CSA E0 312: Secure Computation September 09, 2015

[Lecture 17]

Instructor: Arpita Patra *Submitted by: Pratik Sarkar*

1 Recap

In our previous lecture we saw how the shamir secret sharing can be extended to Verifiable secret sharing (VSS) in order to tackle a malicious adversary.

In a VSS protocol there is a distinguished party known as dealer $D \in P$; who holds an input $s \in F$ (where F is a field, such that $|F| > n$) , referred to as the secret, which he wants to share with n parties such that $t+1$ of them together can uniquely reconstruct it but t or less number of parties together have no information about the secret. We call an n party protocol, with adversary A, dealer D and secret s, an (n, t) VSS protocol if it satisfies the following conditions:

1. **Secrecy:** If D is honest then A's view during the sharing phase reveals no information on s. Formally, A's view is identically distributed for all different values of s.
2. **Correctness:** If D is honest then the honest parties togther output a common secret s at the end of the reconstruction phase. This condition holds true for any choice of the random inputs of the uncorrupted parties and A's randomness.
3. **Strong Commitment:** If D is corrupted, then at the end of the sharing phase there is a unique value $s' \in F$, such that at the end of reconstruction phase all honest parties output s', irrespective of the behavior of the corrupted parties. s' may include *Abort* as output, incase D has dealt inconsistent shares to the honest parties.

The protocol consists of two phases, a sharing phase and a reconstruction phase. In case of an honest dealer, in the first stage (called sharing), the dealer shares a secret so that any $t+1$ parties can later reconstruct the secret, while any subset of t or fewer parties will learn nothing whatsoever about the secret. In the second stage (called reconstruction), a set of $t+1$ or more parties reconstruct the secret. If the dealer is corrupted, the VSS ensures that either the dealer is forced to commit some (n, t) secret s which is uniquely reconstructed in the reconstruction phase, or he is disqualified. The dealer chooses a bivariate polynomial $F(x, y)$ of degree atmost t in x and y where the secret $s = F(0, 0)$. The dealer sends to party P_i two univariate polynomials: $f_i(x) = F(x, i)$ and $g_i(y) = F(i, y), 1 \leq i, j \leq n$. In the last lecture we saw 3 claims whjch hold true for VSS:

Claim 1 *The knowledge of t F(x, i) and t F(i, x) does not leak any information about F(x, y)*

Claim 2 *The knowledge of t+1 F(x, i) and t+1 F(i, x) completely determine F(x, y)*

Claim 3 $g_i(j) = f_j(i) = F(i, j)$ *and* $g_j(i) = f_i(j) = F(j, i)$

We also saw a 4-round VSS protocol. The parties share 2 random pads for each pair. Then after receiving the shares from the dealer each P_i checks for inconsistency among the shares with every other P_j publicly by broadcasting the shares padded with the random strings in the second round. If inconsistency is found then P_i, P_j and D broadcast their corresponding shares to that inconsistent share and the party whose share matches with $D's$ share is placed in the happy set whereas the other party is placed in the unhappy set. For every unhappy player P_k, D broadcasts his corresponding polynomial $f'_k(x)$. If there are more than t unhappy players then the dealer is disqualified. If the dealer is not disqualified then every party checks for consistency with the polynomials broadcast by D. If for every public polynomial $f_i(x)$, there are $2t+1$ parties for whom it holds that $g_j(i) = f_i(j)$, then the dealer is qualified else disqualify him. If the dealer is still not disqualified then every party P_i, in the happy set, shares his secret $s_i = f_i(0)$ and for every unhappy party P_j, he considers the polynomial $f'_j(x)$ broadcast by D and provides his share as $s_j = f'_j(0)$. Then the t-degree polynomial $g(y)$ is computed on the values $s_0, s_1, \ldots, s_n$ using Reed Solomon codes. The final secret is reconstructed as $g(0)$.

In this lecture we will see how to reconstruct using Reed Solomon codes. But before that let's recall Reed Solomon codes.

2 Reed Solomon Codes

A Reed Solomon (RS) $[n, k, d]$-code over a field $F(|F| > n)$ of size q is a code of length n, where each codeword is a sequence of n field elements and from that k field elements are used as message bits. So there are q^k different codewords), and every two codewords has a distance $d = n-k+1$, i.e. they are atleast d Hamming distance apart from each other. For our needs, we construct a RS code of size n, dimension $k = t+1$, and distance $d = n-t$ as follows:

Let F be a finite field such that $|F| > n$, and let $a_1, a_2, \ldots, a_n$ be n distinct field elements. Let $m = (m_0, m_1, \ldots, m_t)$ be a message to be encoded, where each $m_i \in F$. The encoding of m is as follows:

1. Define a polynomial $p_m(x) = m_0 + m_1 x + \ldots + m_t x^t$ of degree t.
2. Compute the codeword $C(m) =< p_m(a_1), p_m(a_2), \ldots, p_m(a_n) >$

Theorem 4 *A Reed Solomon code with distance $d > 2x$ can correct upto x errors.*

3 Error Correction

In our case, we have $n > 3t$, so the minimum distance of the RS codes are:

$d = n - (t+1) + 1 = 3t + 1 - t - 1 + 1 = 2t + 1$

So we can correct upto t errors in our protocol. We have $n = 3t+1$ parties where each party P_i broadcasts his share $s_i = f_i(0) = F(0, i)$ in the last round of the VSS protocol.

