Project Progress Report

- Arti Gupta 50170010
- Aditya Nalge 50207629

1. Articles or documentation read so far (the type of reading material, the number of documents, etc.) before and after project proposal submission.

BEFORE

1. http://www.cs.cornell.edu/courses/cs754/2001fa/307.PDF
2. https://en.wikipedia.org/wiki/Threshold_cryptosystem
3.https://www.dcl.hpi.uni-potsdam.de/teaching/cloudsec/presentations/threshold-cryptography.pdf

Initially, we referred first paper where we learnt about threshold schemes, ElGamal cryptosystems and using Lagrange's interpolation for modifying the shadow. That paper also talks about proving the security of this approach while highlighting the failures with RSA and decryption we are using this paper. Also, we studied Wikipedia page and PPT to understand concepts thoroughly.

AFTER

For Key Generation and Decryption Algorithm

1. https://www.cryptoworkshop.com/ximix/lib/exe/fetch.php?media=pedersen.pdf
This paper shed light on how to implement a Threshold Cryptosystem without having a trusted third party.

To understand Shamir's Secret Sharing in detail, we are referring following:

2. https://cs.jhu.edu/~sdoshi/crypto/papers/shamirturing.pdf
This paper shows how to divide data D into n pieces in such a way that D is easily re-constructible from any k pieces, but even complete knowledge of k-1 pieces reveals absolutely no information about D.

3. http://www.asecuritysite.com/Encryption/shamir
This website helped us to practically understand how Shamir's Secret Sharing works with real examples.

For Digital Signatures:

4. http://faculty.uml.edu/klevasseur/math/RSA_Signatures/RSA_Signatures.pdf
Paper explaining implementation of Digital Signatures using RSA.

2. Any changes to the original project proposal and/or additional detail about its implementation (anything that you discovered after starting working on the project, but which was not obvious at the project proposal time).

Ans: *After submitting project proposal, we realized that we had some misconceptions and gaps in our knowledge about threshold cryptography.*

1. *We had mentioned that we could maintain Data Integrity using hash functions. However, we learnt that hash functions do not provide sufficient guarantee. That's because the integrity of the data can still be compromised if an attacker modifies the data and calculates a new hash value for this corrupted data. The attacker can then replace the old hash value with the new hash value. As advised by the instructor, the Digital Signature technique, that we are implementing for non-repudiation will provide data integrity guarantee as well.*

2. *Initially, we planned to do encryption using RSA algorithm. But, we figured out that since Lagrange Interpolation cannot be used with RSA [Paper - Threshold Cryptosystems]. We then decided to use ElGamal Encryption.*

3. *One of the major changes that we learnt after submitting the original project proposal was the misconception we had related to key sharing. Initially we were going to assign each user a unique/separate key and then, each user will enter their own key into server and data will be decrypted. However, the key that will be generated using El Gamal Encryption algorithm will be divided into parts and the key share will be provided to each user individually such that no user will have any information regarding other users or their keys. We are planning to implement it in such a way that after receiving its key, each user will keep it in encrypt form and when key shares from a threshold number of users are combined, it can be used for decryption to recover original data.*

3. If the instructor asked for clarifications or additional project details in the feedback to project proposal, include any new information here.

Ans: *The instructor wanted to know what resources we were protecting. "Data" is the main resource we will be protecting using Encryption.*
Is Non-Repudiation also a goal of our project was another query that the instructor had. Yes, using digital signatures to ensure Non-Repudiation is a goal of our project.
The instructor also wanted to know what Cryptographic construction we were going to use. After thorough research and reading about the various pros and cons of different encryption schemes that are used to implement a threshold cryptographic scheme, we decided to use ElGamal Encryption and Decryption as our cryptographic construction. The main stages of our construction are:
i. Key Generation
ii. Encryption using ElGamal Public Key Cryptography.
ii. Implementing Shamir's Secret Sharing.
iv. Implementing Digital Signatures.
v. ElGamal Decryption using the key shares.

Key Generation

Here, n members of a group (P1,...,Pn) can select a public key of the form (p,q,g,h) where g,h $\in G_q$ and the corresponding secret key is $x=\log_g h$, such that a fixed parameter k ($1 \le k \le n-1$), k members must cooperate to use the secret key. It is assumed that $n > 2K - 1$.
As (k,n)-threshold schemes allow at most k-1 cheating participants, this means that a majority of the participants is assumed to be honest. The protocol should be generalized such that L members select the key and distribute it to the n members of the group where $n \ge L \ge 2k-1$. It is assumed that the members of the group have previously agreed on the primes p and q.

