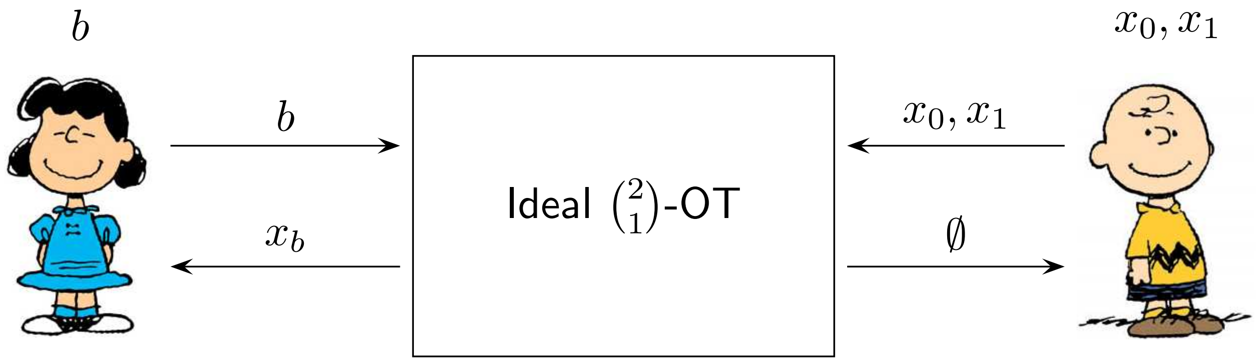
MTAT.07.003 Cryptology II

Oblivious Transfer

Sven Laur
University of Tartu

Ideal implementation



The protocol is always carried out between a client $\mathcal{P}_1$ and a sender $\mathcal{P}_2$.

- ▷ The server $\mathcal{P}_2$ has a database of two elements $x_0, x_1 \in \mathcal{M}$.
- ▷ The client $\mathcal{P}_1$ can fetch either x_0 or x_1 so that the server $\mathcal{P}_2$ cannot detect which element is fetched.
- ▷ The client should not learn anything more than x_b. Moreover, the client should be always aware of his or her choice b.

How to handle large databases?

Theorem. 1-out-of-2^ℓ oblivious transfer protocol for k-bit strings can be implemented using 1-out-of-2 oblivious transfer protocol for $2^\ell \cdot k$-bit strings.

Simplified proof

To encode $x_{00}, \ldots, x_{11}$, generate uniformly matrices Y and Z such that

$$\begin{array}{|c|c|}\hline x_{00} & x_{01} \\ \hline x_{10} & x_{11} \\ \hline \end{array} = \begin{array}{|c|c|}\hline {\color{red}y_{00}} & {\color{blue}y_{01}} \\ \hline {\color{red}y_{10}} & {\color{blue}y_{11}} \\ \hline \end{array} \oplus \begin{array}{|c|c|}\hline {\color{red}z_{00}} & {\color{red}z_{01}} \\ \hline {\color{blue}z_{10}} & {\color{blue}z_{11}} \\ \hline \end{array}$$

Next the client uses 1-out-of-2 oblivious transfer twice.

▷ First, the client must fetch the correct column of Y.

▷ Second, the client must fetch the correct row of Z.

Even a malicious client can learn only a single entry x_{ab} and he or she must be aware of the location ab.

Solution to the millionaires problem

Let $w_1, w_2 \in \{1, \ldots, n\}$ be the total wealth of two millionaires. Then one of them can find out who is richer and nothing more with the help of oblivious transfer protocol. The construction was first published by Yao (1982).

- ▷ The first millionaire creates an n-element table of possible answers

1	2	$\cdots$	n
$w_1 > 1$	$w_1 > 2$	$\cdots$	$w_1 > n$

- ▷ The second millionaire fetches the w_2th entry from the table and thus learns the value $w_1 > w_2$.
- ▷ The protocol is secure only if the first millionaire behaves semi-honestly.

This construction can be generalised for all functions with small input range.

Multiplication $\Leftrightarrow$ Oblivious transfer

Theorem. Given an ideal multiplication protocol, we can implement 1-out-of-2 oblivious transfer. Given an ideal 1-out-of-2 oblivious transfer protocol we can implement multiplication over $\mathbb{Z}_2$ in the semihonest model.

