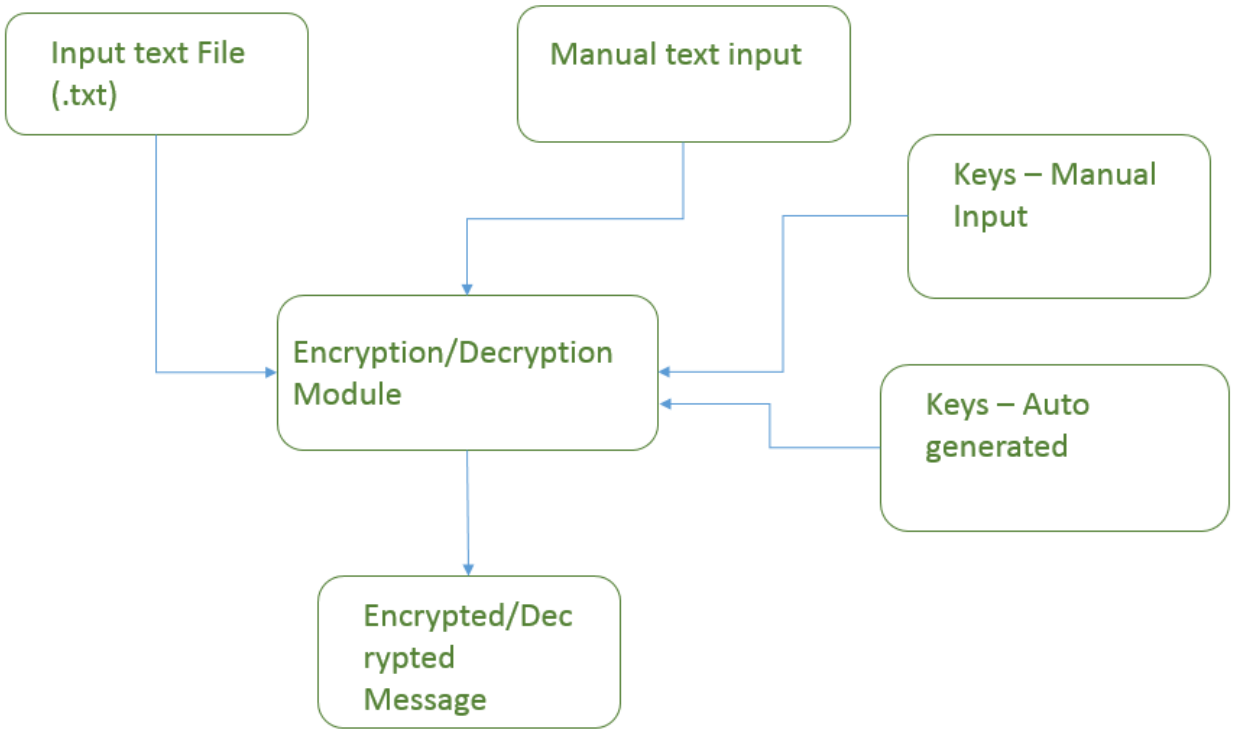
# CS5770: Course project –Project proposal

**Team Members: Akshay Kulkarni, Aravindhan Eswaran, Karthik Sambamoorthy**

We propose to develop a MATLAB based GUI tool, that performs encryption/decryption using three very prevalent and popular Cryptosystems namely, 1) The RSA 2) The ElGamal and 3) The AES (Advanced Encryption Standard). The tool also has an auto key generation functionality, the user has the flexibility of inputting his own keys for encryption and decryption or choose auto generate which will generate fresh keys each time. The overall design and architecture of the module is presented below, all three systems abide by the same design.




1) RSA Cryptosystem: This is one of the most popular Cryptosystems in use today, it enables privacy and authenticity for the data being transmitted. It is used by web servers and browsers to secure web traffic, it is used to ensure privacy and authenticity of Email, it is used to secure remote login sessions, and it is at the heart of electronic credit-card payment systems. It is a public key Cryptosystem, i.e. it allows a user to publish a public key, which is used by the sender for encryption and decrypt using a private key. The following equations describe the properties of the numbers involved and the encryption/decryption procedure:

$i)\ Find\ two\ disctinct\ primes\ p, q\ and\ compute\ N = p * q$
$ii) Find\ the\ Euler's\ totient\ of\ N\ i.e. \emptyset(N) = (p-1) * (q-1)$
$iii) Find\ a\ positive\ Integer\ e\ \ni \gcd(e, N) = \gcd\big(e, \emptyset(N)\big) = 1$
$iv) Find\ a\ second\ Positive\ Integer\ d\ \ni e * d \equiv 1 mod \emptyset(N)$
$v)\ For\ encrypting\ a\ message\ m, compute\ m' = m^e mod N$

$vi) For\ decrypting\ a\ ciphertext\ m', compute\ m = {m'}^{d} mod N$

Proof of why RSA works can be found in [1]. As mentioned in the previous paragraph RSA is used in a wide assortment of applications including day-to-day financial transactions using credit cards, so it is extremely pragmatic and knowing how to implement it is a useful skill.

