CSE 594 : Modern Cryptography 4/13/2017

Lecture 20: Non interactive Zero Knowledge

Instructor: Omkant Pandey Scribe: Venkata Kedarnath Pakala, Sayan Bandyopadhyay

1 Setting

In previous lecture , we discussed about interactive zero knowledge proof. But what if Prover (Alice) is restricted to send only a single message to verifier (Bob). Proof becomes 'Non interactive'. But 1-message zero-knowledge proofs is only possible for languages in bounded probabilistic polynomial (**BPP**) because a simulator that can simulate the single message can use this as witness for x. This is useless as we want to prove statements for languages in **NP**

Fortunately, Alice and Bob have access to common random string generated honestly by someone they both trust. This common random string can be used for proving statements non-interactively.

2 Definition 1 (NIZK)

A non interactive proof system for language L with witness relation R is a tuple of algorithms (K,P,V) such that:

- Setup : $\sigma \leftarrow \mathbf{K}(1^n)$ outputs a common random string
- Prove : $\pi \leftarrow \mathbf{P}(\sigma, x, w)$ takes as input a common random string σ and a statement x $\in \mathbf{L}$ and a witness w and outputs a proof π
- Verify : $\mathbf{V}(\sigma, x, \pi)$ outputs 1 if it accepts the proof and 0 otherwise.

A non interactive proof system must satisfy the properties of completeness and soundness as shown below :

Completeness : $\forall x \in L, \forall w \in R(x)$:

$Pr\left[\sigma \leftarrow K(1^n); \pi \leftarrow P(\sigma, x, w) : V(\sigma, x, \pi) = 1\right] = 1$

Non-Adaptive Soundness : There exists a negligible function *v(.)* s.t. $\forall x \notin L$

$Pr\left[\sigma \leftarrow K(1^n); \exists \pi s.t. V(\sigma, x, \pi) = 1\right] \leq$ *v(n)*

Adaptive Soundness : There exists a negligible function *v(.)* s.t.

$Pr\left[\sigma \leftarrow K(1^n); \exists (x, \pi) s.t. \forall x \notin L \wedge V(\sigma, x, \pi) = 1\right] \leq$ *v(n)*

The reader should note that in non adaptive soundness adversary chooses x before seeing the common random string while in adaptive soundness adversary can choose x based on common random string, implies Adaptive soundness is stronger notion of soundness.

Similar to soundness we have two types of non interactive zero knowledge (NIZK). We can transform a non-adaptive NIZK to one with adaptive soundness in a way similar to hardness amplification.

3 Definition 2(Non adaptive NIZK)

A non interactive proof system (K,P,V) for a language L with witness relation R is non adaptive zero-knowledge if there exists a PPT simulator $\boldsymbol{S}$ *s.t. for every* $x \in L$, $w \in R(x)$*, the output distribution of the following two experiments are computationally indistinguishable* :

$$\begin{array}{c|c} \underline{REAL(1^n, x, w)} & \underline{IDEAL(1^n, x)} \\ \sigma \leftarrow K(1^n) & (\sigma, \pi) \leftarrow \boldsymbol{S}(1^n, x) \\ \pi \leftarrow P(\sigma, x, w) & \\ Output(\sigma, \pi) & Output(\sigma, \pi) \end{array}$$

Here the simulator is allowed to generate both the common random string and the simulated proof for a given input statement x. If simulator $\boldsymbol{S}$ is not allowed to generate σ, the definition would have been trivial as verifier could have convinced himself by running the simulator instead of interacting with P. Allowing $\boldsymbol{S}$ still keeps the definition zero knowledge as verifier sees both σ and π but P and $\boldsymbol{S}$ are treated unequally.

