ETH Zurich, Department of Computer Science
SS 2017

Dr. Martin Hirt
Chen-Da Liu Zhang

Cryptographic Protocols
Solution to Exercise 9

9.1 Shamir Sharings

a) Suppose there is another polynomial f' of degree at most $n-1$ with the property that $f'(\alpha_i) = s_i$ for all $i = 1, \ldots, n$. Then, the polynomial $h := f - f'$ has n roots (namely $\alpha_1, \ldots, \alpha_n$). Since it has degree at most $n-1$, h must be the all-zero polynomial. Thus, $f = f'$.

b) For $T \subseteq \{1, \ldots, n\}$ and $s \in \mathbb{F}$, denote by $S^{T,s}$ the distribution sampled as follows: Choose random coefficients $R_1, \ldots, R_t$, compute $S_i := p(\alpha_i)$ for $p(x) := s + R_1 x + R_2 x^2 + \ldots + R_t x^t$ and set $S^{T,s} := (S_i)_{i \in T}$. That is, $S^{T,s}$ denotes the random variable corresponding to the vector of shares of the players P_i with $i \in T$ when $s \in \mathbb{F}$ is shared.

A sharing scheme reveals no information about s to up to t players if for every $T \subseteq \{1, \ldots, n\}$ with $|T| \leq t$,

$$S^{T,s} \equiv S^{T,s'} \tag{1}$$

for all $s, s' \in \mathbb{F}$.

Consider now a second distribution $\tilde{S}^{T,s}$, which is defined as $S^{T,s}$ except that the sharing polynomial $\tilde{p}(x)$ is obtained by choosing values $\tilde{R}_1, \ldots, \tilde{R}_t$ of $\tilde{p}(x)$ and interpolating the unique polynomial $\tilde{p}(x)$ through the points $(\tilde{\alpha}_i, \tilde{R}_i)$ and $(0, s)$ for some t *arbitrary* distinct non-zero positions $\tilde{\alpha}_i$. It is easily seen that $S^{T,s} \equiv \tilde{S}^{T,s}$ for all T and s, since every choice of coefficients $R_i = r_i$ uniquely determines a polynomial $p(x)$, which in turn uniquely determines the values at the t positions $\tilde{\alpha}_i$ and vice-versa.

Since the t arbitrary positions $\tilde{\alpha}_i$ can be chosen as the $(\alpha_i)_{i \in T}$, $\tilde{S}^{T,s} \equiv \tilde{S}^{T,s'}$ (both distributions are simply $|T|$ uniformly random and independent field elements). This implies (1).

c) Denote by $f(X) = a'X + a$ and $g(X) = b'X + b$ the sharing polynomials of a and b, respectively. In the following we create a system of equations that will allow P_2 to compute a and b from the values which he sees in the protocol:

$$f(\alpha_2) = a_2 \quad \Longleftrightarrow \quad 2a' + a = a_2 \tag{2}$$

$$g(\alpha_2) = b_2 \quad \Longleftrightarrow \quad 2b' + b = b_2 \tag{3}$$

Using the announced shares c_i, one can compute the *unique* polynomial h of degree at most 2 that goes through these points, i.e., $h(1) = c_1$, $h(2) = c_2$ and $h(3) = c_3$:

$$h(X) = h_1 + h_2 X + h_3 X^2 \tag{4}$$

for some coefficients h_1, h_2,and h_3, which can be computed, e.g., using Lagrange's interpolation formula.

Because h corresponds to the polynomial resulting from the multiplication of f and g, it should have the following form:

$$\begin{aligned} h(X) &= f(X) \cdot g(X) \\ &= (a + a'X) \cdot (b + b'X) \\ &= ab + (ab' + a'b)X + a'b'X^2 \end{aligned} \tag{5}$$

Because the coefficients in (4) and (5) should be the same

$$\begin{aligned} ab &= h_1 \\ ab' + a'b &= h_2 \\ a'b' &= h_3 \end{aligned}$$

The above three equations, together with (2) and (3), form a system of 5 equations over GF(5) with 4 unknowns. Solving these equations P_2 can compute the factors a and b.

d) The adversary can use its shares to interpolate a degree-$(t-1)$ polynomial $g' \neq g$, since the degree of the sharing polynomial g is exactly t. Because $g(\alpha_i) = g'(\alpha_i)$ for t indices $i \in \{1, \ldots, n\}$, $g(0) \neq g'(0)$ (since otherwise $g' = g$). Thus, the adversary can exclude $g'(0)$ as the secret, which violates privacy.

9.2 Circuit Evaluation

a) Since the order of the multiplicative group of $\mathbb{F}$ is $p-1$, $x^{p-1} = 1$, which implies that $x^{p-2} \cdot x = 1$, hence $x^{-1} = x^{p-2}$.[1] Note that when $x = 0$, then the computed "inverse" equals 0.

b) Let $c \in \{0, 1\}$. To execute the "if"-statement, compute

$$z := (1 - c) \cdot x + c \cdot y.$$

For an arbitrary $c \in \mathbb{F}$, compute

$$z := (1 - c^{p-1}) \cdot x + c^{p-1} \cdot y.$$

This results in the correct value z since $c^{p-1} = 1$ if $c \neq 0$ and $c^{p-1} = 0$ if $c = 0$.

9.3 Impossibility and Feasibility Proofs

a) If the players want to compute $b_1 \oplus b_2$, then the next protocol is passively secure:
Party P_1 sends b_1 to P_2, then party P_2 sends b_2 to P_1, and both output $b_1 \oplus b_2$.
Observe that P_1 can obtain the input b_2 from his input b_1 and the output $b_1 \oplus b_2$. Also, P_2 can obtain the input b_1 from his input b_2 and the output $b_1 \oplus b_2$. Hence, regardless of who the adversary corrupts, he does not learn anything beyond what is leaked by the output.

b) If the output is constant (0 or 1), the parties can trivially output the constant without exchanging any messages. We only need to focus on the case where the number of ones is two.
If there are two ones, the outputs can be either the first input $(0, 0, 1, 1)$, the second input $(0, 1, 0, 1)$, the XOR of the inputs $(0, 1, 1, 0)$, or the negation of one of the three cases.

[1] This can be implemented efficiently using the square-and-multiply method.

If the output is equal to the input of a party P_i, the protocol consists of one message, where P_i sends his input x_i to the other party, and both output x_i. In the previous subtask we saw a passively secure protocol that computes the XOR.

For the negated functions, it is enough to execute the protocol for the non-negated function, and at the end negate the output.

c) Now we show that if the output vector contains exactly a one, it is impossible to construct a protocol for such a function. We make the argument for the case where the vector is $(1, 0, 0, 0)$, since all cases are similar. In this case, the function is $\neg b_0 \wedge \neg b_1$. We sketch a reduction to the impossibility proof for the AND function presented in the lecture. Assume there is a protocol π that computes the function $\neg b_0 \wedge \neg b_1$. In order to construct a protocol that computes the AND function easily, the parties negate their inputs, and execute the protocol π on inputs $\neg b_0$ and $\neg b_1$.

Finally, if the vector has three ones, it is enough to execute the protocol for the negated function which has only a one, and negate the output.