University of Illinois, Urbana Champaign
CS/ECE 498AC3/4 Applied Cryptography

Instructor: Dakshita Khurana
Scribe: Mohammad Behnia
Date: September 29, 2020

LECTURE
20

Oblivious Transfer

In this lecture we will introduce the concept of secure computation. In a transition out of our discussions of zero knowledge, we will explore methods of performing computation on encrypted data in a secure manner. So far, we have explored ways to facilitate secure communication (various encryption systems) followed by methods to show proof of encryption without revealing any secrets (using zero-knowledge to enable such proofs). Now moving onto secure computation, the high-level goals will be to reveal some partial results to the involved parties while hiding sensitive information we do not want specific parties to recover. The first part of this lecture involves setting up a scenario where Alice and Bob want to engage in an oblivious transfer. We will explore this model using an ideal trusted party as a mediator and a real-world view without the mediator. We will look at different types of adversaries that will challenge any protocols we construct. Finally, we will look at a secure construction and begin to formulate our security proofs.

20.1 Oblivious Transfer

Oblivious transfer is a cryptography primitive under which a sender transfers one of many choices of information but remains oblivious to which one was chosen. Let's take a first look at an example of oblivious transfer. Consider the scenario with a sender Alice, and receiver Bob depicted in Figure 20.1. Alice has two messages m_0 and m_1, and Bob has some binary choice denoted by bit b. The goal is for Bob to receive exactly one of m_0 and m_1 depending on bit b. The chosen message will be denoted as m_b. Under the ideal construction, we will have a trusted authority that will mediate this exchange. Alice will send m_0 and m_1 to the authority and in return receive an acknowledgement that the exchange has completed. Bob will send bit b to the authority and receive a message m_b. In this scheme, both Alice and Bob only receive partial information from the trusted authority (Alice receives the acknowledgement signal and Bob receives only m_b). Beyond the multiplexer functionality, we care about the security of the oblivious transfer. In particular, we would like Alice to have no idea about the value of bit b (which we should be able to formally prove using the two-game adversary model), and Bob to have no idea about the contents of the message that was not selected: m_{1-b}. Under the trusted authority model both of these goals are satisfied, but the question remains if we can guarantee these security goals without the trusted authority.

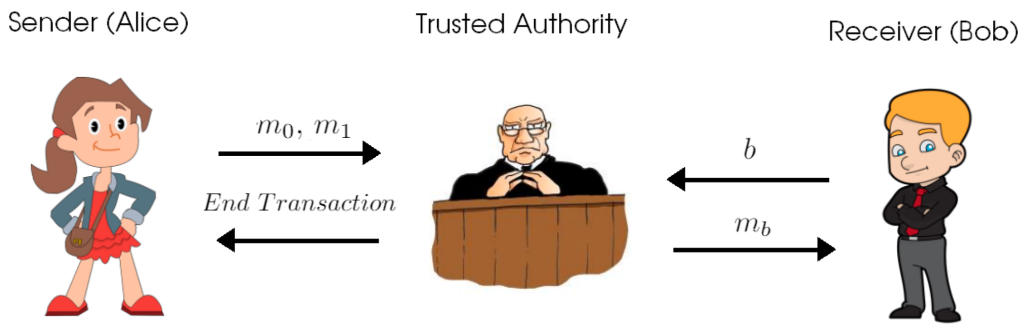

FIGURE 20.1: An ideal model of oblivious transfer between Alice and Bob.

20.2 An Honest Protocol

This will be our first attempt at creating an oblivious transfer protocol for Alice and Bob assuming they are honest parties. We will explore the definition of "honesty" and alternative adversary models after completing this construction. The goal of this construction is to create a groundwork for a functional protocol.

The high-level idea is to have Bob sample two public keys pk_0 and pk_1. Alice will then encrypt each message using its corresponding public key and return ciphertexts to Bob. Bob, however, should only be able to decrypt one of the encrypted messages. Using the El-Gamal key exchange scheme we will define the construction as follows:

1. For choice of bit b, Bob will sample $pk_b = g^{\alpha_b}$. Set $pk_{1-b} = g'$, a random group element for which we do not know the discrete log. By hardness of discrete logarithm and Diffie-Hellman problem, Bob should not be able to have access to corresponding private key.

2. Bob sends Alice the two public keys: pk_b and pk_{1-b}.

3. From Alice's view, she receives $pk_0 = g^{\alpha_0}$ and $pk_1 = g^{\alpha_1}$. Alice will encrypt each message with a different public key. Using g^{β_0} and g^{β_1}, compute $H(g^{\alpha_0\beta_0}) \oplus m_0$ and $H(g^{\alpha_1\beta_1}) \oplus m_1$.

