CSA E0 312: Secure Computation September 09, 2015

[Lecture 9-10]

Instructor: Arpita Patra *Submitted by: Pratik Sarkar*

1 Summary

In this lecture we will introduce the concept of Public Key Samplability and use it alongwith CPA security to build the Oblivious Transfer protocol. Then we prove the receiver end and sender end security using key samplability and CPA security by applying them to hybrid models. At the end of the lecture we discuss about the GMW protocol and give its security proof.

2 Oblivious Transfer Protocol

We can visualise the OT as a blackbox with two parties, sender S and receiver R. S gives the inputs m_0 and m_1. R gives a input bit b and receives the message m_b. The OT functions in such a way that S doesn't know anything about b and m_b whereas R doesn't know anything about m_{1-b}. The diagram for a OT is in Fig 1.

In this section we define the concepts of CPA-secure public key encryption scheme and key samplability and use them to develop a 1-out-of-2 OT protocol by Even-Goldreich-Lempel [EGL85] scheme.

2.1 Public Key Encryption

A public key encryption scheme π consists of three probabilistic polynomial time (PPT) algorithms (Gen, Enc, Dec) where

1. $Gen(1^n)$ is the key generation algorithm that takes in input 1^n (where n is the security parameter) and outputs the public key pk and the secret key sk. $(pk, sk) \leftarrow Gen(1^n)$
2. $Enc(m, pk)$ is the encryption algorithm that takes in input a message m and the public key pk and outputs the ciphertext $c \leftarrow Enc_{pk}(m)$
3. $Dec(c, sk)$ is the decryption algorithm that on input a ciphertext c and secret key sk outputs the message $m = Dec_{sk}(c)$.

It is necessary that for every n and every pair of $(pk, sk) \leftarrow Gen(1^n)$ and every message m from the message space, it holds that $Dec_{sk}(Enc_{pk}(m)) = m$.

2.2 CPA security

Let π be a public key encryption scheme $(Gen(1^n), Enc(m, pk), Dec(c, sk))$ and A be the adversary turing machine claiming to distinguish between the encryption of two messages

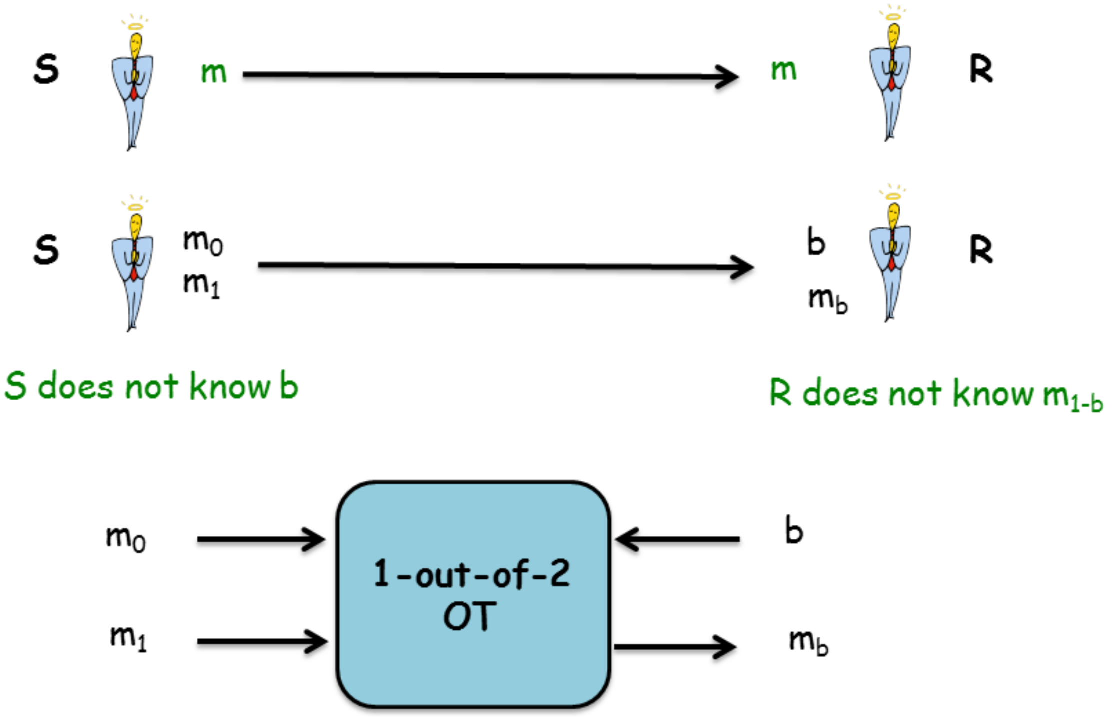


Figure 1: 1-out-of-2 OT

m_0 *and* m_1 sampled from $\{0,1\}^n$. π is said to be CPA secure if,
$|Pr[A(Enc_{pk}(m_0))] - Pr[A(Enc_{pk}(m_1))]| \leq negli(n)$,
where $negli(n)$ is a negligible function.

