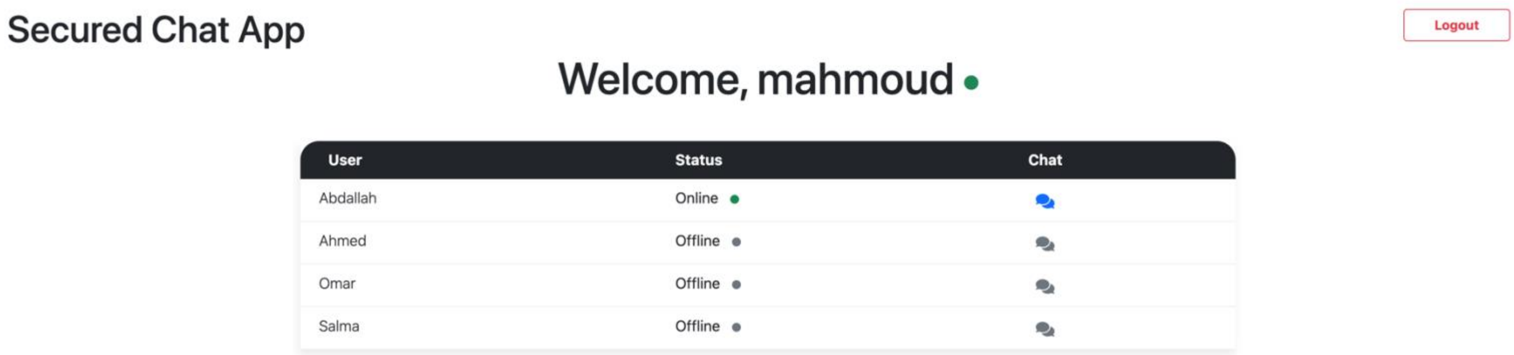
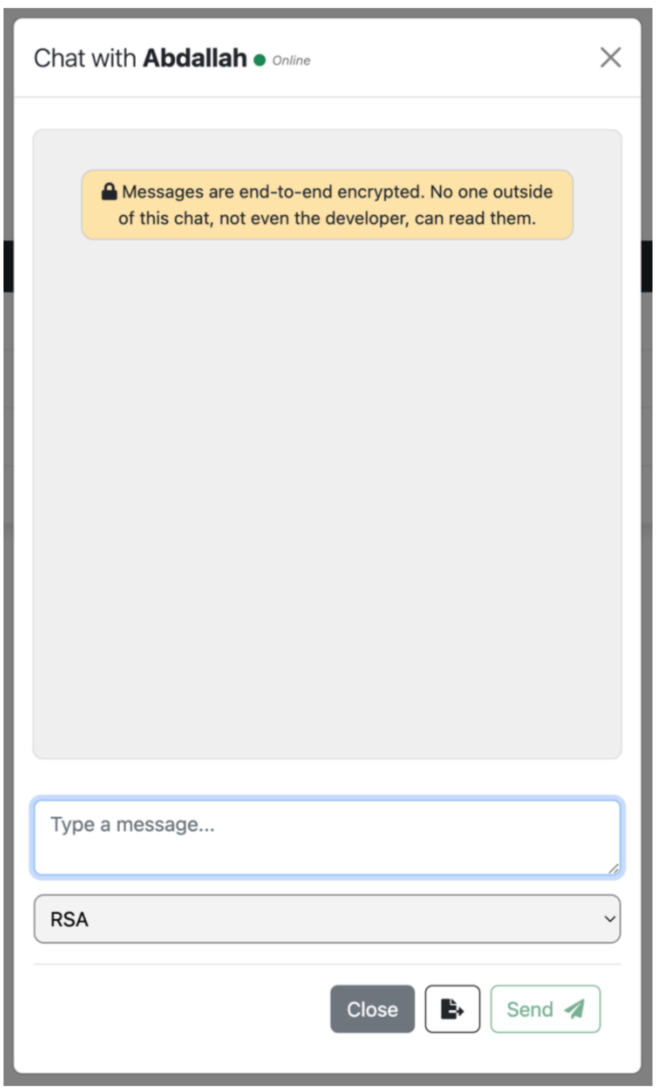
I. Introduction:

