6.876/18.426: Advanced Cryptography March 3, 2003

Lecture 7: Variations on ZK

Scribed by: Joël Alwen

1 Recap

1.1 Current Definition of Zero Knowledge Proof Systems

We say that an interactive proof system $\mathsf{P},\mathsf{V})$ is a Zero Knowledge Proof System ($ZKPS$) for a language L iff:

1. **Completeness:** $\forall x \in L\ Pr[(\mathsf{P},\mathsf{V})[x] = \text{“YES”}] \geq 1 - neg(k)$
2. **Soundness:** $\forall x \notin L\ \forall \mathsf{P}'\ Pr[(\mathsf{P}',\mathsf{V})[x] = \text{“YES” }] \leq \frac{1}{2}$
3. **Zero Knowledge:** $\forall \mathsf{V}'_{\mathsf{PPT}}\ \exists \mathsf{S}_{\mathsf{PPT}}\ \forall x \in L\ \forall a \in \{0,1\}^{|x|^c}\ VIEW^{\mathsf{PV}'}_{\mathsf{V}'(a)} \approx S(x,a)$

For an in depth discussion of the details of this definition see the previous lectures, notably lecture 3 for a discussion on the importance of the advice string a and lecture 4 for the reasoning behind why the simulator S only needs to approximate the $VIEW^{\mathsf{PV}'}_{\mathsf{V}'(a)}$. Note though that for this lecture no knowledge of any of the subtleties is required as we will be covering new, yet related, concepts rather then deepening any understanding of the above idea itself.

1.2 Existence and Properties of $ZKPS$

In previous lectures we looked at the relation of various complexity classes to ZK. Our first such result, $NP \subset ZK$, showed us that our definition of $ZKPS$ is in fact a practical and that such proofs exists for a class of interesting problems.

In the previous lecture we greatly expanded this result showing that $IP \subset ZK$ which gives a lot of *power* to $ZKPS$ as IP is defined as the set of all languages L such that both **completeness** and **soundness** hold true. This was shown by useing the result of Goldwasser and Sipser that $AM = IP$ and $AM \subset ZK$.

The last important result from the previous lecture tells us that $ZKPS$ can also be *efficient*. This was demonstrated with an example, namely Blum's $ZKPS$ for graph 3-colorability, which gives a $\frac{1}{2}$ probability (that $x \in L$) in 3 rounds. A natural question which arises from the quest for efficiency is “What happens when we parallelize ZK proofs?”

2 Parallelization of $ZKPS$

2.1 3-Colorability and Parallelization

First we describe a parallel version of Blum's protocol $(\mathsf{P}^*,\mathsf{V}^*)$ for the 3-colorability of G:

1. P^* begins by choosing a set of random permutations π_i for $i \in \{1, ..., k\}$ and applying them to the graph G resulting in a new set of isomorphic graphs H_i. The permutation only affects the labeling of the nodes thus all H_i are still 3-colorable iff G is. Next P^* encrypts the coloring of every vertex of every H_i and sends all this information to V^*

2. V^* flips enough coins to choose a random edge in each graph H_i, and send the selection back to P^*.

3. P^* reveals the coloring for the 2 nodes in H_i on each of the edges selected by V^* in the previous step, along with a proof that these colors where truly the information encrypted in the first step. V^* then checks this proof, and if the colors of each pair of nodes revealed are not equal then V^* accepts G's 3-colorability.

Though **completeness** and **soundness** are relatively straightforward to show (as they are analogous to the serial protocol), **Zero Knowledge** is not. The proof of ZKness no longer works as it does in the serial case since rewinding becomes a problem. The problem begins with a malicious verifier V', which could, for example, use the hash of the first sequence of encrypted colorings as a reply in step two. This raises another interesting point. V' now has a so called "receipt" which it can use to prove that it has in fact talked to P^*, and can thus use the transcript of the conversation to prove $x \in L$ to any third party. However, this does not *automatically* imply that any information has been leaked, yet it does in some sense contradict the intuitive definition of a $ZKPS$. In other words we have been presented with an interesting question: Does not having the property of Zero Knowledge imply that some information *must* be leaked or is our definition of ZKness too strong?

