Robust Oblivious Transfer

Objective

- Build a robust routing scheme where the sender and the receiver have n different connections/routes and the task is to send k blocks of data successfully even if up to any e of the n connections are corrupt.
- Using a public-key cryptosystem, say El Gamal, design a Robust Oblivious Transfer protocol between a client A (who has the index i) and server B (who has the array) such that A and B are part of a large network and reliably communicate via the above robust routing mechanism.

Robust Routing Scheme

First, let's define an algorithm to break k blocks of information into n blocks, such that even if e get corrupted, we can reconstruct the original k blocks from the remaining $n - e$ blocks.
To store k blocks of data/information, coding theory suggests that we need $n \geq k + 2e$ blocks, such that if at most e blocks are corrupted, we can still reconstruct the k blocks of information.
Using digital signatures, we can achieve this with $n \geq k + e$ blocks.
This scheme is very similar to Shamir's Secret Sharing scheme.

1. Construct a $k-1$ degree polynomial, with the values of the k blocks of information (assumed to be integers), as the coefficients of the polynomial.

   $$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1}$$

   where a_i is the value of the i^{th} block.
   This is similar to Shamir's Secret Sharing scheme, in which a_0 is the secret and rest of the coefficients are generated randomly.

2. Generate n points on this polynomial.

   $$(x_1, f(x_1)), (x_2, f(x_2)), ..., (x_n, f(x_n))$$

   where $x_i' s$ are chosen randomly.

3. Now, we can use digital signatures to tag these values, in order to check when a block has been corrupted. Thus, the n blocks will have the values $c_i = (x_i, f(x_i), sign(hash(f(x_i))))$.

4. Each of these n blocks is transmitted over the channels between the sender and the receiver. Thus, each of the n channels will be used to send one block of the n c_i blocks.

Reconstruction

For each block received at the receiver, we can generate the signature of the values again and check whether it matches the stored value. If it does not match, the block is corrupted.
If e blocks are corrupted, then $(n - e)$ can be used for reconstruction. Since we need at least k points to reconstruct a polynomial of degree $k - 1$, at least k blocks should not be corrupted.

$$\implies n - e \geq k$$
$$\implies n \geq k + e$$

Thus, with verification through digital signatures, it is possible to achieve fault tolerant storage with $n \geq k + e$ blocks.

To get the values of the k blocks, we need the k coefficients of the polynomial, which can calculated by solving the over-determined system of linear equations obtained through the known $n - e$ points on the polynomial.
Even if $e \leq n - k$ channels fail, the receiver can still reconstruct the data using the remaining blocks.

Robust Oblivious Transfer Protocol

Assumptions

- Server B has an Array D of data $[d_1, d_2, ..., d_k]$
- Client A has index i, and wants the block d_i from B.
- El Gamal algorithm is used for encryption and decryption of the data blocks.
- A know's B's public key.

The Protocol

1. A chooses k random integers $(r_1, r_2, ..., r_k)$
2. A encrypts r_i using El Gamal with B's public key.
3. A sends $Z = (r_1, r_2, ..., f(r_i), ..., r_k)$, where f is the El Gamal encryption, using the robust routing scheme designed before.

4. B reconstructs and decrypts Z and obtains
   $Y = (f^{-1}(r_1), f^{-1}(r_2), ..., r_i, ..., f^{-1}(r_k))$

5. B performs $D \oplus Y = d_1 \oplus f^{-1}(r_1), ..., d_i \oplus r_i, ..., d_k \oplus f^{-1}(r_k)$

6. B sends $D \oplus Y$ to A using the robust routing scheme.

7. A obtains $d_i = (D \oplus Y)[i] \oplus r_i = (d_i \oplus r_i) \oplus r_i = d_i$

Since A cannot see d_j for $j \neq i$ due to the presence of f^{-1}, and B cannot know which index A has requested, since r_i and $f(r_i)$ have the same range, this is an Oblivious Transfer Scheme

El Gamal Encryption

Key Generation

1. Chose a cycle group Z_p^*, and compute the generator generator g for the prime p.

2. Generate a random integer x from $\{1...(p-1)\}$. This is the private key.

3. Calculate $h = g^x \ mod \ p$.

4. Share (h, p, g) as the public key.

Encryption

1. Generate a random integer y from $\{1...(p-1)\}$.

2. Compute $c_1 = g^y \ mod \ p$

3. Compute Shared secret $s = h^y \ mod \ p = g^{xy} \ mod \ p$

4. Compute $c_2 = (m * s) \ mod \ p$, where m is the message to be sent

5. Share (c_1, c_2) as the ciphertext.

Decryption

1. Receive the ciphertext (c_1, c_2)

2. Calculate the shared secret $s = c_1^x \ mod \ p = g^{xy} \ mod \ p$

3. Get the message $m = (c_2 * (s^{-1} \ mod \ p)) \ mod \ p$, where $s^{-1} \ mod \ p = s^{p-2} \ mod \ p$ (Using Fermat's Little Theorem)