600.641 Special Topics in Theoretical Cryptography January 23, 2007

Lecture 2: Complexity Theory Review and Interactive Proofs

Instructor: Susan Hohenberger *Scribe: Karyn Benson*

1 Introduction to Cryptography

Question 1 *What is cryptography?*
When most people think of cryptography, they think encryption. The field also includes authentication, signatures, trust relationships, elimination of trusted parties and more.

Question 2 *Are these tools useful in other fields?*
Cryptography, in the 1980's, was far from becoming an extinct field of study. Modern cryptographers apply their tools and techniques to branch into other fields. This expansion was possible through breakthrough work done in 1976-1986, shown in Figure 1.

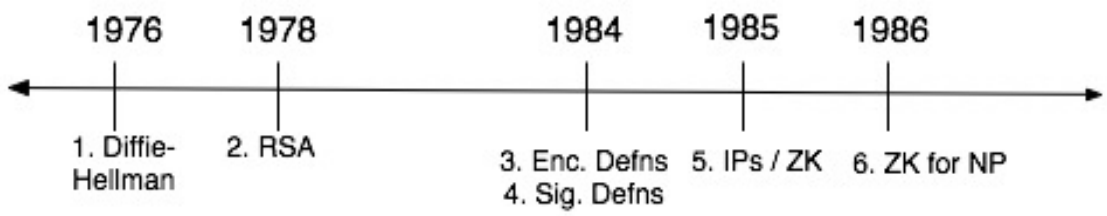

Figure 1: Some Important Results in Modern Cryptography

1. Diffie and Hellman [DH76], explained in "New Directions in Cryptography," a key exchange protocol, which can be performed over an insecure channel. Before Diffie-Hellman, people believed that cryptography required a shared secret. Now two parties who have never met can carry on a private conversation even if an adversary observes all the messages passing between them!

2. Rivest, Shamir, and Adelman (RSA) present public key encryption and signature schemes [RSA78]. These techniques are efficient and (now) commonly used.

3. Goldwasser and Micali [GM84] studied the question "What does it mean to encrypt something securely?" They presented a definition of security for probabilistic encryption schemes and a construction that meets their definition. (The basic RSA scheme is deterministic and does not meet this definition.)

4. Goldwasser, Micali, and Rivest published a paper defining security for signature schemes with a construction meeting their definition [GMR88].

5. Goldwasser, Micali, and Rackoff studied in detail interactive proofs and introduced the concept of the "zero-knowledge proof" [GMR85].

6. Goldreich, Micali, and Wigderson further refined the concept of a zero-knowledge (ZK) proof and showed that there is a ZK proof for all NP statements [GMW86b,GMW86a].

There are, of course, many other great results not included in this timeline. This class will start with the results listed above from 1985 and move forward. We will then revisit some of the topics prior to 1985.

2 Review of Complexity Hierarchy

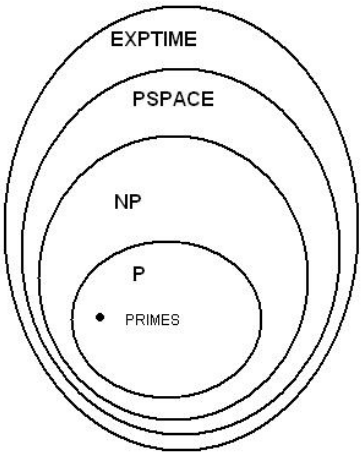

Figure 2: Complexity Hierarchy

- NP, which stands for Nondeterministic Polynomial Time, is the class of languages where membership can be verified quickly: for all $x \in L$, this fact can be verified in polynomial time.
- $|x|$ will be used to denote the size of the binary representation of x (i.e., the value x can be represented compactly in roughly $log_2(x)$ bits.
- The only separation between these classes of languages that we know is: EXPTIME $\neq$ P.

