CSA E0 312: Secure Computation	23/09/2015
[Lecture 13-14]	
Instructor: Arpita Patra	*Submitted by: Nithin V Nath*

In the previous lectures, we saw secure two-party computation using Yao's protocol. In this lecture, we are going to see how Yao's two-party protocol can be achieved using existing cryptographic tools as well as its proof of security.

1 Recap

In Yao's protocol a garbled circuit is constructed by one party (*constructor*, P_1) and sent to the other party (*evaluator*, P_2). The constructor assigns to each wire, identical looking *keys*(labels) corresponding to each possible input. For each gate, the keys corresponding to the input gates are *locked* (double encrypted) using the input keys. The keys corresponding to the input values of the *evaluator* are sent by the *constructor* using OT.

2 Yao's Garbled Circuit from "Special SKE"

In this section we will see how a *special* symmetric key encryption (SKE) can be used for Yao's protocol.

- Keys of SKE can be used as the keys for each wires.
- SKE can be used to encrypt the resulting key of each gate.
- For example, in an AND gate, each of the two input wires will have two SKE keys associated with them. The output wire also will have two SKE keys associated.

2.1 Syntax of SKE

A Symmetric Key Encryption has 3 algorithms $Gen()$, $Enc_k()$ and $Dec_k()$

1. **Key-generation Algorithm** $Gen()$: This algorithm outputs a key k chosen according to some probability distribution determined by the scheme. To ensure security, this must be a randomized algorithm.
2. **Encryption Algorithm** $Enc_k(m)$: This algorithm takes as input the message $m \in \{0,1\}^*$ and returns c, the ciphertext. This can be a deterministic or a randomized algorithm.
3. **Decryption Algorithm** $Dec_k(c)$: This algorithm takes as input the ciphertext and outputs the corresponding message $m := Dec_k(c)$.

Also associated with each algorithm are the following.

1. **Key space** ($\mathcal{K}$): Set of all possible keys output by $Gen()$

2. **Plain-text message space** ($\mathcal{M}$): Set of all possible *legal* messages. In other words, the valid inputs to the $Enc()$ algorithm.

3. **ciphertext space** ($\mathcal{C}$): Set of all ciphertexts output by Enc.

The sets $\mathcal{K}$ and $\mathcal{M}$ together define the set $\mathcal{C}$.

We will now see that not just any SKE will work in case of Yao's protocol.

2.2 Garbling of Wires

Every wire is associated with a pair of **identical looking** SKE keys. Both keys are chosen independently at random.

2.3 Garbling the Gates

The keys corresponding to output wires are double encrypted using the keys from input wires. For instance if it were an AND gate, then the key corresponding to 1 from each input wire will be used to double encrypt the key corresponding to 1 in the output wire. Each pair will decrypt exactly one ciphertext.

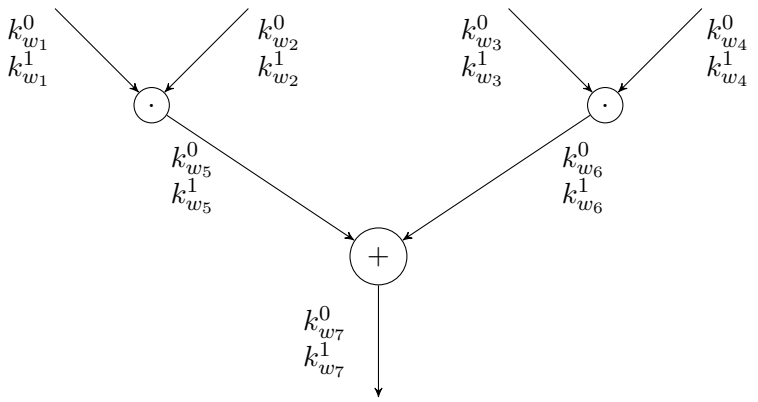

Figure 1: An example circuit

Input wire 1	**Input wire 2**	**Double Encryptions**
$k^0_{w_1}$	$k^0_{w_2}$	$Enc_{k^0_{w_1}}(Enc_{k^0_{w_2}}(k^0_{w_5}))$
$k^0_{w_1}$	$k^1_{w_2}$	$Enc_{k^0_{w_1}}(Enc_{k^1_{w_2}}(k^0_{w_5}))$
$k^1_{w_1}$	$k^0_{w_2}$	$Enc_{k^1_{w_1}}(Enc_{k^0_{w_2}}(k^0_{w_5}))$
$k^1_{w_1}$	$k^1_{w_2}$	$Enc_{k^1_{w_1}}(Enc_{k^1_{w_2}}(k^1_{w_5}))$