Clarification

- ▷ Observe that $x_b = (1-b)x_0 + bx_1$ and thus any multiplication protocol that provides shares is sufficient to implement oblivious transfer.
- ▷ Oblivious transfer is sufficient to implement multiplication, since the sender can use columns of the multiplication table as the input.

Kilian proved in 1988 that zero-knowledge proofs and commitments can be constructed using only oblivious transfer protocol. Hence, we can use commitment and zero knowledge proofs to eliminate malicious behaviour.

Homomorphic Oblivious Transfer

Homomorphic encryption

A public key cryptosystem $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is an additively homomorphic cryptosystem if for any two message $m_1, m_2 \in \mathcal{M}$ the distributions

$$\mathsf{Enc}_{\mathsf{pk}}(m_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2) \equiv \mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2)$$

coincide even if we fix a ciphertext $\mathsf{Enc}_{\mathsf{pk}}(m_1)$.

Multiplying a ciphertext $\mathsf{Enc}_{\mathsf{pk}}(m)$ with a newly generated $\mathsf{Enc}_{\mathsf{pk}}(0)$ completely destroys all extra information besides the value m.

We can compute also crypto-compute multiplication

$$\mathsf{Enc}_{\mathsf{pk}}(m_1)^{m_2} \cdot \mathsf{Enc}_{\mathsf{pk}}(0) \equiv \mathsf{Enc}_{\mathsf{pk}}(m_1 \cdot m_2) \enspace .$$

Famous examples

The Goldwasser-Micali cryptosystem is additively homomorphic over $\mathbb{Z}_2$.

The lifted ElGamal cryptosystem is additively homomorphic over $\mathbb{Z}_p$

$$\overline{\mathsf{Enc}}_{\mathsf{pk}}(m) = \mathsf{Enc}_{\mathsf{pk}}(g^m) = (g^r, g^m y^r)$$

$$\overline{\mathsf{Dec}}_{\mathsf{sk}}(c_1, c_2) = \log_g[\mathsf{Dec}(c_1, c_2)] = \log_g \left[\frac{c_2}{c_1^x}\right]$$

For obvious reason, the decryption rule $\overline{\mathsf{Dec}}_{\mathsf{sk}}(\cdot)$ can be efficiently computed for few ciphertexts or otherwise the cryptosystem would not be secure.

The Paillier cryptosystem uses lifting with together with a trapdoor that allows us to efficiently compute discrete logarithms. The corresponding message space is $\mathbb{Z}_n$ where n is RSA modulus.

Aiello-Ishai-Reingold oblivious transfer

b | x_0, x_1

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$
$c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(b)$

$\xrightarrow{\mathsf{pk}, c}$

$\xleftarrow{d_0, d_1}$

Halt if $c \notin \mathcal{C}$
$r_0, r_1 \xleftarrow[u]{} \mathcal{M}$
$d_0 \leftarrow c^{r_0} \cdot \mathsf{Enc}_{\mathsf{pk}}(x_0)$
$d_1 \leftarrow \left(\frac{c}{\mathsf{Enc}_{\mathsf{pk}}(1)}\right)^{r_1} \cdot \mathsf{Enc}_{\mathsf{pk}}(x_1)$

$m_b \leftarrow \mathsf{Dec}_{\mathsf{sk}}(d_b)$

If $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an additively homomorphic cryptosystem then

$$d_0 \equiv \mathsf{Enc}_{\mathsf{pk}}(b)^{r_0} \cdot \mathsf{Enc}_{\mathsf{pk}}(x_0) \equiv \mathsf{Enc}_{\mathsf{pk}}(x_0 + br_0) \enspace ,$$

$$d_1 \equiv \mathsf{Enc}_{\mathsf{pk}}(b-1)^{r_1} \cdot \mathsf{Enc}_{\mathsf{pk}}(x_1) \equiv \mathsf{Enc}_{\mathsf{pk}}\bigl(x_1 + (b-1)r_1\bigr) \enspace .$$

If the message space has prime order then br_0 has uniform distribution if $b \neq 0$ and $(b-1)r_1$ has uniform distribution if $b \neq 1$.

