CSE 5854: Class 13

Benjamin Fuller

March 1, 2018

1 Building multi party functionality

Following the previous class's formalization of security we are going to extend to a private multi party protocol. It is natural to try and extend the protocol we used for the two party case (Yao's garbled circuits). Let us suppose that we have parties $P_1, ..., P_n$ and we assume that P_1 constructs the Garbled circuit. While a player P_2 can be given the Garbled circuit its unclear how to deal with the inputs of other players. We could try and design a variant of oblivious transfer when there are three players P_1 specifies the strings, P_2 receives the string, and P_i provides the bit. However, it is impossible for this construction to be secure if P_1 and P_2 are acting maliciously and collaborating. This is because they can easily determine the bit b by combining their values. Other variants of an oblivious transfer like protocol suffer from similar issues, small number of participants can learn other parties inputs (making simulation impossible).

Instead the first multi party protocol [GMW87] takes a different approach that combines oblivious transfer with a new primitive called secret sharing. The idea is to split up each input in a way that no party learns any information about the input. We will then show how to compute through a computation circuit gate by gate. This will be done on the "shared" values directly. The final step of the protocol will then be to reconstruct the output of the circuit.

2 Secret Sharing

Secret sharing was introduced by Shamir in 1979 [Sha79]. A t-out-of-n sharing of a secret x is an encoding of the secret into n pieces, called **shares**, such that any t shares together can be used to reconstruct x but fewer than t shares give no information at all about x. It consists of two algorithms: Share and reconstruct or Rec.

1. $(s_1, ..., s_n) \leftarrow \mathsf{Share}(x)$ takes in a secret x and produces n shares.
2. $\tilde{x} \leftarrow \mathsf{Rec}(s_{i_1}, ..., s_{i_t})$ takes in t secret shares and returns the reconstructed secret $\tilde{x}$.

Let $\mathcal{I} \subset \{1, ..., n\}$, define the random variables $S_{\mathcal{I},x}$ as the restriction of $\mathsf{Share}(x)$ to the values in $\mathcal{I}$. We want two properties out of a secret sharing scheme. These are naturally correctness and secrecy.

1. **Correctness:** For all x, for all sets $\mathcal{I} \subset \{1, ..., n\}$ such that $|\mathcal{I}| = t$ then

$$\Pr[\mathsf{Rec}(S_{\mathcal{I},x}) = x] = 1.$$

2. **Security:** For any two x_1, x_2 for all sets $\mathcal{I} \subset \{1, ..., n\}$ such that $|\mathcal{I}| < t$. For all $s_{i_1}, ..., s_{i_t}$,

$$\Pr[S_{\mathcal{I},x_1} = s_{i_1}, ..., s_{i_t}] = \Pr[S_{\mathcal{I},x_2} = s_{i_1}, ..., s_{i_t}].$$

Note the drastic change we ask from the scheme, when only $t-1$ shares are presented the secret should be independent of the shares. However, once t shares are present the secret becomes completely determined. It is possible to define secret sharing for other "access structures" for example a player i must participate for reconstruction to work properly.

2.1 A basic scheme

We'll now define a very simply secret sharing scheme that is based on a generalization of the one-time pad. The idea is to construct $n-1$ random keys distributed these to players $1, ..., n-1$ and the encrypted value to the last party. That is:

$\mathsf{Share}(x)$:

1. Generate random $s_1, ..., s_{n-1}$.
2. Set $s_n = x \oplus s_1 \oplus ... \oplus s_{n-1}$.
3. Output $s_1, ..., s_n$.

Reconstruction follows by simply adding all values. Correctness of the scheme is immediate. Security is also simple. Suppose that $\mathcal{I}$ (where $|\mathcal{I}| \leq n-1$) does not include n then the shares have no information about x and thus are independent. However, if $n \in \mathcal{I}$ there must be some i^* that is not included in $\mathcal{I}$ thus the value s_n can be viewed as $s_n = x \oplus r_{i^*} \oplus y$ for some (possibly) known value y. Since r_{i^*} is uniformly distributed the value s_n is independent of x.

