CS 294 – Special Topic in Cryptography: Secure Computation Feb 2, 2016

Lecture 5: BGW – Malicious Case

Instructor: Sanjam Garg *Scribe: Lynn Tsai*

1 A verifiable secret sharing scheme

Over finite field F: $\alpha_1, \ldots, \alpha_n$.

1. Sample $f(x)$ s.t. $f(0) = s$.
   - random
   - degree t
2. Sample $s(x, y)$ s.t. $s(0, z) = f(z)$.
   - random
   - degree t
3. $s(x, \alpha_i)$ and $s(\alpha_i, y)$ to P_i
   $f_i(x)$ $g_i(y)$
4. $p_i, p_j \iff f_i(\alpha_j) = g_j(\alpha_i)$

If checks pass, prove degree t polynomial and all parties can recover the same polynomial.

Proof.
Let K be a set of honest parties. Let L be some set in K such that it contains at least $t+1$ parties.

1. $L \leq k$ s.t. $|L| = t + 1$, consistent.
   - For every honest party, shares are correct for the defined polynimal.
   - Define $s(x, y)$ for L.
2. $\forall k \in K, g_k(y)$ is consistent with $s(x, y)$.
3. $\forall k \in K, f_k(x)$ is consistent.

■

A more detailed proof:

Proof.

$$\begin{aligned}\forall k \in K, l \in L g_k(\alpha_l) =& f_l(\alpha_k)\\ & \vdots \\ & f_{l_{t+1}}(\alpha_k) \\ & \text{(there are } t+1 \text{ choices for l)}\end{aligned}$$

We have fixed g_k at $t+1$ points. g_k is a degree-t polynomial (and must be a unique polynomial). The points are $f_{l_1}(x), \ldots, f_{l_{t+1}}(x)$ and check consistency with g.

- $\forall k \in L$, we are done.
- $\forall k \in K \setminus L$, analogous argument.
- $\forall k \in K, j \in K, f_k(\alpha_j) = g_j(\alpha_k) = s(\alpha_j, \alpha_k)$
  - defines $> t+1$ points, so also fixed polynomial

We want to next prove that an attacker does not learn anything more.

- Argue that s is hidden
- Stronger claim: if q_1, q_2 are two degree-t polynomials over F s.t. $q_1(\alpha_i) = q_2(\alpha_i) \forall i \in I, |I| \leq t$, then $S_i = \{(i, S_1(x, \alpha_i), S_1(\alpha_i, y))\}$.
  - q_1 does not have to be exactly the same as q_2
- Looking for all possible choices consistent with the polynomials.
- If you picked some polynomial, it is equally likely that it could have come from q_1 or q_2

Adversary sees:

$$z : \{l_i, f_i(x), g_i(y)\}_{i \in I}$$

s_1 polynomials exist with q_1 and s_1 polynomials exist with q_2, so you get z with the same probability from both q_1 and q_2.
We need to prove that the number of polynomials in both settings (q_1, q_2) are the same.

1. S_1 is a set of polynomials, pick an $S(x, y)$ randomly with equal probability.
2. Only pick S consistent on i points.

3. Prove attacker that with information in z, does not learn anything about S. Gets to view z regardless of which q is chosen.

4. $S_1 \neq S_2$ necessarily, they don't need to be the exact same sets (if size $t+1$ then it's true, but not needed in proofs).

5. Attempt to reduce $z \to S$:

$$\text{constraints: } \begin{cases} \forall i \in I f_i(0) = q_i(\alpha_i) = q_2(\alpha_i) \\ \forall i,j \in I f_i(\alpha_j) = g_j(\alpha_i) \end{cases}$$

6. How many polynomials are in S_1? $|I|$ polynomials $f_i(x)$, need $t+1-|I|$ more to fix S_1.

7. Limited by constraints

8. For every $f_1(x)$ that I add, there are $I+1$ constraints:

$$\begin{aligned} f_j(\alpha_i) = g_i(\alpha_i) \forall i \in I \\ f_i(0) = g_1(\alpha_i) \end{aligned}$$

   Number of degrees of freedom: $(t+1) - (|I|+1) = t - I$.

9. Same if argue the other side.

■

Highlights of the construction:

- So far, all shares are distributed with degree t polynomial
- Error correction
- Enforce honest behavior in malicious
- Accomplished verifiable secret sharing
  - Linearity of Reed-Solomon code means that addition is fine.

$$P_1, \ldots, P_n$$

- Share their shares with verifiable secret sharing
- Compute linear function: $f_{s_1}(\alpha_i) + f_{s_2}(\alpha_i) = f_{s_1+s_2}(\alpha_i)$
- Reed-Solomon decoding, detect malicious input and corrects it if up to $t < \frac{n}{3}$, now decoded (doesn't reshare share correctly)

- Uses distance between codewords
- The number of errors does not go outside boundary if less than $\frac{n}{3}$.
- Must protect against incorrect inputs during computation!
- Malicious parties can only make it a non-codeword, not another codeword. (The number of codewords is small).

2 Multiplication

- What can go wrong?
- Need to compute product.
- Parties may not multiply a_i, b_i incorrectly.
  - Mechanism of sharing a_i, b_i product that is consistent.
  - Degree larger so can't use Reed-Soloman decoding
- Trying to share a_i, b_i as degree-t polynomial s.t. a_i, b_i consistent from before.

Share $a_i, b_i, a_i \cdot b_i$, and a bunch of things that collectively guarantee a_i, b_i correct.

$$D(x) = a \cdot b + \sum_{k=1}^{2t} d_k x^k$$

$$k \in \{1, \ldots, t\} D_k(x) = \sum_{l=U}^{t-1} r_{k,l} \cdot x^l \left(d_{k+t} - \sum_{j=k+1}^{t} r_{j,t+k-j} \right) \cdot x^t$$

$$C(x) = D(x) - \sum_{k-1}^{t} x^k D_k(x)$$

- Wouldn't need this if $< \frac{n}{4}$ malicious people.
- Now linear, can do on own.
- Also share $C(x)$, D's higher order terms set to zero and also randomized.
- $C(0) = D(0)$ is all we want; C is also a degree-t polynomial.
- Can check $C(x)$ relationship for all α_i on your own.