ETH Zurich, Department of Computer Science
SS 2020

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

Cryptographic Protocols
Solution to Exercise 9

9.1 ElGamal Commitments

a) We are to show that the commitment function

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

is homomorphic. This can be seen as follows:

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

b) Given a pair $(g_1, g_2) = (g^\alpha, g^a h^\alpha)$, 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 $x = \log_g(g_2 h^{-\alpha})$.

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 $\tilde{C}_a := (g^\alpha, g^a k)$.

Using the triangle inequality 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 $(x, (y, g^a z))$ 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 $(x, (y, g^a z))$ 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 $(x, (y, g^a z))$ to D is distributed identically to $(h, \tilde{C}_a)$. Thus,

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

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

and, finally,

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

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

9.2 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.[2]

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. If the broadcasted value is 0, P can broadcast the opening information.

Again, the broadcast properties ensure that, in the end, either all honest players consider P' committed to y or reject the protocol. Some player P_i accepts the protocol run and considers P' committed by y if and only if the broadcast value by P' is 1 or the broadcast value was 0 and P broadcasts a correct opening information.

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 $e = ab$ 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 $E = C(e, \epsilon)$ for some $\epsilon \in_R R$ and broadcasts E. Then, he executes a *distributed* (see below) zero-knowledge proof of knowledge of a pre-image of (A, E) with respect to the homomorphism

$$\psi : \ \mathcal{X} \times \mathcal{R} \times \mathcal{R} \to \mathcal{B} \times \mathcal{B}, \ \ (x, \xi, \rho) \mapsto (C(x, \xi), C(xb, x\beta + \rho)) \, .$$

One can intuitively interpret the proof as showing that $e = ab$; a bit more precisely, it shows that P can open A to a value a and E to ab.

In order to build such a zero-knowledge proof, one makes use of the one-way group-homomorphism protocol seen in the first part of the lecture. To use such a protocol, one needs that the function is homomorphic (the reader should verify this!), and can

[2] 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.

be efficiently evaluated: any party can evaluate the function without knowing b (only using the publicly known blob B):

$$\psi(x, \xi, \rho) = (C(x, \xi), B^x \cdot C(0, \rho)) .$$

The pre-image of (A, E) that P uses in the protocol is $(a, \alpha, \epsilon - a\beta)$. A player P_i accepts the protocol if and only if P succeeds in the zero-knowledge proof.

The proof is a zero-knowledge proof of knowledge if there exists u, l such that:

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

and also $|C|$ is poly-bounded and $1/|C|^s$ is negligible, where s is the number of repetitions of the 3-round OWGH protocol. Such conditions are satisfied for example with $u = 0$, $l = |\mathcal{B}|$, $\mathcal{C} = \{0, 1\}$ and s is large enough.

So far the OWGH protocol introduced in the first part of the lecture was a 2-party protocol. Here, we need that P proves the statement towards all parties, and all parties have agreement on whether the proof was successful or not. For that, P broadcasts all of his messages. The challenge is chosen as follows: Each player P_i chooses a random value r_i, and they jointly compute the sum of the random values.

Since P broadcasts E 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.