two scenarios:

> You walk down the street and some person approaches you saying that he will sell you a 100-bit random string for \$1000. Do you accept the offer?
>
> You walk down the street and some person approaches you saying that he will sell you the product of two prime numbers of your choice for \$1000. Do you accept the offer?

We would definitely reject the first offer on the basis that we can generate our own 100-bit random string for **free** by simply flipping a coin! Similarly, we would also reject the second offer on the basis that the proposed product can be easily computed for **free**; everyone can efficiently multiply two prime numbers.

What do all these mean? That we do not need to "buy" something that can be computed efficiently, since we can simply compute it ourselves. Recall that efficiently means in probabilistic polynomial time. And that is exactly what the Knowledge Act states: (a) probabilistic (randomness) and (b) polynomial time come for free. In the same way that we would not "buy" something that is PPT computable, by the Knowledge Act anything that is PPT computable does not convey any knowledge.

So far we have defined what does not convey knowledge. To make things totally clear let us consider yet another scenario:

> You walk down the street and some person approaches you saying that he will sell you the output of an exponential time computation for \$1000. Assume for example that the person is selling an isomorphism between two (isomorphic) graphs. Do you accept the offer?

Now we should think carefully before rejecting this offer. In fact we want to consider this offer (although we might want to bargain for the price!). An exponential time computation is hard to perform and we could not simply carry it out by ourselves.[3]

[2]Note that information and knowledge are two completely distinct concepts. Indeed, from Information Theory we know that a random bit string contains much information. On the other hand, by the Knowledge Act a random bit string does not convey any knowledge. So we should be careful when using the words "information" and "knowledge", as they are not synonyms. Nevertheless, since this course is concerned with knowledge and not information, we might some times use the word "information" when actually referring to knowledge.

[3]Here we assume that we are acting as a PPT algorithm, since ultimately the goal is to have the verifier of an IPS play the role of the potential buyer.

What this example illustrates is that hard computations (i.e., not PPT computable) do not come for free. In the same way that something that is not PPT computable is not free, by the Knowledge Act anything that is not PPT computable conveys knowledge.

3 Proving Zero-Knowledgeness

Having provided the required background and intuition, we proceed to show how one might go about proving that an IPS satisfies the ZK property. *We will not present the final result from the beginning. Rather we will iteratively improve our proposed scheme and in parallel refine and formalize the definition of the ZK property.*

3.1 Knowledge in an Interactive Proof System

As we have discussed, ultimately we wish to prove that an IPS does not convey any knowledge, other than the fact that an element belongs in a language. This raises the following question. What else could possibly be conveyed by a proof other than the fact that the proven result is actually a valid one? This question can be answered without even considering Interactive Proof Systems. Indeed, consider a static proof (i.e., a traditional proof) of a statement. What knowledge do you get by reading the proof? Well, for one you learn that the statement proven is actually true. But you get more than that. You also learn a way to prove that the statement is correct. After reading the proof you can go and convince another person that the statement is correct. This ability or knowledge was acquired by reading the proof.

In a more abstract way, we can view this additional knowledge being conveyed in an interactive proof. Consider for instance the ISO IPS we have seen earlier. The outcome $(\mathsf{P}, \mathsf{V})[x]$ of the interaction of the prover P and the verifier V on an element $x \in L$ is: (a) a proof that the element indeed belongs in L and (b) a set of messages (collectively called a *conversation*) that were exchanged between the two interacting parties. We will denote this set of messages with $VIEW_{\mathsf{P},\mathsf{V}}[x]$.[4] Since we are concerned with what the verifier learns from such an interaction, we will restrict $VIEW_{\mathsf{P},\mathsf{V}}[x]$ to the messages send by the prover P, since the verifier's messages are already known to the verifier!

The proper way to view a conversation produced by the interaction of two parties on some common input, is as if the two parties choose a string from a set of strings over some distribution, through their interaction. Thus, a conversation is a string being chosen from some distribution. $VIEW_{\mathsf{P},\mathsf{V}}[x]$ then is to be viewed as a distribution over all the possible sequences of messages sent by the prover P in the set of interactions that can be produced by $(\mathsf{P}, \mathsf{V})[x]$ and not as the sequence of a specific interaction.

Based on the above, in order to obtain zero knowledge from an IPS $(\mathsf{P}, \mathsf{V})[x]$ we need to prove that $VIEW_{\mathsf{P},\mathsf{V}}[x]$ can be easily (i.e., PPT computable) generated by the verifier V, in the case where $x \in L$. Notice how the Knowledge Act allows us to compute anything in PPT for free. Also notice how the asymmetry we have encountered in our first intuitive definition of the ZK property, carries on to this slightly more formal definition.

