6.876/18.426: Advanced Cryptography March 17, 2003

Lecture 11: Generalizing Non-interactive Zero Knowledge Proofs

Scribed by: Scott Russell

1 Summary of NIZK So Far

Last class we developed a NIZK protocol to prove membership in 3SAT. We made use of our previous NIZK protocol for the "special" language $\mathcal{L} = \{(n, y) : n \in 2PP \wedge y \in \mathcal{NQR} \cap J_{+1}\}$ where $J_{+1} = \{y : (\frac{y}{n}) = 1\}$, i.e. the set of all remainders mod n with Jacobi symbol $+1$). We observed that the resulting protocol in fact satisfied a slightly more general definition of NIZK than we had started with. Specifically we were able to weaken the assumptions on the soundness condition without sacrificing zero-knowledgeness. Recall our current definition of a NIZK proof system. We say that (P, V) is a NIZKPS for language L if is satisfies the following 3 conditions:

1. **Completeness:** $\forall x \in L$, $\forall w \in W_x$, $Pr[\sigma \leftarrow \{0,1\}^{|x|^c}; \pi \leftarrow \mathsf{P}(x, \sigma, w) : \mathsf{V}(x, \sigma, \pi) =$ "YES" $) > 1 - negl(|\sigma|)$
2. **(Strong) Soundness:** $\forall \mathsf{P}'$, $Pr[\sigma \leftarrow \{0,1\}^{|n|^c}; (x', \pi') \leftarrow \mathsf{P}'(\sigma) : \mathsf{V}(x', \sigma, \pi') =$ "YES"$) < negl(|\sigma|)$
3. **ZKness:** $\exists \mathsf{S}_{PPT}$, $\forall x \in L, \forall a$, $\mathsf{S}(x, a) = VIEW(x, a)$.

What are the strengths and weaknesses of our definition and the protocol we gave for membership in 3SAT last class?

- Strengths
  1. Allowing a malicious P' to choose x', π' after seeing σ in the soundness requirement is a more general formulation.
- Weaknesses
  1. To construct a ZK simulator, it seems theorem x must be given before the random string σ
  2. Can only prove 1 theorem with a single σ.
  3. Need many random bits to prove a single theorem[1] $|\sigma| \gg |x|$

Currently in order to prove ZKness we need to have σ follow x so that the simulator S can come up with an appropriate value. This may not seem like a problem right now, but we may later encounter situations where this works to our disadvantage. Thus it would be nice if the proof of theorem x was independent of σ.

[1] As explained in [1], for a size n clause ϕ, a total of $8n^3 + 2n^4$ random bits are required.

The number of theorems we can prove with a given σ is 1 (possibly $O(1)$) since each theorem we want to prove imposes restrictions on σ (from S's perspective) and it will quickly become impossible to find a σ capable of simultaneously satisfying all of these constraints.

TODAY'S GOAL: To continue to generalize our definition of NIZK and to improve our 3SAT protocol with respect to the not so nice features identified above.

2 Review of NIZK for 3SAT (Version 1)

TOOLS:

- 1 generic lie can prove 1 specific "truth"
- Randomization of the goal

The basic idea was that (P, V) has as input the theorem $\phi \in$ 3SAT and a shared random string (think of it as coming from the sky) whose length is polynomial in the security parameter $k = |\phi|$. This random string is divided into two parts call them σ and τ. First the prover chooses a random k-bit integer n, the product of two primes, and a random element y of Z_n^* such that y is a non-square mod n and has Jacobi symbol $+1$. P uses σ (in a purified form) to construct a NIZK proof that $(n, y) \in \pounds$ (using the protocol for $\pounds$ we discussed in lecture 9) with reference string σ. Then it uses (n, y) to prove that $\phi \in$ 3SAT. It does this by generating an encoding $\vec{e}$ of the witness $\vec{w}_\phi$ (a satisfying assignment) for ϕ and sending this to the verifier. Each component of $\vec{e}$ is random element of Z_n^* where $\Psi(e_i) = x_i$ for all variable assignments x_i in $\vec{w}_x$. Recall that last class we defined the function $\Psi : J_{+1} \to \{0, 1\}$. For $a \in J_{+1} \subset Z_n^*$, $\Psi(a) = 0$ if a is a square (mod n), i.e. $a \in \mathcal{QR}$, and $\Psi(a) = 1$ if a is not a square, i.e. $a \in \mathcal{NQR}$.

