Crypto Library Project

Caesar Cipher:

➢ Functions used in enc , dec , cryptanalysis

const string allAlphabet{ "ABCDEFGHIJKLMNOPQRSTUVWXYZ"}

- Constant string variable carries all alphabet characters from a to z

void textToUpper()

- Input : plaintext
- Convert plaintext to upper characters

allAlphabet.find()

- Find function search for character in string and returns index of it

int mod(int k, int n)

- To find character before shifting in alphabet
- Input : index of character , length of alphabet(26)
- Output : index of character before shifting

(1) Encryption

string EncryptCaeser(string plainText, int keyValue)

- Input : plaintext , key
- Output : Cipher text

(2) Decryption

string DecryptCaeser(string EncryptedText, int keyValue)

- Input : Cipher text , key
- Output : Plain text

(3) CryptAnalysis

void bruteforceCaeser(string EncryptedText)

- Input : Cipher Text
- Print all possible keys with its own plaintext

int cryptanalysis(string EncryptedText, string plain_txt)

- Input : cipher text , plaint text
- Output : key

Mono Cipher :

(1) Encryption

String encrypt(string plaintext, string key)

- Input : plain text , key
- Output : cipher text

(2) Decryption

String decrypt(string ciphertext, string key)

- Input : cipher text , key
- Output : plain text

(3) Analysis

void AnalyseUsingCharFrequency(string ciphertext)

- Input : cipher text
- Output : display the result of frequency analysis on cipher text

PlayFair :

void create_matrix()

- Input : key as string
- Create matrix of key

void encrypt(string plain_txt)

- Input : plain text
- Print cipher text

void decrypt(string cipher_txt)

- Input : Cipher Text
- Output : plain Text

Rail Fence :

plain_txt.erase(remove(plain_txt.begin(), plain_txt.end(), ' '), plain_txt.end())

- Remove spaces from string

transform(plain_txt.begin(), plain_txt.end(), plain_txt.begin(), toupper)

- Convert string to upper case

ceil(int)

- returns the smallest possible integer value which is greater than or equal to the given argument.

(1) Encryption

Encrypt_RainFence(string plain_txt, int key)

- Input : Plain text , key
- Output : Cipher text

(2) Decryption

Decrypt_RainFence(string cipher_txt, int key)

- Input : Cipher text , key
- Output : Plain text

(3) CryptAnalysis

int Analyse_RainFence(string plain_txt, string cipher_txt)

- Input : Plain text , Cipher text
- Output : key steps

Vigenere with Auto key :

void textToUpper(string& text)

- Input : plaintext
- Convert plaintext to upper characters

int mod(int k, int n)

- To find character before shifting in alphabet
- Input : index of character , length of alphabet(26)
- Output : index of character before shifting

(1)Encryption

string Autokeystream(string PlainTEXT, string Key);

- Fill the difference in length between key and plain (key = key +plain[i])
- Input : plain text , key
- Output : key with same length of plain

string Encrypt(string plainText, string key)

- Input : plain text , key
- Output : Cipher Text

(2)Decryption

- string Decrypt(string cipherText, string key);
- Input : Cipher text , key
- Output : Plain text

(3)CryptAnalysis

- string Analyse(string plainText, string cipherText)
- Input : Plain text , Cipher text

- Output : key string

Vigenere with Repeating key :

void textToUpper(string& text)

- Input : plaintext
- Convert plaintext to upper characters

int mod(int k, int n)

- To find character before shifting in alphabet
- Input : index of character , length of alphabet(26)
- Output : index of character before shifting

(1)Encryption

string Repeatingkeystream(string PlainTEXT, string Key)

- Fill the difference in length between key and plain (key = key +key[i])
- Input : plain text , key
- Output : key with same length of plain

string Encrypt(string plainText, string key)

- Input : plain text , key
- Output : Cipher Text

(2)Decryption

string Decrypt(string cipherText, string key)

- Input : Cipher text , key
- Output : Plain text

(3)CryptAnalysis

string Analyse(string plainText, string cipherText)

- Input : Plain text , Cipher text
- Output : key string

Columnar :

int key_numOf_Eelemnts

- Number of key elements (Cin)

text.erase(std::remove_if(text.begin(), text.end(), ::isspace), text.end())

- Remove space from string

transform(plaintext.begin(), plaintext.end(), plaintext.begin(), ::toupper)

- Convert string to upper case

ceil(float)

- returns the smallest possible integer value which is greater than or equal to the given argument.

