CSE 5854: Class 03

Benjamin Fuller

January 23, 2018

1 Knowledge Gained by a Proof

In the previous set of notes we described an interactive proof for graph non-isomorphism which is a language not known to be in BPP or NP (so a trivial interactive proof system is not known). We stated that there are two primary uses for interactivity in a proof system. The first is convincing verifiers of statements that they cannot verify themselves (with or without a witness). The second is convincing verifiers without completely revealing a witness. In this class we will try and formalize this notion. Roughly, the minimum amount that could be communicated by an interactive proof is whether $x \in L$. One might hope that the verifier V cannot convince another person of the truth of the statement, the proof is only "for them."

We'll start to formalize this notion using the security of encryption which had a similar flavor, we should learn "nothing" from an encryption.

Definition 1 (Perfect Secrecy). *[SWBH49] Let $\mathcal{M}$ be a message space. Let K be a distribution.* Enc *satisfies perfect secrecy if for any m and for any message distribution M over $\mathcal{M}$, $\Pr[M = m | \mathsf{Enc}(K, M) = c] = \Pr[M = m]$. That is, M is independent of $\mathsf{Enc}(K, M)$.*

In computational setting this definition was more complicated as we could not talk about pure independence of the message and the ciphertext. This lead us to the definition of semantic security.

Definition 2 (Semantic Security). *[GM84] Let $\mathcal{M}$ be a message space. Let K be a distribution.* Enc *is* semantically-secure *if for all PPT $\mathcal{A}$ there exists a simulator $\mathcal{A}'$ such that for any message distribution M over $\mathcal{M}$ and for any $f, h : \mathcal{M} \to \{0,1\}^*$,*

$$\Pr[\mathcal{A}(C, h(M)) = f(M))] - \Pr[\mathcal{A}'(h(M)) = f(M)]| < \epsilon.$$

In this definition we guarantee that any function of m that can be computed by an adversary seeing the ciphertext can be computed by an adversary A' with almost the same probability without seeing the ciphertext. The point of this definition was that the ciphertext did not give the adversary any knowledge about the underlying message m.

Discussion Question 1: Consider an alternative definition that says there exists a machine $\mathcal{A}'$ that on input $h(m)$ can produce messages identically distributed as c. Why does this definition imply semantic security?

We will use this second formulation as our starting point for defining knowledge gained by an interactive proof. The rough intuition is that a proof carries no knowledge if as a verifier we could have produced all the same messages that the prover sent. As a reminder the notation $< P, V >$ is used to represent the output of V when interacting with P.

Definition 3 (Zero knowledge attempt 1). *Let $\mathcal{L}$ be some language. We say that a P, V is a* zero-knowledge *proof system if it is an interactive proof system (satisfying completeness and soundness) and for all PPT A there exists a "simulator" S such that $\forall x \in \mathcal{L}$:*

$$< P, A > (x) \overset{d}{=} S(x).$$

Note the similarities between this definition and semantic security. First we define zero knowledge for all PPT A, we do not assume that the verifier that is trying to gain knowledge correctly follows security. This is because we are defining security for an honest prover. Importantly, the machine S is allowed to depend on A which means that S can have the entire code of A embedded.

While the machine V outputs only a single bit (that represented whether $x \in \mathcal{L}$), the algorithm A is allowed to output an arbitrary string. This string could depend on the messages sent by P, random coin tosses, the statement $x \in \mathcal{L}$, and some hidden values in the description of A. Because of this "arbitrary" behavior it seems like the only way to design a machine S is to try and run the algorithm A.

However, this requirement is slightly too strong so we allow the algorithm S to report a failure, that is we allow the machine to output $\perp$ some fraction of the time.

