ETH Zurich, Department of Computer Science
SS 2018

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

Cryptographic Protocols
Solution to Exercise 13

13.1 Hyper-Invertible Matrices

a) Consider a hyper-invertible matrix M. Denote by $f : \mathbb{F}^n \to \mathbb{F}^m$ the linear function defined by M and let $\vec{y} = (y_1, \ldots, y_m) = f(x_1, \ldots, x_n)$ for arbitrary values $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}^n$. Consider two sets $A, B \subseteq \{1, \ldots, n\}$ with $|A| + |B| = n$. Then, we have $\vec{y} = M\vec{x}$ and $\vec{y}_B = M_B\vec{x} = M_B^A\vec{x}_A + M_B^{\overline{A}}\vec{x}_{\overline{A}}$. Since M is hyper-invertible, $M_B^{\overline{A}}$ is invertible, and $\vec{x}_{\overline{A}} = (M_B^{\overline{A}})^{-1}(\vec{y}_B - M_B^A\vec{x}_A)$. The values $\vec{y}_{\overline{B}}$ can be computed similarly.

b) Let $f : \mathbb{F}^n \to \mathbb{F}^m$ be a hyper-invertible linear function. Due to linearity, it can be described by a matrix M. To show that M is hyper-invertible, consider sets $R \subseteq \{1, \ldots, n\}$ and $C \subseteq \{1, \ldots, n\}$ with $|R| = |C| > 0$ and the submatrix M_R^C of M. Since $|R| = |C|$, it suffices to show that M_R^C is surjective. To that end, consider an arbitrary $\vec{y}_R$. We are to show that there exists an $\vec{x}_C$ such that $\vec{y}_R = M_R^C\vec{x}_C$. This is equivalent to the existence of an $\vec{x}$ such that $\vec{y}_R = M_R\vec{x}$ with $\vec{x}_{\overline{C}} = \vec{0}$. Consider the values $x_i = 0$ for $i \in \overline{C}$ and the values y_i for $i \in R$. By the hyper-invertibility of f, $\vec{x}_C$ can be computed (linearly) from these values. Thus, M_C^R is invertible.

c) Since every (1×1)-sub-matrix must be invertible, a hyper-invertible matrix cannot contain any zeros. The only other element in GF(2) is 1. Clearly, the all-one matrix is invertible if and only if $n = 1$. Hence, for $n > 1$, there are no hyper-invertible matrices over GF(2).

13.2 Sharings of Zero

In the following, denote by $\mathbb{F}$ the field used in the sharing.

a) The protocol proceeds similarly to the passively-secure protocol that allows the players to create $n - t$ double-sharings of random values.

1. Each player P_i t-shares $s_i = 0$ among all players, resulting in sharings $[s_1], \ldots, [s_n]$.
2. The players (by local computation) compute the sharings

$$([r_1], \ldots, [r_{n-t}]) = M([s_1], \ldots, [s_n]),$$

   where M is some hyper-invertible $(n - t) \times n$-matrix over $\mathbb{F}$.
3. The players use $[r_1], \ldots, [r_{n-t}]$ as sharings of zero.

We show that the outputted sharings are random, correct degree-t sharings of 0.

Essentially, we use the fact that the 0-sharing property "survives" applying a hyper-invertible matrix.[1]

[1] Actually, the sharings of zero form a vector space.

Let f be the hyper-invertible function induced by the matrix M. Denote the indices of the honest players by $H \subseteq \{1, \ldots, n\}$, $|H| = n - t$. By hyper-invertibility of f, for any t fixed sharings $\{[s_i]\}_{i \notin H}$, there exists a bijective linear function $f' : \{[s_i]\}_{i \notin H} \to \{[r_j]\}_j$. Hence, the outputted sharings are the result of applying a bijective linear function f' (chosen by the adversary) to random, correct degree-t sharings of 0. Since any linear combination of degree-t sharings of 0 is again a degree-t sharing of 0, the outputted sharings are correct and random.

b) The protocol proceeds similarly to the actively-secure (with abort) protocol for generating $T = n - 2t$ random double-sharings.

