ETH Zurich, Department of Computer Science
SS 2019

Prof. Ueli Maurer
Dr. Martin Hirt
Chen-Da Liu Zhang

Cryptographic Protocols
Solution to Exercise 8

8.1 Types of Oblivious Transfer

a) The reduction is straight-forward: the sender sends $(b_0, b_1, 0, \dots, 0)$ via 1-out-of-k OT, and the receiver picks $c \in \{0,1\}$.

b) Alice and Bob use the following protocol:

$$\begin{array}{lcl}
\textbf{Alice} & & \textbf{Bob} \\
r_1 \in_R \{0,1\},\ e_1 := b_1 & \xrightarrow{[e_1 \mid r_1]_{\text{1-2-OT}}} & \text{if } c = 1, \text{ pick } e_1, \text{ else } r_1 \\
r_2 \in_R \{0,1\},\ e_2 := b_2 \oplus r_1 & \xrightarrow{[e_2 \mid r_2]_{\text{1-2-OT}}} & \text{if } c = 2, \text{ pick } e_2, \text{ else } r_2 \\
r_3 \in_R \{0,1\},\ e_3 := b_3 \oplus r_1 \oplus r_2 & \xrightarrow{[e_3 \mid r_3]_{\text{1-2-OT}}} & \text{if } c = 3, \text{ pick } e_3, \text{ else } r_3 \\
\vdots & \vdots & \vdots \\
e_k := b_k \oplus r_1 \oplus \dots \oplus r_{k-1} & \xrightarrow{e_k} & b := e_c \oplus r_1 \oplus \dots \oplus r_{c-1}
\end{array}$$

Alice trivially does not learn any information about Bob's choice $c \in \{1, \dots, k\}$. If Bob wishes to learn bit b_c, he needs to know all preceding one-time pads $r_1, \dots, r_{c-1}$ as well as the value e_c. Hence, he cannot choose any of the values $e_1, \dots, e_{c-1}$, and he has to choose the bit e_c. However, in that case he does not learn r_c and learns no information about b_i for $i > c$. Hence, even when Bob does not follow the protocol, he learns at most one of the k bits.

c) Alice and Bob use the following protocol:

$$\begin{array}{lcl}
\textbf{Alice} & & \textbf{Bob} \\
i \in_R \{0,1\} & & j \in_R \{0,1\} \\
b_i := b,\ b_{1-i} := 0 & \xrightarrow{[b_0 \mid b_1]_{\text{1-2-OT}}} & \text{pick } b_j \\
 & \xrightarrow{i} & \text{if } i = j, \text{ set } b := b_j, \\
 & & \text{else } b := \perp
\end{array}$$

Alice trivially does not learn any information about whether Bob receives the bit or not. Moreover, it is obvious that Bob receives the bit with probability $\frac{1}{2}$ and otherwise has no information about it.

d) Let κ be a security parameter. Alice and Bob use the following protocol:

Alice		**Bob**
$r_1, \ldots, r_\kappa \in_R \{0,1\}$	$\xrightarrow{\forall i : [r_i]_{\text{Rabin-OT}}}$	$\forall i$: receive $r'_i \in \{r_i, \perp\}$
		$T_c := \{i \mid r'_i \neq \perp\}$
$t_0 := \bigoplus_{i \in T_0} r_i$, $t_1 := \bigoplus_{i \in T_1} r_i$	$\xleftarrow{T_0, T_1}$	$T_{1-c} := \{i \mid r'_i = \perp\}$
$e_0 := b_0 \oplus t_0$, $e_1 := b_1 \oplus t_1$	$\xrightarrow{e_0, e_1}$	$t_c := \bigoplus_{i \in T_c} r_i$
		$b_c := e_c \oplus t_c$

Alice does not learn any information about Bob's choice $c \in \{1, \ldots, k\}$ since T_0 and T_1 do not reveal which instances of the underlying Rabin OT were successful. Furthermore, with probability $1 - 2^{-\kappa}$ there is at least one bit r_i the receiver does not learn, and, therefore, at least one of the one-time pads t_0 and t_1 is uniformly random. Therefore, except with probability $2^{-\kappa}$, the receiver learns at most one of the bits b_0 and b_1.

