6.876/18.426: Advanced Cryptography 10 March 2003

Lecture 9: A Bounded NIZK Proof System for a Special Language

Scribed by: Matthew Lepinski

1 Introduction to Non-interactive Zero-Knowledge

1.1 What is this lecture about?

So far, in our study of zero-knowledge, we have seen that we can attain zero-knowledge proof systems by exploiting two very powerful tools: (A) The interaction between the prover and the verifier and (B) The unpredictability of the verifier's coin flips. Today we will see the amazing result that it is possible to achieve zero-knowledge in a setting where: (A) The verifier does not send messages to the prover and (B) All of the randomness used by the verifier is known to the prover in advance.

1.2 The Non-interactive Zero-Knowledge Model

Like in a standard zero-knowledge proof system, we consider a situation where there are two parties, a prover and a verifier. The prover would like to prove the theorem "$x \in L$" to the verifier, where x is a common input to both parties. Additionally, the prover has an input w_x which is a witness to the fact that $x \in L$[1].

The NIZK proof consists of the prover sending a single message, $\pi(w_x, \sigma)$. Upon seeing this single message, the verifier will be convinced that the theorem is true but will gain no additional knowledge besides the truth of the theorem. What makes this miraculous goal feasible is that NIZK proofs take place in *the common random string model.* That is, it is assumed that there is a truly random string, σ, which is visible to both the prover and the verifier, but which neither the prover nor the verifier can control. The length of this string is some fixed polynomial in $|x|$ (the theorem to be proved) and k (the security parameter). (Since the length of the random string required depends on the length of the theorem being proved, we refer to this type of NIZK proof system as a *Bounded* Non-Interactive Zero-Knowledge proof system. In a future lecture we will consider the case where we want to prove very long theorems (or very many theorems) with a single short string, σ).

1.3 Examples of Common Random Strings

1. Most libraries include a table of random strings published by the Rand corporation. Since it is believed that neither the prover nor the verifier was able to influence the publication of these tables, a string from the Rand corporation could serve as a common random string.

[1] This assumes that $L \in$ NP, but we can define the same concept with respect to an unbounded prover for languages outside of NP, with the idea that the stronger prover makes up for the lack of a witness.

2. The least significant digit of the daily trading volume from the New York Stock Exchange starting on January 1, 1921. Since it seems unlikely that either the prover or the verifier had a significant impact on the trading volume of the NYSE eighty years ago, these digits could serve as a common random string.

3. Some physical phenomena like the amount of radiation given off by some star many light years away could also serve as a common random string. (Both the prover and the verifier could take readings of the radiation levels from the star, but neither party could force the star to give off more or less radiation).

Note: In all of these examples, it is important to remember that the entire string σ is known to the prover before he begins his proof. What is important is NOT that the string is unpredictable to the prover (or verifier) but that the string is truly random and outside the influence of the prover (and the verifier).

Note: An NIZK proof statement is a transformation that proves

$$\sigma \text{ is truly random"} \Rightarrow x \in L\text{"}$$

2 Definition of NIZK

Completeness:

$$\forall\, x \in L\ \forall\, w \in W_x$$

$$\Pr[\sigma \leftarrow \{0,1\}^{k^c}; PROOF \leftarrow \mathsf{P}(w,\sigma) : \mathsf{V}(\sigma, x, PROOF) = 1] > \frac{2}{3}$$

Note: In an NIZK proof system, V is deterministic. Furthermore, $NIZK \subseteq AM[2]$, the class of languages provable in an Arthur-Merlin scenario in two rounds. Finally, we know that all AM proof systems can be made to have completeness 1. Therefore, all NIZK proof systems can be made to have completeness 1.[2]

Soundness:

$$\forall\, x \notin L\ \forall\, P'$$

$$Pr[\sigma \leftarrow \{0,1\}^{k^c}; PROOF' \leftarrow \mathsf{P}'(x,\sigma) : \mathsf{V}(\sigma, x, PROOF') = 1] < \frac{1}{3}$$

Clearly, we can make the probability of soundness closer to 0 by increasing the length of σ (and composing the NIZK proofs in parallel).

