Zero Knowledge

CS6230: Topics in Information Security
Lecture 5 (Sep 7, 2021)
Lecture and notes: Prashant Nalini Vasudevan

Today, we will start studying a kind of interactive proof called a *Zero Knowledge (ZK) proof*. These are interactive proofs where the prover can convince the verifier that a statement is true without revealing anything else to the verifier. Consider, for example, the IP we saw for Graph Non-isomorphism. We state now a slightly modified, but equivalent, protocol (P, V) that works as follows given a graphs G_0, G_1 over n vertices:

1. V samples a uniformly random relabelling permutation $R : [n] \to [n]$, and a uniformly random bit $b \leftarrow \{0, 1\}$. It sends $R(G_b)$ to the prover.
2. If G_0 and G_1 are isomorphic, P sets $b' = \perp$. Else, P sets the bit $b' = 0$ if G' is isomorphic to G_0, and $b' = 1$ otherwise. It sends b' to V.
3. V accepts if $b = b'$, and rejects otherwise.

Note that, in any execution of this protocol with the honest prover P, the verifier V does not learn anything that it does not already know, except for whether G_0 and G_1 are isomorphic. If $G_0 \equiv G_1$, then the honest P only sets $\perp$ to V, which clearly conveys nothing. If $G_0 \not\equiv G_1$, the only information V receives from P is the bit b' which, if P is behaving honestly, is equal to b, which V already knows! (Note, however, that these arguments are only valid if V behaves honestly and follows the protocol.)

1 Perfect Zero Knowledge

Defining the zero-knowledge property is a non-trivial task. We require not that some specific piece of information is hidden from the verifier, but that nothing at all is revealed to it. Note that anything that the verifier already knows or can compute can be excluded – we need to capture the fact that the verifier learns nothing *new* from the proof. An elegant fomulation of this was discovered by Goldwasser, Micali, and Rackoff [GMR85]. They asked that the verifier be able to *simulate* anything that it learnt in the course of the protocol on its own, without having to talk to a prover.

In order to state their definition, we will need the notion of the *view* of a party in an interactive protocol. For our purposes, we only need to consider the view of the verifier, defined as follows.

Definition 1.1. In an execution of an interactive proof (P, V) with input x, the *view* of the verifier, denoted $\mathit{view}_V^P(x)$, consists of the random string used by the verifier in the execution, the set of all messages sent and received by the verifier, and its output (that is, accept or reject).

We would like our zero-knowledge property to hold not just when the verifier behaves honestly according to the protocol, but also when it deviates from the protocol. We will denote such verifier strategies by V^*, and their interaction with the prover by (P, V^*). For the purpose of zero-knowledge, we will only be concerned with its interaction with the honest prover – think of zero-knowledge as a form of protection for the prover; if the prover deviates from the protocol, the protocol is no longer obliged to provide this protection. Consideration of cheating verifiers will be especially useful when we are interested in cryptographic protocols that use zero-knowledge to protect secrets that the prover possesses.

For ease of notation, we will use $\mathit{view}_{V^*}^P(x)$ to also denote the random variable corresponding to the distribution of the view of the verifier V^* over the randomness of V^* and P in the protocol. The following is a modern statement of a version of the GMR definition of zero-knowledge.

Definition 1.2 (Perfect Zero Knowledge)**.** An interactive proof (P, V) for a language L is said to be *perfectly zero knowledge (PZK)* if, for every possible PPT verifier algorithm V^*, there exists a PPT *simulator* algorithm S_{V^*} such that:

- On any input x, $S_{V^*}(x)$ outputs $\perp$ with probability at most $1/2$
- For any input $x \in L$, the distribution of $S_{V^*}(x)$ conditioned on $S_{V^*}(x) \neq \perp$ is identical to $\mathit{view}_{V^*}^P(x)$

Note that the simulator above is required to run in polynomial time, and does not have access to the prover. This captures the intuitive notion of the verifier being able to simulate on its own anything it learnt in the course of the protocol. We allow the simulator to fail with some probability because it turns out that without this allowance the definition is too strict to be of much use. The probability of failure can be made exponentially small by repetition.

We make the simulation requirement only for $x \in L$. This is because, for $x \notin L$, the honest prover's algorithm may be taken to simply be to send $\perp$ as its only message and terminate, as in the above protocol for GNI. Think, for a minute, of an alternative simulator that needs to work for all inputs x, but is given as additional input a bit that tells it whether $x \in L$ or not. This capture more precisely our initial idea that the protocol is allowed to reveal whether $x \in L$ and nothing else. Since the protocol when $x \notin L$ is just $\perp$, it is easy to see that this definition would be equivalent to Definition 1.2.