4. Alice sends Bob g^{β_0}, $H(g^{\alpha_0\beta_0}) \oplus m_0$ and g^{β_1}, $H(g^{\alpha_1\beta_1}) \oplus m_1$.

5. Bob will be able to decrypt m_b using the key-pair for pk_b he sampled. Bob will not be able to recover the message that was encrypted using pk_{1-b}.

If Bob follows the protocol for generating the two public keys and Alice is not able to distinguish the two public keys, this will be a sound protocol. However if Bob is not honest and samples both key-pairs, the adversarial Bob will be able to recover both messages.

20.3 Adversarial Model

We will now define two types of adversaries.

- Semi-honest or Honest-but-curious adversary (follows protocol specifications)
- Malicious adversary (does not necessarily follow protocol specifications)

The gold standard for achieving security for oblivious transfer is against the malicious adversary (also sometimes referred to as byzantine adversary in some contexts).

Before we dive into a secure construction against a malicious adversary, we will define what it means to be secure in such a context. Consider two games, one being the ideal game with a trusted authority, and the second being a real game with Bob and Alice directly interacting. Intuitively we want for Bob to only be able to recover m_b and Alice to only know m_0 and m_1 in the transcript of communication in the real game. In other words, we would like the **view** of Bob and Alice to be secure in the real game and indistinguishable from their **view** in the ideal game. To define **view**, we will consider Bob and Alice to be interactive Turing machines (each with an initial state, some inputs, a series of state updates, etc).
Let's take a closer look at adverserial Bob's **view**:

$\textbf{View}_{Bob} = \textbf{State}_{Bob} + \textbf{Transcript}$ (sum total of all messages exchanged in the protocol)

The goal is for the **view** of Bob (and Alice) to not break guidelines for secure oblivious transfer. However, the view of Bob in the ideal game and real game are fundamentally different (for example, Bob cannot directly send bit b to Alice in the real game as it can send it to the directed authority in the ideal game). To address this, in the ideal model, we will place a simulator in between Bob and the ideal authority. We will aim to maintain indistinguishable views from Bob's transcript with the simulator and Bob's transcript with Alice in the real game. The role of the simulator is depicted for a malicious Bob in Figure 20.2 and for a malicious Alice in Figure 20.3.

Some observations about this simulator:

- Simulator needs to pretend to be Alice
- Simulator needs to "find" implicit bit b, since Bob is sending encrypted messages to hide the choice of this bit
- Simulator must generate the view without knowledge of m_{1-b}, because it must create a transcript that looks like Alice's while only receiving m_b

Using this simulator model, if there is a real protocol where such a simulator exists, Alice can rest assured anything Bob can recover in the real view, Bob can recover in the ideal view. In other words, since the ideal view is secure, the real view will not contain any information about m_{1-b} by virtue of indistinguishability. To enable this, the simulator will have certain capabilities such as being able to program a random oracle.

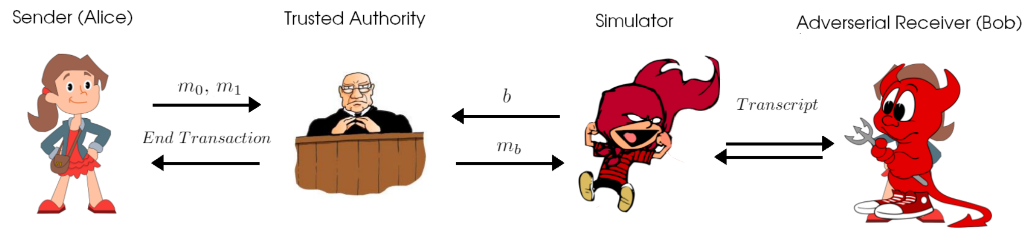

FIGURE 20.2: An adversarial receiver interacting with the simulator.

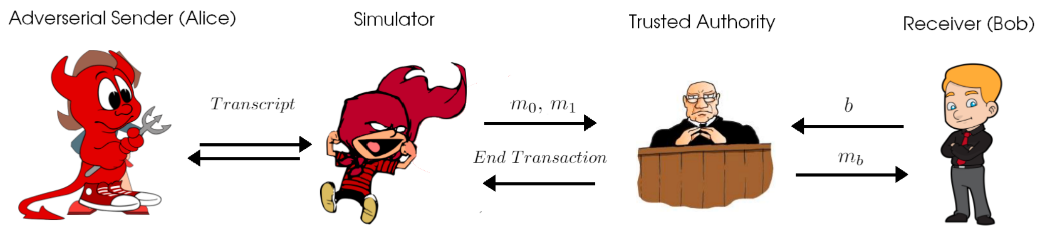

FIGURE 20.3: An adversarial sender interacting with the simulator.