3 Co-NP

Definition 1 Co-NP - *If $L \in NP$, then $\bar{L} \in$ Co-NP, where $\bar{L}$ is the set of strings (defined over some alphabet) which are not in L.*

Example 1 *Recall that $SAT \in NP$ and that SAT is the set of strings representing all satisfiable formulas. An example of a satisfiable formula is $x_1 \vee x_2 \vee x_3$, which can be satisfied by setting $x_1 = x_2 = x_3 = 1$.*

$\overline{SAT} \in$ Co-NP, where $\overline{SAT}$ is the set of all unsatisfiable formulas. An example of an unsatisfiable formula is $x_1 \wedge \overline{x_1}$.

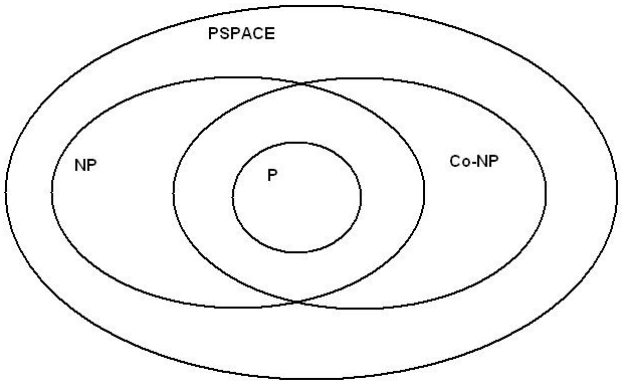

Figure 3: The Classes P, NP, Co-NP, and PSPACE

The intersection of Co-NP and NP includes P and is included in PSPACE. It is *possible* that there exists a language L which is in both NP and Co-NP and yet not in P, but this isn't known for sure.

Example 2 *PRIMES, which contains all prime numbers, is an example of a language known to be in both NP and* Co-NP*. Last lecture, we discussed that $PRIMES \in P$, $P \subseteq NP$ and $P \subseteq$* Co-NP*. Also, $\overline{PRIMES}$ is a language in P.*

4 Introduction to Interactive Proof Systems

In a proof system, there are two parties: a prover and a verifier.

4.1 Classical Proofs

Classical proofs are proven using steps like the following:

1. Start with a statement: A.
2. Make an inference: A $\rightarrow$ B.
3. Draw a conclusion: B.

In a classical proof, a prover typically writes down all that he has to say and a verifier then checks this statement. The prover and the verifier have no interaction. (This is like a written exam.)

4.2 Interactive Proofs

In an interactive proof, however, the prover and the verifier are allowed to send multiple messages back and forth to each other. This requires that the prover and verifier be online together, but perhaps there is some power in interacting. Perhaps there are theorems that a prover cannot prove to the verifier via a classical proof, but could prove via an interactive proof. (This is like an oral exam.)

Let's say more about these interactions. There are two parties involved:

1. Prover: can have infinite running time - can think of as a powerful wizard.
2. Verifier: has polynomial running time, and has the ability to generate random numbers. This is known as probabilistic polynomial time, abbreviated ppt.

The Prover and Verifier interact for a polynomial time. Eventually the Verifier accepts that $x \in L$ or rejects since it is not convinced that $x \in L$. (Note that x may be in L, but rejection means that the prover did not *convince* the verifier.)

Example 3 *Graph Isomorphism (ISO).*
Suppose an (undirected) graph is represented as a number of nodes n and a set of edges, where an edge between node i to node j is represented as (i, j). Given two graphs G and H in this representation, one question one might ask about them is: are G and H isomorphic? That is, is it possible to relabel the nodes of H so that G and H are exactly the same? In essence, we are asking if G and H are the same graph.

Formally, ISO = { (G,H) : $G \cong H$ }

$ISO \in NP$. Intuitively, there are lots of reorderings ($n! > 2^n$) but if given instructions on how to relabel the nodes, the isomorphism is easy to verify. It is key that these instructions be at most polynomial in size, and indeed, they are.