[4]This is not our final definition of a view, but rather a bad one used for educational purposes. However, it will suffice for now.

3.2 Subconscious / Simulator

Striving for a more formal definition of the ZK property we proceed to examine the single most important notion in proving that an IPS conveys zero knowledge: that of the subconscious or the simulator. We present this notion by examining how one would apply it in proving that the *ISO* IPS satisfies the ZK property, or equivalently that if the two input graphs G_0 and G_1 are isomorphic then the verifier can easily generate the conversation that results from $(\mathsf{P}, \mathsf{V})[(G_0, G_1)]$.

As a first step we "kill the prover". Since we are trying to show that the verifier can single-handedly produce a conversation, then the prover should not be used and we can safely dispense of it. Next, we consider that the verifier is in a "subconscious" state, call it $\mathsf{S}_{\mathsf{PPT}}$. In this state, the verifier acts as a PPT algorithm who is trying to construct a conversation, whose behavior we define below. To avoid confusion, from now on we will refer to the conversation produced by the *ISO* IPS we have seen earlier as the "original conversation" and to the conversation produced by $\mathsf{S}_{\mathsf{PPT}}$ as the "simulated conversation". Also remember that we work under the assumption that the two input graphs G_0 and G_1 are isomorphic.

In the original conversation, the prover P selects a random graph C isomorphic to G_0 and G_1, then the verifier V chooses a random bit b and finally the prover P returns a permutation π. In the simulated conversation the order of events is slightly different. $\mathsf{S}_{\mathsf{PPT}}$ first chooses a random permutation π, then chooses a random bit b and finally computes $C = \pi(G_b)$. Notice that in both conversations C is a randomly chosen graph that is isomorphic to both G_0 and G_1. The bit b is also chosen at random in both conversation. The fact that the two events occur in reverse order in the two conversations does not matter, since the random choices are independent of each other and thus they cannot affect each other, regardless of the order they are computed. The only open end is that while in the simulated conversation the permutation π is selected randomly, we do not know how π is selected in the original conversation. Since the protocol does not specify it, it would be possible that the prover selects the first permutation in lexicographical order. To completely close this end, we simply "polish" our original *ISO* protocol so as to select C at random, by choosing a random permutation π. In total, the original and the simulated conversation have the same probability of occurring.[5] Actually, what we have presented here is one round of a conversation. It clear that one could repeat the same line of reasoning to create k rounds.

Once more we are tempted to ask what happens in the case where the two input graphs G_0 and G_1 are not isomorphic. We can still produce a valid simulated conversation, but such a conversation is meaningless since it will never appear. Recall that we are considering an honest prover and we have already seen that such a prover will simply abort the protocol, thus not producing any conversation. In any case, whatever our treatment is for this case, such valid simulated conversations occur with negligible probability.

Summing up, we do not know what the behavior of the prover will be in the case where $x \notin L$ (since our protocol does not define it) and thus the best we can deliver is for $\mathsf{S}_{\mathsf{PPT}}$ to produce the same distribution over conversations in the case where $x \in L$.

A corollary of our preceding discussion is the following.

[5] The notions outlined here are considered to be hard and will be dealt with in the next several lectures.

Corollary: The simulated (false) conversation and the original conversation are indistinguishable.

Intuitively, if the conversations were distinguishable, then we could write down a conversation and check whether it is produced by the prover P or not (i.e. whether it is an original or simulated conversation) and accordingly decide whether the two input graphs G_0 and G_1 are isomorphic in PPT.

3.3 Formal Definition of Zero-Knowledgeness

Our first attempt for a formal definition comes from our discussion in the previous subsection, where only an honest verifier was considered.

Definition (First Attempt): An IPS (P,V) for a language L has the *zero-knowledgeness* property, if $\exists \mathsf{S}_{\mathsf{PPT}}, \forall x \in L, VIEW_{\mathsf{P},\mathsf{V}}[x] = \mathsf{S}[x]$.

What this definition says is that, there is a way ($\exists \mathsf{S}_{\mathsf{PPT}}$) to imitate the prover P such that if $x \in L$ the probability distributions of the simulated ($\mathsf{S}[x]$) and the original ($VIEW_{\mathsf{P},\mathsf{V}}[x]$) conversations are indistinguishable (equal). To see why this is the correct way to define the ZK property, consider the following scenario:

> You are a criminal reporter and a crime has just taken place. You are considering whether you should even bother getting out of bed and interview the inspector for the crime. Under which circumstances you would decide that staying in bed is the best solution?