Figure 2: Computing the garbling of 1st AND gate

2.3.1 Need for random permutation

The garbled AND gate sent will be the 3rd column of Figure 2. However, it is not ideal to send the GATE in any fixed order. Suppose the constructor sends the garbling in the order present in the table itself. Then suppose the evaluator is able to *open* the second row. The evaluator can deduce that the k_{w_1} he possess corresponds to 0 and k_{w_2} corresponds to 1. Thus some intermediate result is leaked. In order to avoid this, the rows are randomly permuted and sent.

2.3.2 Need for special SKE

In Yao's protocol one pair of keys should open exactly one *box*. If we are using SKE for *locking* (encrypting), then opening a box corresponds to decryption. However in usual SKE, a wrong key will lead to a wrong message - decryption does not fail. As a consequence, the circuit evaluator cannot know which decrypted value is the intended output key and therefore **correctness of two-party computation will fail.**

The SKE should have to additional properties

1. The ciphertext spaces must be distinct under distinct keys with high probability. That is, the encryption under one key will fall in the range of another key with negligible probability.
2. There must be a mechanism to *efficiently* verify if a given ciphertext belongs to the ciphertext space of a given key.

Formally the requirements can be specified using the following definition.

Definition 1 Let (G, E, D) be a private-key encryption scheme and denote the range of a key in the scheme by $\mathsf{Range}_n(k) \stackrel{\text{def}}{=} \{E_k(x)\}_{x\in\{0,1\}^n}$.

1. We say that (G, E, D) has an elusive range if for every probabilistic polynomial-time machine A, every polynomial $p(\cdot)$ and all sufficiently large ns

$$Pr_{k\leftarrow G(1^n)}[A(1^n) \in \mathsf{Range}_n(k)] < \frac{1}{p(n)} \tag{1}$$

2. We say that (G, E, D) has an efficiently verifiable range if there exists a probabilistic polynomial-time machine M such that $M(1^n, k, c) = 1$ if and only if $c \in \mathsf{Range}_n(k)$

By convention, for every $c \notin \mathsf{Range}_n(k)$, we have $D_k(c) = \perp$. ♢

Example SKE: $\mathcal{F} = \{f_k\}$ be a family of pseudorandom functions, where $f_k : \{0,1\}^n \to \{0,1\}^{2n}$ for $k \in \{0,1\}^n$. Define

$$E_k(x) =< r, f_k(r) \oplus x0^n > \tag{2}$$

where $x \in \{0,1\}^n, r \leftarrow_R \{0,1\}^n$ and $x0^n$ denotes concatenation of x and 0^n

2.3.3 Drawbacks

1. The evaluator has to do multiple trial decryption which is computationally expensive.
2. Huge ciphertext size for the special SKEs.
3. The correctness proof is more involved.

2.3.4 Point-and-Permute

Point-and-permute is a method to reduce the evaluation time for a garbled circuit. The wire labels for each wire are randomly given *permutation bits* 0 and 1. Since the assignment of this bit is independent of the wire labels' association with true(1) and false(0), the permutation bits can be safely revealed to the evaluator. The evaluator can simply use the permutation bits he sees as pointers to index the appropriate ciphertext rather than performing trial decryption on all 4. In summary, it works in the following way

1. A **random** bit called *permutation bit* will be associated with each wire in the construction phase.
2. The *permutation bits* corresponding to input wires of a gate are used to permute the ciphertexts
3. For the example given in Figure 3, $Enc_{k^0_{w_1}}(Enc_{k^0_{w_2}}(k^0_{w_5}))$ will be placed at the $p_1 p_2^{th}$ row.

This also has the added advantage of not requiring any special property from SKE. Since the evaluator exactly knows which row to decrypt.