π is CPA-secure if for every PPT attacker A taking part in the above experiment, the probability that A wins the experiment is at most negligibly better than $\frac{1}{2}$.

2.3 Public Key Samplability

Next we introduce the concept of public key samplability in a public key encryption scheme. It contains 5 PPT algorithms:

1. $Gen(1^n)$ is the key generation algorithm that takes in input 1^n (where n is the security parameter) and outputs the public key pk and the secret key sk. $(pk, sk) \leftarrow Gen(1^n)$
2. $Enc(m, pk)$ is the encryption algorithm that takes in input a message m and the public key pk and outputs the ciphertext $c \leftarrow Enc_{pk}(m)$
3. $Dec(c, sk)$ is the decryption algorithm that on input a ciphertext c and secret key sk outputs the message $m = Dec_{sk}(c)$.

4. $oGen(1^n)$ generates a public key (pk) and the corresponding randomness (r) given the security parameter (n) as input. It doesnt return the secret key corresponding to the secret key. $(pk, r) \leftarrow oGen(1^n)$

5. $fGen(1^n)$ is the randomness (r') generator corresponding to the public key pk as input. $r' \leftarrow oGen(pk)$

   The (pk, r) from $oGen()$ and (pk, r') from $fGen()$ looks indistinguishable to an adversary. This can be shown using the key samplability experiment described below.

2.3.1 Key Samplability Experiment

There is a $\pi(Gen, Enc, Dec, oGen, fGen)$ public key encryption scheme satisfying the public key samplability property $PubK^{ksamp}_{A,\pi}(n)$ for adversary A. There is an Adversary A who claims to the challenger C that he can break the scheme and distinguish between two public keys generated from $oGen()$ and $fGen()$. The experiment proceeds as follows:

(a) The Challenger C randomly chooses a bit b and based on that he runs either the $(Gen, fGen)$ algorithm or the $oGen$ algorithm.
Suppose for example,
for b=0, $(pk, sk) \leftarrow Gen(1^n), r \leftarrow fGen(pk)$, and returns (pk, r)
for b=1, $(pk, r) \leftarrow oGen(1^n)$
He then sends (pk, r) to the adversary A.

(b) Now the adversary knows that the challenger will always run the $(Gen, fGen)$ for $b = 0$ and $oGen$ for $b = 1$. So seeing (pk, r) he will guess the value of b as b', inorder to distinguish whether the (pk, r) pair came from $(Gen, fGen)$ or $oGen$. He sends the value of b' to the challenger.

(c) If$(b == b')$ then the adversary wins else he loses. The adversary wins only when he can correctly distinguish between the public keys and randomness of the two algorithms.

π is key-samplable if for every PPT attacker A taking part in the above experiment, the probability that A wins the experiment is at most negligibly better than $\frac{1}{2}$.
$Pr[PubK^{ksamp}_{A,\pi}(n) = 1] \leq \frac{1}{2} + negli(n)$.

The Elgamal PKE satisfies a trivial sampleable public keys, where the randomness is the bit representation of the public keys.

2.4 Construction of 1-out-of-2 Oblivious Transfer

The construction of an OT (shown in Fig.2) using a CPA secure and key sampleable π encryption scheme is as follows:

(a) The receiver R generates a random bit b and generates a (pk_b, sk_b) pair from Gen algorithm and a (pk_{1-b}, r_{1-b}) from $oGen$ algorithm. He sends (pk_0, pk_1) to sender S who has two messages m_0, m_1.

(b) The sender encrypts $m_{b'}$ into $c_{b'}$, for $b' = 0$ and 1 using the public keys $pk_{b'}$ and sends the cipher texts to R.

(c) R decrypts m_b using s_b and gets the required message.

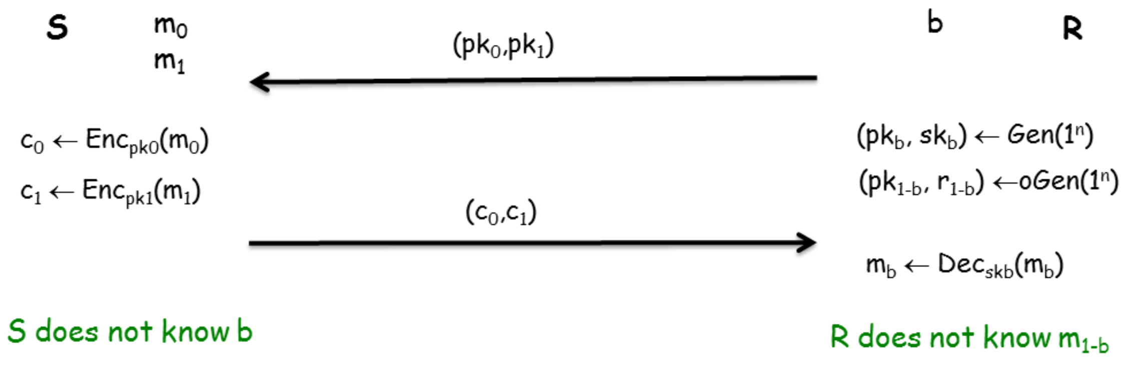


Figure 2: Construction of OT

By using this construction, the receiver securely receives the message m_b without revealing his b choice to the sender. Also the sender's m_{1-b} input was hidden from the receiver. Thus an OT is successfully implemented using the properties of key samplability and CPA security. Now we show the security at both the sender and receiver ends for semi-honest adversaries.