4 Definition 3(Adaptive NIZK)

A non interactive proof system (**K,P,V**) for a language **L** with a witness relation **R** is adaptive zero knowledge if there exists a PPT simulator **S**=(S_0, S_1) s.t. for every x $\in$ **L**, w $\in$ R(x), the output distribution of the following two experiments are computationally distinguishable :

$$\begin{array}{c|c} \underline{REAL(1^n, x, w)} & \underline{IDEAL(1^n, x)} \\ \sigma \leftarrow K(1^n) & (\sigma, \tau) \leftarrow S_0(1^n) \\ \pi \leftarrow P(\sigma, x, w) & \pi \leftarrow S_1(\sigma, \tau, x) \\ Output(\sigma, \pi) & Output(\sigma, \pi) \end{array}$$

Here τ is the "trapdoor" for simulated common random string σ that is used by simulator S_1 to generate an accepting proof for x without knowing the witness.

Here τ should be considered as the local state stored by the simulator.

Remarks on NIZK :

1. In NIZK, the simulator gets seemingly "extra power" in choosing common random string along with trap door to enable simulation without a witness.
2. While in interactive ZK, the simulator's extra power was the ability to reset the verifier.
3. It turns out that, simulator must always have extra power over the normal prover else it would be impossible to realize the definition in the languages other than BPP.
4. In NIZK, the extra power is justified as we require the indistinguishability of the joint distribution over σ and π

Now, let us show that adaptive soundness is much harder to achieve by constructing it from NIZK with non adaptive soundness with a procedure similar to hardness amplification.

Lemma : Given a NIZK(K,P,V) with non-adaptive soundness, we can construct NIZK(K,P,V) with adaptive soundness.

Proof: Let us consider a σ "bad" for x_0 if (for $x_0 \notin L$) then $\exists$ a false proof π for x_0 using random string σ s.t. $V(\sigma, x, \pi) = 1$ Let $\ell(n)$ be the length of the statements Now, if we repeat the non-adaptive NIZK polynomially many times each time choosing fresh random string σ, the probability of σ being "bad" for x_0 decreases to $2^{-2\ell(n)}$. By using union bound we can determine the probability of σ being "bad" for all statements ($x \in L$) as follows :
$Pr[\exists(x, \pi) s.t. V(\sigma, x, \pi) = 1]$
$= Pr[\sigma \text{ bad for some x }]$
$\leq 2^{\ell(n)} * Pr[\sigma \text{ bad for } x_0]$
$= 2^{\ell(n)} * 2^{-2\ell(n)}$
$=2^{-\ell(n)}$
So, this repeated scheme becomes adaptively sound.

5 NIZK for NP

NIZK for NP is constructed first from non-adaptive zero-knowledge property and then convert non-adaptive NIZK to adaptive NIZK

Steps to construct NIZK for NP from non-adaptive zero-knowledge property are :

1. Construct a NIZK proof system for **NP** in the **hidden bit model**. this step is unconditional.

2. Using trapdoor permutation, transform any NIZK proof system for language in hidden bit model to a non-adaptive NIZK proof system in the common random string model.

Next transform non-adaptive NIZK to adaptive NIZK for **NP** using one-way functions which are implied by trap door permutations.

Putting all the steps together, we get adaptive NIZKs for **NP** using trapdoor permutations.

6 NIZK in Hidden-Bit Model

6.1 Syntax

A non-interactive proof system for a language L with witness relation R in the hidden-bit model is a tuple of algorithms

- Setup : $\sigma \leftarrow \boldsymbol{K_{HB}(1^n)}$ outputs the hidden random string
- Prove : $(I, \pi) \leftarrow \boldsymbol{P_{HB}(\sigma, x, w)}$ generates the indices $I \subseteq [|r|]$ of r to reveal, along with a proof π
- Verify : $\boldsymbol{V_{HB}(I, \{r_i\}_{i \in I}, \pi)}$ outputs 1 if it accepts the proof and 0 otherwise.