This type of secret sharing scheme is known as linear which means that it is possible to perform operations on shares that result in a linear operation on the underlying value. For example if $s_1, ..., s_n$ is sharing of y and $r_1, ..., r_n$ is sharing of x then $r_1 \oplus s_1, ..., r_n \oplus s_n$ is a sharing of $x \oplus y$. This is because $x \oplus y = (r_1 \oplus ... \oplus r_n) \oplus (s_1 \oplus ... \oplus s_n) = (r_1 \oplus s_1) \oplus ... \oplus (r_n \oplus s_n)$. Similarly, for some fixed value a is possible to produce a sharing of ax by operating on the shares. The scalar a is just multiplied by each value. Then

$$ax = a(r_1 \oplus ... \oplus r_n) = ar_1 \oplus ... \oplus ar_n.$$

2.2 Shamir's Scheme

The previous scheme works well for the case where we want privacy when all other players may be malicious. However, sometimes we will want to continue if some players stop participating in the protocol. An n-out-of-n secret sharing is incapable of helping with this task. Shamir showed how to construct a t-out-of-n secret sharing for any $2 \leq t \leq n$. We consider some finite field $\mathbb{F}$.

As a reminder, a finite field supports multiplication, addition, subtraction, and division (by things other than 0). For our purposes we can think of $\mathbb{Z}_p^*$ but there is a finite field for binary strings of each length. The core of the scheme is to generate polynomials with an intercept of x. Suppose the polynomials generated are of degree $t-1$. Polynomials have two important properties:

1. A polynomial of degree at most $t-1$ is uniquely defined by any collection of t points.
2. In a finite field, knowing the value of any $t-1$ points does not tell you the value of the polynomial at any other point. In particular, it does not tell you the value of the polynomial's intercept.

These two properties correspond to correctness and security respectively. We first present the basic scheme:

$\mathsf{SShare}_t(\alpha)$:

1. Generate $a_1, ..., a_{t-1} \xleftarrow{\$} \mathbb{F}$.
2. Define the polynomial $f(z) = \alpha + a_1 \cdot z + ... + a_{t-1} \cdot z^{t-1}$.
3. For all $1 \leq i \leq n$, set $s_i = f(i)$.
4. Release to player i, (i, s_i).

To describe the reconstruct algorithm requires an algorithm called Lagrangian interpolation. Lagrangian interpolation allows one given t points to exactly construct the *unique* polynomial of degree at most $t-1$. That is, given $(x_1, y_1), ..., (x_t, y_t)$ where all x_i are distinct one can construct the polynomial as:

$$L(z) \stackrel{def}{=} \sum_{j=0}^{t} y_j \ell_j(z).$$

here each ℓ_j is known as a Lagrange basis polynomial and is defined as:

$$\ell_j(z) \stackrel{def}{=} \prod_{0 \leq m \leq t, m \neq j} \frac{z - x_m}{x_j - x_m}.$$

The idea behind each of these ℓ_j is to define a polynomial which is 0 at all points other than j and has a value of 1 at j (which is then scaled to y_j in $L(z)$).

This then immediately leads to the reconstruction algorithm:

$\mathsf{Rec}(i_1, s_{i_1}, ..., (i_t, s_{i_t}))$:

1. Find the Lagrangian polynomial $L(z)$.
2. Output $L(0)$.

To show correctness of the protocol we must show that whenever $L(z)$ agrees with $f(z)$ at t locations they must be the same. This is proved under the assumption that both $L(z)$ and $f(z)$ are of degree at most $t-1$.

Proof. By construction both L and f are of degree at most $t-1$. Furthermore by construction $L(i_j) = f(i_j)$ for $1 \leq j \leq t$. It suffices to show that $L(z) - f(z) = (L - f)(z)$ is the zero polynomial. The polynomial $(L - f)$ is of degree at most $t - 1$ as it is the sum of two degree $t - 1$ polynomials. Furthermore, at each i_j, $L(i_j) = f(i_j)$ and thus $(L - f)(i_j) = 0$. That is, $L - f$ has t distinct zeros. Since it is of degree at most t by the fundamental theorem of algebra L_f is the zero polynomial. □

We now move on to showing privacy which is that given $t - 1$ points on the polynomial this gives no information about the shared value α. That is we seek to show that for any α_1, α_2 any subset $\mathcal{I}$ of size at most $t - 1$ is independent of the shared value. Suppose that the adversary has points $(1, y_1), ..., (t - 1, y_{t-1})$ without loss of generality. For any value of the intercept $(0, \alpha)$ there is a unique polynomial f_α of degree at most $t - 1$ that goes through these points. This is by the same Lagrangian interpolation argument that we used before. Since for this set of points there is one possible polynomial that has each intercept, the adversary is unable to infer any information about the intercept based on these points.

References

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.