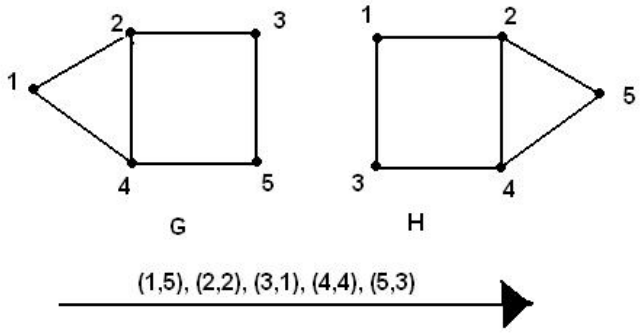

Figure 4: Example of two isomorphic graphs

We have just seen that membership in ISO has an efficient classical proof. Let's now consider a related language.

Example 4 *Graph Non-Isomorphism (NISO). The set of all pairs of graphs that are not isomorphic. NISO= {(G,H) : $G \ncong H$ }*

How would you prove, using a classical proof, that two graphs are not isomorphic? What short proof of this can you give? We don't know of an algorithm that works all the time, and the verifier can't check all the possibilities in polynomial time. So we don't know how to prove that NISO ∈ NP. However, by definition, NISO ∈ Co-NP*. So, NISO ∈ PSPACE and since PSPACE=IP we know NISO has an interactive proof (IP) [Sha90].*

Let's consider the following interactive proof for NISO. The statement being prove is $(G, H) \in$ NISO. Recall that the computational power of the prover is unbounded, but that the verifier must run in ppt.

P	$(G, H) \in$ *NISO*	***V***
		Pick F= G or H at random.
		Pick a random permutation π*.*
		$C = \pi(F)$
	$\xleftarrow{C}$	
Decide if $C \cong G$*, or* $C \cong H$		
	$\xrightarrow{\text{“}G''\ or\ \text{“}H''}$	
		Verify the answer from the prover is F.

Why does this protocol work? Intuitively, if G and H are not isomorphic, then an honest P can always correctly answer the verifier. If G and H are isomorphic, then even a malicious P is correct at most half the time.

For a better soundness probability, run this protocol k times and have the verifier reject if the prover is ever wrong. Then, the probability that any *malicious Prover cheats the verifier is* $\leq (\frac{1}{2})^k$*. When* $k = 32$*, the prover has roughly 1 in a billion chance of cheating the verifier. When* $k = 80$*, the prover has a better chance of winning the lottery ten times in a row!*

5 Defining Interactive Proof Systems (IPS)

Now that have seen an interactive proof and have some appreciation for the potential power of these proofs, let's define this idea more formally. First, let's start with some intuition on what these proof systems should do.

5.1 Informal Definition

Definition 2 *Any proof system should have the following two properities:*

1. *Completeness: "Any true theorem can be proven"*

2. *Soundness: "Any false theorem cannot be proven"*

We say that a pair of algorithms (P, V) form an Interactive Proof System (IPS) for membership in a language L if the IPS is:

1. Complete with a small probability of error.
2. Sound with a small probability of error.

5.2 Formal Definition

Definition 3 *(P,V) is an Interactive Proof System (IPS) for a language L, and security parameter k if:*

1. **Completeness***: $\forall x \in L$, $Pr[(P,V)[x] = accept] \geq 1$ - $negl(k)$.*
2. **Soundness***: $\forall x \notin L, \forall P^*$, $Pr[(P^*,V)[x] = accept] \leq negl(k)$.*

Here $negl(\cdot)$ denotes a negligible function. The security parameter is a value we pick to control the probability of error. Typically, $k << |x|$.

When we write P and V we mean the honest algorithm, but by P^* we denote a possibility malicious prover.