2) ElGamal Cryptosystem: This is an asymmetric key encryption cryptosystem used for achieving public-key cryptography, it is based on the Diffie-Hellman key exchange. This encryption consists of 3 components: the key generator, the encryption algorithm and the decryption algorithm.

**Key Generation:**

The key generator works as follows:

- Alice generates an efficient description of a cyclic group ***G*** of order ***q*** with a generator ***g***.
- Alice chooses a ***x*** randomly from ***{1,…,q-a}***.
- Alice computes $h := g^x$ .
- Alice publishes ***h***, along with the description of ***G***, ***q***, ***g*** as her public key. Alice retains ***x*** as her private key, which is to be kept secret.

**Encryption:**

The encryption algorithm works as follows:

- Bob chooses a random y from ***{1,…, q-1}***, then calculates $c1 := g^y$ .
- Bob calculates the shared secret $s := h^y$ .
- Bob maps his secret message m onto an element ***m`*** of ***G***.
- Bob calculates $c2 := m` * s$ .
- Bob sends the cipher-text ***(c1, c2)*** = $(g^y, m` * h^y)$ = $(g^y, m` * (g^x)^y)$ to Alice.

If an adversary knows ***m`*** then he/she can find $h^y$ . So, a new ***y*** is generated for every message for security.

**Decryption:**

The decryption algorithm works as follows:

- Alice calculates the shared secret $s := c1^x$ **.**
- Alice then computes $m` := c2 * s^{-1}$ which she then converts back into plaintext message m, where $s^{-1}$ is the inverse of ***s*** in the group ***G***.

The decryption algorithm produces the intended message as:

$C2 * s^{-1} = m` * h^y * (g^{xy})^{-1} = m` * g^{xy} * g^{-xy} = m`$

3) Advanced Encryption System (128 bit AES) : AES is based on the Rijndael Cipher, which is based on a design principle known as 'Substitution-Permutation Network'. The Rijndael cipher uses triple discreet invertible uniform transformations (layers). Specifically, these are: Linear Mix Transform, Non-linear Transform and Key Addition Transform. So typically the AES has the four steps during encryption:

- Substituting Bytes
- Shifting Rows
- Mixing Columns
- Adding Round keys

Also the AES involves dealing with bytes of data rather than bits of data. The main advantage of AES compared to DES and other encryption algorithms is that the speed of encryption is fast both in the software and hardware.

$$\begin{bmatrix} Byte_0 & Byte_4 & Byte_8 & Byte_{12} \\ Byte_1 & Byte_5 & Byte_9 & Byte_{13} \\ Byte_2 & Byte_6 & Byte_{10} & Byte_{14} \\ Byte_3 & Byte_7 & Byte_{11} & Byte_{15} \end{bmatrix}$$

**ENCRYPTION:**
A 128 bit AES encryption involves 10 rounds of the above mentioned steps. The input plain text is split into bytes of size 8 bits. So we typically have 16 bytes of (block) data to deal with in each round of operation. The bytes are typically arranged in the format shown below:

Substituting Bytes: A substitution step where each byte is replaced with another byte within the matrix based.

Shifting Rows: This is a called a 'Transposition Step' where the last three rows of the matrix are shifted cyclically based on some pre-defined number of steps.

Mixing Columns: The four bytes in a column are combined using an invertible linear combination.

In all the above steps the involvement of keys is nil. It is only in the last step the keys are XOR-ed with the text values.

Adding Round Keys: The keys for adding to the state matrix are generated using the 'Rjindael Key Schedule'. For 126 bit (16 Byte) encryption a 16 Byte key is generated and XOR-ed with the State matrix in each round. The strength of this algorithm also lies in the fact that the key being injected in each round is slightly different from the one used in the previous round. This is accomplished using the 'Key Schedule', which performs some mix and match and cyclic procedures to increase the randomness of the keys in each round of encryption.

We are implementing 128 bit AES encryption in this project, so 10 rounds are encryption are carried out.

The Decryption of the cipher text will be performed for 10 rounds, with all the steps in reverse order. The speed of this method lies in the fact that there is no requirement to calculate the inverse key value, as XOR-ing the cipher value again with the same key will give us the plain text. Essentially both the sender and receiver use the same key to encrypt and decrypt. So, AES falls into the category of Symmetric Key Cryptography.

[1] R. L. Rivest, A. Shamir and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” Communications of the ACM, Vol. 21, No. 2, 1978, pp. 120-126. doi:10.1145/359340.359342