2.4.1 Security for the Receiver

We prove the security of this scheme using the real world and ideal world indistinguishability. Let us first prove it for against a corrupted sender. The protocol has been shown in Fig. 3. In the real world, the receiver follows the normal protocol and the adversary A at S tries to know the value of b chosen by R so that he can know which message R has chosen. A receives pk_0, pk_1 and finds randomness r_0^S, r_1^S for the two encryptions and sends the ciphertexts c_0, c_1 to R. In ideal world A has the following view:
$View_S^{Real}(m_0, m_1, b, k) = \{m_0, m_1, pk_0, pk_1, r_0^S, r_1^S\}$
In the ideal world, there is a simulator SIM_s which has the input and output of the corrupted sender A and it tries to simulate the view of the honest receiver to A. We run the SIM_s and the sender S turing machines to find the ideal world view.

$SIM_s(1^n, m_0, m_1)$:

(a) Run $(pk_0, sk_0) \leftarrow Gen(1^n)$ and $(pk_1, sk_1) \leftarrow Gen(1^n)$

(b) Output (pk_0, pk_1).

A performs the same protocols in the ideal world as in the real world. In the ideal world A has the following view: $View_S^{Ideal}(m_0, m_1, b, k) = \{m_0, m_1, pk_0, pk_1, r_0^S, r_1^S\}$

