1 Zero-Knowledge Definitions

Definition 1 We say $(P,V)$ is an efficient prover zero-knowledge proof system for a language $L$ and relation $R_L$ if

1. The prover $P$ runs in polynomial time.

2. The protocol is complete. That is, for every $x \in L$ there exists a witness $w \in R_L(x)$ such that
   \[
   \Pr[P(x,w) \leftrightarrow V(x) \text{ accepts}] = 1
   \]

3. The protocol is sound against unbounded provers. That is, for $x \notin L$, we have
   \[
   \Pr[P^*(x,w) \leftrightarrow V(x) \text{ rejects}] \geq \frac{1}{2}
   \]
   for any prover $P^*$ of arbitrary complexity and any witness $w \in R_L(x)$.

4. There exists an expected polynomial time probabilistic machine $S$ (a simulator) such that for all PPT $V^*$, for all $x \in L, w \in R_L(x), z \in \{0,1\}^*$ we have
   \[
   \{\text{View}_{V^*}(P(x,w) \leftrightarrow V^*(x,z))\} \simeq_c \{S^{V^*}(x,z)\}
   \]

   Of course, the soundness probability can be amplified to be greater than any $1 - 1/2^k$, for arbitrary $k > 0$, by repeating the simulation multiple times.

   The order of the quantifiers in item 4 matters. If we did not quantify over simulators before quantifying over other strings, then we could hard-code information about the verifiers or strings $x$ and $w$ into our simulator. Since we would like our simulator to be universal, then this is not acceptable.

If we quantified first over verifiers $V^*$ and then over simulators $S$, then this variant is considered black-box zero-knowledge. Our standard definition is considered non black-box zero-knowledge.

There are also exist variants that use statistical indistinguishability rather than computational indistinguishability.

The $z$ in item 4 is considered auxiliary input.

We will discuss the importance of requiring expected polynomial time in the next section.

2 Graph Isomorphism

Recall our protocol for graph isomorphism: the interaction is $P(x,w) \leftrightarrow V(x)$ where $x$ represents graphs $G_0 = (V,E_0)$ and $G_1 = (V,E_1)$ and $w$ represents a permutation $\pi$ on $V$ such that $\pi(G_0) = G_1$.

1. $P$ samples a random permutation $\sigma : V \to V$ and sends the graph $H = \sigma(G_1)$ to $V$. 

8-1
2. $V$ samples a random bit $b$ and sends it to $P$.

3. If $b = 1$, then $P$ defines a permutation $\tau$ to be $\sigma$. If $b = 0$, then instead $\tau = \sigma \circ \pi$. $P$ then sends $\tau$ to $V$.

4. $V$ verifies that $\tau(G_b) = H$ and accepts if so.

We will show that this is an efficient prover zero-knowledge proof system. It is clear that if $G_0$ and $G_1$ are isomorphic, then this protocol will succeed with probability 1.

For soundness, observe that if $G_0$ is not isomorphic to $G_1$, then the graph $H$ that $P$ sends to $V$ in step 1 of the protocol can be isomorphic to at most one of $G_0$ or $G_1$. Since $V$ samples a bit $b$ uniformly at random in step 2, then there is a probability of at most $1/2$ that $P$ can produce a valid isomorphism in step 3.

For zero knowledge, consider the following simulator $S$ with input $G_0$ and $G_1$ (with vertex set $V$) and verifier $V^*$:

1. Sample a bit $b$ uniformly at random.
2. Sample a permutation $\pi : V \rightarrow V$ uniformly at random.
3. If $V^*$ outputs $b$ on input $\pi(G_b)$, then output $(\pi(G_b), b, \pi)$ and terminate $S$. Otherwise, restart at step 1.

Note that if $G_0 \simeq G_1$, then $\pi(G_b)$ is statistically independent of $b$ because $b$ and $\pi$ are sampled uniformly. Thus, with probability $1/2$, $V^*$ will output $b$ so on average, two attempts will be needed before $S$ terminates. It follows that $S$ will terminate in expected polynomial time.

Since $b$ is sampled uniformly at random, $\pi(G_b)$ is uniformly distributed with all graphs of the form $\sigma(G_1)$ where $\sigma$ is sampled uniformly at random from permutations on $V$. Thus, the output $\pi(G_b)$ in our simulator will be identically distributed with the output $H$ in our graph isomorphism protocol. In step 3 of our graph isomorphism protocol, note that $\tau$ is distributed uniformly at random. This is because composing a uniformly random permutation with a fixed permutation will not change its distribution. Thus $\tau$ will be identically distributed with $\pi$ in our simulator. It follows that the transcripts outputted by our simulator will be identically distributed with the transcripts produced by the graph isomorphism protocol.

3 Zero-Knowledge for NP

An $n$-coloring of a graph $G = (A,E)$ is a function $c : A \rightarrow \{1, \ldots, n\}$ such that if $(i,j) \in E$, then $c(i) \neq c(j)$. So we want to paint each vertex of a graph a certain color so that the endpoints of any edge are colored differently.

