Implementation of the Mutual-Restraining E-Voting System

1 Project Description

1. The purpose: besides learning, using, and mastering cryptography and cybersecurity principle/knowledge and skills, learn and master Internet programming and protocol-based message exchange to conduct inter-communication of different processes running on different machines using socket (or secure socket).
2. Use the following location anonymization scheme for voters to obtain their respective unique secret locations.
3. Implement your e-voting system by using the standardized protocol as described
4. Start your project with minimum/core components.
    - Collectors: share generation and distribution of shares to voters.
    - Voters: computation of their own shares and ballots and transmission of their ballot to collectors.
    - Collectors: aggregation of the ballots to get the tallied voting vector, then the converted plain votes, and finally the tally of the candidates votes.
5. Do not need to consider the following at the beginning but may add them one by one later to enhance the functionality and usability of your implemented e-voting system.
    - No GUI is required. Of course, GUI-based or browser-based implementation is great if your team is able to do so.
    - No election creation is required. You can predefine an election involving 2 or more candidates and such a predefined election can be provided to each voter client as an command-line input option/parameter. Of course, if your team implements random election creation, that would be great.
    - Do not need multiple machines. Using 127.0.0.1 (loop-back address) run multiple processes on the same machine to inter-communicate with each other. On one machine, open multiple terminals and each terminal runs one process. The process can be one of the collectors or one of the voters. Of course, if you have and can run your e-voting system on multiple machines , that would be great.
    - No in-process enforcement (sub protocols 1 and 2) is required. of course, if your team implement it, that would be great.
    - Enclosed is Paillier cryptosystem. You can create your own Paillier public and private keys by simply generating two primes of same length.

2 An efficient Location Anonymization Scheme (LAS)

2.1 LAS principle

Location anonymity can be achieved using a variation on STPM [1] nicely as follows.

- First, one of the collectors, such as $\mathcal{C}_1$ creates a Paillier cryptosystem [2] and gives its public key to the other collector, such as $\mathcal{C}_2$.
- $\mathcal{C}_1$ also creates a random permutation π_1 of the integers from 1 to n, where n is the number of voters. $\mathcal{C}_1$ encrypts these values using its public key to which only $\mathcal{C}_1$ itself has the private key, resulting in $\langle y_1, y_2, \ldots, y_n \rangle$, where $y_i = E(\pi_1(i))$ for $1 \leq i \leq n$. $\mathcal{C}_1$ passes these values to $\mathcal{C}_2$.
- $\mathcal{C}_2$ re-permutes them with random permutation π_2 and generates n random numbers $r_{2,i}$ $(1 \leq i \leq n)$. $\mathcal{C}_2$ then computes
  $\langle y_{\pi_2(1)} E(r_{2,1})^{-1}, y_{\pi_2(2)} E(r_{2,2})^{-1}, \ldots, y_{\pi_2(n)} E(r_{2,n})^{-1} \rangle$.
  $\mathcal{C}_2$ sends them back to $\mathcal{C}_1$.
- $\mathcal{C}_1$ decrypts each value to obtain $\langle r_{1,1}, r_{1,2}, \ldots, r_{1,n} \rangle$.
- Each collector j $(1 \leq j \leq 2)$ needs to send voter i $(1 \leq i \leq n)$ the value $r_{j,i}$ at some point (e.g., following the order of registration or logging into to vote). Then, the voter's secret location is $r_{1,i} + r_{2,i}$.

This technique is similar to STPM [1], i.e., $x_1 * x_2 = r_1 + r_2$; $\mathcal{C}_1$ has each x_1 in $(1, n)$ and $\mathcal{C}_2$'s value x_2 is effectively 1 always. Also, unlike STPM, $\mathcal{C}_2$ permutes the received values before sending any information back. Thus, $r_1 + r_2$ is a permuted value in $(1, n)$ by permutations π_1 (only known and done by $\mathcal{C}_1$) and π_2 (only known and done by $\mathcal{C}_1$). However, neither $\mathcal{C}_1$ nor $\mathcal{C}_2$ knows what $r_1 + r_2$ is.