The class of languages that have PZK proofs is denoted by PZK. If the zero-knowledge property of a protocol (P, V) only holds with respect to the honest prover V, the protocol is said to be *honest-verifier zero-knowledge* (HVPZK). The class of languages that have HVPZK proofs is denoted by HVPZK. In general, honest-verifier zero-knowledge does not imply zero-knowledge, but we will see certain settings in which it does.

2 A PZK protocol for Graph Isomorphism

As pointed out earlier, the protocol we saw for GNI is honest-verifier perfect zero knowlege. In order to prove this, we will need to present a PPT simulator S that is such that for any (G_0, G_1) that are not isomorphic, the distribution of $S(G_0, G_1)$ is identical to that of the view $view_V^P(G_0, G_1)$ of the honest verifier V when interacting with the honest prover P. This view consists of: the random relabelling permutation R, the random bit b, the verifier's message $R(G_b)$, and the prover's response b'.

Recall the reason we thought this protocol was zero-knowledge to begin with – the bit b' is something that the verifier already knows. More precisely, if G_0 and G_1 are not isomorphic, and the prover is honest, the bit b' is always equal to b. Thus, the simulator is simple: sample R and b at random as the verifier would, compute $R(G_b)$, and output $(R, b, R(G_b), b)$. This is identical to the verifier's view in the original protocol.

Why is this protocol not zero-knowledge against cheating verifiers? Consider a verifier V^* that has a third graph H, and it wants to know whether H is isomorphic to G_0. Suppose G_0 is not isomorphic to G_1. Then, if V^* simply sends H as its message to P, the response b' will tell whether this is the case. Here, V^* has learnt something that it did not know before and, if graph isomorphism cannot be solved in polynomial time, something that it could not have computed on its own. Thus the protocol is no longer zero-knowledge.

This protocol can be made PZK against cheating verifiers under a slightly weaker definition of the zero-knowledge property that allows the simulator to run in *expected* polynomial time. We will see instead a PZK protocol for its complement, Graph Isomorphism.

2.1 Graph Isomorphism

The language GI consists of pairs of graphs (G_0, G_1), both on the same number n of vertices, such that G_0 and G_1 *are* isomorphic. It is easy to see that GI is in NP – the isomorphism itself is a witness. However, this NP proof does not give a zero-knowledge proof of this isomorphism. Consider, instead, the protocol (P, V) that works as follows given input (G_0, G_1):

1. If G_0 and G_1 are not isomorphic, P sends $\perp$ to V and terminates.

2. If not, P samples a uniformly random relabelling permutation $R : [n] \to [n]$, and a uniformly random bit b. It sends $H = R(G_b)$ to V.

3. V samples a random bit b' and sends it to P.

4. P finds a uniformly random relabelling R' such that $R'(G_{b'}) = H$, and sends R' to V.

5. V accepts iff $R'(G_{b'}) = H$.

Completeness and Soundness. When G_0 and G_1 are isomorphic, any H that is isomorphic to one of them is also isomorphic to the other. Thus, P can always find an R' as necessary, and the protocol is perfectly complete. If they are not isomorphic, then there is at most one of G_0 and G_1 that is isomorphic to any H, and so no P^* can make V accept with probability more than 1/2.

Perfect Zero Knowledge. In order to prove that protocol is PZK, for any possible verifier strategy V^*, we will have to show there exists a simulator S_{V^*}. Note that, when G_0 and G_1 are isomorphic, the view of the verifier consists of: its own randomness r_{V^*}, the prover's message H, its message b', the relabelling R', and its own output. The simulator S_{V^*} will rely on the fact that the bit b is hidden from V^* if the graphs are isomorphic (a fact that we used to argue soundness of the protocol for GNI). It operates as follows given input (G_0, G_1):

1. Generate R, b, and $H = R(G_b)$ as P does.
2. Sample r_{V^*} uniformly at random from the appropriate domain, and compute $b' \leftarrow V^*((G_0, G_1), H; r_{V^*})$.
3. If $b \neq b'$, output $\perp$ and terminate.
4. if $b = b'$, compute the output of the verifier $out \leftarrow V^*((G_0, G_1), H, b, R; r_{V^*})$, and output (r_{V^*}, H, b, R, out).

To prove PZK, we need to show two properties of the simulator S_{V^*} on input G_0 and G_1 that are isomorphic: that it outputs $\perp$ with probability at most 1/2, and that, conditioned on this not happening, its output is distributed identically to $view^P_{V^*}(G_0, G_1)$.