In the graph 3-coloring problem (3COL), we are given a graph and asked if there exists a 3-coloring. In this section, we will provide a computational zero knowledge proof for 3COL. It is a fact that 3COL is NP-complete, so any problem in NP has a polynomial time reduction to 3COL. Thus, by giving a zero knowledge proof for 3COL, we will show that there are zero knowledge proofs for all of NP.

We will first give a high-level description of a zero-knowledge protocol for 3COL. Suppose a prover $P$ wants to convince a verifier $V$ that his graph $G$ is 3-colorable without revealing what the coloring
c actually is. If the three colors we use are red, green, and blue, then note that if we colored all the red vertices blue, all the green vertices red, and all the blue vertices green, we would still have a valid 3-coloring. In fact, if \( \phi \) was any permutation on the color set of red, green, and blue, then \( \phi \circ c \) would be a valid 3-coloring of \( G \).

\( P \) asks \( V \) to leave the room and then samples a random permutation \( \phi \) of the three colors. He colors the vertices of \( G \) according to \( \phi \circ c \), then covers all the vertices with cups. At this point, \( P \) invites \( V \) back into the room. \( V \) is allowed to pick one edge and then uncover the two endpoints of the edge. If the colors on the two endpoints are the same, then \( V \) rejects \( P \)'s claim that the graph is 3-colorable.

If the colors on the two endpoints are different, then \( V \) leaves the room again, \( P \) samples \( \phi \) randomly, and the process repeats itself. Certainly if \( G \) is actually 3-colorable, then \( V \) will never reject the claim. If \( G \) is not 3-colorable, then there will always be an edge with endpoints that are colored identically and \( V \) will eventually uncover such an edge.

Note that \( V \) does not gain any information on the coloring because it is masked by a (possibly) different random permutation every time \( V \) uncovers an edge. Of course this protocol depends on \( P \) not being able to quickly recolor the endpoints of an edge after removing the cups. This is why we need commitment schemes.

3.1 Commitment Schemes

We want to construct a protocol between a sender and a receiver where the sender sends a bit to the receiver, but the receiver will not know the value of this bit until the sender chooses to "open" the data that he sent. Of course, this protocol is no good unless the receiver can be sure that the sender was not able to change the value of his bit in between when the receiver first obtained the data and when the sender chose to open it.

**Definition 2** A commitment scheme is a PPT machine \( C \) taking input \((b,r)\) that satisfies two properties:

- (perfect binding) For all \( r,s \), we have \( C(0,r) \neq C(1,s) \).
- (computational hiding) \( \{C(0,U_n)\} \simeq \{C(1,U_n)\} \)

So for the sender to "open" the data, he just has to send his value of \( r \) to the receiver. We say that \( r \) is a decommitment for \( C(x,r) \). Why do we require perfect binding instead of just statistical binding? If there existed even a single pair \( r,s \) where \( C(0,r) = C(1,s) \), then the sender could cheat. If he wished to reveal a bit value of 0 then he could just offer \( r \) and if he wished to reveal a bit value of 1 then he could just offer \( s \).

We can use injective one-way functions to construct commitment schemes.

**Theorem 1** If injective one-way functions exist, then so do commitment schemes.

**Proof.** We can let \( f \) be an injective one-way function. Recall from Lecture 3 that \( f'(x,r) := (f(x),r) \) will also be an injective one-way function with hard-core bit \( B(x,r) := \langle x,r \rangle \). We claim that \( C(b,x,r) := (f'(x,r),b \oplus B(x,r)) \) is a commitment scheme.

If \((x,r) \neq (y,s)\) then \( C(0,x,r) \neq C(0,y,s) \) because \( f' \) is injective. Since \( C(0,x,r) = (f'(x,r),B(x,r)) \neq (f'(x,r),\overline{B(x,r)}) = C(1,x,r) \), then \( C \) satisfies perfect binding.

8-3
Suppose $D$ can distinguish $C(0, U_n)$ from $C(1, U_n)$. Then we can distinguish $B(x, r)$ from $\overline{B(x, r)}$ given $f'(x, r)$ which contradicts the fact that $B(x, r)$ is a hard-core bit for $f'(x, r)$. Thus, $C$ has the computational hiding property.

We can extend the definition of commitment schemes to hold for messages longer than a single bit. These commitment schemes will work by taking our commitment schemes for bits and concatenating them together. For the extended definition, we require that for any two messages $m_0$ and $m_1$ of the same length, the ensembles $\{C(m_0, U_n)\}$ and $\{C(m_1, U_n)\}$ are computationally indistinguishable.