Let C(m,T) denote a commitment to m $\in \{0,1\}^*$ using the random string T . Then the keys are selected as follows:

1. *Pi chooses $X_i \in Z_q$, at random (uniform distribution) and computes $h_i = g^{x_i}$. Then a random string r_i is chosen and $C_i = C(h_i, r_i)$ is broadcast to all members.*
2. *When all n members have broadcast a commitment, each Pi opens C_i.*
3. *The public key, h, is computed as h = product of h_i (from i == 1 till i == n).*

Now all members know public key but they cannot find the secret key x = sum of Xi (from i == 1 till i == n)

unless they all work together (or some of them can find discrete logarithms). If Pi chooses Xi at random then the distribution of the secret key is polynomial indistinguishable from the uniform distribution.

X can be distributed such that any K members can find it if necessary. The proposed method is as follows :

1. *P_i chooses at random a polynomial $f_i(z) \in Z_q(z)$ of degree at most k-1 such that $fi(0) = X_i$.*
   Let, $f_i(z) = f_{i0} + f_{i1} + .. + f_{i,k-1}z^{k-1}$ Where $f_{i0} = X_i$.
2. *P_i computes $F_{ij} = g^{fij}$ for j = 0,....,k-1 and broadcasts (F_{ij}) j =1,...,k-1 ($F_{i0} = h_i$ is known beforehand)*
3. *When everybody have sent these k-1 values, P_i sends $S_{ij} = f_i(j)$ secretly and a signature on S_{ij} to P_j for j = 1,...,n*
4. *P_i verifies that the share received from , $P_j(S_{ji})$ is consistent with previously established values by verifying that*

$$g^{s_{ji}} = \prod_{l=0}^{k-1} F_{jl}^{i^l}.$$

   ***If** this fails, P_i broadcasts that an error has been found, publishes **S_{ij}**, and the signature and then stops.*
5. *P_i computes **his** share of x **as** the **sum** of **all** shares received in step 3 **S_i** = (Sum of all S_{ji} from j=1 to j=n). Finally Pi signs h.*

*When all members have signed **h,** a key authentication center verifies the signatures, and if they are correct, it makes a certificate showing that **h** is the public key of the group.*

***[REFERENCE* - https://www.cryptoworkshop.com/ximix/lib/exe/fetch.php?media=pedersen.pdf]**

Encryption using ElGamal Public Key Cryptography

The global elements of ElGamal are a prime number q and α, which is a primitive root of q. User A generates a private/public key pair as follows:

- *Generate a random integer X_A, such that $1 < X_A < q - 1$.*
- *Compute $Y_A = a^{XA} \bmod q$.*
- *A's private key is X_A and A's public key is {q, α, Y_A}.*

Any user B that has access to A's public key can encrypt a message as follows:

- *Represent the message as an integer M in the range $0 \le M \le q - 1$. Longer messages are sent as a sequence of blocks, with each block being an integer less than q.*
- *Choose a random integer k such that $1 \le k \le q - 1$.*
- *Compute a key $K = (Y^A)k \bmod q$.*
- *Encrypt M as the pair of integers (C1, C2) where $C1 = \alpha^k \bmod q$; $C2 = KM \bmod q$*

Shamir's Secret Sharing Methodology

*By using (k,n) threshold scheme (k = threshold number of keys, n = total number of parts) with **n = 2k – 1**, we get a very robust key management scheme. We can recover original key even when [n/2] = k-1 of the n parts are destroyed, but our opponents cannot reconstruct the key even when security breaches expose*

[n/2] = k-1 of the remaining k parts.

***Explanation:** Based on polynomial interpolation; suppose we are given k-points in the 2-dimensional plane $(x_1,y_1),....,(x_k,y_k)$ with distinct x_i's, there is one and only one polynomial q(x) of*

degree k-1 such that $q(x_i) = y_i$ for all i. Without loss of generality, we can assume that data D is a number. To divide it into parts D_i, we can pick a random k-1degree polynomial $q(x) = a_0 + a_1x^{k-1} + \ldots + a_{k-1}x^{k-1}$ in which $a_0 = D$, and we will calculate:

$$D_1 = q(1), \ldots, D_i = q(i), \ldots, D_n = q(n)$$

Now, for any given subset of k of these D_i values, coefficients of q(x) and D = q(0) by interpolation can be calculated, however, by knowing just k-1 values, does not suffice in order to calculate D. Since, by using modular arithmetic, if we choose a prime p greater than both D and n and compute values $D_1, \ldots, D_n$ in terms of modulo p, we find that for each candidate D` in [0,p) integers, the opponent can construct one and only one polynomial q`(x) of degree k-1 such that q`(0) = D` and q`(i) = D_i for given k-1 arguments and these possible p are equally likely by construction, thus, the opponent will never be able to deduce anything about the real value of D. Thus, our system will be secure.