The first follows from the fact that the distribution of H is independent of b. That is, the distributions of random relabellings of G_0 and G_1 are identically distributed. In other words, given H, b is completely random. Thus, $b' = b$ will happen with probability 1/2, irrespective of how V^* computes b'.

To see the second, we argue for the terms in the view in order. Note, first of all, that $b = b'$ happens with probability 1/2 independently of anything in the view. That is, even if we fix some values of r_{V^*}, H, b', this happens with probability 1/2. Thus, conditioning on $b = b'$ does not change the joint distribution of these variables. This implies that the (r_{V^*}, H, b) output by the simulator is distributed identically to the corresponding parts of the actual protocol. (Do you see how this argument works? Try spelling it out in more detail yourselves.)

Next, conditioned on (r_{V^*}, H, b), the R that the simulator has is uniformly distributed over all relabellins R' such that $R'(G_b) = H$. This is because R was originally sampled uniformly at random from all distributions. Finally, the output of V^* is a deterministic function of all these variables. Thus, the entire output of S_{V^*}, conditioned on $b = b'$, is identically distributed to the view of V^*. This proves that the protocol is PZK.

These two ZK protocols – the HVPZK protocol for GNI, and the above for GI – are archetypes of two paradigms that will recur in the construction of ZK protocols. We will see a more general instance of the GNI one later, and an instance of the GI approach now.

3 Computational Zero Knowledge

While PZK is useful as a conceptual exercise, its use is limited by its stringent requirement that the simulated distribution is identical to the actual view of the verifier. For instance, it is known that only languages in $\mathsf{AM} \cap \mathsf{coAM}$ can have PZK protocols. In particular, no NP-complete problem can have such protocols unless the polynomial hierarchy collapses. Almost always, relaxations of this requirement are good enough, both in theoretical and practical applications. We will see two such relaxations that are of significance. We start with a relaxation to requiring that the simulated and actual distributions be just computationally indistinguishable.

Computational Indistinguishability. In order to define computational indistinguishability, we will need to think of *ensembles* of distributions rather than individual distributions. This is simply a family of distributions $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$, where each D_λ is over some domain $\{0,1\}^{\ell(\lambda)}$ for some function ℓ. Normally, this λ represents a "security parameter" that represents the level of security provided by using the distribution D_λ for some purpose.

Definition 3.1. Two ensembles of distributions $D_0 = \{D_{0,\lambda}\}$ and $D_1 = \{D_{1,\lambda}\}$ are said to be *computationally indistinguishable* if, for any PPT algorithm A, and any constant $c \in \mathbb{N}$, there exists a $\lambda_c \in \mathbb{F}$ such that for all $\lambda > \lambda_c$:

$$\left|\Pr_{x \leftarrow D_{0,\lambda}}\left[A(1^\lambda, x) = 1\right] - \Pr_{x \leftarrow D_{0,\lambda}}\left[A(1^\lambda, x) = 1\right]\right| < \frac{1}{\lambda^c}$$

We denote this by the notation $D_0 \approx_c D_1$. In other terms, consider the following function that captures the *advantage* of an algorithm A in distinguishing between D_0 and D_1:

$$Adv_A^{D_0,D_1}(\lambda) = \left|\Pr_{x \leftarrow D_{0,\lambda}}\left[A(1^\lambda, x) = 1\right] - \Pr_{x \leftarrow D_{0,\lambda}}\left[A(1^\lambda, x) = 1\right]\right|$$

The ensembles D_0 and D_1 are computationally indistinguishable if, for any PPT A, this advantage is $o(1/\mathsf{poly}(\lambda))$. We say that any such function is *negligible* in λ. We use this as the bound because such low advantage cannot be amplified to be a constant with any polynomial number of repetitions of A.

The definition of computational zero-knowledge may be obtained as a natural relaxation of Definition 1.2, though without needing to explicitly allow the simulator to fail with small probability, as this can be absorbed into the distinguishing probability. We will need to adjust the setup of our protocols to allow for the prover and verifier (and simulator) to be given an additional input 1^λ that specifies the level of security we require of the zero-knowledge. We can define the corresponding distribution ensembles $S(x) = \left\{S(1^\lambda, x)\right\}_{\lambda \in \mathbb{N}}$ and $view_{V^*}^{P}(x) = \left\{view_{V^*(1^\lambda)}^{P(1^\lambda)}(x)\right\}_{\lambda \in \mathbb{N}}$

Definition 3.2. An interactive proof (P, V) for a language L is said to be *computationally zero knowledge (CZK)* if, for every possible PPT verifier algorithm V^*, there exists a PPT *simulator* algorithm S_{V^*} such that for any input $x \in L$, the ensembles $S_{V^*}(x)$ and $view_{V^*}^{P}(x)$ are computationally indistinguishable.