As a result of not being able to rewind, we are not able to prove that $(\mathsf{P}^*, \mathsf{V}^*)$ has the Zero Knowledge property. Specifically we could not show

$$\exists \mathsf{S}_{\mathsf{PPT}}\ \forall \mathsf{V}'_{\mathsf{PPT}}\ \forall x \in L\ \forall a \in \{0,1\}^{|x|^c}\ VIEW^{\mathsf{PV}'}_{\mathsf{V}'(a)} \approx S^{\mathsf{V}'}(x, a)$$

which is called **Black Box Zero Knowledge**. $BBZK$ is in fact stronger then our definition of ZK as it allows for only a single S for all V's. It is not intuitive however that not being able to prove $BBZK$ implies not being able to prove the weaker ZK.

Goldreich and Krawczyk [GK96] proved that if 3-round $BBZK$ can be proven then the language L is trivial. (Here trivial means $L \in \{\mathsf{PPT}, \mathsf{PP}\}$), though this only refers to "3-round" $BBZK$ where the soundness probability is negligible rather than 1/2. Thus we can conclude that the parallel 3-colorability protocol is not ZK.

3 Exercises

This brings us the first problem set of the semester (Exercises 1). However subsequently an entire alternate problem set is given (Exercises 2). Aditionally there is one more problem which can be optionally substituted for any problem in either of the problem sets (Exercise 3). The assignment is due on Monday the 10 March, 2003.

3.1 Exercises 1

1. **Define Low Enough Knowledge**
   The problems encountered above with parallelization prompt us to look for new definitions as a work around. Intuitively LEK can be understood as being an IP where P proves $x \in L$ via a witness w_i for $i = 1ori = 2$. However V does not know which of the two witnesses was used. Give a formal definition of LEK using the appropriate notation.

2. LEK **and Parallelization**
   Prove that LEK is closed under parallel composition.

3. **Examples of** LEK
   Exemplify LEK.

3.2 Exercises 2

Under our definition zero knowledge interactive proofs are slanted. I.e. they are proofs for sure, however the ZK property only holds under a complexity assumptions. The dual of ZKPs are ZK **arguments**, which are slanted in the opposite direction. I.e. they are ZK for sure, however they are only maybe proofs.

1. **Define** ZK **Arguments**
   Give a formal definition of ZK arguments using the proper notation.

2. **Requirements of** ZK **Arguments**
   Give a detailed definition of what is needed to implement ZK Arguments? (Not just "Perfect bit commitment is required." but a detailed definition of how such a bit commitment should work, for example.)

4 ZK Argument Example

4.1 Bit Commitment

Intuitively, a commitment scheme can be understood as a protocol between two parties A and B, where, in the initial phase, A *commits* i.e, sends a value to B with the property that B does not know what the value is until the second phase has been completed. This property is often called *hiding.* In the second phase A reveals what the value was, along with a proof that this is in fact what was originally committed too. A commitment scheme must be *binding* in the sense that A can only decommit to one value. A *bit commitment* protocol, is a protocol where A commits to a single bit.

The following is an example $(A_{\mathsf{PPT}}, B_{\mathsf{PPT}})$ of a bit commitment protocol. It is based on the assumption that the discrete logarithm problem (DLP) is hard. I.e. find an x such that $g^x \equiv r (\text{mod } p)$

1. **Intialization** B begins by selecting a random prime p, and a corresponding generator g, as well as a random integer $r \in \{1, ..., p-1\}$. B sends all three values to A.

2. **Commitment** A chooses a random integer z. Then A commits to her bit b by sending $g^z * r^b \bmod p$ to B.

3. **Decommitment** Later on when A wants to decommit she sends z to B. B then checks that the value he received z' is the same as the z that was originally committed to by checking that $g^z * r^b \equiv g^{z'} * r^b \bmod p$.