Security in the semi-honest model

Lemma 1. If the cryptosystem is additively homomorphic over $\mathbb{Z}_p$, then for any t-time *semi-honestly* corrupted receiver $\mathcal{P}_1^*$ there exists $t + \mathrm{O}(1)$-time ideal world adversary $\mathcal{P}_1^\circ$ such that the joint output distributions are identical in the real and ideal world.

Proof

- ▷ For fixed value b, the messages received by $\mathcal{P}_1^*$ have the following distribution: $d_b = \mathsf{Enc}_{\mathsf{pk}}(x_b)$ and $d_{b-1} = \mathsf{Enc}_{\mathsf{pk}}(m)$ where $m \leftarrow \mathcal{M}$.
- ▷ Given x_b from the trusted third party, we can perfectly simulate the reply in the real world.
- ▷ Since the output of $\mathcal{P}_2$ is $\perp$ in both worlds the joint output distribution coincides in both worlds.

□

Security in the semi-honest model

Lemma 2. If the cryptosystem is (t, ε)-IND-CPA secure, then for any τ-time *semi-honestly* corrupted sender $\mathcal{P}_2^*$ there exists $(\tau + \mathrm{O}(1))$-time ideal world adversary $\mathcal{P}_2^\circ$ such that the joint output distributions in the real and ideal world are $(t - \tau, \varepsilon)$-indistinguishable.

Proof

- ▷ The sender receives an encryption of b that we cannot simulate, since the trusted third party sends only $\emptyset$ to $\mathcal{P}_2^\circ$.
- ▷ However, we can replace c with $\mathsf{Enc}_{\mathsf{pk}}(0)$. Since $\mathcal{P}_1$ outputs m_b in both worlds then the output distributions must be $(t - \tau, \varepsilon)$-indistinguishable.
- ▷ Otherwise, we can construct a new adversary $\mathcal{A}$ from the participant $\mathcal{P}_2^\circ$ and the output distinguisher $\mathcal{B}$ that wins the IND-CPA game.

□

Interpretation of the results

Semi-honest receiver can carry out only the attacks that are possible against ideal implementation. The only benefit the receiver may gain in the real world is a marginal $\mathrm{O}(1)$ speed-up compared to the ideal world.

Let us consider a specific security goal. Then any of those can be formalised as a predicate $\mathcal{B}(\cdot)$ that indicates whether $\mathcal{P}_2^*$ was successful or not.

Lemma 2 indicates that is we consider specific $(t-\tau)$-*time* security goals $\mathcal{B}(\cdot)$, then for any τ-time semi-honest sender $\mathcal{P}_2^*$

$$\Pr\left[\mathcal{P}_2^* \text{ wins}\right] \leq \Pr\left[\mathcal{P}_2^\circ \text{ wins}\right] + \varepsilon$$

where $\mathcal{P}_2^\circ$ is $(\tau + \mathrm{O}(1))$-time adversary. In other words, $\mathcal{P}_2^*$ can achieve only marginal increase ε in success and a marginal $\mathrm{O}(1)$ speed-up.

Security against malicious receivers

Lemma 3. If the cryptosystem is additively homomorphic over $\mathbb{Z}_p$ and validity of the public key can be tested, then for any t-time *maliciously* corrupted receiver $\mathcal{P}_1^*$ there exists *unbounded* ideal world adversary $\mathcal{P}_1^\circ$ such that the joint output distributions are identical in the real and ideal world.

Proof

- ▷ Given a valid public key pk we can always find the corresponding secret key by looking through all valid $(\mathsf{pk}, \mathsf{sk})$ pairs.
- ▷ Hence, we can decrypt c and find out the true input of $\mathcal{P}_1^*$.
- ▷ If $b \notin \{0, 1\}$ then the received messages d_0 and d_1 are both random encryptions. Thus we can always perfectly simulate the replies.
- ▷ Other steps in the proof are analogous.

Interpretation of the results

Lemma 3 indicates that for each real world attack there is a matching ideal world attack. Hence, the adversary can learn nothing that cannot be computed form the intended output m_b.

However, the participation in the real world protocol might give a huge computational speedup compared to the ideal world.