The above proof must satisfy completeness and soundness like above

6.2 Definition

A non-interactive proof system (K_{HB}, P_{HB}, V_{HB}) for a language L with witness relation R in the hidden-bit model is (non-adaptive) zero-knowledge if there exists a PPT simulator S_{HB} s.t. for every $x \in L, w \in R(x)$, the output distributions of the following two experiments are computationally indistinguishable:

$REAL(1^n, x, w)$	$IDEAL(1^n, x)$
$\sigma \leftarrow K_{HB}(1^n)$	$(I, \{r_i\}_{i \in I}, \pi) \leftarrow S_{HB}(1^n, x)$
$(I, \pi) \leftarrow P_{HB}(r, x, w)$	
$Output(I, \{r_i\}_{i \in I}, \pi)$	$Output(I, \{r_i\}_{i \in I}, \pi)$

7 Conversion from NIZK in HB to NIZK in CRS

7.1 Intuition

How to transform a public random string into a hidden random string?
Suppose the prover samples a trapdoor permutation (f, f^{-1}) with hardcore predicate h. Given a common random string $\sigma = \sigma_1, \sigma_2, ..., \sigma_n$ the prover can compute $r = r_1, ..., r_n$ where:

$$r_i = h(f^{-1}(\sigma_i))$$

.
If f is a permutation and h is a hard-core predicate, then r is guaranteed to be random. Now r can be treated as the hidden random string: V can only see the parts of it that the prover wishes to reveal

7.2 Construction

Let $F = \{f, f^{-1}\}$ be a family of 2^n trapdoor permutations with hardcore predicate h. Let (K_{HB}, P_{HB}, V_{HB}) be a NIZK proof system for L in the hidden-bit model with soundness error 2^{2n}

Construction of (K,P,V):

$K(1^n)$: Output a random string $\sigma = \sigma 1, ..., \sigma n$ s.t. $\forall i, |\sigma_i| = n$

$P(\sigma, x, w)$: Execute the following steps:

- Sample $(f, f^{\text{-}1}) \leftarrow F(1^n)$
- Compute $\alpha_i = f^1(\sigma_i)$ for $i \in [n]$
- Compute $r_i = h(\alpha_i)$ for $i \in [n]$
- Compute $(I, \phi) \leftarrow PHB(r, x, w)$
- Output $\pi = (f, I, \{\alpha_i\}_{i \in I}, \Phi)$

$V(, x,)$: Parse $\pi = (f, I, \{\alpha_i\}_{i \in I}, \phi, \Phi)$ and:

- Check $f \in F$ and $f(\alpha_i) = \sigma_i$ for every $i \in I$
- Compute $r_i = h(\alpha_i)$ for $i \in I$
- Output $VHB(I, r_{ii \in I}, x, \Phi)$

Notes:

- Completeness $\rightarrow$ α is uniformly distributed since f^1 is a permutation and σ is random. Further, since h is a hard-core predicate, r is also uniformly distributed. Completeness follows from the completeness of (K_{HB}, P_{HB}, V_{HB})
- Soundness $\rightarrow$: For any $f = f_0$, r is uniformly random, so from (non-adaptive) soundness of (K_{HB}, P_{HB}, V_{HB}), we have:

  $$\Pr_{\sigma}[\text{Pcan cheat using f0}] \leq 2^{-2n}$$

  Since there are only 2^n possible choices of f (verifier checks that $f \in F$), by union bound, it follows:

  $$\Pr_{\sigma}[\text{Pcan cheat}] \leq 2^{-2n}$$

7.3 Proof of zero knowledge: Simulator

Let SHB be the simulator for (K_{HB}, P_{HB}, V_{HB})

procedure Simulator S(1n, x)
  $(I, \{r_i\}_{i \in I}, \Phi) \leftarrow S_{HB}(1^n, x)$
  $(f, f^{-1}) \leftarrow F$
  $\alpha_i \leftarrow h^{-1}(r_i),\ \forall i \in I$
  $\sigma_i = f(\alpha_i),\ \forall i \in I$
  $\sigma_i \xleftarrow{\$} \{0,1\}^n,\ \forall i \notin I$
  Return $(\sigma, f, I, \{\alpha_i\}_{i \in I}, \Phi)$
end procedure

Note: $h^1(r_i)$ denotes sampling from the pre-image of r_i, which can be done efficiently by simply trying random α_i 's until $h(\alpha_i) = r_i$