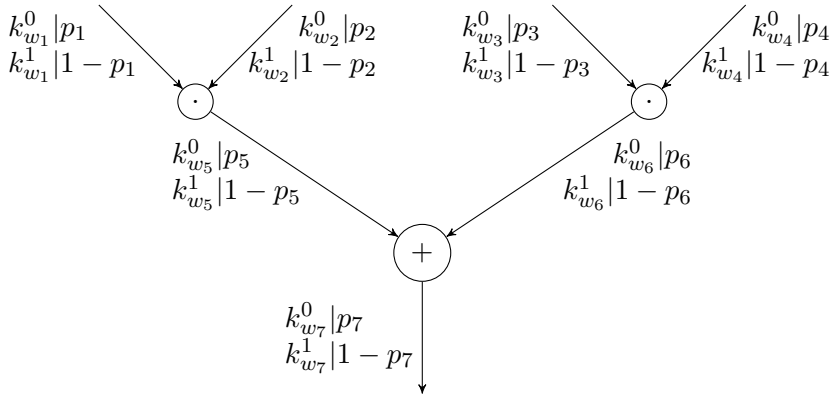

Figure 3: An example circuit with point and permute

The key space, message space and ciphertext space are now related as

$$\mathcal{K}|\{0,1\} = \mathcal{M} = \mathcal{C} \tag{3}$$

Garbled AND Gate
$Enc_{k^1_{w_1}}(Enc_{k^1_{w_2}}(k^1_{w_5}\|1-p_5))$
$Enc_{k^1_{w_1}}(Enc_{k^0_{w_2}}(k^0_{w_5}\|p_5))$
$Enc_{k^0_{w_1}}(Enc_{k^1_{w_2}}(k^0_{w_5}\|p_5))$
$Enc_{k^0_{w_1}}(Enc_{k^0_{w_2}}(k^0_{w_5}\|p_5))$

Figure 4: ciphertext table assuming $p_1 = p_2 = 1$

Now the SKE must be such that a bad evaluator should have no information about what the three unopened ciphertext contains. To see the problem, consider the example of AND gate. If the evaluator is able to see that the three unopened messages are same, then it clearly means the decrypted key corresponds to 1. So we need a very subtle security definition.

2.3.5 Double-encryption security

Let (G, E, D) be a private-key encryption scheme and assume without loss of generality that $G(1^n)$ returns a string of length n. The experiment definition is

---

$\mathsf{Expt}^{\mathsf{double}}_{\mathcal{A}}(n, b)$

1: The adversary $\mathcal{A}$ is invoked upon input 1^n and outputs two keys k_0 and k_1 of length n and two triples of messages (x_0, y_0, z_0) and (x_1, y_1, z_1) where all messages are of same length

2: Two keys $k'_0, k'_1 \leftarrow G(1^n)$ are chosen for the encryption scheme.

3: $\mathcal{A}$ is given the challenge ciphertext

$$\langle E_{k_0}(E_{k'_1}(x_b)), E_{k'_0}(E_{k_1}(y_b)), E_{k_0}(E_{k'_1}(z_b))\rangle$$

as well as **oracle access** to $E_{\cdot}(E_{k'_1}(\cdot))$ and $E_{k'_0}(E_{\cdot}(\cdot))$

4: $\mathcal{A}$ outputs a bit b' and this is taken as the output of the experiment

---

Note that in the ciphertexts that $\mathcal{A}$ receives, at least one of the keys used is unknown to $\mathcal{A}$. The oracle access here means that $\mathcal{A}$ can provide any k amd m to $E_{\cdot}(E_{k'_1}(\cdot))$ and receive back $E_k(E_{k'_1}(m))$. Similarly for $E_{k'_0}(E_{\cdot}(\cdot))$

Definition 2 An encryption scheme (G, E, D) is secure under chosen double encryption if for every non-uniform probabilistic polynomial-time machine $\mathcal{A}$, every polynomial $p(\cdot)$ and all sufficiently large n,

$$|Pr\left[\mathsf{Expt}^{\mathsf{double}}_{\mathcal{A}}(n,1) = 1\right] - Pr\left[\mathsf{Expt}^{\mathsf{double}}_{\mathcal{A}}(n,0) = 1\right]| < \frac{1}{p(n)}$$

♢

Theorem 1 *An encryption scheme that has indistinguishable encryptions under chosen-plaintext attacks, is secure under chosen double encryption. In other words, every CPA-secure scheme is also CDE-secure.*

2.4 Completing the picture

Garbled Circuit: Garbled gates along with output decryption table

1: The constructor P_1 constructs the garbled circuit using SKE keys and sends it to the evaluator P_2 along with the input keys corresponding to P_1's input.
2: For every input wire of P_2, P_1 and P_2 runs a 1-out-of-2 oblivious transfer protocol so that P_2 gets the keys corresponding to the input. (This can be run in parallel)
3: For every gate, P_2 uses the *permutation bits* from the input wire and decrypts the ciphertext pointed by it. This will give the output key and the permutation bit of output wire.
4: For the output gate, the key corresponding to the output value for given inputs is obtained and is translated to correct output using the decryption tables.