Definition 4 (Perfect Zero Knowledge). *[GMR89] Let (P, V) be an interactive proof system for some language L. We say that (P, V) is* perfect zero knowledge *if for every PPT A there exists a PPT S such that for every $x \in L$ the following two conditions hold:*

1. *With probability $\leq 1/2$ on input x, S outputs $\perp$.*

2. *Let $s^*(x)$ be a random variable describing the distribution of $S(x)$ conditioned on $S(x) \neq \perp$. Then the following holds:*

$$< P, A > (x) \overset{d}{=} s^*(x).$$

Machine S is called a perfect simulator *for the interaction of A with P.*

While this definition is relatively simple to state it provides little intuition about how to actually construct a simulator. Towards this end we'll restate the zero knowledge proof introduced in class.

Recall that two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a permutation $\pi : V_1 \to V_2$ such that $\pi(V_1) = V_2$ and $\forall u, v$, $u, v \in E_1$ if and only if $\pi(u), \pi(v) \in E_2$. We first assume that P is given as input a permutation π as it allows a polynomial time description. Call this permutation ϕ.

1. P inputs G_1, G_2 and ϕ. P picks a random π and computes a graph G' that is π applied to G_2. P sends G' to V.

2. The verifier upon receiving G' flips a bit $b \in \{1, 2\}$ and asks P to provide a permutation between G' and G_b.

3. The prover receives b from V. If $b = 2$ then P sends π^{-1}. Otherwise P sends $\phi^{-1} \circ \pi^{-1}$ to V.

4. If the message, denote ψ received from the prover is an isomorphism between G' and G_b then the verifier outputs 1 otherwise it outputs 0.

Theorem 1. *The language graph isomorphism has a perfect zero-knowledge interactive proof system. The programs specified satisfy the following:*

1. *If G_1 and G_2 are isomorphic then the verifier accepts with probability* 1.

2. *If G_1 and G_2 are not isomorphic then no matter what machine V interacts with it will reject the input with probability at least* $1/2$

3. *The prover is perfect zero knowledge.*

Proof. We first show the programs are an interactive proof system. If the graphs G_1 and G_2 are isomorphic then the G' is isomorphic to both graphs. Thus, P can always answer the challenge of V which will always output 1. If G_1 and G_2 are not isomorphic then there is no graph that is isomorphic to both G_1 and G_2 so no matter what graph G' is sent in the first step the prover will not be able to respond correct with probability at least $1/2$.

We now turn to showing that the prover is perfect zero-knowledge. We first note that the honest prover outputs 1 on messages $x \in L$ thus the simulator can just output 1. What is difficult here is that we don't know what A is going to do. We will focus on providing the machine A with properly distributed inputs so it output whatever it likes. Consider the following program for S which is able to run A:

1. Input two graphs G_1 and G_2. Pick a bit $b' \in \{1, 2\}$. Pick a permutation π from vertex $V_{b'} \to V_{3-b'}$ and compute $G' = \pi(G_{b'})$. Send G' to A.

2. If A does not respond with a bit output $\perp$. Without loss of generality we assume that A responds with $b \in \{1, 2\}$. If $b' \neq b$ abort. Otherwise, output π^{-1}.

3. Output whatever A outputs.

Here we note that the probability that the simulator aborts is exactly $1/2$ as if the two graphs are isomorphic the adversary A cannot predict the b' with probability greater than $1/2$. The crucial claim here is that the graph G' is statistically independent of the bit b'. Note that this is not true when the two graphs are not isomorphic. The "zero-knowledge" property is only required to hold when $x \in L$. Further note that S runs in polynomial time if A runs in polynomial time. It should be clear that S outputs $\perp$ with probability $\leq 1/2$. When it doesn't output $\perp$ it provides A with messages distributed exactly the same way as the honest prover. □

Discussion Question 2: It seems very weak that the simulator is allowed to fail with probability $1/2$. How can we reduce the probability with which S fails?

References

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[SWBH49] Claude E. Shannon, Warren Weaver, Richard E. Blahut, and Bruce Hajek. *The mathematical theory of communication*, volume 117. University of Illinois press Urbana, 1949.