One might consider a variation of this definition where $PROOF'$ is chosen before σ. (That is, we might consider inserting a $\forall PROOF'$ outside of the Probability statement). Observe that the definition given above is no weaker than the alternative definition because we can always consider a P' that ignores σ and always outputs the same $PROOF'$. Additionally, if we consider the case where the verifier always accepts if $\sigma = PROOF'$, we can see that the above definition is in fact strictly stronger than the alternative definition.

[2] It is not obvious to me that the AM transformation preserves the zero-knowledge property of an NIZK proof system.

(Observe that the case where the verifier always accepts when $\sigma = PROOF'$ is allowed by the alternative definition but not by the above definition.)

Zero-Knowledgeness

$$\exists\ S_{\mathsf{PPT}}\ \forall\ x \in L\ \forall a,\ VIEW_{\mathsf{V}}[x,a] = S(x,a)$$

3 NIZK for a special language

After defining non-interactive zero-knowledge proof systems, the big question is: "Do NIZK proof systems exist?" We will spend the rest of this lecture providing an affirmative answer to this question.

3.1 The Example Language

Let $2PP$ be the language consisting of integers which are divisible by exactly two distinct primes. (That is, all numbers n such that $n = p^\alpha q^\beta$, where p and q are prime).

Note that $x^2 \mod p^\alpha$ has exactly two square roots. $x^2 \mod p^\alpha q^\beta$ has exactly four square roots. $x^2 \mod p^\alpha q^\beta r^\gamma$ has exactly eight square roots. Further, note that $1PP \in BPP$ (Actually, it was recently shown to be in P). If $n \in 2PP$, then Z_n^* consists of $1/4$ squares. If $n \in 3PP$, then Z_n^* consists of $1/8$ squares.

3.2 How to Prove that $n \in 2PP$

Without loss of generality, assume that n has k bits. First we parse sigma into

$$\sigma = y_1|y_2|y_3|y_4|y_5|\ldots|y_{k^c}$$

Where each y_i is a $k - bit$ string. Next we go through and throw out all of the y_i such that $y_i > n$. (Note: This will throw out at most half the y_i's).

We are then left with a set of random elements from Z_n^*. (Actually, there is a negligible probability that one of the y_i's has the property that $gcd(y_i, n) \neq 1$. In this case, the verifier can easily factor n and the proof is trivial. Therefore, we consider only the case, where all the y_i's are either greater than n or in Z_n^*). We denote the elements which remain after throwing out the $y_i > n$ as:

$$\sigma = x_1|x_2|x_3|x_4|x_5|\ldots$$

Where each $x_i \in Z_n^*$.

Next, for each x_i which is a square mod n, the prover provides the verifier with a square root of x_i. If the prover is able to do this for approximately $1/4$ of the x_i's then the verifier is convinced that $n \in 1PP$ or $n \in 2PP$. This is because if n has three distinct prime factors, it is amazingly unlikely that σ will happen to consist of $1/4$ squares. (Since this is twice as many squares as we would expect). The verifier can easily check for himself that $n \notin 1PP$ and thus is convinced that $n \in 2PP$.

3.3 A Simulator for the 2PP Proof System

On input n, the simulator must output a random looking string σ and a proof using σ that $n \in 2PP$. Observe that since the simulator does not know the factorization of n, the simulator cannot take square roots mod n. Our first thought is to have the simulator construct the string σ by flipping random coins to get R_i and then setting σ to be:

$$R_1^2(= x_1)|R_2^2(= x_2)|R_3^2(= x_3)|R_4^2(= x_4)\ldots\ ...$$

Unfortunately, this won't work because every x_i will have Jacobi symbol 1. (Where as if σ is truly random then half the x_i should have Jacobi symbol -1).

We can fix this problem by having the simulator do the following:

- For each i flip a random coin C_i.
- If C_i is 0, choose a random T_i with Jacobi Symbol -1 and let $x_i = T_i$.
- If C_i is 1, choose a random R_i and let $x_i = R_i^2$.
- For each i with $C_i = 1$, flip another coin and if this coin comes up 1, put R_i in the proof.