### 3.2 3COL Protocol

Below we describe the protocol $P(x, z) \leftrightarrow V(x)$, where $x$ describes a graph $G = (\{1, \ldots, n\}, E)$ and $z$ describes a 3-coloring $c$:

1. $P$ picks a random permutation $\pi : \{1, 2, 3\} \to \{1, 2, 3\}$ and defines the 3-coloring $\beta := \pi \circ c$ of $G$. Using a commitment scheme $C$ for the messages $\{1, 2, 3\}$, $P$ defines $\alpha_i = C(\beta(i), U_n)$ for each $i \in V$. $P$ sends $\alpha_1, \alpha_2, \ldots, \alpha_n$ to $V$.

2. $V$ uniformly samples an edge $e = (i, j) \in E$ and sends it to $P$.

3. $P$ opens $\alpha_i$ and $\alpha_j$.

4. $V$ will accept only if it received valid decommitments for $\alpha_i$ and $\alpha_j$, and if $\beta(i)$ and $\beta(j)$ are distinct and valid colors.

It is clear that this protocol is PPT. If $G$ is not 3-colorable, then there will be at least a $1/|E|$ probability that $V$ will reject $P$’s claim in step 4. Since $|E| \leq n^2$ we can repeat the protocol polynomially many times to increase the rejection probability to at least $1/2$.

We will now show that this protocol is zero-knowledge. We describe a simulator $S$ below, given a verifier $V^*$:

1. Sample an edge $e = (i, j) \in E$ uniformly at random.

2. Assign $c_i$ and $c_j$ to have distinct values from $\{1, 2, 3\}$ and do so uniformly at random. Set $c_k := 1$ for all $k \neq i, j$.

3. Compute $n$ random keys $r_1, \ldots, r_n$ and set $\alpha_i = C(c_i, r_i)$ for all $i$.

4. Let $e' \in E$ be the response of $V^*$ upon receiving $\alpha_1, \ldots, \alpha_n$.

5. If $e' \neq e$, then terminate and go back to step 1. Otherwise, proceed. If $S$ returns to step 1 more than $2n|E|$ times, then output fail and halt the program.

6. Print $\alpha_1, \ldots, \alpha_n, e$, send $r_i$ and $r_j$ to $V^*$ and then print whatever $V^*$ responds with.

By construction, $S$ will run in polynomial time. However, sometimes it may output a fail message. We will show that this occurs with negligible probability.

Suppose that for infinitely many graphs $G$, $V^*$ outputs $e' = e$ in step 4 with probability less than $1/2|E|$. If this is true, then it is possible for us to break the commitment scheme $C$ that we use in $S$. Consider a modified version of $S$ called $\tilde{S}$, where in step 2 we set $c_i = 1$ for all $i$. Note that in
this case, $V^*$ cannot distinguish between any of the edges so the probability that it returns $e' = e$ is $1/|E|$.

If we gave $V^*$ a set of commitments $\alpha_k = C(1, r_k)$ for random keys $r_k$, then we would be in the setting of $\tilde{S}$. If we gave $V^*$ the commitments $\alpha_k$ but with two of the values set to $C(c, r)$ and $C(c', r')$ where $c, c'$ are distinct random values from $\{1, 2, 3\}$ and $r, r'$ are random keys, then we are in the setting of $S$. This implies that it possible to distinguish between these two commitment settings with a probability of at least $1/2|E|$ which is non-negligible. It follows that $V^*$ outputs $e' = e$ with probability less than $1/2|E|$ for only finitely many graphs $G$. Thus, the probability that $S$ outputs fail in the end is less than $(1 - 1/2|E|)^{2n|E|} < 1/e^n$ which is negligible.

Now we need to argue that the transcripts generated by $S$ are computationally indistinguishable from the transcripts generated by $P \leftrightarrow V^*$. Again, we consider a modified version of $S$, called $S'$, given a 3-coloring of its input $G$ as auxiliary input. In step 2 of the simulation, $S'$ will choose a random permutation of the colors in its valid 3-coloring for the values of $c_i$ rather than setting all but two values $c_i$ and $c_j$ equal to 1. Note that this is how our protocol between $P$ and $V$ behaves. Observe that $P \leftrightarrow V^*$ is computationally indistinguishable from $S'$ because $S'$ outputs fail with negligible probability. Thus, it suffices to show that $S$ and $S'$ are computationally indistinguishable.

Again, we will suppose otherwise and argue that as a result we can distinguish commitments. We consider two messages $m_0$ and $m_1$ of the same length where $m_0$ consists of $n - 2$ instances of the message 1 and two committed colors $c_i$ and $c_j$ (for a random edge $(i, j) \in E$) and $m_1$ consists of a committed random 3-coloring of $G$ (with a random edge $(i, j) \in E$) chosen. Observe that by feeding the former message to $V^*$ we are in the setting of $S'$ and by feeding the latter message to $V^*$ we are in the setting of $S$. If we could distinguish those two settings, then we could distinguish the commitments for $m_0$ and $m_1$. This contradiction completes our argument that our 3-coloring protocol is zero-knowledge.