2.2 LAS protocol/message exchange

To perform LAS, collector 1 first creates a random permutation of $\{0, 1, \ldots, N-1\}$. Let this permutation be represented by $\pi_1(i)$. They initialize a Paillier cryptosystem, encrypt each permuted value, and send the encrypted and permuted list to collector 2, $\langle y_0, y_1, \ldots, y_{N-1} \rangle$, where $y_i = E(\pi_1(i))$ for $0 \leq i < N$.

Signed and Encrypted	
message_type	TYPE_LAS1
key_hash	64 bytes
election_ID	16 bytes
n_length	4 bytes
n	var. length
value1_length	4 bytes
value1	var. length
...	...

Collector 2 applies their own random permutation and sends collector 1
$\langle y_{\pi_2(0)} E(r_{2,0})^{-1}, y_{\pi_2(1)} E(r_{2,1})^{-1}, \ldots, y_{\pi_2(N-1)} E(r_{2,N-1})^{-1} \rangle \pmod{n^2}$,
where for each $0 \leq i < N$, $r_{2,i}$ is a random number such that $0 \leq r_{2,i} < n$.

Signed and Encrypted	
message_type	TYPE_LAS2
key_hash	64 bytes
election_ID	16 bytes
value1_length	4 bytes
value1	var. length
...	...

Finally, collector 1 takes the received vector $\langle z_0, z_1, \ldots, z_{N-1} \rangle$ and computes $r_{1,i} = D(z_i)$ for $0 \leq i < N$.

Both collectors obtain $R_{j,i}$ from $r_{j,i}$ $(0 \leq i < N)$ by subtracting n from $r_{j,i}$ if $2r_{j,i} \geq n$ (otherwise $R_{j,i} = r_{j,i}$).

2.3 Paillier Cryptosystem

The Paillier cryptosystem is an asymmetric cryptosystem. Importantly, it has the homomorphic property that two ciphertexts can essentially be multiplied together to get an encryption of the sum of the two corresponding plaintexts. Also note that unlike with RSA, with the Paillier cryptosystem, there are multiple ciphertexts for each possible plaintext.

2.3.1 Details

This material is based on [3]. See that page for more details. Here we will focus on the Paillier cryptosystem as it will be used in the e-voting system.

To generate a key, two large primes p and q can be chosen of equal length. For the e-voting system, choose these so that they are 2048 bits each.

Let $n = pq$, $\lambda = (p-1)(q-1)$, and $\mu = \lambda^{-1} \pmod{n}$.

n can be used as the public key, whereas λ and μ together can be used as the private key.

To encrypt a message m where $0 \leq m < n$, first select random $0 < r < n$ such that r and n are coprime (their greatest common denominator is 1). The ciphertext is then $(n+1)^m r^n \pmod{n^2}$. Note that we are letting $g = n + 1$.

To decrypt a ciphertext c where $0 < c < n^2$ and c is coprime to n, compute $\mu \lfloor \frac{(c^\lambda n^2)-1}{n} \rfloor \pmod{n}$.

Note that in Java, floor division on positive numbers can be performed using the `BigInteger.divide(BigInteger)` method.

2.3.2 Example

$$p = 46183$$

$$q = 48907$$

$$n = 2258671981$$

$$\lambda = 2258576892$$

$$\mu = 698630339$$

$$m = 123456$$

$$r = 37404609$$

$$c = 2035022520095106479$$

The decryption of c is 123456.

References

[1] S. Samet and A. Miri, "Privacy preserving ID3 using Gini index over horizontally partitioned data," in *Proc. of the 2008 IEEE/ACS*, ser. AICCSA '08, Washington, DC, USA, 2008, pp. 645–651. [Online]. Available: http://dx.doi.org/10.1109/AICCSA.2008.4493598

[2] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology — EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer Berlin / Heidelberg, 1999, vol. 1592, pp. 223–238, 10.1007/3-540-48910-X_16. [Online]. Available: http://dx.doi.org/10.1007/3-540-48910-X$_1$6

[3] Wikipedia contributors, "Paillier cryptosystem — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Paillier$_c$*ryptosystemoldid* = 1000112107, 2021, [*Online*].