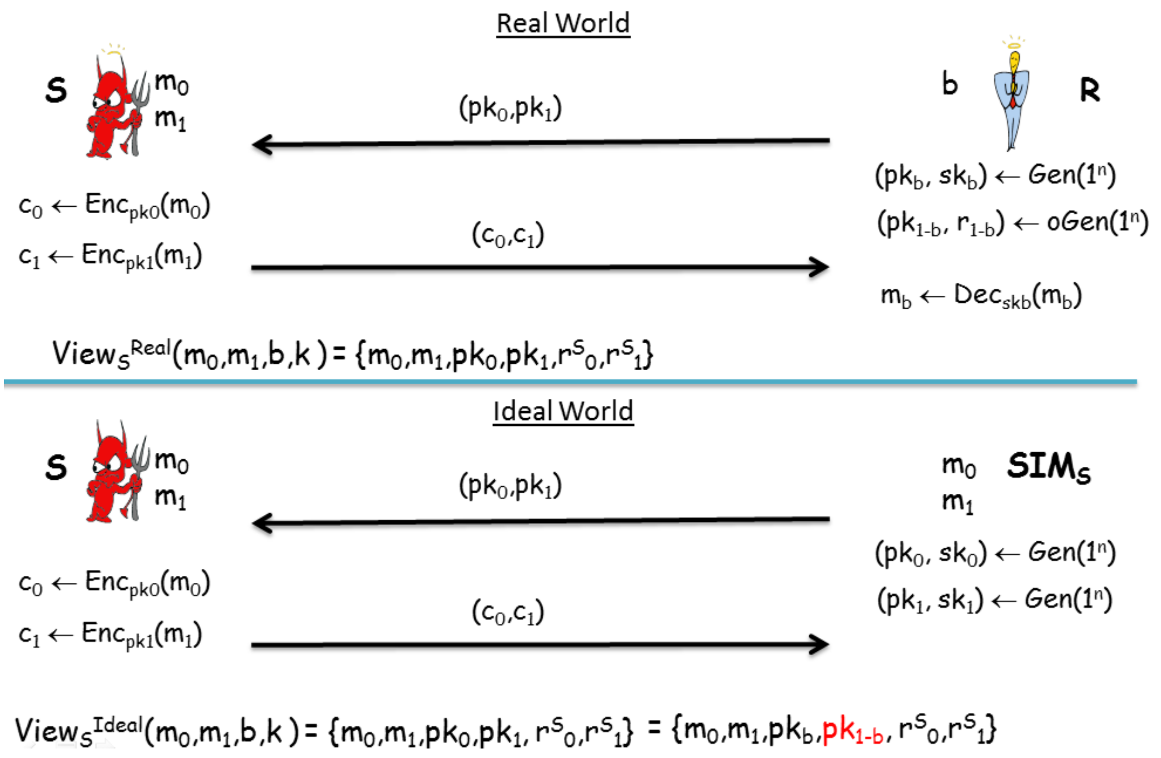

Real World
S
m_0
m_1
b
R
(pk_0,pk_1)
$(pk_b, sk_b) \leftarrow Gen(1^n)$
$(pk_{1-b}, r_{1-b}) \leftarrow oGen(1^n)$
$c_0 \leftarrow Enc_{pk0}(m_0)$
$c_1 \leftarrow Enc_{pk1}(m_1)$
(c_0,c_1)
$m_b \leftarrow Dec_{skb}(m_b)$
$View_S^{Real}(m_0,m_1,b,k) = \{m_0,m_1,pk_0,pk_1,r^S_0,r^S_1\}$
Ideal World
S
m_0
m_1
m_0
m_1
SIM_S
(pk_0,pk_1)
$(pk_0, sk_0) \leftarrow Gen(1^n)$
$(pk_1, sk_1) \leftarrow Gen(1^n)$
$c_0 \leftarrow Enc_{pk0}(m_0)$
$c_1 \leftarrow Enc_{pk1}(m_1)$
(c_0,c_1)
$View_S^{Ideal}(m_0,m_1,b,k) = \{m_0,m_1,pk_0,pk_1, r^S_0,r^S_1\} = \{m_0,m_1,pk_b,pk_{1-b}, r^S_0,r^S_1\}$

Figure 3: Security for Receiver

The public and the secret key pair (pk_b, sk_b) corresponding to the choice bit b is identically distributed in both cases above as they are both generated by running the key generation algorithm of the public key encryption scheme. The public keys pk_{1-b} are also indistinguishable in both ideal world and real world since the OT has been constructed using π, which is a public key ecryption scheme satisfying key samplability property. And so public keys generated from $(Gen, fGen)$ and $oGen$ are indistinguishable. So the p_{1-b} in the real world (generated by $oGen$) and p_{1-b} in the ideal world (generated by Gen) are indistinguishable. And hence both the real and ideal world views are same. Thus it is secure against corrupt sender.

2.4.2 Security for the Sender

The receiver is corrupted by adversary A and the sender is the honest party. We show the protocol in Fig4. The view of the adversary in the real world is:
$View_R^{Real}(m0, m1, b, k) = \{b, m_b, pk_b, r_b, pk_{1-b}, r_{1-b}, c_0, c_1\}$

In the ideal world we consider a simulator SIM_R who simulates the view of the sender against the receiver adversary. The adversary follows the normal receiver end protocol. The simulator has the inputs (b, m_b) and it does the following:
$SIM_R(k, m_b, b)$:

(a) $c_b \leftarrow Enc_{pk_b}(m_b)$

(b) $c_{1-b} \leftarrow Enc_{pk_{1-b}}(0^n)$

(c) Send c_b, c_{1-b} to receiver

The view of the receiver in the ideal world is:
$View_R^{Ideal}(m0, m1, b, k) = \{b, m_b, pk_b, r_b, pk_{1-b}, r_{1-b}, c_b, c_{1-b}\}$ Only the c_{1-b} is different for the receiver of the ideal and the real world. We cannot apply CPA security here directly, because c_{1-b} is encrypted using the public key generated by $oGen$ algorithm and not a Gen algorithm.