Note, we can also create this simulator model for the context of a malicious sender (Alice). In this case, the simulator will be between Alice and the trusted authority, with a transcript with Alice and sending m_0 and m_1 and receiving acknowledgement from the trusted authority. One observation from this construction is that the simulator must be able to find m_0 and m_1 independent of b, since simulator has no knowledge of bit b. It follows, that Alice's behavior in the real game must also be independent of the Bob's choice of bit b.

20.4 Oblivious Transfer Protocol

This construction is similar to a work [1] in which El-Gamal encryption is employed. Further security analysis under adversarial Alice/Bob are presented in the paper.

Consider Bob has some input bit b.

1. Sample (pk, sk) key-pair using encryption algorithm of choice.
2. Sample $r_0 \in \{0,1\}^n$ uniformly at random, and compute $H(r_0)$. (where $H(x)$ refers to the hash output of a random oracle)
3. Set $r_1 = H(r_0) \oplus p_k$. (Note: $p_k = r_1 \oplus H(r_0)$)
4. Set $p'_k = r_0 \oplus H(r_1)$. (Note: Bob does not know s'_k)
5. If $b == 0$, set $s_0 = r_0$ and $s_1 = r_1$, else if $b == 1$, set $s_0 = r_1$ and $s_1 = r_0$.

Alice receives s_0 and s_1 from Bob.

1. Compute $pk_0 = H(s_0) \oplus s_1$
2. Compute $pk_1 = H(s_1) \oplus s_0$
3. Ciphertext $ct_0 = Enc_{pk_0}(m_0)$
4. Ciphertext $ct_1 = Enc_{pk_1}(m_1)$

Alice sends ct_0 and ct_1 to Bob.

Bob knows exactly one $(pk,\ sk)$ pair, corresponding to m_b. For $b == 0$, Bob can recover m_0, and for $b == 1$ Bob can recover m_1.

We will address some underlying assumptions and initial questions about this protocol.

- Why is Alice not able to distinguish a regularly sampled key-pair as opposed to a seemingly random n-bit string?

  Assumption 20.1. A public key encryption scheme with pseudorandom public keys (or mapping thereof) is selected.

- Can Bob send pk_0 and pk_1 instead of s_0 and s_1? Alice needs to be able to confirm these public keys are generated using this XOR scheme otherwise a malicious Bob can send two public keys for which he knows both private keys.
- Why does r_0 need to be a random sample? To ensure indistinguishability of s_0 and s_1 from Alice's perspective. There would be issues if Bob had some deterministic choice of r_0.

20.5 Motivating the Proof

We will begin to look at what the simulator must do and look at some constructions that we will revisit next lecture to finalize our proof of view-indistinguishability.

Consider security against a malicious Alice:

The simulator sends $\widetilde{s_0}$ and $\widetilde{s_1}$ to Alice and receives $\widetilde{ct_0}$ and $\widetilde{ct_1}$. The simulator should be able to decrypt both ciphertexts and send m_0 and m_1 to the trusted authority. To enable this, the simulator will have control of the random oracle (programmability of oracle hash output values). This can be achieved if the simulator samples $\widetilde{s_0}$ and $\widetilde{s_1}$ such that they will know the $\widetilde{sk_0}$ and $\widetilde{sk_1}$ corresponding to $\widetilde{pk_0}$ and $\widetilde{pk_1}$.
One approach is for the simulator to randomly sample $\widetilde{s_0}$ and $\widetilde{s_1}$ and set $H(\widetilde{s_0}) = \widetilde{pk_0} \oplus \widetilde{s_1}$ and $H(\widetilde{s_1}) = \widetilde{pk_1} \oplus \widetilde{s_0}$. To prove indistinguishability, one would need to show the view of Alice observing $\widetilde{s_0}$ and $\widetilde{s_1}$ sampled uniformly at random and her interactions with the random oracle (with the outputs set accordingly) is indistinguishable from the real world transaction with Bob (next lecture).

Consider security against a malicious Bob:

Bob will need to send $\widetilde{s_0}$ and $\widetilde{s_1}$ to the simulator. The simulator must be able to find the implicit bit b send it to the trusted authority and receive m_b. One approach could be to watch the order Bob samples the random oracle. From the protocol, we know Bob will first ask for $H(\widetilde{r_0})$ and then query $H(\widetilde{r_1})$. Using this, one could potentially recover bit $b == 0$ if $\widetilde{s_0} = \widetilde{r_0}$ is queried first and $b == 1$ if $\widetilde{s_0} = \widetilde{r_1}$ is queried first. This concludes this lecture, we will revisit this proof of indistinguishability in the next lecture.

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.

References

[1] P. S. L. M. Barreto, B. David, R. Dowsley, K. Morozov, and A. C. A. Nascimento. *A Framework for Efficient Adaptively Secure Composable Oblivious Transfer in the ROM.* Cryptology ePrint Archive, Report 2017/993, 2017. https://eprint.iacr.org/2017/993, 2017.