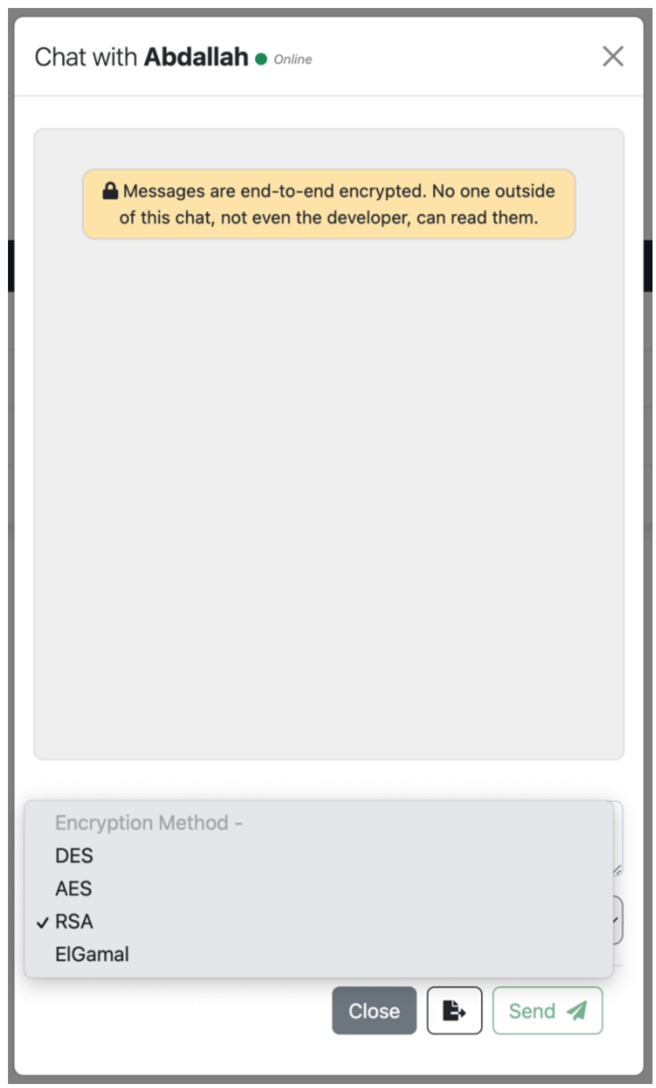
- A Web Application that provides a secured chatting experience with an end-to-end encryption.
- The interface is designed using HTML, CSS, Bootstrap, JavaScript, and jQuery.
- There are 4 encryption algorithms and 1 key exchange algorithm included.
- All the algorithms are implemented from scratch, without using any libraries or outsource code.
- Algorithms are developed using PHP and JavaScript, and then, all the information are sent to a secured MySQL database using PHP/AJAX.
- Users have to register with a unique username and a password to be able to chat with another user.
- Users can send text-messages and files.
- Files are then can be downloaded by both users after its sent.
- There is a select option element that let the user choose his preferred encryption algorithm for the current message to be sent.
- The select option element includes DES, AES, RSA, and El-Gamal Algorithms.