Unlike PZK, it is known that all of NP has CZK protocols under reasonable assumptions. In fact, all of IP does. That is, under these assumptions, any language that has an interactive proof has a zero-knowledge interactive proof.

4 CZK for NP

We will see now how to use the high-level ideas behind the PZK proof for GI to construct, under computational assumptions, a CZK protocol for any language in NP. Originally due to Goldreich, Micali, and Wigderson [GMW86], and these protocols work by first reducting the NP-language to a specific NP-complete problem, and then using a CZK protocol for that NP-complete language. This language is 3-Colouring, which consists of all graphs that can be coloured using at most 3 in such a way that on any edge of the graph, the two vertices are coloured differently. We first state the cryptographic primitive whose existence is the computational assumption that the protocol uses, and then present the protocol for 3-Colouring itself.

4.1 Commitments

A cryptographic commitment is a way for a "sender" S to convey to a "receiver" R a commitment to a message m in such a way that:

1. Until the sender chooses to open the commitment at a later point, the message m is hidden from the receiver.
2. For almost all commitments, there is a unique message m that the sender can open it to.

There are many flavours of commitments, and we will define more precisely now the one that is useful to us. The commitment scheme consists of an interactive protocol (S, R), where S and R are polynomial-time algorithms. Both parties are given a security parameter 1^λ, S uses a random string s, and R uses a random string r. In addition, S is given a secret input $m \in M$ from some domain M. After any execution of the protocol, the view of the receiver constitutes the commitment that the sender makes to its secret input m. It can, at a later time, open this commitment by sending to the receiver m and the random string s that it used in the protocol. The receiver then simulates the execution of the protocol on its own using s as the sender's randomness, and checks that it results in the view that it actually had.

Denote by $view_{R(1^\lambda;r)}^{S(1^\lambda,m;s)}$ the view of R in the protocol when R and S use the specified strings r and s, respectively, as randomness. Denote by $view_{R(1^\lambda)}^{S(1^\lambda,m)}$ the random variable corresponding to selecting the r and s uniformly at random, and define the ensemble $view_R^{S(m)} = \left\{view_{R(1^\lambda)}^{S(1^\lambda,m)}\right\}_{\lambda\in\mathbb{N}}$. The above two properties can now be formalised as follows.

Definition 4.1 (Computational Hiding)**.** A commitment scheme (S, R) is *computationally hiding* if, for any distinct $m, m' \in M$, and any possible strategy R^* of the receiver, the views of R^* in the protocol when S is given m and m' are computationally indistinguisable. That is,

$$view_{R^*}^{S(m)} \approx_c view_{R^*}^{S(m')}$$

Above, we are satisfied with computational security because, in our construction later, we will be using this scheme with the prover as S and the verifier as R. As the verifier is computationally bounded, computational security against it is good enough. The next property, however, represents security against malicious senders, and so needs to hold against computationally unbounded adversaries.

Definition 4.2 (Statistical Binding)**.** A commitment scheme (S, R) is *statistically binding* if, for all but a negligible fraction of random strings r, there do not exist distinct $m, m' \in M$ and random strings s, s' for the sender such that $view_{R(1^\lambda,r)}^{S(1^\lambda,m;s)}$ and $view_{R(1^\lambda,r)}^{S(1^\lambda,m';s')}$ are that same. That is,

$$\Pr_r\left[\exists m \neq m' \in M, s, s' : view_{R(1^\lambda,r)}^{S(1^\lambda,m;s)} = view_{R(1^\lambda,r)}^{S(1^\lambda,m';s')}\right] \leq \text{negl}(\lambda)$$

Such commitment schemes can be constructed using pseudorandom generators (PRGs) or, equivalently, one-way functions (OWFs), which are basic cryptographic assumptions whose existence is a rather mild assumption.

Exercise 1. *A PRG $G : \{0,1\}^n \to \{0,1\}^m$, where $m > n$ is an efficiently computable function such that the distribution $G(x)$ where $x \leftarrow \{0,1\}^n$ and the distribution $y \leftarrow \{0,1\}^m$ are computationally indistinguishable. Show how to construct a 2-message commitment scheme using a PRG of the form $G : \{0,1\}^n \to \{0,1\}^{3n}$.*

4.2 Protocol for 3-Colouring

With the above notion of commitments in hand, the protocol (P, V) for 3-Colouring works as follows given as input a graph $G = ([n], E)$ on n vertices:

1. If G is not 3-colourable, P sends $\perp$ and terminates.
2. Else, P samples a random 3-colouring $\phi : [n] \to \{0, 1, 2\}$ of G.
3. For each $v \in [n]$, P and V run the commitment protocol with P as the sender and V as the receiver, where P commits to $\phi(v)$.
4. V picks a random edge $(u, v) \in E$ and sends it to P.
5. P opens the commitments to $\phi(u)$ and $\phi(v)$.
6. V checks that the two opened values are different, and contained in $\{0, 1, 2\}$. If so, it accepts, otherwise it rejects.