2.5 Security

Intuitively, the security of the protocol comes from the security of the oblivious transfer protocol and the security of the encryption scheme.

After constructing the garbled circuit, P_1 only learns the output. P_1 cannot learn the inputs of P_2 because of the security of the oblivious transfer protocol.

P_2 receives exactly one key per circuit wire (OT security) and is therefore able to decrypt exactly one value in each gate. Furthermore, since the keys corresponding to 1 and 0 are randomly chosen, P_2 cannot learn if the value obtained in this decryption corresponds to a 0 or a 1. Thus P_2 learns only the output from the computation.

2.5.1 Case 1 - P_1 is corrupted

P_1's view of the OT contains the execution of the OTs and a single message containing $f(x, y)$ received from P_2. Since the OTs are secure, there is a simulator which simulates P_1's view in the OT given its input to them alone.

Simulator S_1: $(x, f(x, y))$

1: Choose randomness r_c and generate the garbled circuit using r_c.
2: Let $k^0_{n+1}, k^1_{n+1}, \ldots, k^0_{2n}, k^1_{2n}$ be the keys corresponding to P_2's input in the garbled circuit.
3: Let S_1^{OT} be the simulator for party P_1 in the oblivious transfer protocol. Since OTs are secure, this will exist.
4: For every $i = 1, \ldots n$, invoke simulator S_1^{OT} with input (k^0_{n+i}, k^1_{n+i}) to obtain P_1's view in the ith oblivious transfer.
5: Output $(x, r_C, S_1^{OT}(k^0_{n+1}, k^1_{n+1}), \ldots, S_1^{OT}(k^0_{2n}, k^1_{2n}))$

This view is indistinguishable from the real view since the OTs are all secure. Define a hybrid distribution H_i in which the first i oblivious transfers are simulated and the last $n - i$ are real. A hybrid argument can now be used to prove this indistinguishability.

2.5.2 Case 2 - P_2 is corrupted

We need to construct a simulator S_2 that is given input $(y, f(x,y))$ and generates the view of P_2. It needs to send a garbled circuit to P_2 which when evaluated must output $f(x,y)$. Since S_2 doesn't know x, it cannot honestly create a circuit. Therefore the simulator creates a circuit that always evaluates to $f(x,y)$, irrespective of the keys used. This is achieved using gate tables in which all four entries encrypt the same key.

Simulator S_2: $(y, f(x,y))$

1: For every wire w_i in the circuit C, simulator S_2 chooses two random keys k_i and k'_i.

2: The gates are computed: Let g be a gate with input wires w_i. w_j and output wire w_l. g contains encryptions of the single key k_l under *all four combinations* of the keys k_i, k'_i, k_j, k'_j that are associated with the input wires to g. That is, S_2 computes the values

$$c_{0,0} = E_{k_i}(E_{k_j}(k_l)), \qquad c_{0,1} = E_{k_i}(E_{k'_j}(k_l)),$$
$$c_{1,0} = E_{k'_i}(E_{k_j}(k_l)), \qquad c_{1,1} = E_{k'_i}(E_{k'_j}(k_l)),$$

3: Let n-bit output $f(x,y)$ be denoted by $z_1 \cdots z_n$. Let the output wires be $w_m - n + 1, \ldots, w_m$. Let k_{m-n+i} be the (single) key encrypted in the gate from which wire w_{m-n+i} left, and let k'_{m-n+i} be the other key.

4: Output decryption table for wire w_{m-n+i} is given by

- if $z_i = 0$, then $[(0, k_{m-n+i}), (1, k'_{m-n+i})]$
- if $z_i = 1$, then $[(0, k'_{m-n+i}), (1, k_{m-n+i})]$

This ends the construction of garbled circuit $\tilde{G}(C)$. Generating the view remains.