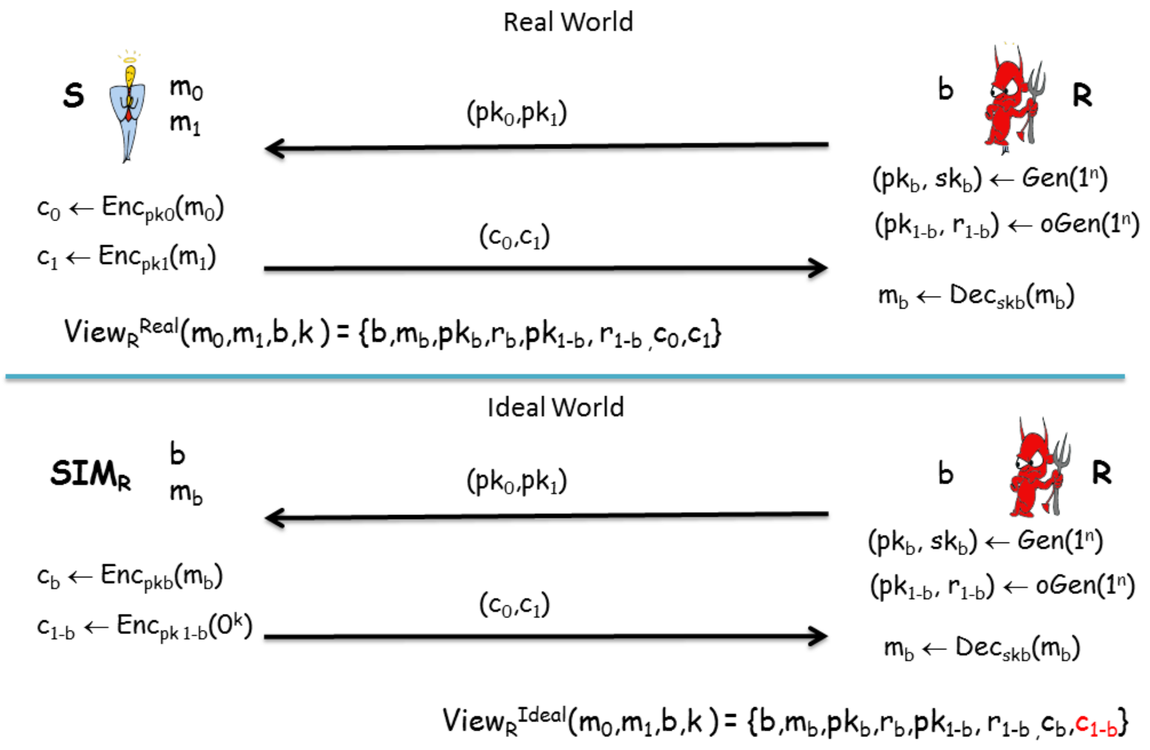


Figure 4: Security for Sender

So we need 2 hybrid views to prove the security of this scheme. We show the hybrid models in the Fig5 The first hybrid model does the follwoing:
$View_R^{HYBRID1}(k, m_0, m_1, b)$:

(a) $(pk_b, sk_b) \leftarrow Gen(1^n)$

(b) $(pk_{1-b}, sk_{1-b}) \leftarrow Gen(1^n)$

(c) $r_{1-b} \leftarrow fGen(pk_{1-b})$

(d) $c_b \leftarrow Enc_{pk_b}(m_b)$

(e) $c_{1-b} \leftarrow Enc_{pk_{1-b}}(m_{1-b})$

$View_R^{Hybrid1}(m_0, m_1, b, k) = \{b, m_b, pk_b, r_b, pk_{1-b}, r_{1-b}, c_b, c_{1-b}\}$

The $View_R^{HYBRID1}$ and $View_R^{Real}$ are identical as the only difference is the (pk_{1-b}, r_{1-b}) but they are indistinguishable in both the cases due to key samplability. Now we consider the second hybrid model from the first.

$View_R^{HYBRID2}(k, m_0, m_1, b)$:

(a) $(pk_b, sk_b) \leftarrow Gen(1^n)$

(b) $(pk_{1-b}, sk_{1-b}) \leftarrow Gen(1^n)$

(c) $r_{1-b} \leftarrow fGen(pk_{1-b})$

(d) $c_b \leftarrow Enc_{pk_b}(m_b)$

(e) $c_{1-b} \leftarrow Enc_{pk_{1-b}}(0^k)$

$View_R^{Hybrid2}(m_0, m_1, b, k) = \{b, m_b, pk_b, r_b, pk_{1-b}, r_{1-b}, c_b, c_{1-b}\}$ The $View_R^{HYBRID1}$ and $View_R^{HYBRID2}$ are same because they differ only in c_{1-b}, but they are indistinguishable due to CPA security. Now we show the ideal world view:

$View_R^{Ideal}(k, m_0, m_1, b)$:

(a) $(pk_b, sk_b) \leftarrow Gen(1^n)$

(b) $(pk_{1-b}, sk_{1-b}) \leftarrow oGen(1^n)$

(c) $c_b \leftarrow Enc_{pk_b}(m_b)$

(d) $c_{1-b} \leftarrow Enc_{pk_{1-b}}(0^k)$

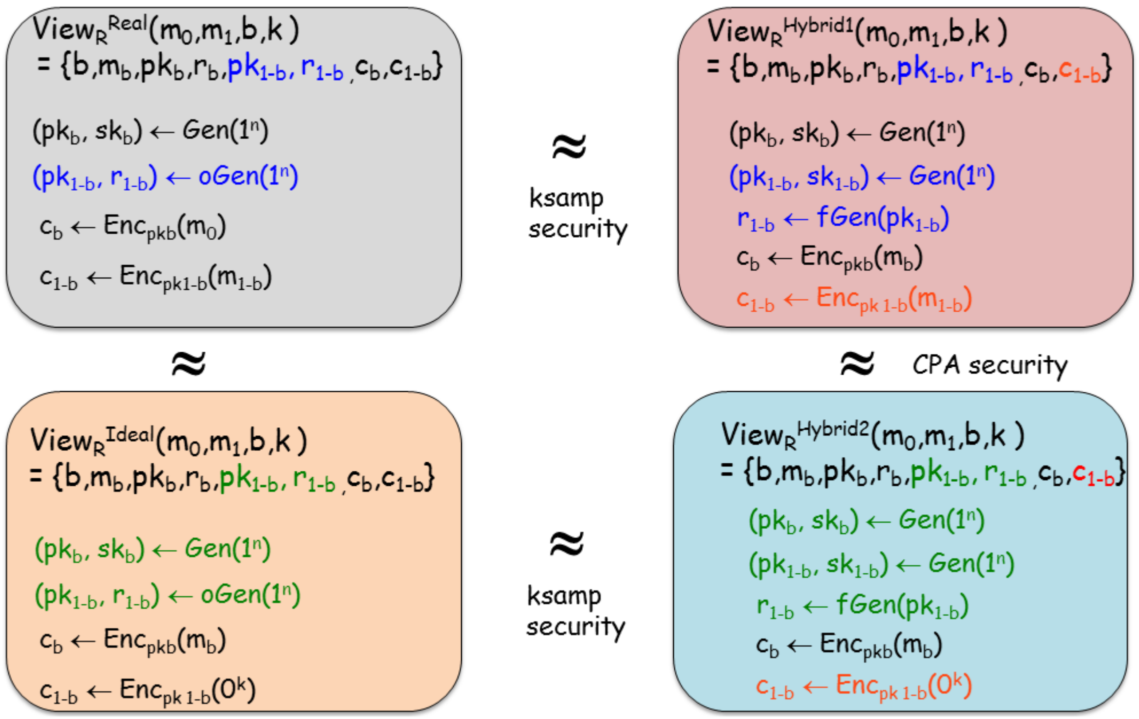
$View_R^{Ideal}(m_0, m_1, b, k) = \{b, m_b, pk_b, r_b, pk_{1-b}, r_{1-b}, c_b, c_{1-b}\}$ Again both the $View_R^{HYBRID2}$ and $View_R^{Ideal}$ are same as they differ in only (pk_{1-b}, r_{1-b}), which cannot be distinguished due to key samplability property. Thus the $View_R^{Real}$ and $View_R^{Ideal}$ are indistinguishable.

