CSA E0 312: Secure Computation October 14, 2015

Guest Lecture 2-3

Guest Instructor: C. Pandu Rangan *Submitted by: Cressida Hamlet*

1 Introduction

Till now we have seen only semi-honest parties. From now onwards, we will give more power to the adversary in the sense that, it can corrupt the parties as well as make the parties to violate from the agreed protocol. Zero Knowledge Proofs are one of the important tool we have in this setting. As the name suggests, in ZKP, no new knowledge is allowed to gain. Before going into the security definition of ZKP, let us look into the security definition of encryption systems which we have been seeing for a long time to get an insight about the requirements of a security definition.

1.1 Security of Encryption Systems

We can consider encryption as a two party protocol, one will encrypt the data and the other will decrypt it. The encrypter's role is over when he/she encrypts and sends the encrypted data. The decrypter may not be online when this happen. The decrypter can decrypt this whenever he/she need, doesn't depend whether or not the encrypter is online. If we look into public key encryption, decrypter doesn't even need to know who the encrypter is. What is the security goal required in this situation - An eavesdropper should not be able to get the message. To achieve this what constraints are we required? Say we are encrypting m to get a ciphertext c. One obvious requirement is: from c one should not get m or the function we use for encryption should not be invertible.

Now we need to look if this is the only requirement we need so that an eavesdropper don't get m. Real world attacker may have more information than c and public key (pk) - **Auxiliary Information**. We need to define these auxiliary information in the security definition, that is where the concept of oracle access comes. Consider the case where eavesdropper can't get m from c and AI (auxiliary information), but it can check if m' is hidden in c or not. In this case, if the message space is small (eg. in the case of auction) one can check with all the messages and find the correct message. Here comes the concept of indistinguishability - even if eavesdropper knows that the message is either m_1 or m_2, it shouldn't be able to guess what the message is (not negligibly more than $\frac{1}{2}$). One may select highly distinguishable m_1 and m_2, but the internal random choices blind the messages such that the encryptions (the random variables corresponding to the encrypted messages) are indistinguishable. The indistinguishability definition gives the correct notion of security.

2 Zero Knowledge Proof

ZKP help us to prove we know some information, without revealing the information. One may think that why do we need to prove it by not leaking anything about the information,

i.e, why not just give the information and thus prove the possession of the information.

2.1 Need for ZKP

Consider the use of identity card (like passport). Identity cards are used in places where some authority needs to verify the identity. One can check the photo or id number in the card and confirm the person. Nowadays possession of information also serves the purpose of uniquely identifying a person/organization (eg. passwords, security pins). In this case if we disclose the information (to prove our identity), an eavesdropper who got the information can use it to impersonate. We may can consider the case where one encrypts the information and the verifier decrypts and verifies the identity, but the verifier may also be corrupted and the problem of impersonation still persists. Here we need to consider a malicious verifier, which we don't need to consider in the case of encryption. These possibilities were nearly not there in the case of physical identification cases (duplication of id cards are there to consider there). Now we should be able to convince that we hold an information, but should not reveal the information.

2.2 ZKP properties

We have a prover and a verifier and need a protocol which help the verifier to verify the prover's possession of some information without learning anything about the information. The protocol should also be complete and sound.
Complete: Every time a honest prover who executes the protocol should get accepted.
Sound: Every dishonest prover who executes the protocol should be rejected.
Similar to the case of encryption there is a public information and a private information, and verifier should not learn the private information after the execution of protocol. To understand what can be a public and private information, consider the following case:

$$y = \sqrt{x} \bmod N$$

here, y can be a private information while x, N public information.

We now need to think how to model these conditions properly. If we need to say a protocol doesn't give any knowledge about the private information to the verifier, we need to formalize the concept of "knowledge". Let's look at the case where in the protocol, verifier sends a to the prover and the prover sends back $2a$. Can we say the execution of protocol gives a new knowledge to the verifier? No, because verifier could calculate $2a$ by itself in *polynomial time*. So this interaction didn't give verifier any knowledge. Any information which we can obtain in polynomial time using the publicly known information are considered to be possessed. If the information obtained through (polynomial time) interaction can be obtained in polynomial time without the interaction, then the interaction is said to give **"zero knowledge"**.

To formalize the security of ZKP, we need the simulator paradigm. Without interaction, a polynomial time algorithm (simulator) with only publicly known information as input should be able to give a computationally indistinguishable distribution as that of a verifier's output. This can model that the verifier doesn't gain any knowledge.

Transcript is whatever that is communicated in the protocol execution. Transcript is available to both the verifier as well as any eavesdropper. Eavesdropper doesn't have

access to the randomness used by the verifier, but the verifier has access to the randomness also. *Honest but curious verifier* will try to find extra information, but follow the protocol. *Dishonest verifier* doesn't follow the protocol, but tries to send syntactically correct values so that the prover won't find out the dishonesty. The security definition where simulator tries create a transcript which is indistinguishable from the protocol's transcript is called *transcript oriented view* and the security definition where simulator tries to create an output which is similar to (may be corrupted) verifier's output is called *output oriented view*. It has been proved that both views are essentially the same, but transcript oriented view is more commonly used.