An obvious answer (and the one we are looking for) to the question is that you would not bother interviewing the inspector if you could produce by yourself (simulate) an interview (conversation) that cannot be distinguished from an actual (original) interview with the inspector. Since no one could tell the difference between the two interviews, you might as well continue your sleep and later create a simulated interview for the newspaper you work for.

It seems that our first attempt contains some correct ideas, but is not still completely correct. Since we view the verifier as trying to get as much as possible out of a conversation with the prover, we wish to relax the condition that the verifier follows a predetermined protocol (i.e. that the verifier is honest). Conceivably, a devious verifier could extract more knowledge from a conversation. Since we wish to define zero-knowledgeness, then no one (not even a dishonest verifier) should be able to get additional knowledge from a conversation. These ideas lead to the following refined definition.

Definition (Second Attempt): An IPS (P,V) for a language L has the *zero-knowledgeness* property, if $\forall \mathsf{V}', \exists \mathsf{S}'_{\mathsf{PPT}}, \forall x \in L, VIEW_{\mathsf{P},\mathsf{V}'}[x]\mathsf{S}'[x]$.

Unlike the previous definition, this allows for an arbitrary verifier ($\forall \mathsf{V}'$). For each such verifier, the definition is identical to our earlier definition. That is, there is a way ($\exists \mathsf{S}'_{\mathsf{PPT}}$) that depends on the verifier, to imitate the prover P such that if $x \in L$ the probability distributions of the simulated ($\mathsf{S}'[x]$) and the original ($VIEW_{\mathsf{P},\mathsf{V}'}[x]$) conversations are indistinguishable (equal). Notice that when we refer to the simulated and original conversation here, we mean conversations in which V' plays the role of the verifier.

Notice the same motive we have in the definition of soundness for an IPS. Both in the definition of soundness and in the definition of the ZK property we treat the prover and the verifier asymmetrically. In soundness we require that the result holds for all provers (no prover can prove a wrong statement). In zero-knowledgeness we require that the result holds for all verifiers (no verifier can gain additional knowledge).

Continuing our previous example, the reporter would have no reason to talk to the inspector, if the former was behaving according to some known (to the inspector) protocol, since then the inspector would be in position to answer in such a way as to reveal no knowledge. However, if the reporter could behave maliciously and use some tricks, then it could be possible that the inspector would fall into the trap and accidentally reveal some knowledge. What our new definition says is that even in this case the inspector P can outsmart the malicious reporter and convey no knowledge, other than what the smart and easily computable (V' is still assumed to be PPT) reporter can compute in a lazy way (using S', thus V' can stay in bed!) and easily computable (S' is assumed to be PPT) way.

So, our second definition for the ZK property seems to be the correct one, in the sense that it captures what we intuitively expect, according to the Knowledge Act. It is as if we have now defined the perfect island, but can we ever hope to go there, or is our island a utopia[6]? Are there Interactive Proof Systems that can satisfy the ZK property, or can we never achieve this stricter definition of the ZK property (as opposed to our first attempt which is attainable as our example on the ISO IPS shows in the previous subsection)?

It turns out that we can attain the ZK property and in fact in most cases by using the same ideas as in proving the weaker (first attempt) definition of ZK property. That is, by trying to construct an IPS that satisfies the property than an honest verifier cannot gain any additional knowledge from the protocol, we often end up with an IPS that satisfies more than that and in particular it satisfies the stricter / stronger / more constraint definition of the ZK property.

3.4 Black-Box Simulators

Before seeing a way in which one can prove that an IPS satisfies the ZK property, we will try to get some insight on what a zero knowledge IPS looks like. As always we will proceed by examining the ISO IPS.

Notice in the ISO IPS that what the prover sends to the verifier is a random isomorphism C of one of the input graphs and a random permutation π. That is, all the verifier gets to see from the prover is just the output of a PPT computable function, i.e. either random strings, or things that can be efficiently computed. We have intuitively argued that this IPS satisfies the ZK property. This is consistent with our definition of the Knowledge Act, since the prover conveys only information that comes for free (i.e. can be easily computed by the verifier) and thus this information conveys no knowledge. In fact, this is not a simple coincidence. In any zero knowledge IPS we expect that the messages coming from the prover will only comprise of the outputs of PPT computable functions, for otherwise we

[6]The word "utopia" comes from the Greek work "$o\upsilon\tau o\pi\acute{\iota}\alpha$" which means "a non-existent place".

cannot hope to simulate the prover by a PPT algorithm, namely $\mathsf{S}_{\mathsf{PPT}}$, and thus we cannot hope to prove that the IPS satisfies the ZK property. To state it once more in a more direct way, in every IPS that satisfies the ZK property, the prover only sends messages that are PPT computable.

