ETH Zurich, Department of Computer Science
SS 2018

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

Cryptographic Protocols
Notes 8

Scribe: Sandro Coretti (modified by Chen-Da Liu Zhang)

About the notes: These notes serve as written reference for the topics not covered by the papers that are handed out during the lecture. The material contained therein is thus a *strict* subset of what is relevant for the final exam.

This week, the notes treat a generic MPC protocol secure against active adversaries as well as an instantiation thereof leading to cryptographic security against an (efficient) attacker corrupting up to $t < n/2$ players. Next week, we discuss a second instantiation that results in information-theoretic security.

8.1 Actively Secure Multi-Party Computation: Basic Idea

In this section we modify the protocol secure against passive adversaries from Section 7.2 to provide security against actively corrupted players. We will do so in a generic fashion and then provide two instantiations which lead to cryptographic and information-theoretic security, respectively.

There are three forms of active misbehavior: divulging secret information, not sending values, and sending incorrect values. In the following we will incrementally improve the passive protocol to be secure against each of these forms.

Divulging secret information. In our protocol, the adversary could, e.g., use non-random coefficients to share values, or he could send values he is not supposed to send. Clearly, the worst-case scenario is if up to t players publish all of their information. But since the adversary already knows this information and honest players must ignore it, this kind of active corruption does not pose a problem.

Not sending values. There are three points in the protocol at which players can refuse to send values:

1. While sharing a value, a corrupted dealer might not send a share to some of the players.
2. In the multiplication protocol, a corrupted player might not send his product share to some of the players.
3. During reconstruction a corrupted player might not send his share to some of the players.

Case 3 is easy to solve: As $n > 2t$, an honest player receives at least $n - t \geq t + 1$ shares, which suffices to reconstruct a polynomial of degree at most t.

In Case 1, a player who did not receive a share accuses the dealer publicly, using broadcast. Then, the accused must broadcast the corresponding share. If he refuses to comply, he is disqualified and a default value is assumed as his input. This can be done using a degree-0 Shamir sharing, i.e., a polynomial that evaluates to this default value at every point. Note that the publication of the share using broadcast does not constitute a problem as the value broadcast is already known to the adversary (either the dealer is corrupted, or the accusing player is corrupted and has already received the value from the dealer).

To deal with Case 2, one first proceeds as in Case 1, i.e., by accusation and broadcast. However, in this case, if the accused player refuses to broadcast the necessary value, the remaining players cannot just assume a default value since they need his product share d_i in the multiplication protocol.

There is a number of different ways to handle this:

- The entire protocol is repeated without the fallible player (decreasing both the number of total players and the number of cheating players by one. Since the fault occurred during a multiplication, no players have received information about the output yet.
- The factor shares a_i and b_i of the cheating player (and only those) are reconstructed. Then, each player can use a degree-0 sharing of the product share $d_i = a_i b_i$ in remainder of the computation. The cheater is, however, not eliminated. How an uncooperative player's share can be reconstructed by the remaining players is discussed in Exercise 09.1.

Sending wrong values. After having introduced measures that make our protocol secure against players who refuse to send certain values, it now remains to deal with players who send wrong values. The idea will be to make sure that the sending of incorrect values is detected by honest parties who then treat this as if no value had been sent (using the solutions described above).

Roughly speaking, to ensure that players cannot send incorrect values without being detected, every player will be *committed* to every value he knows at any given time. Whenever a player is supposed to send some value to some other player, the commitment must be *transferred* to the recipient. When a value is to be broadcast, the sender must open the commitment to all players. Finally, whenever some player performs some internal computation, he commits to the result and proves (e.g., using a general zero-knowledge proof of knowledge) that the result was computed correctly.

Depending on the level of security one wants to achieve, different commitment schemes are used: If one desires computational security only, one uses normal, cryptographically secure commitment schemes like, e.g., Pedersen or ElGamal Commitments (see Section 8.3). To obtain information-theoretic security, one uses special *distributed* commitment schemes (which will be introduced next week).

8.2 Generic Actively Secure MPC

Motivated by the discussion in the previous section, we postulate a commitment scheme with the following sub-protocols:

- Commit: A player can *commit* to some value (w.r.t. to all players).
- Open: A player committed to some value can *open* the commitment to a single or to all players.
- CTP (Commitment Transfer Protocol): A commitment of some player to some value can be *transferred* to some other player.

Commitment Sharing Protocol

Starting point: Dealer is committed to some value s.
Goal: Every player has a share of s and is committed to it.

1. The dealer chooses the random coefficients used in the secret sharing scheme and commits to them.
2. Each player (locally) computes the commitments to all shares (using the homomorphic property).
3. For every player, the dealer transfers the commitment to the share corresponding to that player using the CTP.

Figure 1: *The commitment sharing protocol.*

- CMP (Commitment Multiplication Protocol): Two commitments of some player to two values can be transformed into a commitment of the *product* of these two values.

Recall that the basic idea is to detect the sending of wrong values and treat it as if none had been sent. Therefore, it is imperative that for each of the sub-protocols, the honest parties are in agreement whether it was executed successfully or not.

Finally, we require that the scheme be *homomorphic*. That is, from two commitments of some player to two values, one can compute a commitment to the *sum* of these two values (without communication).

The operations Commit and Open are part of any commitment scheme, except that now a player commits to value w.r.t. to all players (instead of just a single one). The Open protocol must allow a single player to open his commitment either to a single or to all players.

The protocol to transfer commitments will be used whenever some party is supposed to send a value to some other party in the passive protocol. If the transfer protocol is successful, the recipient will be committed to the same value (w.r.t. all players) and know how to open the commitment to that value.

Finally, the protocol for commitment multiplication allows some player already committed to two value to commit to the product of these two values.

We are now ready to present the generic protocol, which we later instantiate with cryptographically and information-theoretically secure commitment schemes. First, we construct (from the postulated commitment scheme) a commitment sharing protocol (CSP, see Figure 1), which allows a dealer committed to some value to "share" this value in such a way that every player ends up being committed to his share.

The actively secure MPC protocol is outlined in Figure 2.

8.3 Cryptographic Security

Given the considerations above, to obtain cryptographic security, it suffices to use some homomorphic commitment scheme that satisfies the requirements given in Section 8.2. Such an instantiation results in (a variant of) the protocol by [GMW87]. If the scheme is of type H (e.g., Pedersen Commitments), then the *secrecy* of the resulting protocol holds unconditionally. Conversely, if the scheme is of type B (e.g., ElGamal Commitments), then the *correctness* of the resulting protocol is unconditional. For further discussion, in particular for the construction of the protocols Commit, Open, CTP, and CMP from a normal (two-party) homomorphic commitment scheme, see Exercise 9.2.

Actively Secure MPC

Whenever some other player refuses to execute any of the steps below, proceed as described in Section 8.1 (sending wrong values).

Input

Each player gives input by committing to it and then executing the CSP to commit all other players to their share of the input.

Addition Gates (and Linear Functions)

Two values are added by having each player commit to the sum of his two shares using the homomorphic property of the commitment scheme (i.e., without communication).

Multiplication Gates

To multiply two values, each player first uses the CMP to commit to the product of his shares. Then he uses to the CSP to share this value. Finally, the players compute the product using Lagrange interpolation (as in the passive protocol), which is a linear function and can be evaluated using the homomorphic property.

Output

To reconstruct a value towards some player, all other players open their commitment to the share of that value to that player. Since there are at least $t+1$ honest players, the player receives enough shares.

Figure 2: *Actively secure MPC protocol tolerating at most $t < n/2$ corrupted players (or less if the commitment scheme requires a lower threshold).*

References

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.