In particular,

1. Each player P_i shares $s_i = 0$ among all players, resulting in sharings $[s_1], \ldots, [s_n]$.
2. The players (by local computation) compute the sharings

$$([r_1], \ldots, [r_n]) = M([s_1], \ldots, [s_n]),$$

   where M is some hyper-invertible $(n \times n)$-matrix over $\mathbb{F}$.
3. For $i = T+1, \ldots, n$, every player P_j sends his share of $[r_i]$ to P_i, who checks that all shares lie on a polynomial g of degree at most t and that $g(0) = 0$. If any of these conditions does not hold, P_i broadcasts a complaint and the protocol *aborts*.
4. The players use $[r_1], \ldots, [r_T]$ as sharings of zero.

It is easily seen that the protocol succeeds if all players follow the instructions (that is, the protocol is *complete*).

It remains to show that if the protocol does not abort, then the resulting sharings $[r_1], \ldots, [r_T]$ are indeed random degree-t sharings of zero. Indeed, if the protocol does not abort, then at least t of the sharings $[r_i]$ opened in Step 3 are correct degree-t sharings of zero. Moreover, $n - t$ of the sharings $[s_i]$ were created by honest players and thus are degree-t sharings of zero as well. Since M induces a hyper-invertible function, there exists a bijective linear function f mapping these n correct sharings of zero to the remaining sharings. Hence, all sharings are correct degree-t sharings of zero.

Given this, one can argue about the randomness of the sharings in the same fashion as above.

13.3 Passive Packed Secret-Sharing

a) Intuitively, we need that even given t shares of the corrupted players *and* $l-1$ shared values, the last shared value remains secret. This happens if $d = l + t - 1$. A formal proof follows.

Let $\mathbf{s} = (s_1, \ldots, s_l)$ be the vector of shared values. Without loss of generality, consider the shares $p(\alpha_1), \ldots, p(\alpha_t)$ of players $P_1, \ldots, P_t$.

By the Lagrange interpolation, it follows that with $\mathbf{s}$ fixed, there exists a bijection, mapping the vectors of t shares $p(\alpha_1), \ldots, p(\alpha_t)$ to the sharing polynomials of degree at most $d = l + t - 1$. Since the sharing polynomial is chosen uniformly at random, the shares are distributed uniformly at random, irrespective of the actual secret $\mathbf{s}$.

b) We assume that packed sharings with degree d and degree $2d$ are given for a random vector $\mathbf{r}$. The protocol to multiply vectors of shares is very similar to the one presented in the lecture. However, instead of a public reconstruction of the value $a \cdot b - r$, we will reconstruct this value towards a single player, e.g. P_1, who will share it with degree r:

0. Given: Packed sharings $[\mathbf{a}]_d, [\mathbf{b}]_d, [\mathbf{r}]_{d,2d}$.
1. Each P_i computes $d_i = a_i \cdot b_i \rightarrow [\mathbf{d}]_{2d}$.
2. Each player locally computes $[\mathbf{s}]_{2d} = [\mathbf{d}]_{2t} - [\mathbf{r}]_{2d}$, and sends this share to P_1.
3. P_1 reconstructs $\mathbf{s}$ from the shares received and shares with degree d the value $\mathbf{s} \rightarrow [\mathbf{s}]_d$.
4. Each P_i computes $[s]_d + [r]_d$.

The above protocol requires that $2d < n$. This means that $2(l - 1 + t) < n$, that is $t < \frac{n}{2} + 1 - l$. On the other hand, standard secret sharing (with $l = 1$) only requires $t < \frac{n}{2}$.

Therefore, packed secret sharing allows to decrease complexity of protocols, at the cost of a sub-optimal corruption threshold.