# Evaluation 5

To develop a fault tolerant routing, we can encode the message using a $k-1$ degree polynomial over a sufficiently large field $F_p$, st $p$ is a prime, $p > 2^b$ and $|f_p| > n$

To send the message,

1. Split message into $k$ blocks
   $m = m_0, m_1, \ldots m_k$
2. Construct a $k-1$ degree polynomial
   $M(x) = m_0 + m_1 x + m_2 x^2 + \ldots m_{k-1} x^{k-1}$
3. Take $n$ points on the polynomial where $n > k+e$
4. Sign each point digitally
5. Send each point and sign through a separate channel.

If any block is corrupted during transmission, we can verify the signature.

When the message is recieved, if we are left with atleast $k$

blocks which are not corrupted, we can reconstruct the polynomial and get back the coefficients.

Now to design an oblivious transfer protocol, we assume that we have a robust routing scheme available. Also, the public keys of the server and client are known.

Protocol:

1. Client makes a random array
$R = \{ Enc_{PK(B)}(r_1), Enc_{PK(B)}(r_2), \ldots, Enc_{PK(B)}(r_k) \}$
where $PK(B)$ = public key of server
2. Client then sends this to the server using our robust routing protocol.
3. Server then reconstructs the array R

~~4. Server creates a new array B st.~~

4. Server then decrypts the array and XOR each element with the corresponding data element.
$\{ Dec_{PK(B)}(r_1 \oplus b_1), \ldots, (r_0 \oplus b_0), \ldots Dec(r_k \oplus b_k) \}$

5. Server then sends this new array again using the robert routing to the client.

6. client then XORs the $i^{th}$ element which is required with $r_i$ which gives
$(r_i \oplus b_i) \oplus r_i = b_i$

- This protocol requires that both public keys of server and client are known and that only $r_i$ is encrypted from the client side otherwise the client can retrieve other data elements too.

(a) and (c) when public key of both are known or public key of server is known to client, then the algorithm provided works and we can encrypt using public key of server.
The maximum permissible error
$e \leq (n - k)$

(b) and (d). public key of server is not known to client.

In this case, client first needs to gets server's keys and for that we can use our robust mechanism. So if our public key is of s blocks, then. $e \leq \min(n-k, n-s)$. where n is total number of channels and k is the number of message blocks to be transferred.

For encryption / decryption mentioned before, we can use El-Gamal.