3 GMW(Goldreich-Micali,Wigderson) protocol: Approach to semi-honest two party computation

The GMW protocol is used by n parties to securely compute a function provided there is an honest majority. For n parties we require a (n,n) - secret sharing for semi-honest adversaries. The adversaries cannot leak anything if there are less than n shares available to them.

In this lecture we consider a 2-out-of-2 secret sharing scheme for 2 parties. This protocol is applied for boolean circuits and the interactive computations are done using OTs. Each party holds a secret share. Each i^{th} party holds shares (x_i, y_i). There are 3 types of operations which are done in the boolean computations- XOR, NOT and AND. We show that the parties can locally compute the shares for XOR and NOT operations but not for AND:

1. **XOR :** No communications are required for an XOR gate - each party can construct the shares of the output using their existing shares of the inputs.
   Suppose the parties need to compute $z = x \oplus y$ then each party can locally compute $z_i = x_i \oplus y_i$.



$\text{View}_R^{\text{Real}}(m_0,m_1,b,k)$
$= \{b,m_b,pk_b,r_b,pk_{1-b},r_{1-b},c_b,c_{1-b}\}$
$(pk_b, sk_b) \leftarrow Gen(1^n)$
$(pk_{1-b}, r_{1-b}) \leftarrow oGen(1^n)$
$c_b \leftarrow Enc_{pkb}(m_0)$
$c_{1-b} \leftarrow Enc_{pk1-b}(m_{1-b})$
$\approx$
ksamp security
$\text{View}_R^{\text{Hybrid1}}(m_0,m_1,b,k)$
$= \{b,m_b,pk_b,r_b,pk_{1-b},r_{1-b},c_b,c_{1-b}\}$
$(pk_b, sk_b) \leftarrow Gen(1^n)$
$(pk_{1-b}, sk_{1-b}) \leftarrow Gen(1^n)$
$r_{1-b} \leftarrow fGen(pk_{1-b})$
$c_b \leftarrow Enc_{pkb}(m_b)$
$c_{1-b} \leftarrow Enc_{pk\,1-b}(m_{1-b})$
$\approx$
$\approx$ CPA security
$\text{View}_R^{\text{Ideal}}(m_0,m_1,b,k)$
$= \{b,m_b,pk_b,r_b,pk_{1-b},r_{1-b},c_b,c_{1-b}\}$
$(pk_b, sk_b) \leftarrow Gen(1^n)$
$(pk_{1-b}, r_{1-b}) \leftarrow oGen(1^n)$
$c_b \leftarrow Enc_{pkb}(m_b)$
$c_{1-b} \leftarrow Enc_{pk\,1-b}(0^k)$
$\approx$
ksamp security
$\text{View}_R^{\text{Hybrid2}}(m_0,m_1,b,k)$
$= \{b,m_b,pk_b,r_b,pk_{1-b},r_{1-b},c_b,c_{1-b}\}$
$(pk_b, sk_b) \leftarrow Gen(1^n)$
$(pk_{1-b}, sk_{1-b}) \leftarrow Gen(1^n)$
$r_{1-b} \leftarrow fGen(pk_{1-b})$
$c_b \leftarrow Enc_{pkb}(m_b)$
$c_{1-b} \leftarrow Enc_{pk\,1-b}(0^k)$

Figure 5: Security proof via Hybrid Arguments

2. **NOT :** Each party can flip the bit of the share, whose secret value needs to be flipped. Suppose the parties need to find $z = x'$ (where x' is the complement of x) then each party can locally compute $z_i = x_i'$.

