ETH Zurich, Department of Computer Science
SS 2019

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

Cryptographic Protocols
Solution to Exercise 7

7.1 Permuted Truth Tables

a) Peggy chooses a random permuted truth table for the $\wedge$-function and commits to its elements. Vic chooses a random challenge bit c and sends it to Peggy. If $c = 0$, then Peggy opens the whole table and Vic checks if it is a permuted $\wedge$-table. If $c = 1$, Peggy takes the blobs (d_1, d_2, d_3) from the row corresponding to the triple (b_1, b_2, b_3) and proves (using the ZK protocol for equality) that $\forall i \in \{1, 2, 3\}$ d_i and c_i are commitments of the same value.

Note that the commitments used in the above construction are of type B (i.e., perfectly binding). We show that the above protocol is a zero-knowledge proof of the statement "the committed values (b_1, b_2, b_3) corresponding to the commitments (c_1, c_2, c_3) satisfy the relation $b_1 \wedge b_2 = b_3$."

Completeness: Follows immediately from the completeness of the protocol for blob equality.

Soundness: Assume that $b_1 \wedge b_2 \neq b_3$. If Peggy commits to a valid permuted truth table in the first step, Peggy cannot answer the challenge $c = 1$ as there is no row in this table with with commitments corresponding to b_1, b_2, b_3. If Peggy commits to an invalid table, then she cannot answer the challenge $c = 0$, as the commitment is binding. Hence, the cheating probability of Peggy for each round is approximately $1/2$ (the "approximately" stems from the fact that, in case $c = 1$, Peggy might still be able, with some small probability, to cheat in the equality proof).

Zero-Knowledge: We prove the (computational) zero-knowledge property only informally. We need to show that there exists an efficient simulation S producing a transcript which is (computationally) indistinguishable from the transcript resulting from a real protocol execution between the prover P and (a possibly dishonest) verifier V'.

The simulator S can produce a transcript as follows: First, S computes a valid permuted truth table and commits to it. If V' sends the challenge $c = 0$, the simulator opens the committed table. If V' sends $c = 1$, S uses the simulator S' for the blob equality protocol to compute a transcript of a proof of equality for $c_i = d_i$ ($i = 1 \ldots 3$), where the d_i's are commitments corresponding to a randomly chosen row of the permuted truth table. Note that, by the computational hiding property of the commitments, the transcript produced by S' is computationally indistinguishable from the real interaction even if the d_i's are commitments to different values than those in the c_i's.

b) If Peggy knows the input to the circuit, then she can compute (by evaluating the circuit in a gate-by-gate manner) the bits on the wires. She commits to all those bits and sends the blobs to Vic. Subsequently, she uses the protocol from **a)** for each gate ($\neg$-gates are treated similarly to $\wedge$-gates) to prove that the committed values are consistent with the circuit. To convince Vic that the output of the circuit is in fact 1, Peggy and Vic use a fixed commitment of 1, i.e., a commitment that is hard-coded into the protocol.

c) In the BCC protocol from the lecture, when processing the circuit, Peggy blinds every wire using a random bit. In the protocol from **b)**, this is not necessary, but we need the additional zero-knowledge proofs of equality of committed values.

7.2 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 an adversary in Protocol 3 who chooses the input x_1 of party P_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).