[Reference:https://cs.jhu.edu/~sdoshi/crypto/papers/shamirturing.pdf]

Implementing Digital Signatures

For any RSA public/private key triple (e, d, n) where the public key is (n,e) and the private key is (n,d) the key mathematical fact is that the encryption and decryption functions are inverses of one another. That is, if

$f(m) = m^e \bmod n$ is the encryption function (which is public)
$g(m) = m^d \bmod n$ is the decryption function (which is private)
then, $f(g(m)) = m$ and $g(f(m)) = m$

The idea behind a digital signature using RSA is that f is a function that is known to everyone, but only you know your decryption function. When Alice signs a message, m, she sends $g_A(m)$ together with an indication that the message is from her. When Bob gets it, he sees that the message is from Alice and applies her public encryption function, f_A, to $g_A(m)$ and will get m.

Here a disadvantage is that any user who knows Alice's public key can decrypt the message. As a result, encryption is applied in addition to signing the message. To do that, we just encrypt the message using the public key of the receiver before performing the step mentioned above.

Assume,

Alice Private Key - g_A *Bob Private Key - g_B*
Alice Public Key - f_A *Bob Public Key - f_B*

If Alice wants to send an encrypted and signed message to bob, she will:
i. Encrypt the message say 'm' using Bob's public key - $f_B(m)$
ii. Encrypt this with own private key - $g_A(f_B(m))$

Bob will receive the message and he will:
i. Use Alice's public key - $f_A(g_A(f_B(m))) = f_B(m)$ *[since $f_A * g_A$ is an identity function]*
ii. Use own private key - $g_B(f_B(m)) = m$ *[since $f_B * g_B$ is an identity function]*

Thus, the first step helps authenticate that the received cipher was indeed sent from Alice because Bob was able to decrypt it using Alice's public key. The second step helped maintain confidentiality as only Bob would be able to decrypt the cipher using his private key. This is how Digital Signatures can be implemented using RSA

ElGamal Decryption using key shares

Using the key shares obtained from a threshold number of users, the original key can be recovered. ElGamal uses public key {q,α,Y_A} for encryption and {X_A} for decryption. X_A will be divided into key shares which would be given to the different users. Shamir's Secret Sharing will be used to generate these key shares.
For decryption, these key shares will be obtained from a threshold number of users and will be used to recover the private key X_A temporarily; which will be used to decrypt the data.

Data will be decrypted using X_A and the Cipher Text (C1,C2) as:

- *Recover K by computing $K = (C1)^{XA} \bmod q$*
- *Compute $M = (C2\ K^{-1}) \bmod q$*

The ElGamal Cryptosystem

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

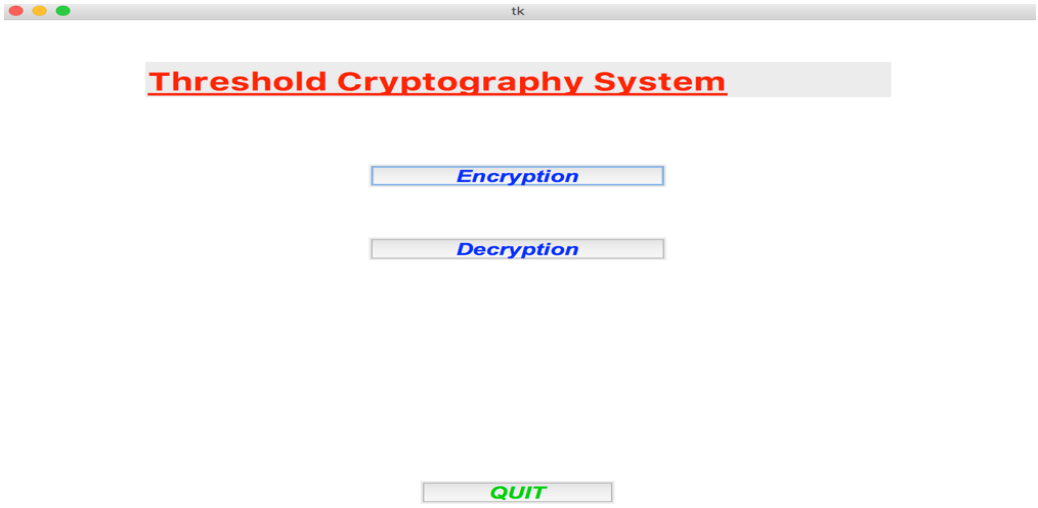
Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