3. **AND:** For AND gate computation interaction is required between the parties via 2 1-out-of-2 OTs.
   Suppose the parties need to compute $z = x.y = (x_0 + x_1).(y_0 + y_1) = (x_0.y_0 + x_0.y_1 + y_0.x_1 + x_1.y_1)$.
   $x_0.y_0$ and $x_1.y_1$ can be computed locally by the 1st and 2nd party respectively without any interaction. $x_0.y_1$ and $y_0.x_1$ has to be computed via OTs.
   The AND computation has been shown in the Fig 6. For the first OT, P_0 is at the sender end and P_1 is at the receiver end.

   (a) P_1 gives y_1 as input to the OT from the receiver end
   (b) P_0 selects a random value r_0
   (c) P_0 gives r_0 and $r_0 \oplus x_0$ as input to the OT from the sender end such that P_1 will receive r_0 for bit choice 0 and $r_0 \oplus x_0$ for bit choice 1
   (d) P_1 receives $r_0 \oplus x_0.y_1$ as output

   Now P_0 is at the receiver end and P_1 at the sender end and they do the following:

   (a) P_0 gives y_0 as input to the OT from the receiver end

(b) P_1 selects a random value r_1

(c) P_1 gives r_1 and $r_1 \oplus x_1$ as input to the OT from the sender end such that P_0 will receive r_1 for bit choice 0 and $r_1 \oplus x_1$ for bit choice 1

(d) P_0 receives $r_1 \oplus x_1.y_0$ as output

P_0 has $(x_0.y_0 \oplus r_0 \oplus (r_1 \oplus y_0.x_1))$ and P_2 has $((r_0 \oplus x_0.y_1) \oplus r_1 \oplus x_1.y_1)$. They share these values among themselves and sum them up to obtain $x.y$.
$x_0.y_0 \oplus r_0 \oplus (r_1 \oplus y_0.x_1) \oplus (r_0 \oplus x_0.y_1) \oplus r_1 \oplus x_1.y_1) = x.y = z$

This scheme can be extended to multiparty where each secret must be (n, n) Shamir secret shared. The AND gate computations has to be done for each pair of parties using 2 OTs. We will see the n party scenario in the next lecture.

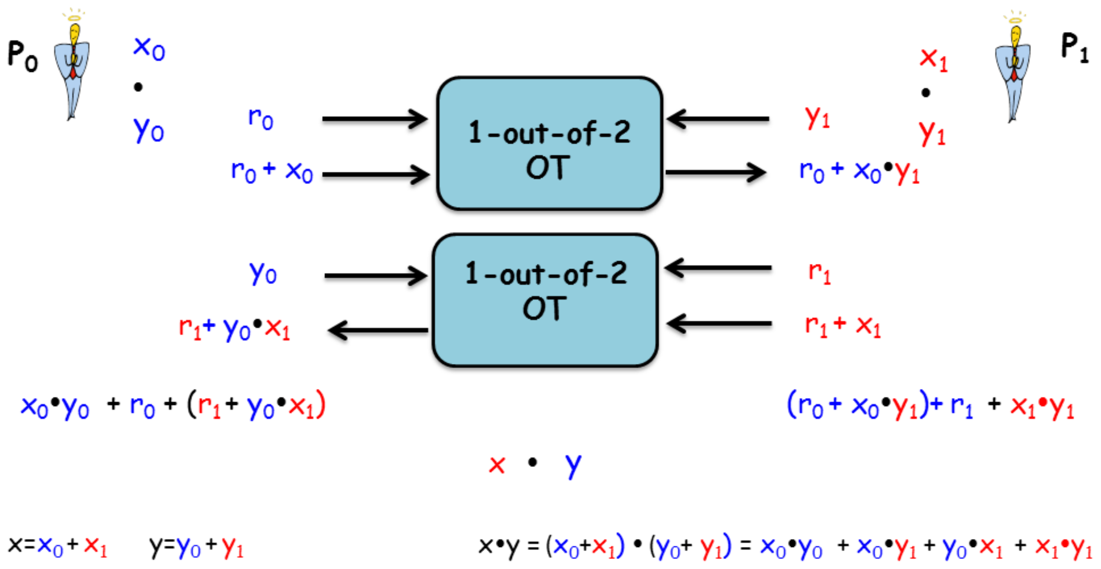


Figure 6: GMW protocol - AND Gate Evaluations

3.1 Security Proof of 2 party computations

The GMW protocol involves interaction between the parties only in case of AND gate evaluation. The other two gates can be computed locally. In the security proof we will consider the case where P_1 is the adversary A on the receiver side for the 1^{st} OT in the AND gate evaluation only. It is same for P_0 too. The 2 OTs in 1 AND gate evaluation are similar so we consider only 1 for security analysis.
During the AND gate evaluation, the real world view of P_1 in the first OT (as seen in Fig 6) is: $View_1^{Real} = (r_0 \oplus x_0 y_1, x_1, y_1, r_1, x.y)$

