ETH Zurich, Department of Computer Science
SS 2017

Dr. Martin Hirt
Chen-Da Liu Zhang

Cryptographic Protocols
Solution to Exercise 10

10.1 Not Sending Values

Note first that, given at least $t+1$ of the n shares, a_ℓ can be computed using Lagrange interpolation. That is, let $S \subseteq \{1,\ldots,n\} \setminus \{\ell\}$ with $|S| \geq t+1$. Then,

$$a_\ell = \sum_{j\in S} w_j a_j \qquad \text{where} \qquad w_j = \prod_{\substack{k\in S\\ k\neq j}} \frac{\alpha_\ell - \alpha_k}{\alpha_j - \alpha_k}.$$

Since the weights w_j are constant, a_ℓ is a linear function of the shares a_j for $j \in S$.

The above leads to the following protocol idea: Each player P_i re-shares his share a_i as $[a_i] = (a_{i1},\ldots,a_{in})$ among the players. This is followed by an accusation phase to make sure that all honest players agree on the same set S of players who have performed the re-sharing. Then, the players compute a sharing of a_ℓ (using only local operations on their respective shares) as shown above and reconstruct the value.

The actual protocol:

1. Sharing: Every player P_i shares his share a_i among all players. Let a_{ij} denote P_j's share of a_i.
2. Accusations: If a player P_i does not receive his share a_{ji} from some player P_i, then P_i complains about this by broadcasting an accusation. As an answer, P_j must broadcast the value a_{ji}. If P_j does not do so, he is disqualified. Denote by S the set of players that have *not* been disqualified.
3. Computing the share of a_ℓ: Every player P_i (locally) computes his share $a_{\ell i}$ of the value a_ℓ as follows:
$$a_{\ell i} = \sum_{j\in S} w_j a_{ji} \qquad \text{where} \qquad w_j = \prod_{\substack{k\in S\\ k\neq j}} \frac{\alpha_\ell - \alpha_k}{\alpha_j - \alpha_k}.$$
4. Reconstructing a_ℓ: Every P_i sends his share $a_{\ell i}$ to all other players. Let S_i denote the set of players from which P_i has received a share. P_i computes a_ℓ as follows:
$$a_\ell = \sum_{j\in S_i} w_j a_{\ell j} \qquad \text{where} \qquad w_j = \prod_{\substack{k\in S_i\\ k\neq j}} \frac{\alpha_k}{\alpha_k - \alpha_j}.$$

It is important that all (honest) players choose the same set S (as done in Step 2). Otherwise, they would execute Step 3 on different sharings and reconstruction would not work anymore. It is easy to verify that this protocol does not violate privacy.

10.2 ElGamal Commitments

a) We are to show that the commitment function

$$C: \ \mathbb{Z}_q \times \mathbb{Z}_q \to G \times G, \ \ (a, \alpha) \mapsto (g^\alpha, \gamma^a h^\alpha)$$

is homomorphic. This can be seen as follows:

$$\begin{aligned} C(a,\alpha) \cdot C(a',\alpha') &= (g^\alpha, \gamma^a h^\alpha) \cdot (g^{\alpha'}, \gamma^{a'} h^{\alpha'}) \\ &= (g^{\alpha+\alpha'}, \gamma^{a+a'} h^{\alpha+\alpha'}) \\ &= C(a+a', \alpha+\alpha'). \end{aligned}$$

b) Given a pair $(g_1, g_2) = (g^\alpha, \gamma^a h^\alpha)$, one can recover a (inefficiently) as follows:

1. Compute $\alpha = \log_g g_1$, the discrete logarithm to basis g of g_1.
2. Compute $x = \log_\gamma (g_2 h^{-\alpha})$.

c) For $a \in \mathbb{Z}_q$, denote by C_a the random variable corresponding to a commitment to a, i.e., for α chosen uniformly at random.

Recall that part of the commitment scheme is the publicly known but randomly chosen $h \in G$. Thus, to prove that ElGamal commitments are computationally hiding, it needs to be shown that, for every a and a', (h, C_a) is computationally indistinguishable from $(h, C_{a'})$.[1]

To that end, for $a \in \mathbb{Z}_q$, consider first an additional random variable $\tilde{C}_a$ defined by choosing $\alpha \in \mathbb{Z}_q$ and $k \in G$ uniformly at random and setting $\tilde{C}_a := (g^\alpha, \gamma^a k)$.

Using the triangle inequality[2] and the fact that $\tilde{C}_a \equiv \tilde{C}_{a'}$ for all $a, a' \in \mathbb{Z}_q$, one obtains that

$$\Delta^D((h, C_a), (h, C_{a'})) \leq \Delta^D((h, C_a), (h, \tilde{C}_a)) + \Delta^D((h, \tilde{C}_{a'}), (h, C_{a'})).$$

The value $\Delta^D((h, C_a), (h, \tilde{C}_a))$ (and similarly $\Delta^D((h, \tilde{C}_{a'}), (h, C_{a'}))$) can be bounded by a reduction to the DDH problem, i.e., transforming the distinguisher D into a distinguisher D'_a for DDH triples as follows: D'_a receives as input a triple (x, y, z) (which is either of the form (g^u, g^v, g^{uv}) or (g^u, g^v, g^w) for randomly chosen $u, v, w \in \mathbb{Z}_q$). Then, D'_a calls D on $(x, (y, \gamma^a z))$ and outputs whatever bit D outputs.