It is clear that A can not decommit a commitment to both a 0 and a 1 since this would mean A has found both an x and a z such that $g^x \equiv y \equiv r * g^z \bmod p$. That would imply solving the DLP for a random integer $r \in \{1, ..., p-1\}$ since $r \equiv g^{x-z} \bmod p$.

However there is still one problem that needs fixing. The distribution of the commitment can be skewed if g is not in fact a generator. The problem arises from the fact that A is in PPT and with out knowing the factors of $\phi(p)$ there is no known efficient algorithm for determining if g is a generator for p. r is chosen at random and, because multiplication by an element of $\{1, ..., p-1\}$ mod a prime p is a permutation, $r * g^z$ is again a random number between 1 and $p-1$. However if g is not a generator then g^z will only range over a (possibly small) subset of the numbers in $\{1, ..., p-1\}$. Thus if B chooses $r \notin \{g^i | i = 1, ..., p-1\}$ then, if the commitment of A is divisible by r, it is definitely a 1. (i.e. B now has an easy distinguisher.)

To fix this B first has to generate p in a such a way that B knows the factors of $\phi(p) = p-1$. This can be done in PPT using either Bach's algorithm [B88], or by looking for what are called *safe primes* (also known as co-Sophie Germain primes). I.e. primes $p = q * 2 + 1$ where q is also prime. Experiments have shown that such primes are quite common, and have the added benefit that the complexity of solving the DLP mod such primes *increases* for all known DLP solving algorithms.

The revised algorithm works as follows:

1. **Initialization** B begins by generating a random prime p along with the factors of $p-1$, a corresponding generator g, as well as a random integer $r \in \{1, ..., p-1\}$. B sends all of these numbers to A.

2. **Commitment** A checks whether g is a generator (see Exercise 3) and if so, chooses a random integer z. Then A commits to her bit b by sending $g^z * r^b \bmod p = z'$ to B.

3. **Decommitment** Later on when A wants to decommit, she sends z to B. B then checks that z' is actually a commitment to z by checking that $g^z * r^b \equiv z' \bmod p$.

Since raising a generator to a random $r \in \{1, ..., p-1\}$ is a permutation, A can be sure that the distributions of a commitment to 0 and a commitment to 1 are truly even and the same. Note also that this scheme has perfect hiding. I.e. even a B_∞ could not find b given only the commitment since for any commitment y there exists a pair (x, z) such that $g^x \equiv y \equiv r * g^z \bmod p$.

4.2 Exercise 3

This exercise may replace any one problem in exercises 1 or 2. Prove that, given the factors of $p-1$, it can efficiently be tested whether g is a generator for a given prime p.

4.3 3-COL ZK Argument

Using the above protocol for bit commitment we can now construct a ZK argument (P, V) for 3 colorability of a graph G.

1. V begins by generating a prime p along with the factors of $p-1$, a corresponding generator g, and a random integer $r \in \{1, ..., p-1\}$. V then sends all these values to P.
2. P checks that g is in fact a generator (see Exercise 3) and, if this is the case, generates a random 3-coloring of the vertices and commits to this coloring. (i.e. if a node is red then P commits to 0 and 0 for that node, if it is white then to 0 and 1 and if it is blue then to and 1 and 0. P sends its commitments for all nodes to V.
3. V chooses a random edge e and sends this back to P.
4. P sends the decommitment for the coloring of the two nodes on e. If they are different and valid colors and they are the colors which P originally committed to then V accepts the proof.

Thus if P cheats then they have a $\frac{1}{|E|}$ chance of being caught.

The fact that this protocol is a ZK argument relies on how the bit commitment works. Namely even B_∞ can not find b given only the commitment. However an A_∞ can cheat (decommit to both a 1 and a 0).

References

[B88] E. Bach. How to Generate Factored Random Numbers. SIAM Journal of Computing, vol. 17, no. 2, pp. 179–193, 1988.

[GK96] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. SIAM Journal of Computing, vol. 25, no. 1, pp. 169–192, 1996.