Today’s Agenda

- Secret sharing
- Threshold Encryption
- Multi-party computation

1 Secret Sharing

We have \( n \) participants and we want to split a secret \( s \) among the participants. Some \( t < n \) out of these participants may be controlled by an adversary \( A \). So we have an algorithm \( \text{Share}(1^k, s) \) which outputs the shares \( x_1, \ldots, x_n \) for each participant. Also we should be able to reconstruct the secret with \( t + 1 \) shares.

\[
\text{Reconstruct}(1^k, x_i_1, \ldots, x_i_{t+1}, i_1, \ldots, i_{t+1}) = s
\]

Intuitively we need correctness to ensure that it is possible to reconstruct the secret \( s \) from \( t + 1 \) shares, we also need security that any \( t \) participants colluding together should not be able to retrieve the secret.

Let us look at an example of \( n \)-out-of-\( n \) secret sharing: Let \( s \in \{0, 1\}^l(1^k) \) and shares \( x_i \leftarrow \{0, 1\}^l(1^k) \) for \( 1 \leq i < n \) and \( x_n = s \oplus x_1 \oplus \cdots \oplus x_n^{-1} \). So we have correctness here which is that \( \forall s, \forall (x_1, \ldots, x_n \in \text{Share}(1^k, s)) \), we have that \( \oplus_{i=1}^n x_i = s \)

What is our intuitive requirement for security? The adversary controls \( t \) participants. So we want that the adversary cannot learn much from the \( t \) shares that it sees. Or we can say that \( \exists \text{Sim} \) s.t. for all secrets \( s \) and for all \( A \), \( \forall i_1, \ldots, i_t, \text{A cannot distinguish between } (1^k, x_i_1, \ldots, x_i_t) \leftarrow \text{Share}(1^k, s) \text{ from } \text{Sim}(1^k, i_1, \ldots, i_{n-1}). \)

1.1 Shamir’s secret sharing

Let us construct a 2-out-of-\( n \) scheme: \( \text{Share}(1^k, s) \) will pick a prime \( q > 2^{l(k)} \), pick a slope \( s \) and each share \( x_i = a_i s + b \). So if you have two points you can construct the line, get the slope and reconstruct the secret. But if you have only one point, it gives you no idea about the secret as each slope is equally likely.

Let us generalize this to have \( t + 1 \)-out-of-\( n \) secret sharing. Now instead of line, we will use a polynomial of degree \( t \) which can be uniquely defined by \( t + 1 \) points. so now \( \text{Share}(1^k, s) \):

- Pick a prime \( q > 2^{l(k)} \)
- Pick \( t \) coefficients \( a_1, \ldots, a_t \leftarrow \mathbb{Z}_q^* \). So now our secret \( s \) can be represented as an element of \( \mathbb{Z}_q \)
- Let \( f(x) = s + \sum_{i=1}^t a_i x^i \) mod \( q \) and each share \( x_i = f(i) \)
Reconstruct algorithm for Shamir’s secret sharing:

\[ x_{i_1} = f(i_1) = s + a_1i_1 + a_1i_1^2 + a_3i_1^3 + \cdots + a_ti_1^t \]
\[ x_{i_2} = f(i_1) = s + a_1i_1 + a_1i_1^2 + a_3i_1^3 + \cdots + a_ti_2^t \]
\[ \vdots \]
\[ x_{i_{t+1}} = f(i_{t+1}) = s + a_1i_{t+1} + a_1i_{t+1}^2 + a_3i_{t+1}^3 + \cdots + a_ti_{t+1}^t \]

So now we have \( t+1 \) unknowns and \( t+1 \) linear equations and the point is that we need \( t+1 \) of these equations to reconstruct the polynomial and the secret and any less than \( t+1 \) shares will not help in reconstructing the secret.