It is easily verified that if (x, y, z) is of the form (g^u, g^v, g^{uv}), then the input $(x, (y, \gamma^a z))$ to D is distributed identically to (h, C_a), and if (x, y, z) is of the form (g^u, g^v, g^w), then $(x, (y, \gamma^a z))$ to D is distributed identically to $(h, \tilde{C}_a)$. Thus,

$$\Delta^D((h, C_a), (h, \tilde{C}_a)) = \Delta^{D'_a}((g^u, g^v, g^{uv}), (g^u, g^v, g^w)),$$

and, finally,

$$\begin{aligned} \Delta^D((h, C_a), (h, C_{a'})) \leq \ \ & \Delta^{D'_a}((g^u, g^v, g^{uv}), (g^u, g^v, g^w)) \\ & + \Delta^{D'_{a'}}((g^u, g^v, g^{uv}), (g^u, g^v, g^w)), \end{aligned}$$

where $D'_{a'}$ is defined analogously to D'_a. Thus, under the DDH assumption, ElGamal commitments are computationally hiding.

[1] To be explicit: the randomness involved here is over the choice of h and α.

[2] See Section 1.7 of the lecture notes for a definition of computational indistinguishability and of the distinguishing advantage.

10.3 Multi-Party Computation from Homomorphic Commitments

a) Some player P can commit to a value $x \in \mathcal{X}$ among *all* players using the following protocol: The input of the sender P is x and the other players P_i have no inputs. P chooses a value $r \in_R R$ and broadcasts $y = C(x, r)$. P's output of the subprotocol is r, and the other players output the value y' received in the broadcast protocol. The protocol is always considered successful.[3]

b) Suppose some player P wants to open some commitment y for which he knows (x, r) such that $C(x, r) = y$ to some other player P'. The input of P is the opening information (x, r). P''s input is the commitment y. In order to open y, P just sends (x, r) to P', who accepts and outputs x if and only if $y = C(x, r)$.

If P wants to open y to all players, he simply broadcasts (x, r).

Since the first is a subprotocol for output (i.e., the value is not used further in the computation) between two players, there is no need for the players to agree on whether it succeeded. For the second subprotocol, broadcast again ensures that all honest players agree on whether the protocol was successful. Note that, here, just *sending* (x, r) is not sufficient.

c) A player P committed by y can transfer this commitment to some other player P' using the following protocol: The input of P is the opening information (x, r) and the input of all other players (including P') is the commitment y. Player P sends (x, r) to P', who checks if $y = C(x, r)$. If so, he broadcasts 1 and otherwise 0. Some player P_i accepts the protocol run and considers P' committed by y if and only if the broadcast value is 1.

Again, the broadcast properties ensure that, in the end, either all honest players consider P' committed to y or reject the protocol.

d) Suppose some player P is committed to a and b by $A = C(a, \alpha)$ and $B = C(b, \beta)$. Then, he can commit to the product $c = ab$ using the following protocol: The inputs of P are a, b, α, β and the inputs of the other players P_i are A, B. First, P computes $C = C(c, \gamma)$ for some $\gamma \in_R R$ and broadcasts C. Then, he executes a *distributed* (see below) zero-knowledge proof of knowledge of a pre-image of (A, C) with respect to the homomorphism

$$\psi : \; X \times R \times R \to Y \times Y, \; (x, \xi, \rho) \mapsto (C(x, \xi), C(xb, x\beta + \rho)) .$$

It is easily seen that the function is homomorphic.

The pre-image of (A, C) that P uses in the protocol is $(a, \alpha, \gamma - a\beta)$. Moreover, any party can compute the homomorphism as

$$\psi(a, \alpha, \rho) = (C(a, \alpha), B^a \cdot C(0, \rho)) .$$

A player P_i accepts the protocol if and only if P succeeds in the zero-knowledge proof. P outputs the randomness γ of C, and the other players P_i output C.

Observe that there must exist u, l such that the following two conditions hold:

1. $[u] = (A, C)^l$.
2. $\forall c_1, c_2 \in \mathcal{C}$ with $c_1 \neq c_2$ $\gcd(c_1 - c_2, l) = 1$

Such conditions are satisfied if $|Y| = q$ with q prime number, $u = 0$, $l = q$, and the challenge space is $\mathcal{C} = \{0, \ldots, q-1\}$.

[3] Note that players simply assume a default value if P does not broadcast anything, which is the reason why this protocol is always successful. The important thing is that all honest players have the same value y', which is guaranteed by the broadcast channel.

In the distributed zero-knowledge proof, P broadcasts all of his messages. The challenge is chosen as follows: Each player commits to a random value. Then, the sum of these values is opened and used as the challenge.

Since P broadcasts C as well as all the messages in the zero-knowledge proof and since the challenge is chosen in a distributed fashion, the honest players agree on whether or not the protocol was successful.