Universally Composable Oblivious Transfer from One-Round Key-Exchange

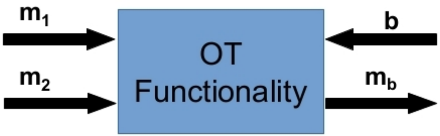
Manuel Goulão

manuel.goulao@tecnico.ulisboa.pt

(work done with Pedro Branco, Jintai Ding, and Paulo Mateus)

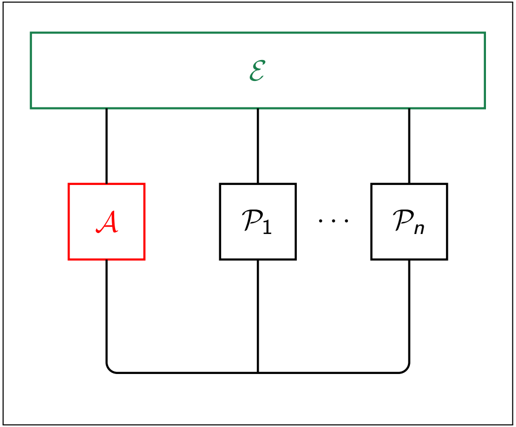
11 February 2019

Oblivious Transfer

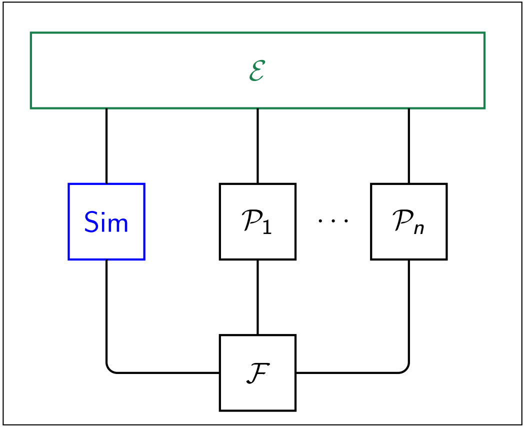


Oblivious transfer (OT) is an important primitive in cryptography as a building block to construct secure multiparty computation.

Universal Composability



Real world execution



Ideal world execution

One-Round Key-Exchange

A-B ORKE structure

Alice		**Bob**
$r_A \leftarrow_{\$} \{0,1\}^{\kappa}$		$r_B \leftarrow_{\$} \{0,1\}^{\kappa}$
$(pk_A, sk_A) \leftarrow Gen(1^{\kappa}, r_A)$		$(pk_B, sk_B) \leftarrow Gen(1^{\kappa}, r_B)$
$m_A \leftarrow Msg^A(r_A, sk_A, pk_B)$	$\xrightarrow{\quad m_A \quad}$	
	$\xleftarrow{\quad m_B \quad}$	$m_B \leftarrow Msg^B(r_B, sk_B, pk_A, m_A)$
$k \leftarrow Key(r_A, sk_A, pk_B, m_B)$		$k \leftarrow Key(r_B, sk_B, pk_A, m_A)$

One-Round Key-Exchange

Properties (intuition)

- *Non-redundant message*: All *parts* of the message must be used to construct the key. I.e., change one *part* and the key changes completely
- *Message indistinguishability*: Given a group action on the space of the messages of the Alice, its codomain must be indistinguishable from the messages of Alice
- *Key indistinguishability*: The key obtained by Bob either using the message from Alice or a random value must be indistinguishable