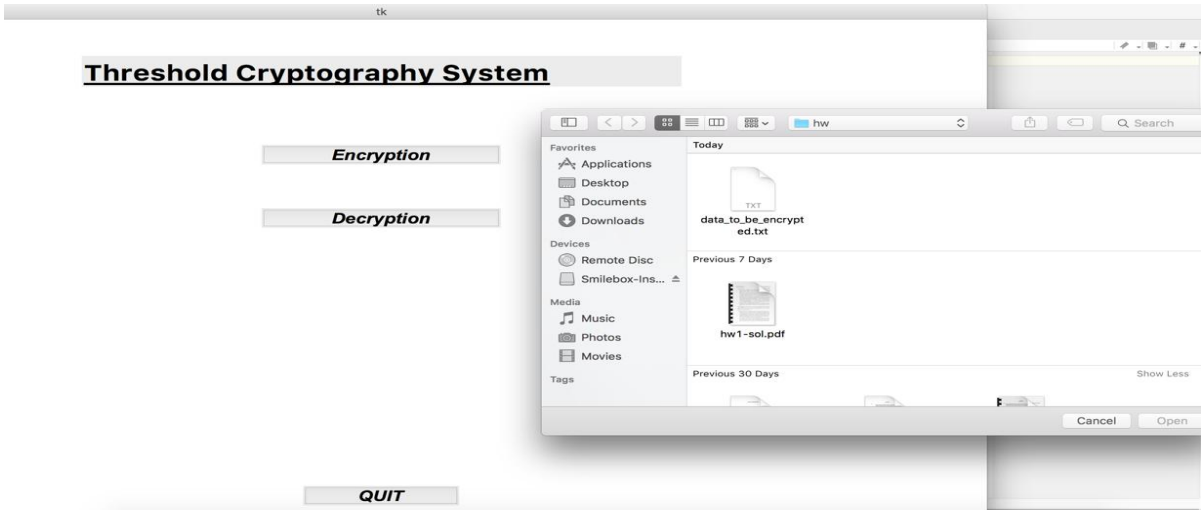
Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

4. Functionality implemented so far (in terms of features of the projects, functions, etc.).
Ans: *We are implementing Threshold Cryptography that will be done in 4 stages and 1 UI phase. We are already done with 1st stage i.e. encryption of Threshold cryptography using ElGamal Algorithm and generating public/private keys using python programming language. Also, we are designing UI using Tkinter library in python.*
The main functionality that we have implemented as of now is Encrypting the data using ElGamal algorithm.
The second functionality we have started implementing is generating the key without using a trusted third party. For this we are referring the paper "A Threshold Cryptosystem without a Trusted Party" We have made a few screens.

Screenshots:



After clicking on Encryption button, it asks for data file which needs to be encrypted:



And then, data gets encrypted using encryption algorithm we are using. Once, encryption and secret sharing is done, following screen will be reflected.

Till now, we are done till this step. Remaining parts, we are planning to do in second half of project submission.
Also, we are half way towards implementing main functionality i.e. right technique to make sure, our project is secure. Other stages will be completed in the latter half of the project.
Libraries used as of now:

1. Crypto.PublicKey
2. ElGamal (This library can be found in Crypto library and ElGamal keys allows us to perform basic signing, verification, encryption, and decryption)
3. Crypto.Hash

5.Any difficulties encountered so far (resolved or unresolved); you can include any questions you have for the instructor.

Ans: *Initially, we had minor issues understanding the crux of the project clearly like Key Generation, Data Integrity and Security. However, we keep meeting the instructor at regular intervals as well as research through several papers, and as a result, we are able to get most of our doubts resolved. Still, we would like to have more clarifications regarding the applications of Digital Signatures.*

6. An estimate on the percentage of the project completed; and note on the major milestones that still remain.
Ans:
Our estimate on the percentage of project that has been completed is 35% We are using Python programming language for our project.

S.No.	***Tasks to be done in Project***	***Weightage of each part***	***Milestones Achieved/Remaining***
1.	*Encrypting Data using Elgamal Algorithm*	*15%*	*Done*
2.	*Key Generation*	*20%*	*Started Implementing*
3.	*Implementing Shamir's Secret Sharing Concept*	*15%*	*Still needs to be done*
4.	*Digital Signatures Implementation*	*15%*	*Still needs to be done*
5.	*Decrypting Data with the collaboration of Threshold no. of k out of n parties (Later, testing of Threshold System to validate if system is secure enough)*	*20%*	*Still needs to be done*
6.	*UI Design (Using Tkinter library in Python)*	*15%*	*For the initial stages; its done*

The major milestones that still remain are:

- *Implementing Shamir's Secret Sharing.*
- *Implementing Digital Signatures.*
- *Decrypting using threshold no. of modified key shares.*
- *Testing our threshold cryptographic system.*

BLOCK DIAGRAM OF THE DIFFERENT STAGES IN THRESHOLD CRYPTOSYSTEMS