1.2 Error-Correcting Codes

Suppose Alice sends the shares of her secret to Bob across a network. The network may be noisy and some of the shares might get corrupted. But we have the guarantee that as long as \( t < n/3 \) shares are corrupted, Bob can still reconstruct the secret \( s \) faithfully. In fact he can also figure out which of the shares got corrupted on their way. This is an area called error-correcting codes. One of the fundamental codes that is even used in DVDs is the Reed-Solomon codes.

2 Threshold Encryption

Let us see a threshold version of Elgamal encryption: We have a group \( G \) of prime order \( q \).

KeyGen : \( \text{KeyGen}(G) \rightarrow (sk = x, pk = y = g^x) \)

Encrypt : \( \text{Enc}(pk, m) \) Pick \( r \leftarrow \mathbb{Z}_q \) and output \( (R, T) = (g^r, y^r \cdot m) \)

Decryption : Output \( m = T/R^x \)

Now if we want to give access to our secret key in a threshold manner such that together your friends should be able to reconstruct the secret key, but any \( t \) of them colluding would not be able to.

Let \( a_0 = x \), pick \( a_1, \ldots, a_t \leftarrow \mathbb{Z}_q^* \) and shares \( x_i \) are computed through Shamir’s secret-sharing:

\[ x_i = \sum_{j=0}^{t} a_j \cdot i^j = a_0 + a_1i + a_2i^2 + \cdots + a_ti^t \]

Now we are ready to define a threshold encryption scheme which lets you Threshold Encryption: Each friend \( i \) has a share \( x_i \)

1. On input \( (R, T) \), broadcast \( R^{x_i} = U_i \)
2. Combine \( U_i \)’s to get \( U = R^x \)
3. We also add verification information \( V_i = g^{x_i} \) for all \( 1 \leq i \leq n \)

Note that each \( U_i = R^{x_i} = R^{\sum_{j=0}^{t} a_j i^j} \) and what we want to compute \( r^{a_0} \) where \( a_0 = \sum_{i=1}^{t+1} \lambda_i x_i \) for appropriate \( \lambda_i \)’s. Hence,

\[ R^{a_0} = R^{\sum_{i=1}^{t+1} \lambda_i x_i} = \prod_{i=1}^{t+1} R^{\lambda_i x_i} = \prod_{i=1}^{t+1} U_i^{\lambda_i} \]
Now we want to make sure that each friend $i$ is outputting the correct share that it is supposed to output: So now along with broadcasting $U_i = R^{+}$, each friend also gives a zero-knowledge proof that $\log_R U_i = \log_g V_i$. So now we are incorporating a share only if the zero-knowledge proof verifies and then we are sure that the share is not tampered with.

For security, we want to argue that there exists a simulator who can simulate shares and verification information for an $\mathcal{A}$ who controls $t$ friends.

3 Multi-Party Computation

Recall that multi-party computation lets $n$ participants compute a function $f$ on their private inputs without revealing their inputs to each other. Cramer-Damgard-Nielson protocol: What we need is an additively homomorphic threshold cryptosystem (a.k.a threshold Paillier) and zero-knowledge proofs.

Let’s say we have $n$ participants such that each participant $P_i$ has input $y_i$. Let $\bar{a}_i = \text{Enc}(y_i)$. They want to compute $f(a_1, \ldots, a_n)$ where $f$ is an arithmetic circuit.

Protocol for multiplication:

1. Each $P_i$ picks $d_i \in \mathbb{Z}_N$, broadcasts $\bar{d}_i$ and zero-knowledge proof that
2. Let $\bar{d} = +\bar{d}_i$ and $\bar{d} + a = \bar{d} + \bar{a}$. Threshold decrypt $\bar{d} + a$
3. $P_i$ computes $a_i = -d_i$ if $i \neq 1$ and $a_1 = d + a - d_1$. Note that $\sum a_i = a$
4. $P_i$ computes $\bar{a}_i \bar{b} = \bar{a}_i \cdot b$. Broadcast this along with zero-knowledge proof.
5. Combine $a_1 \bar{b}, a_2 \bar{b}, \ldots, a_n \bar{b}$ and output $\bar{c} = \bar{ab}$