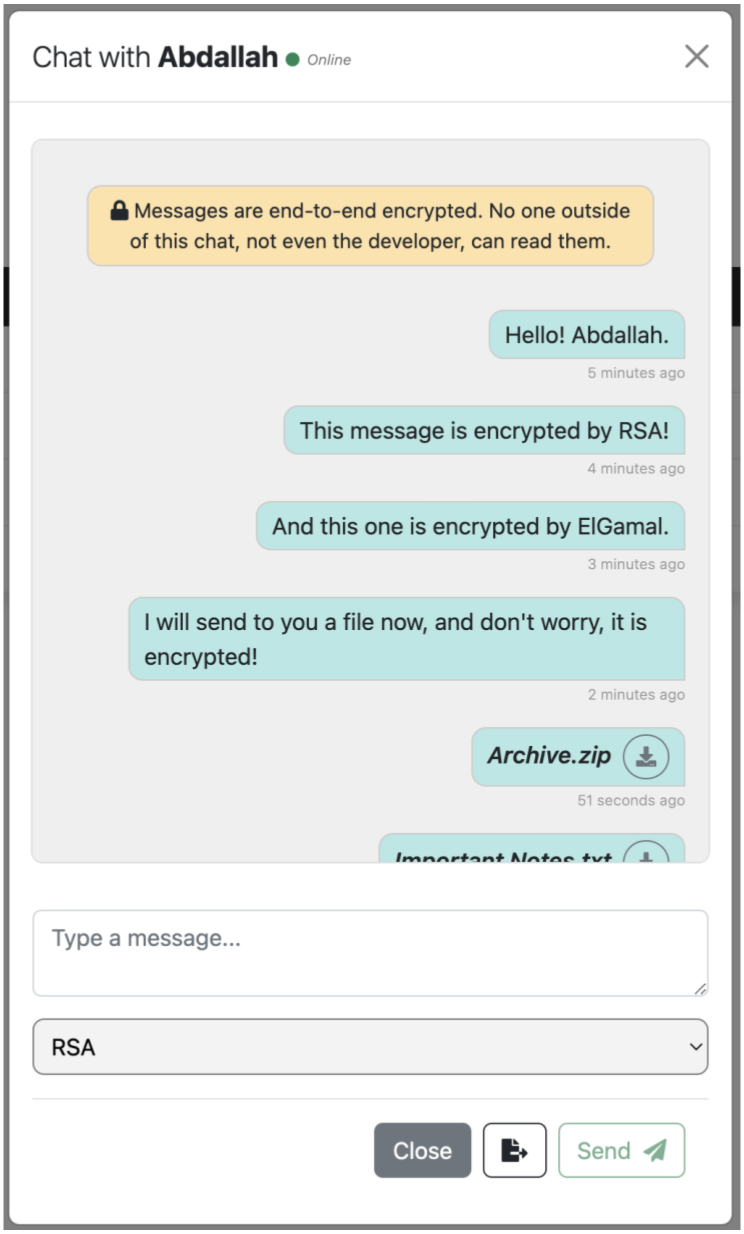
- **Home Page after login:**

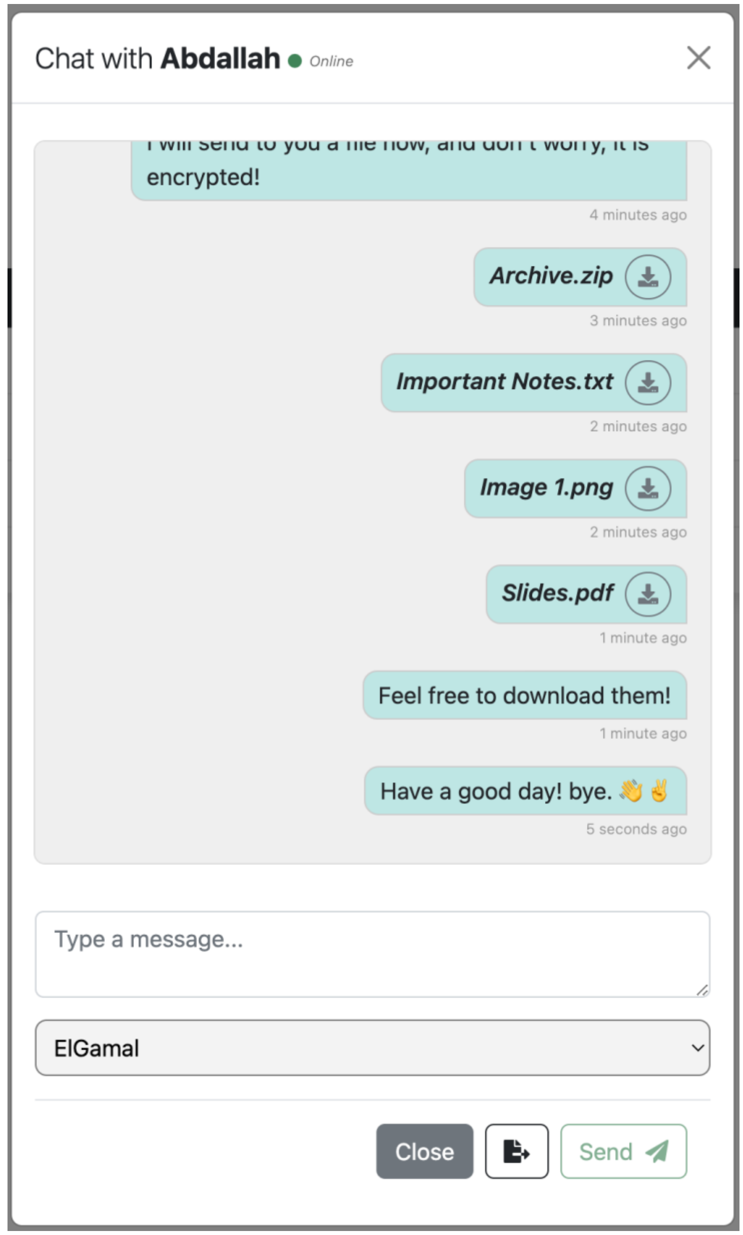


- **Chat Box:**






Chat with Abdallah Online
Messages are end-to-end encrypted. No one outside of this chat, not even the developer, can read them.
Hello! Abdallah.
5 minutes ago
This message is encrypted by RSA!
4 minutes ago
And this one is encrypted by ElGamal.
3 minutes ago
I will send to you a file now, and don't worry, it is encrypted!
2 minutes ago
Archive.zip
51 seconds ago
Type a message...
RSA
Close
Send


Chat with Abdallah Online
encrypted!
4 minutes ago
Archive.zip
3 minutes ago
Important Notes.txt
2 minutes ago
Image 1.png
2 minutes ago
Slides.pdf
1 minute ago
Feel free to download them!
1 minute ago
Have a good day! bye.
5 seconds ago
Type a message...
ElGamal
Close
Send

II. Algorithms:

1. DES:

- The objective of this algorithm encrypts 64-bit data using 56- bit Key. Which takes as an input 64-bits of data and of key. DES involves 16 rounds of identical operations. The rounds in DES are Expansion, XOR operation with round key, Substitution and Permutation. The structure is based on a Feistel network.