It now remains for P to prove to V that the given encoding $\vec{e}$ is in fact a satisfying assignment for ϕ. P does this by subdividing τ into m parts, $\tau_1, \tau_2, \ldots, \tau_m$, one for each clause of ϕ. Each τ_h is then further subdivided into k-bit cells and purified by first removing any cells whose value is greater than n, and then removing any triple of cells containing an element in J_{-1}. The resulting purified τ_h is then used to prove clause C_h of ϕ with the help of the equivalence relation $\approx_n$ that we defined last class. Recall that for $(a_1, a_2, a_3), (b_1, b_2, b_3) \in J_{+1} \times J_{+1} \times J_{+1}$ we write $(a_1, a_2, a_3) \approx_n (b_1, b_2, b_3) \Leftrightarrow \exists \sqrt{a_1 b_1}, \sqrt{a_2 b_2}, \sqrt{a_3 b_3} \bmod n.$, i.e. if and only if a_i and b_i have the same quadratic character for all $i = 1, 2, 3$. For each clause C_h the prover reveals all of the triples in τ_h of the type (sq, sq, sq), corresponding to $(0, 0, 0)$, by giving a random square root for each square component[2]. For a purified τ_h with α triples, there should be close to $\alpha/8$ such triples. Finally, P proves that the encoded assignment of the literals in C_h, say (e_i, ye_j, e_k) for $(x_i, \overline{x_j}, x_k)$ (using multiplication by y mod n for negation), satisfies C_h. This involves proving the equivalence of (e_i, ye_j, e_k) to an additional $\alpha/8$ triples $(a_1, a_2, a_3) \not\approx_n (sq, sq, sq)$, again by revealing the square roots of $e_i a_1, ye_j a_2, e_k a_3$.

In order to prove ZKness, the simulator is able to generate a valid session transcript by lying about the (n, y) pair it chooses, generating a false proof for that pair, and then

[2]Remember to chose a random n, both S and P can chose two random primes and use their product for n. Since they have n's factorization taking square roots is no trouble.

using that pair to prove that the encoding $\tilde{\mathbf{e}}$ corresponding to $(1,1,1,\ldots,1)$ is satisfying. Specifically, S chooses (n,y') with y' a square and transforms a random σ into a $\tilde{\sigma}$ in order to generate false proof π' of $(n,y') \in \mathcal{L}$ (see lecture 10 for details). The simulator's ability to use y' a square (in place of y a non-square) and $\tilde{\sigma}$ (in place of σ) depends on the Quadratic Residuosity Assumption. Then using τ, the second (unmodified) part of the randomly chosen string, S proceeds as P would have to prove the equivalence of the encoding of every clause to a satisfying assignment. Because every element of the encoding $\tilde{\mathbf{e}}$ is non-square, and multiplication by square y' won't change this, S always just proves the equivalence of the C_h's encoding to $(1,1,1)$. Of course V doesn't know which equivalence class was used in the proof only that it wasn't $(0,0,0)$.

Notice that $(n,y) \in \mathcal{L}$ and ϕ are uncorrelated since both the prover and simulator randomly choose (n,y) so there is no relationship between σ and ϕ. Is there any relationship between τ and ϕ? Well, no special properties of τ were used to prove ϕ. We relied entirely on the random choice of (n,y) and the "truth" of its membership in $\mathcal{L}$ substantiated with σ. So, we claim the random string and theorem are completely unrelated which means that getting $\sigma \circ \tau$ first and then ϕ is also all right. If the theorem ϕ to be proven is unknown, we can get the random string of bits first and then use them to prove the theorem ϕ when it is eventually given. Thus we now have greater generality in our ZK property, but can still only prove a single theorem since $|\sigma \circ \tau| \gg |\phi|$.