In the ideal world we have two simulators - the OT simulator (SIM_o) and the simulator (SIM_h) which simulates the view of P_0. The scenario is depicted in the Fig 7 The interactions between A and SIM_h happen through the SIM_o as they are using the

OT for interaction. SIM_h has the inputs and outputs of A. A sends y_1 as his choice bit to SIM_o. The SIM_o receives it and forwards it to the SIM_h. The SIM_h already knows the choice bit of A and the output he should get from the OT. So he provides two inputs m_{y_1} and m_{1-y_1} such that:
$m_{y_1} = r_0 \oplus x_0y_1$ and $m_{1-y_1} = r'$, where r' is some random number. A receives $r_0 \oplus x_0y_1$ as m_{y_1} from the SIM_o. Now at the end of the two OTs, SIM_h can just share a number r'' with A. r'' is defined such that when A XORs it with its inputs then he receives the correct output.
$r'' = (r_0 \oplus x_0y_1) \oplus r_1 \oplus x_1y_1$
$View_1^{Ideal} = (r_0 \oplus x_0y_1, x_1, y_1, r_1, x.y)$
In real world A cannot know the values of shares of P_0 from the $x.y$ only. So A cannot distinguish between the ideal and real world. This happens only if the OT is secure. If the OT is insecure then A will know m_{1-y_1} (from the OT interactions) and from there he can distinguish between the two worlds, as he can compute the shares of the honest parties and reconstruct the actual values of x and y. Thus in the presence of a secure OT, the GMW protocol is secure.

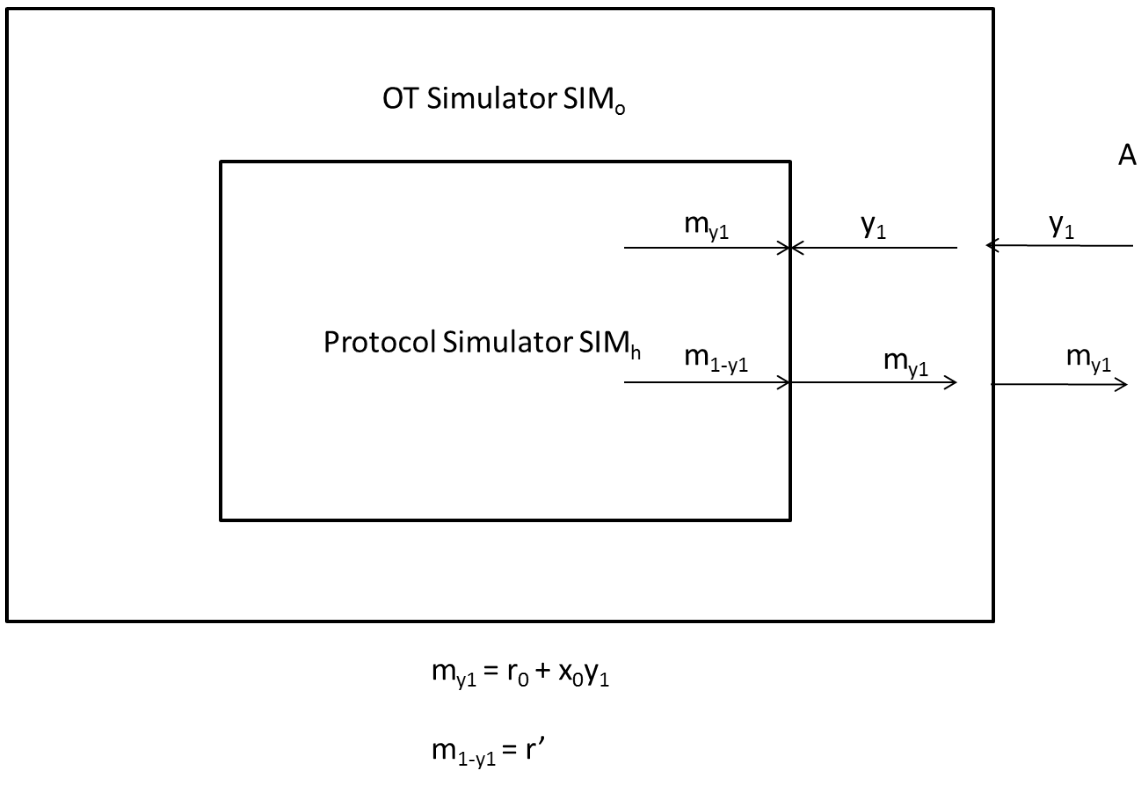


Figure 7: GMW protocol - AND Gate Evaluations

References

[1] Jonathan Katz *https://www.cs.umd.edu/ jkatz/gradcrypto2/f13/lecture3.pdf* Lecture notes.

[2] Arpita Patra *http://drona.csa.iisc.ernet.in/ arpita/SecureComputation15.html* Lecture notes.

[3] Oded Goldreich, Silvio Micali, Avi Wigderson, *How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority.* STOC 1987: 218-229