7.4 Proof of zero knowledge: Hybrid

procedure $H_0(1^n, x, w) := REAL(1^n, x, w)$
  $\sigma \leftarrow K(1^n)$ where $\sigma = \sigma_1, ..., \sigma_n$
  $(f, f^{-1}) \leftarrow F$
  $\alpha_i \leftarrow f^{-1}(\sigma_i),\ \forall i \in [n]$
  $r_i = h(\alpha_i),\ \forall i \in [n]$
  $(I, \Phi) \leftarrow P_{HB}(r, x, w)$
  Return $(\sigma, f, I, \{\alpha_i\}_{i \in I}, \Phi)$
end procedure

procedure $H_1(1^n, x, w)$
  $\alpha_i \xleftarrow{\$} \{0,1\}^n,\ \forall i \in [n]$
  $(f, f^{-1}) \leftarrow F$
  $\sigma_i \leftarrow f(\alpha_i),\ \ \forall i \in [n]$
  $r_i = h(\alpha_i),\ \forall i \in [n]$
  $(I, \Phi) \leftarrow P_{HB}(r, x, w)$
  Return $(\sigma, f, I, \{\alpha_i\}_{i \in I}, \Phi)$
end procedure

$H_0 \approx H_1$: In H_1, we sample α_i at random and then compute σ_i (instead of sampling σ_i and then computing α_i as in H_0). This induces an identical distribution since f is a permutation. So the order of the 2 operations can be reversed.

procedure $H_2(1^n, x, w)$
$r_i \xleftarrow{\$} \{0,1\}^n, \ \forall i \in [n]$
$(f, f^{-1}) \leftarrow F$
$\alpha_i \leftarrow h^{-1}(r_i), \quad \forall i \in [n]$
$r_i = f(\alpha_i), \ \forall i \in [n]$
$(I, \Phi) \leftarrow P_{HB}(r, x, w)$
Return $(\sigma, f, I, \{\alpha_i\}_{i \in I}, \Phi)$
end procedure

$H_1 \approx H_2$: In H_2, we again change the sampling order: first sample $r = r_1, ..., r_n$ at random and then sample α_i from the pre-image of r_i (as described earlier). This distribution is identical to H_1

procedure $H_3(1^n, x, w)$
$r_i \xleftarrow{\$} \{0,1\}, \ \forall i \in [n]$
$(f, f^{-1}) \leftarrow F$
$\alpha_i \leftarrow h^{-1}(r_i), \quad \forall i \in [n]$
$(I, \Phi) \leftarrow P_{HB}(r, x, w)$
$\sigma_i = f(\alpha_i), \quad \forall i \in I$
$\sigma_i \xleftarrow{\$} \{0,1\}^n, \quad \forall i \notin I$
Return $(\sigma, f, I, \{\alpha_i\}_{i \in I}, \Phi)$
end procedure

Here in H_3 we are taking one extra computation step.
$H_2 \approx_c H_3$: In H_3, we output random σ_i for $i \in I$. From security of hard-core predicate h, it follows that:

$$\{f(h^1(r_i)\} \approx_c Un$$

Indistinguishability of H_2 and H_3 follows using the above equation

procedure $H_4(1^n, x)$
$(I, \{r_i\}_{i \in I}, \Phi) \leftarrow S_{HB}(1^n, x)$
$(f, f^{-1}) \leftarrow F$
$\alpha_i \leftarrow h^{-1}(r_i), \quad \forall i \in I$
$\sigma_i = f(\alpha_i), \quad \forall i \in I$
$\sigma_i \xleftarrow{\$} \{0,1\}^n, \quad \forall i \notin I$
Return $(\sigma, f, I, \{\alpha_i\}_{i \in I}, \Phi)$
end procedure

$H_3 \approx_c H_4$: In H_4, we swap P_{HB} with S_{HB}. Indistinguishability follows from the zero-knowledge property of (K_{HB}, P_{HB}, V_{HB})