(1)Encryption

string Encrypt(string plaintext, int key[])

- Input : plain text , array of key elements
- Output : cipher text

(2)Decryption

string Decrypt(string ciphertext, int key[])

- Input : Cipher text , array of key elements
- Output : Plain text

(3)CryptAnalysis

list<int> analyse(string plaintext, string ciphertext);

- Input : Plain text , Cipher text
- Output : list of key elements

Hill Cipher :

int smallerdet(int row, int col, Eigen::Matrix3d M)

- Find determinant of 2x2 partial matrix of 3x3
- Input : row and column to cancel , 3x3 matrix
- Output : integer determinant

Eigen::Matrix3d modular_inverse(Eigen::Matrix3d K)

- Find modular inverse of 3x3 matrix K
- Input : 3x3 matrix
- Output : 3x3 mod inverse matrix

Eigen::Matrix2d modular_inverse2by2(Eigen::Matrix2d K)

- Find modular inverse of 2x2 matrix
- Input : 2x2 matrix
- Output : 2x2 mod inverse matrix

list <int> analyse3by3(list <int> plain, list <int> cipher)

- Preform cryptanalysis on 3x3 matrix plain and cipher “K=P^-1*C”
- Input : cipher and plain as list
- Output : key matrix as list

list <int> analyse(list <int> plain, list <int> cipher)

- Preform cryptanalysis on 2x2 matrix plain and cipher “K=P^-1*C”
- Input : cipher and plain as list
- Output : key matrix as list

list <int> encrypt(list <int> plain, list<int> key)

- Encrypt plain text using given key
- Input : list key , list plain
- Output: list cipher

list <int> decrypt(list <int> cipher, list <int> key)

- decrypt cipher text using given key
- Input : list key , list cipher
- Output: list plain

DES :

bool Length_input(string txt, int size)

- To check size of key
- Input : key , default size(18 ----->0x)
- Output : boolean(True or False)

bool Check_Hex(string txt)

- Check the hexadecimal value
- Input : string of hexadecimal
- Output : boolean (True or False)

Transform(text.begin(), text.end(), text.begin(), ::toupper)

- Convert string to upper case

string hex_to_bin(string sb_wrod)

- Convert hexadecimal value to its binary
- Input : Hexa string
- Output : Binary string

void binstr_to_binarr(string plainText_bin, int* arr)

- Convert binary string to binary array
- Input : binary text , return array
- Fill array with binary text

int** New_keys(int key[])

- Create 16 sub keys from initial key
- Input : Initial key
- Output : 2D array of sub keys (16*48)

void txtpermute(int* pc, int* cipherdtxt, int size1, int* return_arr)

- Permutatuion Function
- Input : permutation array , text , size of new array after permut , return array

void split(int* bigarr, int bigarrSize, int* left, int* right, int partArrSize)

- Split text in each round to right and left side (32bit , 32bit)
- Input : text array , size of text array , left side array , right side array , 32

void Fn(int* arr, int* key_arr, int* returnedarray)

- Expansion : right (32 bit) →right (48 bit)
- XOR between right side and key

void rows(int arr[], int* row)

- Determine the row index
- Input : XORed array , return array with row indexes

void cols(int arr[], int* cols)

- Determine the column index

- Input : XORed array , return array with column indexes

void S_BOX(int rows[], int cols[], int* ret_arr)