Our framework

Sender(x_0, x_1)		**Receiver**$(b \in \{0,1\})$
$(sk_S, pk_S) \leftarrow Gen(1^\kappa)$		$(sk_R, pk_R) \leftarrow Gen(1^\kappa)$
		$h \leftarrow \mathbf{H}(t)$
		$m_R^b \leftarrow Msg^R(r_R, sk_R, pk_S)$
$h' \leftarrow \mathbf{H}(t)$	$\xleftarrow{\quad (m_R^0, t) \quad}$	If $b = 1, m_R^0 \leftarrow \psi(m_R^1, h^{-1})$
$m_R^1 \leftarrow \psi(m_R^0, h')$		
$m_S^0 \leftarrow Msg^S(r_S, sk_S, pk_R, m_R^0)$		
$m_S^1 \leftarrow Msg^S(r_S, sk_S, pk_R, m_R^1)$		
$k^0 \leftarrow Key(r_S, sk_S, pk_R, m_R^0)$		
$k^1 \leftarrow Key(r_S, sk_S, pk_R, m_R^1)$		
$chall \leftarrow \mathcal{C}(m_S^0, m_S^1, \mathbf{H}(k^0), \mathbf{H}(k^1))$	$\xrightarrow{\quad (m_S^0, m_S^1, chall) \quad}$	
		$k \leftarrow Key(r_R, sk_R, pk_S, m_S^1)$
	$\xleftarrow{\quad resp \quad}$	$resp \leftarrow \mathcal{R}()$
if *resp* is correct:	$\xrightarrow{Enc(k^0, x_0), Enc(k^1, x_1)}$	$x_b \leftarrow Dec(k, Enc(k^b, x_b))$

Our framework

Extension to $\binom{n}{1}$-OT

- Sample and send $n - 1$ random values t_i
- Apply group action ψ to m_R^i and output of ROM by t_i. Leave $m_R^b \leftarrow Msg^B()$
  - Are all indistinguishable by *message indistinguishability* property
- Set the challenge accordingly
- In the end, R will only have the key k^b

Security

Receiver

- **First message:** *Message indistinguishability* guarantees the Sender does not know which m_{R}^0 or m_{R}^1 is the message and which is a random string. Thus, it does not know which message R uses to compute its key
- **Second message:** The receiver can compute the response to the challenge regardless of its input, there is no information about the input b

Security

Sender

- **First message:** The security of the KE guarantees the Receiver is not able to derive a key from m_S^{1-b}. And all information from the challenge is output by the ROM, i.e. not correlated with the (other) key
- **Second message:** Security of the SKE assures the impossibility to get the other message without the corresponding key

Universal Composability

Simulating a corrupted receiver

1. The simulator simulates the random oracles $\mathbf{H}_1$, $\mathbf{H}_2$, $\mathbf{H}_3$ and $\mathbf{H}_4$ as usual.
2. Upon receiving $(\mathrm{sid}, t, m_\mathsf{R}^0)$ from the adversary $\mathcal{A}(\mathsf{R})$, the simulator Sim:
   - Follows the protocol and sends $(\mathrm{sid}, m_\mathsf{S}^0, m_\mathsf{S}^1, a_0, a_1, u_0, u_1)$ to $\mathcal{A}$;
   - Sets $b \leftarrow \perp$. When $k_\mathsf{S}^{\bar{b}}$ is asked to the random oracle $\mathbf{H}_2$, it sets $b \leftarrow \bar{b}$;
   - Aborts, if w_{1-b} is asked to the random oracle $\mathbf{H}_3$ before w_b or if k_S^{1-b} is asked to $\mathbf{H}_2$.
3. Upon receiving (sid, ch') from the adversary $\mathcal{A}(\mathsf{R})$, the simulator Sim:
   - Aborts, if $ch \neq ch'$;
   - If $b = \perp$, sets $b \leftarrow_\$ \{0, 1\}$;
   - Sends (sid, b) to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$.
4. Upon receiving (sid, M_b) from $\mathcal{F}_{\mathsf{OT}}$, the simulator Sim:
   - Encrypts $c_b \leftarrow \mathbf{Enc}(k_\mathsf{S}^b, M_b)$ and $c_{1-b} \leftarrow \mathbf{Enc}(k_\mathsf{S}^{1-b}, 0^\lambda)$;
   - Sends (sid, c_0, c_1) to $\mathcal{A}(\mathsf{R})$;