8.2 Multi-Party Computation with Oblivious Transfer

a) A possible generalization of the given protocol to the three-party case could be as follows: A computes the function table of $f(x, \cdot, \cdot)$ and sends it by OT to B, i.e., A and B invoke 1-out-of-m OST, where A inputs the following vectors t_i:

$$\begin{aligned}
t_1 &:= (f(x, y_1, z_1), f(x, y_1, z_2), \ldots, f(x, y_1, z_m)) \\
t_2 &:= (f(x, y_2, z_1), f(x, y_2, z_2), \ldots, f(x, y_2, z_m)) \\
&\vdots \\
t_{|\mathcal{Y}|} &:= (f(x, y_m, z_1), f(x, y_m, z_2), \ldots, f(x, y_m, z_m)).
\end{aligned}$$

B receives t_y, i.e., the function table of $f(x, y, \cdot)$ for an arbitrary y. Subsequently, B sends C the received function table via 1-out-of-m OST, where B inputs m values $f(x, y, z_1), f(x, y, z_2), \ldots, f(x, y, z_m)$, and C receives $f(x, y, z)$ for his input z. Finally, C sends $f(x, y, z)$ to A and B.

b) The above protocol is secure if either A or C are passively corrupted, but not against an adversary who corrupts B. This can be seen by the following example: Consider the function $f : \{0,1\}^3 \mapsto \{0,1\}$ with

$$f(x, y, z) = \begin{cases} 1 & \text{if } x = y = z \\ 0 & \text{otherwise.} \end{cases}$$

In the protocol from **a)**, B learns after the computation of f whether or not $x = y$. However, B should learn this information only when the function evaluates to 1. Hence, the protocol does not achieve the property that the players receive no more information in the execution of the protocol than what they can compute from the output of f.

c) The idea is that A encrypts the function table $f(x, y, \cdot)$ for each possible y using one-time pad encryption and sends the keys to C (but not to B).

More concretely, the improved protocol works as follows: For each $z \in \mathbb{Z}_m$, A chooses a value $r_z \in \mathbb{Z}_m$ uniformly at random (the one-time pad key) and sends it to C.

Subsequently, A sends B the following vector (where x is A's input) by 1-out-of-m OST:

$$\begin{aligned}
t_1 &:= (f(x,y_1,z_1)\oplus r_1, f(x,y_1,z_2)\oplus r_2,\ldots,f(x,y_1,z_m)\oplus r_m)\\
t_2 &:= (f(x,y_2,z_1)\oplus r_1, f(x,y_2,z_2)\oplus r_2,\ldots,f(x,y_2,z_m)\oplus r_m)\\
&\vdots\\
t_m &:= (f(x,y_m,z_1)\oplus r_1, f(x,y_m,z_2)\oplus r_2,\ldots,f(x,y_m,z_m)\oplus r_m).
\end{aligned}$$

That way, B can choose the row corresponding to his input y. Subsequently, B sends C the values $f(x,y,z)\oplus r_z$ (for all $z\in\mathcal{Z}$) via 1-out-of-m OST, and C chooses the value $f(x,y,z)\oplus r_z$ corresponding to his input z and computes $f(x,y,z)$ using the key r_z which he received from A in the first step. Finally, C sends $f(x,y,z)$ to A and B.

It is easily verified that the protocol is secure against a passive adversary B, as the function table of $f(x,y,\cdot)$ that B receives from A in the second step is completely blinded by the one-time pad encryption.

8.3 Trusted Party Operations

a) To generate a random secret value, the trusted party receives a random value r_i from each player P_i and computes $\sum_i r_i$.

b) Since the order of the multiplicative group of $\mathbb{F}$ is $p-1$, $x^{p-1}=1$, which implies that $x^{p-2}\cdot x=1$, we have that $x^{-1}=x^{p-2}$. Then, to compute the inverse x^{-1}, the trusted party can do $p-2$ consecutive multiplications. Note that when $x=0$, then the computed "inverse" equals 0. Using the square-and-multiply method, it is enough to compute $O(\log(p))$ multiplications.

c) The trusted party can generate a secret random value r. Then, using a single multiplication gate it computes $y:=x\cdot r$ and sends this value to each party P_i. Then, each party computes y^{-1} and sends it to the trusted party. Finally, the trusted party computes $r\cdot y^{-1}=r\cdot(x\cdot r)^{-1}=x^{-1}$. Observe that when $r=0$, the inverse is not defined. One can choose the size of the field large enough so that this happens with negligible probability.

When $x=0$, then the players obtain the value $y=0$. In this case, the players learn that the value that is shared is 0.

d) Let $c\in\{0,1\}$. To execute the "if"-statement, compute

$$z:=(1-c)\cdot x+c\cdot y.$$

For an arbitrary $c\in\mathbb{F}$, compute

$$z:=(1-c^{p-1})\cdot x+c^{p-1}\cdot y.$$

This results in the correct value z since $c^{p-1}=1$ if $c\neq 0$ and $c^{p-1}=0$ if $c=0$.