An alternative way to prove ZKness is to have S use σ unchanged (but still purified), chose a random $(n,y) \in \mathcal{L}$, again assign an encoding $\tilde{\mathbf{e}}$ off all non-squares, and transform τ to $\tilde{\tau}$ as necessary to again prove equivalence to the class corresponding to $(1,1,1)$. Of course V still cannot tell which class the encoding is in, only that it isn't (sq,sq,sq). Unfortunately, restricting τ in this way also prevents us from proving more that a single theorem with a single random string. The reason is that once the simulator modifies a portion of τ to prove some clause C_h of ϕ_1, it is very unlikely that the same modification to τ would allow S to also prove a clause C'_h of ϕ_2.

Q: Can't we just take a random σ, use it a as seed for a pseudorandom generator, and use the output as our τ as a way to save on random bits?

A: Possibly if we restrict P to be PPT. We have to be careful since the seed is also publicly known. The security of the PRF depends on the secrecy of the seed, but we are revealing the seed. The question we have to ask ourselves is does V learn anything about the distribution of τ given the seed σ that he doesn't already know?

3 NIZK for 3SAT Version 2

We will now lift the one theorem per random string restriction by showing how to prove an arbitrary *number* of theorems $\phi \in$3SAT. However, the *size* of each individual theorem is still restricted and will need to be "shorter" than in version 1 for the same length random string. As in version 1, completeness will be 1 (or asymptotically close), soundness will strong and negligible in the security parameter, and ZKness will be indistinguishable. Here let k be the length of the shared random string, the security parameter, and let each theorem ϕ have m (or fewer) clauses.

TOOL: Self authenticating history (1 generic lie allows you to prove many truths)

The protocol again divides the random string into parts, this time 3, which we label σ, τ_1, and τ_2. As in version 1, we use σ to prove $(n_0, y_0) \in \mathcal{L}$. Then we use (n_0, y_0) and τ_1 to prove an auxiliary pair $(n_1, y_1) \in \mathcal{L}$ by proving $\phi_{aux} \in$ 3SAT using version 1. ϕ_{aux} is the corresponding 3SAT instance obtained by a Cook reduction of (n_1, y_1). Depending on the specific reduction used, $|\phi_{aux}|$ will be some fixed polynomial in $|n_1|$. Lastly, using (n_1, y_1) and τ_2, P proves the corresponding theorem $\phi_1 \in$ 3SAT, again via version 1.

Notice that our proof of ϕ_1 has gotten longer. We now have to give a proof π_0 for (n_0, y_0), π_1 for (n_1, y_1), and π_2 for ϕ_1. Have we really gained anything? Absolutely, because by using (n_1, y_1) and *re-using* τ_1 to prove a new pair (n_2, y_2) another theorem ϕ_2 can be proved using (n_2, y_2) and *re-using* τ_2. This process can be repeated an arbitrary number of time.

A slightly more detailed description of the protocol (P, V) with input $(\phi_1, \phi_2, \ldots, \phi_t)$ and random string $\sigma \circ \tau_1 \circ \tau_2$ follows. The prover with the additional witness inputs $(\vec{w}_{\phi_1}, \ldots, \vec{w}_{\phi_m})$ does the following:

1. Chooses random m-bit n_0 and $y_0 \in \mathcal{NQR}_{n_0} \cap J_{+1}$
2. Uses σ to prove $(n_0, y_0) \in \mathcal{L}$ via version 1
3. Chooses random m-bit n_1 and $y_1 \in \mathcal{NQR}_{n_1} \cap J_{+1}$
4. Transforms (n_1, y_1) into its corresponding $\phi_{aux} \in$ 3SAT (via a polynomial time Cook reduction)
5. Proves the $poly(m)$ clause $\phi_{aux} \in$ 3SAT using (n_0, y_0) and τ_1 via version 1, thereby proving $(n_1, y_1) \in \mathcal{L}$
6. Proves $\phi_1 \in$ 3SAT via Version 1 using (n_1, y_1) and τ_2
7. Repeats steps 3-6 for each theorem ϕ_i with (n_{i-1}, y_{i-1}) reusing τ_1 to prove a new pair (n_i, y_i), and (n_i, y_i) reusing τ_2 to ϕ_i