2.2.1 Formal Definition of Proofs

Consider a relation and the corresponding language defined as follows:

$$R_L \subseteq \{0,1\}^* \times \{0,1\}^*$$

$$L = \{x | (x,y) \in R_L\}$$

The language L to be in NP, we need to have a polynomial time recognition algorithm for L and so $|y|$ should be upper bounded by polynomial function of $|x|$. y is called the *witness/proof* since y certifies the membership of x in L. So to prove x is in the language one need to find some proof corresponding to x. One protocol which we usually use for proving is *Sigma Protocol*. It is named as sigma protocol because the protocol looks like Σ. At first there will be a commit from the prover to verifier, then verifier sends a challenge and when prover receives challenge, prover will send the response. Verifier should output the result according to the tuple (commit, challenge, response).

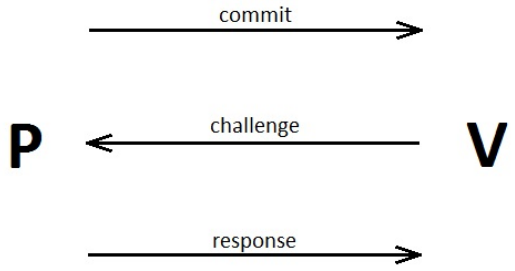

Figure 1: Sigma Protocol

We consider verifier as a polynomial time algorithm but doesn't put any constraints for a prover. We will denote $\mathsf{Output}_{\mathsf{P},\mathsf{V}}(x)$ for the output of the verifier and $\mathsf{View}_{\mathsf{P},\mathsf{V}}(x)$ as everything that has transmitted. So if there are k rounds, then

$$\mathsf{View}_{\mathsf{P},\mathsf{V}}(x) = \{(\mathrm{com}_1, \mathrm{ch}_1, \mathrm{res}_1), ..., (\mathrm{com}_k, \mathrm{ch}_k, \mathrm{res}_k)\}$$

Since we use randomness there can be a small probability of error. So we will define completeness and soundness as follows:

Completeness: If x is valid ($x \in L$, in the context of ZKP, prover is honest) then,

$$\mathsf{Pr}\left[\mathsf{Output}_{\mathsf{P},\mathsf{V}}(x) = a\right] \geq 1 - \varepsilon$$

where a denote accepting case and ε is a negligible function.
Soundness: If x is not valid ($x \notin L$, in the context of ZKP, prover is dishonest) then,

$$\mathsf{Pr}\left[\mathsf{Output}_{\mathsf{P},\mathsf{V}}(x) = a\right] \leq \delta$$

where δ is a negligible function.
This is common for all proofs. Along with completeness and soundness, for ZKP we need to formalize zero knowledge also. As we said earlier there should exists a simulator which is able to give the same view with only access to public information.
Zero-Knowledge:

$$\exists \mathsf{S}(\forall x \in L),\ \mathsf{View}_{\mathsf{P},\mathsf{V}}(x) \equiv \mathsf{S}(x)$$

This definition is fine, if the verifier is honest but curious (i.e. verifier will not deviate from the protocol), then view will be according to the protocol, but what if verifier is dishonest (i.e. verifier may deviate from the protocol), then simulator won't know what kind of view is to be generated. So, according to each verifier we need to have different distributions for view.

$$\forall \mathsf{V}, \exists \mathsf{S} \text{ such that } \forall x \in L,\ \mathsf{View}_{\mathsf{P},\mathsf{V}}(x) \equiv \mathsf{S}(x)$$

which means for any verifier V, there is a corresponding simulator which can produce the same view with only public information. This definition captures all we need, but it is difficult to prove security with this definition. So we will modify the definition such that there is only one simulator, but it will have oracle access (blackbox access) to verifier algorithm.

$$\exists \mathsf{S}, \forall \mathsf{V}, \forall x, \mathsf{View}_{\mathsf{P},\mathsf{V}}(x) \equiv \mathsf{S}^{(\mathsf{V})}(x)$$

There are 3 variants of zero knowledge according to how we define $\equiv$.
Statistical ZK: If the statistical distance between simulator's output and view is less than some small value.

$$\mathsf{d}(\mathsf{S}(x), \mathsf{View}_{\mathsf{P},\mathsf{V}}(x)) \leq \varepsilon$$

Perfect ZK: If the distributions $\mathsf{S}(x)$ and $\mathsf{View}_{\mathsf{P},\mathsf{V}}(x)$ are identical.
Computational ZK: If no polynomial time algorithm can distinguish between the $\mathsf{S}(x)$ and $\mathsf{View}_{\mathsf{P},\mathsf{V}}(x)$ distributions

2.3 ZKP examples

Consider a cyclic group G and a generator g. Let $x = \mathsf{DLog}_g(h)$. Prover needs to prove that it knows x. Here x is the private information while everything else (G, g, h) is public information.