We could have defined our soundness such that Pr[(P^*,V)[x] = accept] = 0. But then our NISO protocol would not have satisfied the definition. Since the probability of P^* and V selecting the same graph is $\frac{1}{2}$; by repeating NISO we can reduce the probability of P^* and V had selected the same graph every time. However, this probability will never be zero since $\forall k > 0$, it holds that $(\frac{1}{2})^k > 0$.

6 Minimizing the Information Revealed by the Prover

In the proof protocols from the previous section, what could the Verifier learn? What if the Prover didn't want the Verifier to learn this extra information? How can we quantify this information?

6.1 A Zero Knowledge Proof for Graph Isomorphism

Recall: ISO = { (G,H) : G $\cong$ H }. Suppose $(G, H) \in ISO$.

Next, suppose Tom (a third party) would like to know if the two graphs G and H are isomorphic, and asks ther verifier V. Although V does not know the answer itself, it can query P to obtain a permutation π such that $\pi(G) = H$. An example of using P in an oracle like fashion is shown below:

P $\overset{(1,5)(2,2)...(5,1)}{\rightarrow}$ V $\overset{(1,5)(2,2)...(5,1)}{\rightarrow}$ Tom

The equates to V learning a lot of information from P, which may be undesirable. The

information that V learns from P could be a credit card number, mother's maiden name,.... V could use this information at a later time to steal P's identity.

Example 5 *Let's try a different protocol for ISO:*

P	*(G,H) ∈ ISO*	***V***
Pick F=G or H at random.		
Pick a random permutation π		
$C = \pi(F)$		
	$\xrightarrow{C}$	
		Choose J= G or H at random
	$\xleftarrow{\text{"}J\text{"}}$	
	Proof (C,J) are isomorphic $\xrightarrow{}$	

In the last step above, the proof that (C,J) are isomorphic can be the classic ISO proof described in the last section. Claim: This solution is both sound and complete with a small probability of error when we run it k times.

This ISP ISO proof reveals less information to the verifier than the classic ISO proof since the Verifier only knows isomorphism between random graphs and G and random graphs and H, but this does not help the verifier learn an isomorphism between G and H.

As a remark, we note that the interactive proof above need not be between an prover with infinite running time and a ppt verifier. If the prover is also ppt *and* the prover knows valid permutation between G and H, then the proof system still works. For example, suppose the prover knows a permutation ϕ such that $\phi(G) = H$. In step one of the protocol, the prover can pick a random permutation π and send to the verifier $C = \pi(G)$. If the verifier asks for J="G" in step two, then in step three, the prover returns π^{-1} since $G = \pi^{-1}\pi(G)$. Otherwise, the verifier has asked for J="H", and the prover sends $\phi \cdot \pi^{-1}$ since $H = \phi(\pi^{-1}\pi(G))$.

It *seems* like the verifier doesn't learn much beyond the fact that $(G, H) \in$ ISO by talking to the prover. Let's now try to capture this intuition in a definition.

6.2 Informal Definition of Zero Knowledge-Interactive Proof System

(P,V) is a Zero Knowledge-Interactive Proof System (ZK-IPS) if it:

1. is complete.
2. is sound.

3. has zero-knowledgeness: the verifier does not learn anything except the truth of the statement.

Question 3 *What does it mean for the verifier not learn anything?*

One idea: The verifier could have created the conversation itself. For each iteration of the proof choose a random $F = H$ or G, random π and apply $\pi^{-1}H = C$, to create the transcript. Placing them in the order C, "F", π creates a "conversation" that looks a lot like a real conversation between the prover and the (honest) verifier. But is this enough? We'll explore this idea more in the next lecture.

References

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

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

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. Earlier versions appeared in FOCS '84 and CRYPTO '84.

[GMW86a] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO '86*, volume 263 of LNCS, pages 171–185, 1986.

[GMW86b] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *FOCS '86*, pages 174–187, 1986.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sha90] Adi Shamir. IP=PSPACE. In *FOCS '90*, pages 11–15, 1990.