Since each ϕ_i has m clauses, each of the n_i's must also be m bits long. Because the number of clauses in ϕ_{aux} is some fixed (dependent on the specific reduction) polynomial in m and version 1 is used to prove ϕ_{aux} and ϕ_i using different parts of the random string, we have to settle for "short" theorems. They are short with respect to the length of the random string and ϕ in version 1.

Are the completeness, soundness, and ZK requirements satisfied by this protocol? Completeness follows directly from the completeness of version 1. Soundness with respect to the proof of each $(n_i, y_i) \in \mathcal{L}$ also follows from version 1. Additionally, soundness when $(n_j, y_j) \in \mathcal{L}$ for all $0 \leq j \leq i$, but $\phi_i \notin$3SAT also holds since version 1 is used and $(n_j, y_j) \in \mathcal{L}$. So how about ZKness? Is it safe to reuse τ_1 and τ_2 in this way? Even though the proof of every theorem ϕ_i uses τ_2, for each the encoding of its witness is with respect to a *different* pair, namely (n_i, y_i). Similarly, even though each pair (n_i, y_i) is used twice, once to prove corresponding theorem ϕ_i and also to prove the next pair, each of these proofs uses

a different part of τ, so it certainly *seems* like nothing about the encoding or correspondence can leak. More formally, as in version 1, S can choose a fake initial pair (n_0, y_0'), generate a false proof π' by transforming σ into σ', and then use that to prove "anything it wants". Specifically it can generate valid looking proofs of pairs (n_i, y_i') where y_i are squares modulo n_i and in turn use these to prove theoremsϕ_i.

REMARK: The full proof of ϕ_i for $0 \leq j \leq t$ now consists of ϕ_i's proof via (n_i, y_i) AND all of the proofs for $(n_0, y_0), \ldots, (n_i, y_i)$ via their predecessors in the chain. In other words the full size is linear the theorem number i, or worst case in the total number of theorems t. Although the proof size has grown, before we could only prove one theorem with one random string. Now because each (n_i, y_i) does double duty, we can prove a polynomial number of theorems. The drawback is that since we are still leveraging our version 1 protocol, we still need significantly more random bits than the length of the theorems we are proving. Still, dealing with one problem at a time we are making progress.

3.1 Version 2a: Tree Variation

Rather than having ϕ_i's proof depend on a chain of prior proofs, leading to a total proof size linear in i, we can instead use a binary tree make the total proof size logarithmic in the number of theorems. Let (n, y) be the initial pair proved using σ. Using (n, y) and τ_1, instead of authenticating a single auxiliary pair, we prove the compound statement $(n_0, y_0) \in \mathcal{L} \wedge (n_1, y_1) \in \mathcal{L}$, again by reducing this statement to the corresponding problem $\phi_{aux} \in$ 3SAT. Proceeding recursively, we use (n_0, y_0) to prove $(n_{00}, y_{00}), (n_{01}, y_{01}) \in \mathcal{L}$ and (n_1, y_1) to prove $(n_{10}, y_{10}), (n_{11}, y_{11}) \in \mathcal{L}$ in both cases reusing τ_1. Each node (n_ρ, y_ρ) is used to prove a pair of children $(n_{\rho 0}, y_{\rho 0})$ and $(n_{\rho 1}, y_{\rho 1})$ and to prove a theorem ϕ_ρ with τ_2 using version 1.

If, the size of each proof is say $O(k^3)$ bits, then the size of the total proof for any ϕ_i is $O(k^3 \log i)$ bits or worst case $O(k^4)$ bits.

Q: Doesn't this mean that we have to remember everything we have proved previously?

