Evaluation 5

2017I076

1. Show how to build a robust, torrent-like routing scheme where sender (A) and receiver (B) have n different routes and the task is to send k blocks such that there is no loss of information even if e routes are corrupted.
2. Using a PKC system design a reliable OT system such that client A (with index) can communicate with server B (with array) with the routing scheme described above.
   Determine maximum tolerable e when:
   i) all PKs public
   ii) B knows PK(A) only
   iii) A knows PK(B) only
   iv) No PKs public.

---

1. With the assumption that PKC exists, and all public keys are known.
   Much like the problem of fault-tolerance, we can encode the message using a $k-1$ degree polynomial over a sufficiently large field F_p, s.t. p is a prime
   and $p > 2^b$
   and $|F_p| > n$

Sending the message

The overall idea is:

- Split message into k blocks.
  $m = m_0, m_1, \ldots m_k$
- Construct a $(k-1)$ degree polynomial $M(x)$ with co-efficients m_i
  $M(x) = m_0 + m_1 x_1 + \ldots + m_{k-1} x^{k-1}$
- Evaluate $M(x)$ at n points ($n > k+e$) and digitally sign each result. (so c_i, $\text{sign}(c_i)$)
- send each of the n c_is on its own channel.

- **Corruption Detection:** corruption detection can be done by verifying the signature of the sent block.

  Proof: given the assumption that PKC exists, and any adversary is at best PPTM, there is negl() probability of tamper going undetected.

- **message reconstruction:** With redundancy of e, there are $\geq k$ blocks left over. We can then recover the message by polynomial interpolation (since it was encoded by a polynomial of degree $k-1$)

2. **Designing an Oblivious Transfer (OT) protocol:**

- Client: A, requesting data for index i. Must not know if $request_i \neq request_j$
- Server: B, has array $b = \{b_1, b_2, \ldots b_k\}$ and must not know requested index i.
- Public key of A, B: PK(A), PK(B)

Protocol:

1. A makes an array $\{r_1, r_2 \ldots Enc_{PK(B)}(r_i), \ldots r_k\}$, $r_j \in_R S$
2. A sends it to B using error-resistent protocol above, so it sends n blocks across n different channels.
3. B gets $n-e$ blocks and reconstructs the original message $\{r_1, r_2 \ldots Enc_{PK(B)}(r_i), \ldots r_k\}$
4. B XORs it with b

   new array $= \{Dec_{K(B)}(r_1) \oplus b_1, \ldots r_i \oplus b_i, \ldots Dec_{K(B)}(r_k) \oplus b_k\}$

   and sends it to A with the protocol, sending n blocks.
5. A gets $n-e$ blocks and reconstructs the message from B $\{Dec_{K(B)}(r_1) \oplus b_1, \ldots r_i \oplus b_i, \ldots Dec_{K(B)}(r_k) \oplus b_k\}$
6. A XORs the i^{th} element of the returned array with r_i (that A knows) to get $(r_i \oplus b_i) \oplus r_i = b_i$

Notes:

1. This protocol requires the client A know the public key of server B
2. It also requires that A only sends r_i encrypted, if it sends all of $r_{1 \to k}$ encrypted then it can retrieve information about the entire array.
   Solution: make A do a ZKP each time it generates the array. If B is not satisfied, it can refuse the transaction.

Scenarios:

1. PKs of both A and B are known

AND

3. PK(B) known to A, PK(A) not known to B
   - A can use PK(B) to Encrypt.
   - For error-free transmission, max permissible $e \leq (n-k)$

2. PK(A) known to B, PK(B) not known to A

AND

4. PK(A),(B), Unknown
   - Since A does not know PK(B), it cannot encrypt r_i.
   - We need to first send PK(B) to A through a fault-tolerant channel.
   - But we cannot use the one described above as PKs are not known, so we cannot digitally sign and verify messages. We can opt for traditional fault tolerance, requiring a fault tolerance of at least $2(n-k)$.
   - Let PK(B) = block size s
   - $\therefore$ $e \leq \min(2(n-k), 2(n-s))$

Appendix: El Gamal Encryption_Decryption

Keygen:

- take p, generator g for $\mathbb{Z}_p$
- pick $x \in_R \{1, 2, \ldots p-1\}$
- $h = g^x \bmod p$
- public key: $\{h, p, g\}$

Enc:

- m will be encrypted
- pick $y \in_R \{1, 2, \ldots p-1\}$
- $s = h^y \bmod p$

  $c_1 = g^y \bmod p$

  $c_2 = ms \bmod p$
- send (c_1, c_2) as encrypted message.

Dec:

- $s = c_1^x \bmod p$
- $s^{-1} = s^{p-2} \bmod p$
- $m = c_2 s^{-1}$