Hence, participation in the protocol might help $\mathcal{P}_1^*$ to compute intractable functions from m_b. For example, if m_b is an encryption, then the protocol might reveal the underlying message.

How to achieve tight security guarantees?

If the receiver proves in zero-knowledge that he knows the secret key sk that corresponds to pk then the possible speedup becomes marginal.

- ▷ We can extract secret key by rewinding $\mathrm{Z}\textsc{k}\mathrm{P}\textsc{ok}_{\mathsf{pk}}\left[\exists \mathsf{sk} : (\mathsf{sk}, \mathsf{pk}) \in \mathsf{Gen}\right]$.
- ▷ The simulation becomes efficient if we learn the secret key sk.

If the sender is assumed to be semi-honest and the protocol uses the ElGamal encryption, then we can use the Schnorr protocol.

To handle malicious senders, we must convert the corresponding sigma protocol $\textsc{pok}_y\left[\exists x : g^x = y\right]$ to zero-knowledge proof of knowledge.

$$\boxed{\mathrm{Z}\textsc{k}\mathrm{P}\textsc{ok}_y\left[\exists x : g^x = y\right] = \textsc{pok}_y\left[\exists x : g^x = y\right] + \text{coin flipping protocol}}$$

Fortunately, we can reuse the same key in many protocol instances.

Security against malicious senders

To handle a malicious sender, we must extract x_0 and x_1 from $\mathcal{P}_2^*$.

▷ We can add zero-knowledge proofs of knowledge

$$\textsc{ZkPok}\left[d_0(x_0, x_1) \text{ and } d_1(x_0, x_1) \text{ are correctly formed}\right]$$

and then we can construct the necessary simulator.

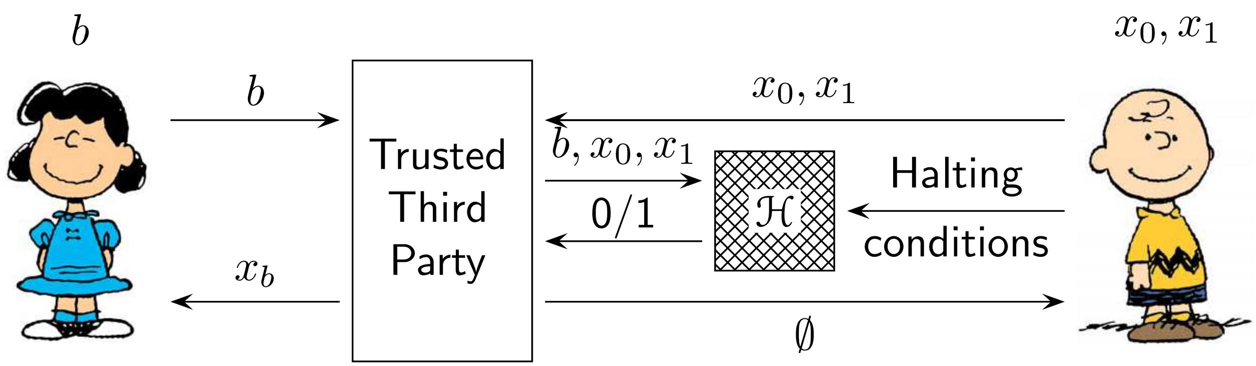
▷ One possibility is to commit x_0 and x_1 and then execute

$$\textsc{ZkPok}\left[\text{commitments are properly formed}\right]$$

and then continue with the certified computation protocol.

As a result, we get a protocol with enormous computational overhead.

Output consistency



If the sender first commits pairs $(0, x_0)$ and $(1, x_1)$ and the oblivious transfer protocol is used to reveal the corresponding decommitment strings, then the malicious sender cannot alter the outputs without getting caught.

$\Rightarrow$ The sender can still cause selective halting.

$\Rightarrow$ Cheating behaviour is detectable with high probability.

$\Rightarrow$ Public complaints reveal information about receiver inputs.

Complete security vs output consistency

Both security levels reveal cheating with high probability:

- ▷ Complete security makes all deviations from the protocol that *could alter* the outcome for some receiver input detectable.
- ▷ Output consistency makes all deviations from the protocol that *alter* the output for this particular receiver input detectable.