Now, let us see how we can prove that an IPS satisfies the ZK property. Recall that we wish to consider any (possibly malicious) verifier V'. Since the verifier V' is not bound by the IPS protocol, it can choose its messages in any way it likes. In particular, in the *ISO* IPS, the verifier V' does not have to select its bit b at random, but can choose it (actually, compute it) based on what it has already observed (i.e. it adapts its behavior based on the previous messages in the conversation). Even so, we will see that there is nothing much the malicious verifier V' can do compared to the honest verifier V and thus no verifier can extract additional knowledge from the *ISO* IPS.

Formally, a verifier V' is a PPT Turing Machine with two tapes; a random tape R and a work tape W. The Turing Machine can only read from tape R and always moving its head towards the right, while it can read or write on tape W moving in both directions. We also wish for the computation to end in a finite number of steps. Notice that this approach treats the verifier V' as a black-box. We do not know and we do not care how the verifier V' works internally. We only care of its output on an input. We can now use this black-box verifier V' as a subprocess of a simulator $\mathsf{S}'_{\mathsf{PPT}}$ and use the latter simulator to prove that an IPS (P, V) has the following property:

$$\exists \mathsf{S}'_{\mathsf{PPT}}, \forall \mathsf{V}', \forall x \in L, VIEW_{\mathsf{P},\mathsf{V}'}[x] \mathsf{S}'^{\mathsf{V}'}[x].[7]$$

Notice that this property is stronger than the ZK property. In other words, if we can prove that an IPS systems has this property (in which case we will say that the IPS has a *black-box simulation*), then this implies that the IPS also has the ZK property. Indeed, if an IPS has a black-box simulation then there exists a global algorithm $\mathsf{S}'_{\mathsf{PPT}}$ (not depending on the verifier V'), which can imitate the prover P, while if an IPS simply satisfies the ZK property then this algorithm depends on the verifier.

We might ask why not use the existence of a black-box simulation as the definition of the ZK property; after all this is what we will be proving. Well, the definition of the ZK property is adequate for capturing the intended behavior and there is no reason to strengthen it. Besides, it is possible that some day we might come up with a way to prove the ZK property for an IPS directly, without being able to prove the stronger property of an IPS having a black-box simulation.[8] Keeping the definition of the ZK property as is ensures that in that case our proof will be sufficient.

3.5 An Example

We finally can apply all the ideas in the previous subsection to see how a black-box simulation can be constructed for the case of the *ISO* IPS. What we need to show is how we define the simulator algorithm $\mathsf{S}'_{\mathsf{PPT}}$, given a black-box for a verifier V', such that the simulator

[7]Here A^B means that algorithm A uses algorithm B as an oracle. That is, A can ask a query and get an answer from B. From a complexity point of view, each query to B is only charged as a signle step of computation, even if B takes exponential time to perform its computation.

[8]In fact, this has been done to a small degree, and we intend to get to this later in the class

produces a conversation indistinguishable from the original conversation between P and V'. On input $(G_0, G_1) \in ISO$, the first step is to fill the random tape R of V' with a random string. Then to produce a single round i of a conversation:

1. The simulator $\mathsf{S}'_{\mathsf{PPT}}$ selects a random bit b_i, a random permutation π_i and computes $C_i = \pi_i(G_{b_i})$. That is the simulator $\mathsf{S}'_{\mathsf{PPT}}$ creates a possible round of a conversation.

2. The simulator $\mathsf{S}'_{\mathsf{PPT}}$ sends (feeds) C_i to the black-box verifier V' and gets the output d_i, which is what the verifier V' would have returned if it were an actual conversation and it received C_i from the prover P.

3. The simulator $\mathsf{S}'_{\mathsf{PPT}}$ checks whether $b_i = d_i$ and if yes, it keeps the created round. Otherwise, it ignores the created round and repeats the scheme. Essentially, what the simulator $\mathsf{S}'_{\mathsf{PPT}}$ does here is that it checks whether the simulated round could have been produced by the interaction of the prover P with the actual verifier V'. It is easy to see that the verifier V' accepts the round if and only if $b_i d_i$.

Remark: In the third step, we could ask that in addition to rejecting a round, the verifier V' should be reset back to the state it had before the round started and thus to forget that the failed round ever took place. This however is unnecessary, given that even if the verifier V' keeps track of all the failed rounds, he still has no way of predicting the random choices that the simulator $\mathsf{S}'_{\mathsf{PPT}}$ (in the simulation) or the prover P (in an actual conversation) send to it.