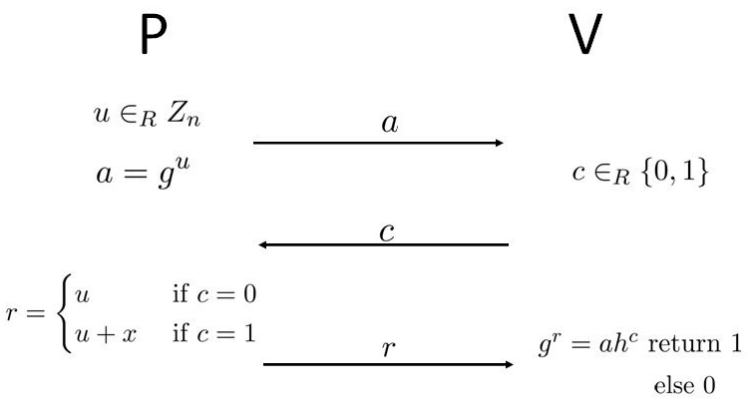

Figure 2: Protocol

2.3.1 Protocol-1:

In the figure 2 we can see a sample protocol for the zero knowledge proof for the above case. We can see that intuitively the protocol works fine, i.e. a honest prover can always win, a dishonest prover will win with a very small probability (if we repeat this k times, then the probability of winning will be $\frac{1}{2^k}$) and a dishonest verifier won't get any information (it gets either u or $u + x$ and since u is random, won't get any information about x).

$$(a, c, r) \in G \times \{0, 1\} \times Z_n \quad \text{such that } g^r = ah^c$$

$$\mathsf{View}_{\mathsf{P},\mathsf{V}}(x) = \{(a_i, c_i, r_i) \quad \text{such that } 1 \leq i \leq k, g^{r_i} = a_i h_i^{c_i}\}$$

Security: To prove the security of this zero knowledge protocol, we need to create a simulator (whose input is G, g, h) which can create an output indistinguishable from the $\mathsf{View}_{\mathsf{P},\mathsf{V}}(x)$.
Consider the triples generated as:

$$(c \in_R \{0, 1\}, r \in_R Z_n, a = g^r h^{-c})$$

Does this give the same distribution as that of the view generated?

$$\mathsf{Pr}\left[(a, c, r) \text{ occur in transcript}\right] = \frac{1}{n}.\frac{1}{2} = \frac{1}{2n}$$

Since $a = g^u$ and $u \in_R Z_n$, probability of taking a particular a is $\frac{1}{n}$. Once a and c is fixed, r (such that $g^r = ah^c$) is a constant.

$$\mathsf{Pr}\left[(a, c, r) \text{ generated as above }\right] = \frac{1}{2}.\frac{1}{n} = \frac{1}{2n}$$

Simialr to the above explanation, r is randomly choosen from Z_n, which means the probability of a particular r is $1/n$ and if r and c is fixed then $a = g^r h^{-c}$ is a constant.

Which give us the conclusion

$$\mathsf{Pr}\left[(a,c,r) \text{ occur in transcript}\right] = \mathsf{Pr}\left[(a,c,r) \text{ generated as above}\right]$$

and thus both the distributions are identical. The above simulator doesn't consider the behavior of a dishonest verifier. In the case of a dishonest verifier, c may not be uniformly random and so these two distributions won't be identical. So we will use the blackbox access of verifier V to create our simulator.

Algorithm

Step 1: $c \in_R \{0,1\}$
Step 2: $r \in Z_n$
Step 3: $a = g^r h^{-c}$
Step 4: $c^* = V(a)$
Step 5: if $c = c^*$ return (a,c,r)
Step 6: if $c \neq c^*$ goto Step 2

Since $c \in_R \{0,1\}$, the probability of $c = c^*$ is $\frac{1}{2}$ and so the expected number of rounds for the algorithm to terminate is 2. Which means the expected running time of the simulator algorithm is polynomial time.

As we mentioned above, to get an error probability $\leq 1/2^k$, we need to run the protocol k rounds. Now we will see another protocol, where we will get an error probability $\leq 1/2^k$ in just one round.

2.3.2 Protocol-2:

The protocol is same as in the previous protocol 2 except the following:

- $c \in_R Z_n$
- $r = u + cx$

Now $(a,c,r) \in G \times Z_n \times Z_n$, and (a,c,r) can be generated as

$$(c \in_R Z_n, r \in_R Z_n, a = g^r h^{-c})$$

$$\mathsf{Pr}\left[(a,c,r) \text{ generated as above }\right] = \frac{1}{n}.\frac{1}{n} = \frac{1}{n^2}$$

For the honest verifier,

$$\mathsf{Pr}\left[(a,c,r) \text{ occur in transcript }\right] = \frac{1}{n}.\frac{1}{n} = \frac{1}{n^2}$$

which makes

$$\mathsf{Pr}\left[(a,c,r) \text{ occur in transcript}\right] = \mathsf{Pr}\left[(a,c,r) \text{ generated as above}\right]$$

Now for a dishonest verifier, who may send c which is not uniformly random from Z_n we will use the same algorithm except in step-1 we will need to take $c \in_R Z_n$ instead of $c \in_R \{0,1\}$. In this case, the probability of $c = c^*$ is $1/n$, and so the expected number of iterations for the algorithm to terminate is n. Now we will get error probability less than $1/n$ with just one round, but with the cost of expected time.