<u>Functions:</u>

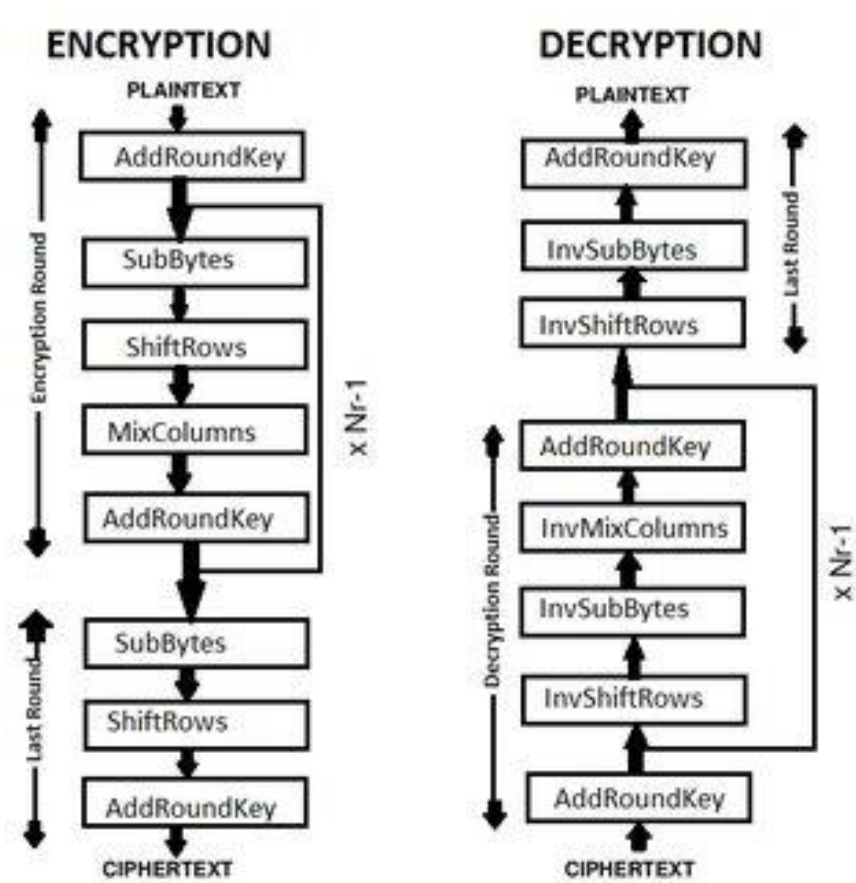
- ConvertHextoBinary( ) → Convert from hexadecimal to binary numbers.
- ConvertBinaryToHex( ) → Convert from binary to hexadecimal numbers.
- ConvertBinaryToDecimal( ) → Convert from binary to decimal numbers.
- ConvertDecimalToBinary( ) → Convert from decimal to binary numbers.
- Permutation( ) → Permute function to rearrange the bits.
- ShiftLeft( ) → Shifting the bits towards left by nth shifts according to the rounds.
- XOR( ) → Calculating the xor of two strings of two binary numbers.
- **Encryption( )** → • Convert from hexadecimal to binary
  - Use the permutation function.
  - Split the 64 bits into two arrays(left, Right)
  - Expanding the 32 bits data into 48 bits
  - Substituting the value from s-box table by calculating row and column
  - After substituting use the permutation function.
  - XOR left and sbox_string.
  - Swap the left with right.
  - Combine the left with right.
  - Final permutation: final rearranging of bits to get cipher text