A: Not really. Suppose we decide in advance we want to be able to be able to provide say 2^k proofs. We only have to remember proofs on the path from the root to the leaf where the proof of the current ϕ_i is located, i.e. the binary representation of i. Alternatively, with seed or key s and a pseudorandom function f_s, for each theorem we could calculate a k-bit path string $R = f_s(\phi_\rho)$ or $R = f_s(i)$. Provided we used the same random bits each time we generate (n_ρ, y_ρ) for node ρ, we could regenerate pairs and reconstruct their corresponding proofs for the appropriate nodes along the path R as necessary. At the very least the prover has to remember the seed and the number of the current theorem. So it seems just as easy to use the theorem number in binary to specify the path down the tree and remember the proofs along the path as long as they are needed and discarding them after that point.

So we have seen how using either a chain or a tree together with the self-authenticating history idea, we were able to remove the restriction on the number of theorems we could prove. Great!

4 NIZK for 3SAT Version 3

Finally we want to lift the last restriction. Namely, we would like it to take only around the same number of random bits as the theorem's length to prove it. This is where we will see why choosing 3SAT as our representative problem is so important.

TOOL: Recursion

We again divide the random string into 3 pieces, σ, τ_1, τ_2. As in version 2a, a random initial pair (n, y) is proved in $\mathcal{L}$ using σ and then using τ_1, the compound statement $(n_0, y_0), (n_1, y_1) \in \mathcal{L}$ is proven. Successive levels of the tree are generated recursively as before. What is new is the way in which we prove $\phi \in$3SAT. First we use (n_0, y_0) to produce an encoding $\vec{e}$ of the witness $\vec{w}_\phi$. Again it remains for P to prove clause by clause that the given encoding corresponds to a satisfying assignment.

Consider clause $C_h = (x_i, \overline{x_j}, x_k)$ with encoding $(e_i, y_0 e_j, e_k)$ where $e_i, e_j, e_k \in Z^*_{n_0}$. P must prove the statement "at least one of $\{e_i, y_0 e_j, e_k\} \in \mathcal{NQR}_{n_0}$". But since this statement is itself a member of an NP language, it can be reduced to 3SAT formula θ_{aux}. Now since e_i, y_0, e_j, e_k, n_0 are all some κ bits each, the resulting θ_{aux} will have $poly(\kappa)$ clauses (where the exact polynomial is fixed with respect to the reduction used). Notice that this does not depend on m the number of clauses in ϕ, but only on κ the length of the n_i used. Version 1 is used to prove $\phi_{aux} \in \mathcal{L}$, but the number of random bits needed depends on κ allowing us to prove ϕ with an arbitrary number of clauses m.

The beauty of 3SAT and the reason it's the "right" problem to choose is this locality. Previously each (n_{rho}, y_ρ) pair proved the entire theorem ϕ_ρ meaning that the number of random bits needed depended heavily on the length of $\sigma \circ \tau_1 \circ \tau_2$. Now, each pair (other than the one used to encode ϕ's witness) shoulders only the burden of proving a single clause's correctness. Consequently the number of random bits needed becomes for the most part independent of the theorem size.

5 Closing Remarks

We have successfully addressed the drawbacks we mentioned at the beginning of the class. However there remain other drawbacks and areas of improvement to NIZK proofs. These include:

1. Replacing the Quadratic Residuosity Assumption with weaker/simpler cryptographic assumptions
2. Allowing the given random string σ to be used *simultaneously* by many provers without some intercommunication
3. Preserving the property of ZK interactive proofs that the transcript of a NIZK proof should not by itself constitute a proof.

The first two we will discuss next class and are due to Shamir, Feige, and Lapidot [3]. The third item is addressed by Dwork, Naor, and Sahai in [2].

References

[1] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero Knowledge. SIAM J. Comput. Vol. 20, No. 6, pp. 1084-1118, 1991.

[2] C. Dwork, M. Naor, A. Sahai. Concurrent Zero-Knowledge. Proceedings of STOC 1998: 409 – 418, 1998.

[3] U. Feige, D. Lapidot, A. Shamir. Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract). Proceedings of FOCS 1990: 308 – 317, 1990.