Sarthak Singhal
(20171091)

PQR-5

3rd May 2020

Question

Recall that in Evaluation II, k blocks of data/information is encoded into n blocks such that if any e of the n blocks are corrupted, it is still possible to retrieve the original k blocks of information.

1. Show how to use this to build a robust (torrent like) 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.
2. Further, 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 (torrent) routing mechanism (you may have to design four different protocols in the four cases outlined below).
3. What do you think would be the maximum tolerable e when
    a. the public-keys of A and B are known to all
    b. public-key of A is known to B, but B's public-key is not known to A
    c. public-key of B is known to A, but A's public-key is not known to B
    d. Neither party knows the public-key of the other party.

Answer

Ans 1)

We can use the same concept as used in the evaluation 2. To send K blocks, we can create a polynomial of degree K-1 with these K blocks as its coefficients. Then we can take N points on the polynomial and we've shown that we can construct the polynomial even if E out of these N blocks are corrupted. So to build a robust routing scheme, we can send each of the N encoded blocks along a channel. Even if E channels are corrupted, we will get blocks from the remaining N-E channels and can easily reconstruct the polynomial hence the original K blocks. Theoretical proofs of the protocol are explained in the previous evaluations.

Ans 2)

Oblivious Transfer

In cryptography, an oblivious transfer (OT) protocol is a type of protocol in which a sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece (if any) has been transferred. Suppose A has an index *i* and B has an array of N elements. A wants to know b_i without revealing *i* to B and B wants to send the required information without revealing anything else. This is a kind of oblivious transfer and can be formally explained as:

- A has an index i
- B has an array $B = [b_1, b_2, ...b_n]$
- A wants to know b_i such that:
    - B doesn't know *i*
    - A doesn't know b_j for $j \neq i$

ElGamal Encryption

In cryptography, the ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange. It consists of three components: the key generator, the encryption algorithm, and the decryption algorithm.

- Key Generation:
    - Construct a cyclic group G of order q with generator g
    - Chose an integer x randomly from {1,...,q-1}
    - Compute $h = (g^x)\%q$
    - Public key consists of <q,g,h> and x is the private key
- Encryption:
    - Map the message M to an element m of G using reversible mapping function

    - Choose an integer y randomly from {1,...,q-1}
    - Compute $s = (h^y)\%q$
    - Compute $c_1 = (g^y)\%q$
    - Compute $c_2 = (m * s)\%q$
    - Send $[c_1, c_2]$ as encrypted message
- Decryption:
    - Compute $s = ({c_1}^x)\%q$
        - Since $c_1 = (g^y)\%q$
        - $\Rightarrow s = (g^{x*y})\%q$
          $= (h^y)\%q \ as \ h = (g^x)\%q$
          $= same \ s \ as \ with \ the \ sender$
    - Compute s^{-1}
    - Compute $m = (c_2 * s^{-1})\%q$
        - Since $c_2 = (m * s)\%q$
        - $\Rightarrow (c_2 * s^{-1})\%q = (m * s * s^{-1})\%q = m$

General two party OT using ElGamal Encryption

Let Alice has an index *i* and Bob has the array B and Alice wants to know b_i

- Alice constructs a random array A and sends it to B with the ith element encrypted using ElGamal scheme.
  $A = [r_1, r_2, ..., r_n] \ where \ r_i = Enc(r)$
- Bob constructs an array D by decrypting the entire array A
  $D = [Dec(r_1), Dec(r_2), ..., r \ , ..., Dec(r_n)]$
- Bob constructs $DB = [D[i] \ \oplus \ B[i] \ ] \ \forall \ i \ \in \ n$ and sends it to Alice
- Alice obtains b_i as $b_i = DB[i] \ \oplus \ r$

OT using ElGamal Encryption over the network

Similar to general two party OT, we can do OT over the network by using the above routing scheme. Basically we want to transfer arrays between

Alice and Bob. Let their size be K. To do so, we take each array element as a data block and create a polynomial of degree K-1 with these blocks as its coefficients. Then we can encode them into N blocks and transfer over N channels using our routing scheme. This is how we’ll implement the OT over a network using the above routing scheme.

Ans 3)

For successful OT between A and B:

- B should know A’s public key and A should know B’s public key.
- When we encode K blocks into N blocks, B needs A’s public key to verify whether the packets received from the channel are corrupted or not i.e. to verify if the signature corresponds to the message.
- For doing OT, A needs B’s public key so that it can encrypt the message and send it to B.

A. **the public-keys of A and B are known to all**
   In this case we can do OT through the above routing scheme easily. Only restriction required is for the routing protocol where we proved that for error free transfer $n \geq k + e$. Thus:
   $e \leq n - k$
   $\Rightarrow e_{max} = n - k$

B. **public-key of A is known to B, but B’s public-key is not known to A**
   Since B knows A’s public key, B can send any message safely to A. Thus first B will send its public key to A and then both parties would know each other’s public keys.
   Let s_b be the size of B’s public key. Thus:
   $n \geq s_b + e$
   $\Rightarrow e \leq n - s_b$
   For normal OT when public keys are known $e \leq n - k$
   Thus combining both we have:
   $e \leq min(n - s_b, n - k)$
   $\Rightarrow e_{max} = min(n - s_b, n - k)$

C. **public-key of B is known to A, but A’s public-key is not known to B**
   Since A knows B’s public key, A can send any message safely to B.

Thus first A will send its public key to B and then both parties would know each other's public keys.
Let s_a be the size of A's public key. Thus:
$n \geq s_a + e$
$\Rightarrow e \leq n - s_a$
For normal OT when public keys are known $e \leq n - k$
Thus combining both we have:
$e \leq min(n - s_a, n - k)$
$\Rightarrow e_{max} = min(n - s_a, n - k)$

D. Neither party knows the public-key of the other party.
Since neither party knows each other's public key, we first need to share one key. Since we do not have access to public keys, we can share the key without using digital signatures. Let us share A's public key. Using this, according to coding theory we have $n \geq s_a + 2e$
$\Rightarrow e \leq (n - s_a)/2$
Now B has A's public key, so it is similar to (B part) where
$e \leq min(n - s_b, n - k)$.
Combining both, we get:
$e \leq min(n - s_b, n - k, (n - s_a)/2)$
$\Rightarrow e_1 = min(n - s_b, n - k, (n - s_a)/2)$
Similarly if we share B's public key, we have:
$\Rightarrow e_2 = min(n - s_a, n - k, (n - s_b)/2)$
$\Rightarrow e_{max} = max(e_1, e_2)$