ETH Zurich, Department of Computer Science
SS 2021

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

Cryptographic Protocols
Solution to Exercise 7

7.1 Protocols and Specifications

a) Protocol 3 does not satisfy Specification 1, since in the protocol P_2 outputs $x_1 \wedge x_2$ and in the specification P_2 outputs x_1, which is different than $x_1 \wedge x_2$ in the case where $x_1 = 1$ and $x_2 = 0$.

Protocol 3 satisfies Specification 2, since the parties output the same in the protocol and in the specification.

b) *P_2 is semi-honest:* Protocol 3 is not secure if P_2 is passively corrupted. We need to argue that there is an adversary in Protocol 3 that achieves something, such that no adversary in the specification achieves the same.

We can see that in Protocol 3 P_2 learns the message x_1, which cannot always be computed from the input and output of P_2. Consider the case where party P_1 chooses x_1 uniformly at random. Furthermore, consider the case where $x_2 = 0$. Then, $x_1 \wedge x_2 = 0$, and the adversary in the specification has to guess x_1, in which it succeeds with probability at most $\frac{1}{2}$.

P_2 is malicious: The protocol is secure in the case where P_2 is actively corrupted. We argue that anything an adversary can do in Protocol 3, there is another adversary in the specification that achieves the same.

In the protocol execution, the adversary obtains the input x_1 of P_1, and then can output an arbitrary value from x_1 and x_2.

In the specification, P_1 sends x_1 to the trusted party. Here, the adversary corrupting P_2 sends 1 to the trusted party. Then, it receives $x_1 \wedge 1 = x_1$, and outputs the same as what the adversary in Protocol 3 outputs.

c) *Two passive corruptions:* If P_1 and P_2 are passively corrupted, the adversary in the specification knows x_1, x_2 and $x_1 \wedge x_2 \wedge x_3$. Hence, it can generate all messages that an adversary in Protocol 5 see (which consists of the messages x_1, $x_1 \wedge x_2$ and $x_1 \wedge x_2 \wedge x_3$). However, when P_1 and P_3 are passively corrupted, the adversary in the specification (who knows x_1, x_3 and $x_1 \wedge x_2 \wedge x_3$) cannot compute $x_1 \wedge x_2$.

If the adversary in Protocol 5 corrupts all players, the adversary in the specification can generate all messages of the protocol, and hence the protocol is secure.

Two active corruptions: If P_1 and P_2 are actively corrupted, the adversary in the specification knows x_1, x_2 and $x_1 \wedge x_2 \wedge x_3$. Hence, it can generate any message that an adversary sees in Protocol 5.

In the case where P_1 and P_3 are corrupted, the adversary in the specification can input 1 to the trusted party on behalf of P_1 and P_3, to learn the input of P_2. Hence, it can generate all messages that can be seen by any adversary of Protocol 5 without changing the output (P_2 has no output).

7.2 Types of Oblivious Transfer

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

b) Alice and Bob use the following protocol:

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

Alice trivially does not learn any information about Bob's choice $c \in \{1, \ldots, k\}$. If Bob wishes to learn bit b_c, he needs to know all preceding one-time pads $r_1, \ldots, r_{c-1}$ as well as the value e_c. Hence, he cannot choose any of the values $e_1, \ldots, e_{c-1}$, 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:

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

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 $r'_i \in \{r_i, \perp\}$
$t_0 := \bigoplus_{i \in T_0} r_i$, $t_1 := \bigoplus_{i \in T_1} r_i$	$\xleftarrow{T_0, T_1}$	$T_c := \{i \mid r'_i \neq \perp\}$ $T_{1-c} := \{i \mid r'_i = \perp\}$
$e_0 := b_0 \oplus t_0$, $e_1 := b_1 \oplus t_1$	$\xrightarrow{e_0, e_1}$	$t_c := \bigoplus_{i \in T_c} r_i$ $b_c := e_c \oplus t_c$

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.

7.3 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.