➢ **Decryption( )** → Do the same steps as encryption by using subkeys in reverse order (SK16 … SK1)

  - 1st round with SK16 undoes 16th encrypt round and then recover the original plain-text.

2. AES:

- **Advanced Encryption Standard (AES)** is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.
- The used secret key size is 128 bits (16 bytes) which means it takes 10 rounds to encrypt/decrypt the plain text.



- **<u>AES key expansion:</u>**

We first split the key into four blocks then on the fourth block "w[3]" we perform:

- Circular byte left shift
- Byte Substitution (S-Box)
- XOR with round constant table = g(w[3])

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Then we perform an XOR operation to generate 4 new blocks from the resulting g(w[4]) block, to form a new key:

- w[4] = w[0] XOR g(w[3])
- w[5] = w[1] XOR w[4]
- w[6] = w[2] XOR w[5]
- w[7] = w[3] XOR w[6]

by combing all of the blocks from w[4] to w[7] we form the round key.

- **<u>In encryption</u>**: we generate new round keys along side each iteration of the state matrix to XOR them by each generated round key.
- **<u>In decryption</u>**: we generate all the round keys first, then we XOR the state matrix with the round keys in reverse order in each iteration (10 iterations since we're using 128 bits).

- **2- Add round key to state matrix:**
- The state matrix is the plaintext (converted to HEX) in a 2d array form.

We simply perform XOR operation with current round key to create a new state matrix each round until a cipher is produced on the 10^{th} round.

- **3- S-Box and Inverse S-Box:**

We simply substitute the bytes using Rijndael' S-box

AES S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Inverse S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

- **4- ShiftRow and Inverse Shiftrow:**

This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

- **<u>5- Mix Columns and Inverse Mix Columns:</u>**

Each row in the state matrix is treated as a four-term polynomial:

- $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$ which are elements within Galois Field $GF(2^8)$.

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- we multiply by this matrix..
- while in the inverse mix columns we multiply by the matrix..