A colorful example that gives some intuition on the above algorithm is the following. We try to achieve a goal. If it succeeds, we simply continue onwards. If it fails, we behave like an ostrich! We put our head in the ground and pretend it never happened.

Using the above algorithm, the simulator $\mathsf{S}'_{\mathsf{PPT}}$ can construct a k-round conversation. Notice that the idea presented in the algorithm is in fact very general. One could pass any test / exam using a similar approach, as it is illustrated in the following scenario:

> You are allowed to take an exam as many times as necessary. What you do then, is you select a random chapter in the book (that contains, say, ten chapters) and you only study that. If the exam happens to be on that chapter, then you pass the test. Otherwise, you repeat the whole scheme from the beginning.

Here the test maker plays the role of the possibly malicious verifier. The test maker is determined to make you fail the test and will use whatever means possible in creating the exam, including keeping a track of all your previous attempts. Now, since you always choose one chapter at random, then no matter what the test maker does, you always have a 1/10 probability of passing the test. In particular, you expect that you will pass the test in 10 tries. In the same way, the simulator $\mathsf{S}'_{\mathsf{PPT}}$ chooses the random bit b_i at random and "passes the test" if the test d_i happens to match its selection. No matter how the verifier V' chooses d_i, there exists a 1/2 probability of "passing the test" and this happens in an expected number of 2 tries. In total, we create a k-round conversation in an expected number of $2k$ tries.

One important distinction is that unlike our scenario where the test maker gets no information from you prior to the exam, here the verifier V' gets a graph $C_i \pi_i(G_{b_i})$ whose creation involves b_i. However, this does not help the verifier V' since C_i is a random graph isomorphic to both G_0 and G_1, since $(G_0, G_1) \in ISO$. Thus, the verifier V' has no way of determining the graph G_{b_i} from which C_i was computed and thus has no way of determining b_i.

The above gives the required black-box simulation. Still there remains an open issue. It is conceivable that the process of eliminating failed rounds might skew the distribution over the possible resulting simulated conversations so it would no longer be indistinguishable from the distribution over the original conversations. For example, if the professor were to ask every student in the class with short hair to leave, then the distribution of the remaining students in the class with respect to the length of their hair is no longer random, but skewed. This is not exactly the case in the pruning of failed rounds. Unlike the "short-hair" pruning, the pruning of failed rounds is done totally at random. Recall that a round is pruned if and only if $b_i \neq d_i$ and b_i is chosen at random. So in some sense we only forget things that we didn't know and thus the distribution is not affected. The following scenario makes things totally clear:

> the TA closes his eyes, while the professor writes a random string on the board. Then the TA (without looking at the board) successively keeps or rejects (at will) each bit in the string. If we only consider the bits that the TA kept, then we have a randomly chosen string!

In the scenario above, the professor plays the role of the simulator $\mathsf{S}'_{\mathsf{PPT}}$ and the random string on the board stands for all the random $b_i's$ the simulator $\mathsf{S}'_{\mathsf{PPT}}$ creates. the TA plays the role of the verifier V' and the fact that he cannot see the bits on the board resembles the fact that the verifier V' has no information about the $b_i's$ of the simulator $\mathsf{S}'_{\mathsf{PPT}}$. The choices of the TA on whether to keep a bit or to reject it correspond to the verifier V' selecting a bit d_i as its output and comparing it with b_i, keeping the former if and only if $b_i = d_i$. As a variation that will bring the example closer to the algorithm, the TA announces a bit value (0 or 1) and the professor keeps the corresponding bit on the board if and only if it matches the TA's bit value. In this variation, the TA's bit value is d_i and the bit b_i on the board is kept if and only if $b_i = d_i$. In the same sense that rejecting bits based on the TA's choices does not affect the distribution of the resulting bit-string, neither does rejecting rounds based on the verifier's response affect the distribution of the resulting conversation. In both cases the pruning is blind and thus preserves the distribution.

4 Summary

Concluding, in today's lecture we have seen that the proposed ISO IPS that was introduced at the end of the last lecture satisfies the ZK property. That is, in addition to the fact that there exists a very easy prover algorithm P and a very easy verifier algorithm V which together constitute an IPS for ISO, we have also seen that there exists a fixed simulator algorithm S' independent of the input $(G_0, G_1) \in ISO$ and the verifier V', that can simulate a conversation. Thus, in the case where $(G_0, G_1) \in ISO$, no one can extract knowledge from the ISO IPS, other than what one could have computed in a "lazy and easy" way.