We sketch the proof that this protocol works, and leave it as an exercise to complete the proof formally.

Completeness. The protocol is perfectly complete because, if a 3-colouring of G exists, then the honest prover will always be able to open the commitments to $\phi(u)$ and $\phi(v)$ in such a way that all the verifier's checks will pass.

Soundness. Suppose G is not 3-colourable. By the statistical binding property of the commitment scheme, with very high probability over the randomness of V, for each $v \in [n]$, there is exactly one value of $\phi(v)$ that P^* can open its commitment to. Suppose this indeed happens and P^* is unambiguously committed to a colouring ϕ. As G has no 3-colouring, there must be some edge $(u, v) \in E$ such that $\phi(u) = \phi(v)$. With probability at least $1/|E|$, the verifier will pick this edge to be opened by P^*, and will reject. Thus, the probability that V accepts is at most $(1 - 1/|E|) + \text{negl}(\lambda)$. This can be amplified by repetition, but there are some caveats here that we will discuss later.

Zero Knowledge. We prove zero-knowledge in a manner similar to our proof for the GI protocol, but using computational indistinguishability in place of the statistical indistinguishability there. Given any verifier V^*, we the corresponding simulator S_{V^*} works as follows on input $G = ([n], E)$:

1. Pick a random edge $(u', v') \in E$, and a random 3-colouring $\phi : [n] \to \{0, 1, 2\}$ such that $\phi(u') \neq \phi(v')$.
2. Pick a random string r_{V^*} for V^*, and run the commitment protocol with it just as P would, committing to $\phi(v)$ for each $v \in [n]$.
3. When V^* outputs an edge (u, v), if this is not the same as (u', v'), output $\perp$ and abort.
4. If $(u, v) = (u', v')$, open the commitments to $\phi(u')$ and $\phi(v')$ as P would have.
5. Run V^* to determine its output, then output the view of V^* in the course of this execution.

We first prove that the probability that the above simulator outputs $\perp$ is bounded away from 1. Then, this probability can be made negligible by repetition. Note that since the commitment protocol is computationally hiding, by the time it has to output (u, v), the verifier has no information at all about the ϕ that has been committed to (in a computational sense). This implies that

Claim 4.1. *The distribution over edges $(u, v) \in E$ output by the V^* for any two ϕ's that are committed to are almost the same.*

Proof Sketch. Suppose there is a pair ϕ_0 and ϕ_1 and an edge (u, v) such that the probabilities that (u, v) is out output when ϕ_0 is committed to the same probability when ϕ_1 is committed to differ by more than some $1/\mathsf{poly}(\lambda)$. Then, this can be detected in polynomial time and used to efficiently distinguish between the commitments to ϕ_0 and ϕ_1, contradicting computational hiding. Thus, the probabilities in the different distributions change by at most $\text{negl}(\lambda)$. □

Let us pretend, for simplicity, that the distributions over the edges are actually the same irrespective of which ϕ is committed to. This means that the distribution of the simulator's (u, v) is independent of the verifier's (u', v'). So, with probability $1/|E|$, these two are equal and the simulator does not output $\perp$.

Next, we prove that, conditioned on the simulator not outputting $\perp$, the simulated view of V^* above is computationally indistinguishable from the actual view. By the computational hiding property of the commitment, the view of V^* in the simulation until it sends (u, v) is computationally indistinguishable from the corresponding parts of its view in the protocol. Further, Claim 4.1 implies that this is true even including the message (u, v) as well. Then, note that the distribution of $(\phi(u'), \phi(v'))$ in the simulation is just a pair of different values from $\{0, 1, 2\}$, which is the same as its distribution in the original protocol (since P was choosing a 3-colouring at random). Finally, the output of V^* is just a deterministic function of all of these and its randomness. Thus, the entire view output by the simulator is computationally indistinguishable from the actual view of V^*.

Exercise 2. *Write out the proof of the zero-knowledge property of this protocol fully, paying particular attention to how exactly we argue that the joint distributions that come up in the last paragraph are still indistinguishable.*

References

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *Proceedings of the 17th*

Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA, pages 291–304. ACM, 1985.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187. IEEE Computer Society, 1986.