- **<u>After encryption/decryption:</u>**

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

  - the outputted text will be converted to a string of hexadecimals then converted back to binary text.
  - The algorithm is developed by **PHP.**

3. RSA:

- RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission. It is also one of the oldest. The acronym "RSA" comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who publicly described the algorithm in 1977.
- RSA is still **un-hackable**, thanks to factorization problem.
- First, **p** and **q** are primely generated using **findRandomPrime()** and **isPrime()** functions, where **p != q**.
- n is calculated by multiplying **p** to **q** using **compute_n()** function
- $\boldsymbol{\phi}(\boldsymbol{n})$ aka **z** is calculated by multiplying **(p-1)** to **(q-1)** using **eular_z()** function.
- **e** is then generated, by using **find_e()** function, which is checking for a number **(e)** > 1 and < $\boldsymbol{\phi}(\boldsymbol{n})$, and with respect that the **gcd(e,** $\boldsymbol{\phi}(\boldsymbol{n})$**)** must be equal 1.
- **d** is generated using **find_d()** function which enters an infinite loop until it finds a number **(d)** such that **(d * e) mod** $\boldsymbol{\phi}(\boldsymbol{n})$ **= 1** (Similar as modular inverse).
- **M**, which represents the message, is then achieved by converting the characters into ASCII numbers by using **ord()** function, with respect that **M < n**.
- **C**, which is the cipher-text, is then generated using **encrypt()** function. Inside the function, **C** concatenate itself with the **bcpowmod()** PHP function using the square and multiply method to power the **e** to the resulted character ASCII number from **M** and then mod **n**, and repeat for every character and concatenate and cipher-text **C**.
- A variable called **everySeparate** is concatenated with the count for every resulted **C** after every calculation of **bcpowmod()**, to help when decrypting, due to the variety of ASCII digits length. (For example, sympols).
- At decrypting, **C**, **d**, **n**, and **everySeparate** are extracted from the database, generating the **M** by using **chr**(**bcpowmod(current(c), d, n))** and concatenate each letter to the M. (chr() function is used to convert from ASCII number to char).
- The algorithm is developed using **PHP**.

4. Diffie-Hellman:

- The purpose of this algorithm is to generate encrypted secret keys that are shared between only two users, which are then used by other text encryption algorithms.
- **<u>Functions:</u>**
- **findRandomPrime()** – Used to generate a random prime number with the help of **isPrime()** function **(q)**.
- **findPrimitives ()** – Used to generate the primitives root base number and then choose one random for them **(a)**.
- **Math.random()** – Used to randomly generate the private keys < **q (Xa, Xb).**
- **mpmod()** – Used to calculate the public keys **(Ya, Yb)** by calculating **(a)** to the power **(xa, xb)** mod to **(q)** based on Square and Multiply method.
- **mpmod()** – Used to compute the secret keys **(_a, _b)** based on Square and Multiply method.
- The secret keys are inserted only once for both users inside the database, and then, they are updated whenever any of the users opens the chat box.
- The algorithm is developed by **JavaScript**.

5. El-Gamal:

- The El Gamal encryption system is a public-key cryptography asymmetric key encryption scheme based on the Diffie–Hellman key exchange. Taher El Gamal described it in 1985.
- It communicates and encrypts messages between two parties using asymmetric key encryption. This cryptosystem is based on the difficulty of finding a discrete logarithm in a cyclic group.
- First, we get random large number 'q', then we get random number 'a' is primitive root of 'q' and smaller than 'q', then we select private key 'Xa' randomly but must be smaller than 'q', then we generate 'Ya' according to it rule which is (a^Xa) mod q.
- **Encryption function:**
- **Parameters: q, a, Ya, message**
- Get random integer 'k' must be smaller than 'q'
- Generates 'K' according to its rule which is (Ya^k) mod q
- Generates C1 = (a^q) mod q
- Generates C2 = (K*message) mod q
- **Decryption function:**
- **Parameters: C1, C2, Xa, q**
- Generate K=(C1^Xa) mod q
- Generate message=(C2*k^-1) mod q.
- Every Separate method is also used to allow more than one character in the message. (See in RSA)
- Letters are converted to its ASCII Decimal code and perform the algorithm and at the end, its converted back to char.
- El Gamal is still **un-hackable**, thanks to Discrete Log problem.
- The algorithm is developed by **JavaScript.**

III. Languages, Tools, and Frameworks:

- *VS-Code*
- ***PHP***
- *HTML, CSS, Bootstrap, and **JavaScript***
- ***AJAX***
- *jQuery*
- ***MySQL***
- *000webhost*