Notice that $t+1$ correct $F(0, i)$ values can be used to interpolate a t-degree polynomial $H(x)$, where $H(0) = F = (0, 0) = s$. We are interested in finding that $H(x)$ polynomial.

Each share is an element in the field F. All the n shares together can be considered as a Reed Solomon code.

The dealer distributed shares according to the t degree polynomial $H(x)$. The $H(x)$ polynomial can be considered equivalent to the $p_m(x)$ polynomial in the RS code definition. The dealer distributes the following shares: $C =< H(a_1), H(a_2), \ldots, H(a_n) >$, where P_i receives $H(a_i)$ share as his $f_i(0)$, which he later broadcasts for reconstruction. Out of the n parties, t of them are corrupted shares and their $f_i(0)$ values(points) do not belong to the $H(x)$ polynomial. $2t+1$ points of the honest parties belong to the $H(x)$ polynomial. The distance of $2t+1$ can be used to find out the t corrupted values and thus the Reed-Solomon reconstruction procedure can be run and the honest parties can all obtain the correct polynomial $H(x)$, and can compute $H(0) = s$.

We demonstrate the Berklekamp-Welch algorithm for error correction in RS codes. We have $3t+1$ distinct points and out of which t are corrupted. The rest $2t+1$ points should interpolate to a t degree polynomial $H(x)$. Let the $3t+1$ points evaluate to a $3t$ degree polynomial $r(x)$. And let us consider the consider the erroneous points as $(e_1, e_2, \ldots, e_t)$. These t erroneous points define a t degree polynomial $e(x)$.

$e(x) = (x - e_1)(x - e_2)\ldots(x - e_t)$

Claim 5 *H(x)e(x) = r(x)e(x) at x = 1, 2, . . . , n.*

The polynomials in the LHS and the RHS in above claim are different. The polynomial in the LHS is of degree atmost $2t$, whereas the polynomial in the RHS is of degree atmost $4t$. If all the parties are honest, then $r(x) = H(x)$. But since there are t corrupted values, the LHS and RHS have same values at only n points. The n points can be divided into two sets:

1. **Honest set**: For all the points corresponding to the honest shares, the $H(x)$ and $r(x)$ values are same, because the honest parties have provided the same shares for both the $H(x)$ and $r(x)$ polynomial construction. $e(x)$ is common for both the sides, and hence LHS=RHS for honest values.

2. **Corrupted set**: For these the $H(x)$ and $r(x)$ values will be different as $r(x)$ has been interpolated on the corrupted points but not $H(x)$. But for them the value of $e(x)$ will be 0 as they will be one of the e_i points. Thus both LHS and RHS will be 0 for corrupted points.

We assume the following:

$$q(x) = H(x)e(x) \tag{1}$$

$$q(x) = r(x)e(x) \text{ at } x = 1, 2, \ldots, n. \tag{2}$$

We have the polynomial $r(x)$, ad we wish to find the value of $H(x)$. The value of $H(x)$ can be found if the value of $q(x)$ and $e(x)$ can be computed. $q(x)$ has $2t+1$ coefficients and $e(x)$ has $t+1$ coefficients. They together have $3t+2$ coefficients. But the coefficient of x^t in $e(x)$ is always 1. We have $3t+1$ values of $r(x)$ and substituting them in Eq. [**?**] gives

$3t + 1$ equations. Upon solving those equations we find the $3t + 1$ coefficients and obtain the value of $q(x)$ and $e(x)$. Then we can find the value of $H(x)$ from Eq. [**?**]. Thus we will obtain the original equation $H(x)$ and also the adversarial points from $e(x)$. Then we can find the value of the secret as $H(0)$. The solving of $3t + 1$ equations can be viewed as solving system of linear equations which reduces to (publicly known) matrix multiplication and this can be done using known methods like Gaussian elimination.

Since all the operations done in the error correction are linear operations, and Shamir Secret Sharing supports linearity property, after the 4^{th} round, the linear operations done above can be locally computed on the $f_i(0)$ broadcast shares of every party. Each party can individually solve all the $3t + 1$ equations and obtain the final secret and the indices of the adversaries. Thus distributed error correction is supported if we are using reed solomon codes.

4 Summary

So we saw how to correctly reconstruct the final secret from the verifiable secret sharing scheme. The previous lecture discussed how the VSS protocol can be used to prevent the dealer from distributing inconsistent shares and commits him him to a secret. It also allowed the honest parties to correctly reconstruct the shared polynomial in the presence of the malicious adversaries. In this lecture we learnt how the reed solomon codes can be used to help in the reconstruction phase of the VSS protocol.

In the next lecture we will see how the adversary can be forced to give valid shares for multiplication gate and thus obtain an information theoretic secure BGW protocol.

References

[1] Reed Solomon Codes *https://math.berkeley.edu/ mhaiman/math55/reed-solomon.pdf.*

[2] Atri Rudra *http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/lectures/lect27.pdf* Lecture notes.

[3] Arpita Patra *http://drona.csa.iisc.ernet.in/ arpita/SecureComputation15.html* Lecture notes.

[4] Gilad Asharovy, Yehuda Lindelly *A Full Proof of the BGW Protocol for Perfectly-Secure Multiparty Computation.* Journal of Cryptology pp 1-94.