5: **Obtaining keys:** Set the keys that P_2 obtains from P_1 to be $k_1, \ldots, k_n$. (Arbitrary. Selecting k'_i instead makes no difference.

6: **OT phase:** Let S_2^{OT} be the simulator for OT protocol. For every $i \in [n]$, invoke $S_2^{OT}(y_i, k_{n+i})$. y_i is the input of P_2 and k_{n+i} is the output to P_2. (This is also arbitrary. Could be k'_{n+i} also.

7: Output

$$\Big(y, \tilde{G}(C), k_1, \ldots, k_n, S_2^{OT}(y_1, k_{n+1}), \ldots, S_2^{OT}(y_n, k_{2n})\Big)$$

The detailed proof shows the computational indistinguishability of the view output by the simulator and the real view $\{view_2^{\pi}(x,y)\}$.

$$\{view_2^{\pi}(x,y)\} \equiv \left\{\left(y, G(C), k_1^{x_1}, \ldots, k_n^{x_n}, R_2^{OT}((k_{n+1}^0, k_{n+1}^1), y_1), \ldots, R_2^{OT}((k_{2n}^0, k_{2n}^1), y_n)\right)\right\}$$

$R_2^{OT}((k_{n+i}^0, k_{n+i}^1), y_i)$ denotes the *real transcript* obtained from the OT.

Proof Details [1]

We need to show that P_2 cannot distinguish the circuit it receives from the correct circuit. First, a hybrid distribution H_{OT} is defined where the real oblivious transfers are replaced with simulated ones. The garbled circuit remains the same.

$$H_{OT}(x,y) = \left(y, G(C), k_1^{x_1}, \ldots, k_n^{x_n}, S_2^{OT}(y_1, k_{n+1}^{y_1}), \ldots, S_2^{OT}(y_n, k_{2n}^{y_n})\right)$$

The only difference from the real view is the use of simulators for oblivious transfers. Therefore indistinguishability follows from the security of the oblivious transfer protocol.

Now consider a series of **hybrids** $H_i(x,y)$ in which one gate at a time is replaced in the real garbled circuit. The gates are replaced in the topologically sorted order. The difference between $H_i(x,y)$ and $H_{i+1}(x,y)$ is that one more real table is replaced with a fake one.

- $H_0(x,y)$ is equal to $H_{OT}(x,y)$ and contains a real garbled circuit.
- $H_{|C|}(x,y)$ contains the fake circuit constructed by S.

We want to show $\{H_0(x,y)\} \stackrel{c}{\equiv} \{H_{|C|}(x,y)\}$. Intuitively, this follows from the indistinguishability of encryptions. We use a hybrid argument. Suppose there exists a non-uniform probabilistic polynomial-time distinguisher D and a polynomial $p(\cdot)$ such that for infinitely many ns,

$$|Pr[D(H_0(x,y)) = 1] - Pr[D(H_{|C|}(x,y)) = 1]| > \frac{1}{p(n)}$$

It follows that there exists an i such that

$$|Pr[D(H_{i-1}(x,y)) = 1] - Pr[D(H_i(x,y)) = 1]| > \frac{1}{|C| \cdot p(n)}$$

If such a distinguisher exists, it violates the security of the encryption scheme used. This is because we can use D and x, y, i in order to construct a non-uniform probabilistic polynomial-time distinguisher $\mathcal{A}_E$ for the encryption scheme (G, E, D).The idea is that $\mathcal{A}_E$ will receive some ciphertexts from which it will construct a partially real and partially fake garbled circuit $G'(C)$. The construction will be such that if the ciphertexts received were of one "type", then the resulting circuit is according to $H_{i-1}(x,y)$. Otherwise the resulting circuit is according to $H_i(x,y)$. This ability to distinguish between H_i and H_{i-1} enables $\mathcal{A}_E$ to distinguish ciphertexts. This violates the security of encryption scheme.

Therefore no such distinguisher can exist and the two-party protocol is secure.

3 References

1. C. Hazay and Y. Lindell. *Efficient Secure Two-Party Computation: Techniques and Constructions*. Springer, 2010.

[1] Please see reference 1 for a complete treatment of proof

2. An Annotated Bibliography of Practical Secure Computation https://web.engr.oregonstate.edu/~rosulekm/scbib/index.php?n=Glossary.PointAndPermute. Accessed October 12, 2015.