- Store intersections of rows and columns(sbox) in array (from 48bit to 32bit)
- Input : row indexes array , columns indexes array , returned array

void dectobinary(int value, int* binaryArr, int arrsize)

- Convert decimal value to binary array
- Input : decimal value , returned binary array , size of array

void swapp(int* arr1, int* arr2)

- Swap elements of two arrays
- Input : array1 , array2

void merge(int* bigarr, int bigarrSize, int* left, int* right, int partArrSize)

- Merge right and left sides in one array
- Input : return merged array , 32 , left side array , right side array , 64

string bin_to_hex2(string key)

- Convert binary to hexa
- Input : key in binary
- Output : key in hexa

void rot(int* X)

- Left circular shift for subkeys
- Input : array of key

(1)Encryption

string Encrypt(string text, string text)

- Input : hex Plain text , hex key
- Output : Cipher Text

(2)Decryption

string Decrypt(string text, string text)

- Input : hex Cipher text , hex key
- Output : Plain Text

AES :

bool Length_input(string txt, int size)

- To check size of key
- Input : key , default size(18 ----->0x)
- Output : boolean(True or False)

bool Check_Hex(string txt)

- Check the hexadecimal value
- Input : string of hexadecimal
- Output : boolean (True or False)

string xoring(string first, string second, int length)

- Input : first string , second string , length of xoring
- Output : XORed binary string

(1)Key Generation

transform(key.begin(), key.end(), key.begin(), ::tolower)

- Convert string to lower case

string rotation(string L_word);

- Left circular shift
- Input : last word
- Output : last word after rotation

string sbox_pick(string L_word)

- Store intersections of rows and columns(sbox) in new string
- Input : last word
- Output :new string picked from sbox

string zerox_rem(string sb_word)

- Remove 0x of sbox elements from sb word
- Input : sb_word
- Output: sb_word without 0x (except initial 0x)

string hex_to_bin(string sb_wrod)

- Convert hexadecimal value to its binary
- Input : Hexa string
- Output : Binary string

string bin_to_hex2(string key)

- Convert binary to hexa
- Input : key in binary
- Output : key in hexa

void key_expansion(string key, int numOfround)

- Input : key , number of round
- Fill key_rounds array with sub keys

(2)Encryption

string first_time(string text, string text)

string Initial_XoR(string plain, string key)

- XORing between plain text and initial key
- Input : plain text , initial key
- Output : XORed value in hexa

void shift_rows(int Matrix_sBox[4][4])

- Input : matrix after sub bytes
- Shifting matrix

string DEC_To_HEX(int dec)

- Convert decimal value to its hexa
- Input : decimal
- Output : hexa string

bool dectobinary(int decimal, int binaryArr[], int mult)

- Input : element of sub bytes , returned array , element of mix columns
- Output : boolean defines xoring with 1b array or not

void XoR_arr(int arr[], int arr2[], int size, int* ret_arr)

- Binary Xoring
- Input : binary array 1 , binary array 2 , size of array , return binary array

string MIX_arr_calc(int temp_arr_mix[4], int S_Box[4][4])

- Multiplication of mix columns matrix by sub bytes matrix
- Input : column of mix matrix , column of sub bytes matrix
- Output : hex value of Cipher Text

string AES_Encrypt(string plaintext, string key)

- Encryption of all rounds except last one
- Input : plaint text , key string
- Output : Cipher text after xoring with key

string Lastround_AES_Encrypt(string plaintext, string key)

- Encryption of last round
- Input : cipher text of round n-1 , key round n
- Output : Cipher text

string AES_Final_ENC(string plain, string initial_key)

- Encryption of all rounds
- Input : plaint text , initial key string

- Output : Cipher text

(3)Decryption

string AES_Decrypt(string cipher, string initial_key)

- Take initial key and find all key rounds then decrypt the cipher
- Input : initial key, cipher text
- Output: string plain text

void shift_rows_right(string Matrix_sBox[4][4])