Complete security has *large* computational overhead.

- ▷ Certified computations require extensive amount of extra steps.

Output-consistent computations have *moderate* computational overhead.

- ▷ Commitments are relatively easy to compute.
- ▷ Selective halting can cause privacy issues.

Bellare-Micali Protocol

Vanilla protocol

b

$(\mathsf{pk}_b, \mathsf{sk}_b) \leftarrow \mathsf{Gen}$

$\mathsf{pk}_{1-b} \leftarrow \mathsf{pk} \cdot \mathsf{pk}_b^{-1}$

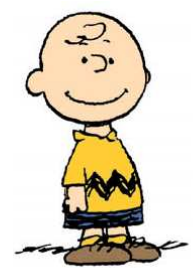
$m_b \leftarrow \mathsf{Dec}_{\mathsf{sk}_b}(d_b)$

$$\xleftarrow{\quad \mathsf{pk} \quad}$$

$$\xrightarrow{\quad \mathsf{pk}_0, \mathsf{pk}_1 \quad}$$

$$\xleftarrow{\quad d_0, d_1 \quad}$$

x_0, x_1



$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$

Halt if $\mathsf{pk}_0 \cdot \mathsf{pk}_1 \neq \mathsf{pk}$

$d_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}_0}(x_0)$

$d_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}_1}(x_1)$

The protocol works under the assumption that all possible public keys form a group and the distribution of public keys is uniform.

- ▷ The ElGamal cryptosystem has such a public key space.
- ▷ Since the public key pk_{1-b} is with correct distribution the corresponding ciphertext d_{1-b} is undecipherable.
- ▷ The protocol can tolerate unbounded senders.

Security in the semi-honest model

Lemma. If the public keys are uniformly distributed over some group, then for any t-time *semi-honestly* corrupted sender $\mathcal{P}_2^*$ there exists $t + \mathrm{O}(1)$-time ideal world adversary $\mathcal{P}_2^\circ$ such that the joint output distributions are identical in the real and ideal world.

Proof

- ▷ In the simulator, we can first compute public keys $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Gen}$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Gen}$ and then set the final key $\mathsf{pk} \leftarrow \mathsf{pk}_0 \cdot \mathsf{pk}_1$.
- ▷ As a result, we can decrypt d_0 and d_1 and send the corresponding messages x_0 and x_1 to trusted third party.
- ▷ The simulation is perfect.

Security in the semi-honest model

Lemma. If the public keys are uniformly distributed over some group and the cryptosystem is (t, ε)-IND-CPA secure, then for any τ-time *semi-honestly* corrupted receiver $\mathcal{P}_1^*$ there exists $\tau + \mathrm{O}(1)$-time ideal world adversary $\mathcal{P}_1^\circ$ such that the joint output distributions in the real and ideal world are $(t - \tau, \varepsilon)$-indistinguishable.

Proof

- ▷ We can simulate the reply d_b with $\mathsf{Enc}_{\mathsf{pk}_b}(x_b)$ and $d_{1-b} \leftarrow \mathsf{Enc}_{\mathsf{pk}_{1-b}}(0)$.
- ▷ As pk_{1-b} can be taken from the IND-CPA game and the message pk defined as $\mathsf{pk} \leftarrow \mathsf{pk}_0 \cdot \mathsf{pk}_1$, any distinguisher $\mathcal{B}$ together with $\mathcal{P}_1^*$ form a successful IND-CPA adversary.

□

How to strengthen the protocol?

Security against malicious receivers:

▷ Sigma protocol that proves that $\mathcal{P}_1$ knows one secret key sk_b is sufficient.

▷ If the protocol uses the ElGamal encryption, then we can use Schnorr protocol to prove $\mathrm{POK}_{y_0,y_1}\left[\exists x : g^x = y_0 \vee g^x = y_1\right]$.

Security against malicious senders:

▷ If we do not care about efficiency, then we can always generate $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Gen}$ and find sk_1 by exhaustive search.

▷ To get efficient simulation, we can always use certified computations. For instance, $\mathcal{P}_1$ and $\mathcal{P}_2$ can jointly generate random coins for pk so that only $\mathcal{P}_2$ learns them but he or she can certify the correctness of pk.