This will create a σ where about half the x_i's have Jacobi symbol 1, and the proof will consist of a square root of half the Jacobi symbol 1 x_i's.

3.4 *Perfect* NIZK

There is one problem with the simulator from the previous section, the string σ created by the simulator is not truly random. This is because, in a truly random string half the x_i's with Jacobi symbol 1 are non-squares whereas the simulator creates a σ where every x_i with Jacobi symbol 1 is a square. Now this isn't a major problem because assuming that the Quadratic Residuosity problem is hard, the σ produced by the simulator is computationally indistinguishable from a truly random string. However, we would ideally like to achieve a *Perfect* NIZK proof system for 2PP instead of merely a *Computational* NIZK proof system for 2PP.

If there is a well known $y \in Z_n^*$ which is a non-square with Jacobi symbol 1, then the simulator can just multiply R_i^2 by y whenever it chooses not to put R_i in the proof and get a σ with the right distribution. That is, replace the simulator's code with the following:

- For each i flip a random coin C_i.
- If C_i is 0, choose a random T_i with Jacobi Symbol -1 and let $x_i = T_i$.
- If C_i is 1, choose a random R_i and let $x_i = R_i^2$.
- For each i with $C_i = 1$, flip another coin and if this coin comes up 1, put R_i in the proof. If this second coin comes up 0, then set $x_i = yR_i$.

Another thought is to make y part of the language. (That is, let L be the language of pairs (n, y) where $n \in 2PP$ and $y \in Z_n^*$ where y is a non-square with Jacobi Symbol 1). Unfortunately, if we do this we get soundness problems because if y is a square mod n, then the prover can prove false theorems. Therefore, in order to use this idea we need to make a small change to guarantee soundness.

3.5 The *Perfect* NIZK Solution

Let L be the language of pairs (n, y) where $n \in 2PP$ and $y \in Z_n^*$ where y is a non-square with Jacobi Symbol 1. As before, we cull all the blocks in σ which are not in Z_n^*. We are left with

$$\sigma = x_1|x_2|x_3|x_4|x_5|\dots$$

Then for each x_i with Jacobi Symbol 1, the prover provides the verifier with either $\sqrt{x_i}$ (if x_i is a square) or $\sqrt{x_i/y}$ (if x_i is not a square).

This proof system is sound. If y is not a square mod n, then when x_i is not a square mod n, the prover will not be able to provide either $\sqrt{x_i}$ or $\sqrt{x_i/y}$.

We only need to make a small modification to our simulator to prove this new proof system is zero-knowledge.

- For each i flip a random coin C_i.
- If C_i is 0, choose a random T_i with Jacobi Symbol -1 and let $x_i = T_i$.
- If C_i is 1, choose a random R_i and let $x_i = R_i^2$.
- For each i with $C_i = 1$, flip another coin and if this coin comes up 0, let $x = yR_i^2$.
- The proof is simply the list of all the R_i's

3.6 An Alternative Definition of Soundness

Note that the above proof system for L actually delivers more than we claimed in our definition. That is, the above proof system satisfies an even stronger notion of soundness.

Alternative Soundness:

$\forall P'$

$$Pr[\sigma \leftarrow \{0,1\}^{k^c}; (PROOF', x') \leftarrow \mathsf{P}'(\sigma) : (x' \notin L) \wedge \mathsf{V}(\sigma, x', PROOF') = 1] < \frac{1}{3}$$

In the standard definition of soundness, the theorem is selected before the random string is chosen. However, in the alternative definition soundness, the cheating prover is allowed to pick the false theorem after he has seen the string σ. This means that the proof system remains sound, even if the prover is proving a theorem about σ. (For example, The prover wants to use page 114 of the Rand Corporation Volume 1 to prove the theorem, "The first 20 digits on page 114 of the Rand Corporation Volume 1 form a number which is the product of two prime powers").