- Take the result from sbox and shif rows according to the rules
- Input : matrix sbox
- Output : no output since the array is passed by reference

void inv_sbox(string matrix[4][4])

- Substitute the give matrix indexes with corresponding values
- Input : matrix
- Output : no output since the array is passed by reference

void inv_mix_col(string matrix[4][4])

- Multiply each column of matrix by the inverse of mix matrix
- Input : matrix
- Output : no output since the array is passed by reference

unsigned char inv0e(unsigned char b)

- Multiply unsigned int by 0e
- Input : unsigned char
- Output: unsigned char

unsigned char inv0d(unsigned char b)

- Multiply unsigned int by 0d
- Input : unsigned char
- Output: unsigned char

unsigned char inv09(unsigned char b)

- Multiply unsigned int by 0e
- Input : unsigned char
- Output: unsigned char

unsigned char inv0b(unsigned char b)

- Multiply unsigned int by 0e
- Input : unsigned char
- Output: unsigned char

unsigned char inv02(unsigned char b)

- Shift unsigned char by one bit ‘’multiply it by 2’’
- Input : unsigned char
- Output: unsigned char

RSA :

int modulo(int a, int b, int n)

- Take the base and power of number and find the modulus n
- Input : a “number” , b “base” , n “ mod n”
- Output : result of mod operation

int Encrypt(int p, int q, int M, int e)

- It encrypt integer message with rsa
- Input : p,q “prime numbers” , M “message” , e
- Output : int cipher

int gcd(int a, int b)

- Find greatest common divisor between two numbers
- Input : two numbers a&b
- Output : GCD (a,b)

int modulusinvesre(int num, int quotient)

- Find modular inverse between two numbers
- Input : two numbers
- Output : mod inverse num of quotient

int Decrypt(int p, int q, int C, int e)

- decrypt given cipher
- input : p ,q , c, e
- output : int plain

MD5 Hash :

string str_to_bin(string input)

- Input : string
- Output : binary string

string str_padded_to_448(string input, int& size)

- Padding to 448 bits
- Input : string wanted to be padded , size of string (by reference →calculated inside function)
- Output : padded string

string add_length(string input, int size)

- Concate length of string with itself
- Input : padded string , length
- Output : 512 bit string

void block_to_32bits(string block_512, string block_32bits[16], int round, bool last_block)

- Split text into blocks of 32 bits (16 blocks)
- Input : text , array of blocks , round number , last block boolean (for little endian)

string littleEndian32Bits(string str)

- Input : text
- Output : text ordered in little endian

uint32_t G_function(uint32_t b, uint32_t c, uint32_t d, int r)

- Combination of math operations based on round number
- Input : initial vector[1] , initial vector[2], initial vector[3] , round number
- Output : unsigned 32 decimal number

void binary_str_to_binary_int_arr(string binary, int ret_bin_arr[], int size)

- Convert binary string to integer array
- Input : string binary , return array , size of return array

uint32_t binary_dec(int arr_bin[], int size)

- Convert binary to decimal
- Input : binary array , size of array
- Output : unsigned 32 decimal value

void CLS(int s_round, int bits[])

- Circular left shift
- Input : round number , array of integer
- Store in bits arrat after shifting

void shifting(int bits[], int num_shifts);

- Circular left shift , Stored in same array
- Input : array of integer , number of shifts based on round number
- Calling inside CLS function

string decToHexa(uint32_t n)

- Input : unsigned 32 decimal value
- Output : hexa string

string littleEndian_hex(string str)

- Input : hexa string
- Output : hexa string ordered in little endian

uint32_t HexToDec(string n)

- Input : Hex string
- Output : unsigned 32 decimal value

string repeated_16(string block_512, int round_num, bool last_block)

- Input : block 512 bits , round number , boolean last block(for little endian)
- Output : CVi for next round (4 rounds)

string MD5_Hash(string plaintext)

- Input : plaintext
- Output : hashing