1. Before activating the adversary, the simulator Sim:
   - Chooses $r_R^0 \leftarrow_\$ \{0,1\}^\kappa$ and $r_R^1 \leftarrow_\$ \{0,1\}^\kappa$;
   - Computes $m_R^0 \leftarrow Msg(r_R^0, \mathsf{sk_R}, \mathsf{pk_S})$ and $m_R^1 \leftarrow Msg(r_R^1, \mathsf{sk_R}, \mathsf{pk_S})$.
2. Upon activating the adversary, the simulator Sim sends (sid, t, m_R^0):
   - Simulates $\mathbf{H}_2$, $\mathbf{H}_3$ and $\mathbf{H}_4$ as $\mathcal{F}_{RO}$;
   - When the adversary queries $\mathbf{H}_1$ with (sid, t), answers h such that $m_R^1 = \psi(m_R^0, h)$.
3. Upon receiving $(\mathsf{sid}, m_S^0, m_S^1, a_0, a_1, u_0, u_1)$ from $\mathcal{A}$, the simulator Sim:
   - Computes $k_R^0 \leftarrow Key(\mathsf{sk_R}, \mathsf{pk_S}, r_R^0, m_S^0)$ and $k_R^1 \leftarrow Key(\mathsf{sk_R}, \mathsf{pk_S}, r_R^1, m_S^1)$;
   - Computes ch' as the honest receiver;
   - Sends (sid, ch') to $\mathcal{A}$.
4. Upon receiving (sid, c_0, c_1) from $\mathcal{A}$, the simulator Sim:
   - Computes $M_0 \leftarrow \mathbf{Dec}(k_R^0, c_0)$ and $M_1 \leftarrow \mathbf{Dec}(k_R^1, c_1)$;
   - Sends (sid, M_0, M_1) to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$.

Efficiency *vs* other frameworks

- Four communication rounds
- One iteration takes $\mathcal{O}(\alpha + \lambda + \kappa)$
  - κ: Security parameter
  - α: Size of messages of the KE
  - λ: Size of the ciphertexts of the SKE
- Only simple computations required
- Few and weak imposed conditions
- First UC framework to be instantiated with RLWE and SIDH

Examples

Diffie-Hellman

Key exchange:

- $(sk = x \in \mathbb{Z}_p^*, pk = g \in \mathbb{Z}_p) \leftarrow Gen(1^\kappa)$
- $g^x \leftarrow Msg(r, x, g)$
- $g^{xy} \leftarrow Key(r, g, x, g^y)$

Required properties:

- Group action: consider $\psi : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*, \psi(y, h) = y * h \bmod p$
- Message indistinguishability: g is a generator, so the output by Msg or ψ are both random
- Key indistinguishability: Keys are of the form g^{xy}, which is a random element in $\mathbb{Z}_p^*$

Examples

RLWE-KE

Key exchange:

- $(s, (a, as + e)) \leftarrow Gen(1^\kappa), \quad s \leftarrow_\$ \chi_\alpha, e \leftarrow_\$ \chi_\alpha, a \leftarrow_\$ R_q = \mathbb{Z}_q[x]/\langle(x^n + 1)\rangle$
- $pk_A \leftarrow Msg^A(r, s, as + e)$
- $(pk_B, w) \leftarrow Msg^B(r, sk_B, pk_B, pk_A)$
- $k \leftarrow Key(r, sk_i, pk_j, m_j)$

Required properties:

- Group action: consider $\psi : R_q \times (R_q, +) \to R_q, \psi(y, h) = y + h$
- Message indistinguishability: message is an RLWE sample. distinguishing would break the RLWE assumption
- Key indistinguishability: From security of KE, to distinguish K from random reduces to deciding the RLWE assumption

Bibliography

For more info about:

- **OT:** Kilian, J.: *Founding cryptography on oblivious transfer.* In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 20–31. STOC '88, ACM, New York, NY, USA (1988), http://doi.acm.org/10.1145/62212. 62215
- **UC:** Canetti, R.: *Universally composable security: A new paradigm for cryptographic protocols.* In: Proceedings of the 42Nd IEEE Symposium on Foundations of Com- puter Science. pp. 136–. FOCS '01, IEEE Computer Society, Washington, DC, USA (2001)
- **ORKE:** Bergsma, F., Jager, T., Schwenk, J.: *One-round key exchange with strong security: An efficient and generic construction in the standard model.* In: Katz, J. (ed.) Public-Key Cryptography – PKC 2015. pp. 477–494. Springer Berlin Heidelberg, Berlin, Heide