Introduction:

Cryptography is the science of securing information by converting it into an unreadable format that can only be deciphered by authorized parties using specific keys. Cryptography ensures confidentiality, integrity, authentication and non-repudiation in communication. This lab focuses on three cryptographic algorithms:

1. Vigenère Cipher: a polyalphabetic substitution cipher that improves upon classical Caesar ciphers by using a repeating key. It uses substitution alphabets that help make it more secure than a Caesar cipher. Encryption of the Vigenère cipher consists of alphabets shifted using a key.
2. S-Box Substitution: a process that helps to enhance security by adding a substitution using a symmetric key encryption. It is used to improve security by making the encryption process less linear to make it harder to decode the encrypted data.
3. RSA Cipher: it is an asymmetric algorithm that works using a public and private key. It uses the public key for encryption and the private key for decryption and is only available to the receiver. The algorithm has three steps, key generation, encryption and decryption. The RSA Cipher is based on the idea that it is hard to factorize a large integer. It has a few advantages such as security, it uses two keys for encryption and decryption, and it can be used for sending keys. Some of the disadvantages it has include that it has a slow processing speed, and that having a large key size may require more computational resources to compute.

Some of the objectives include the implementation of each algorithm from scratch in python, understanding the mechanics of encryption, decryption and key encryption and comparing ciphertexts and plaintexts to verify correctness.

**Vigenère Cipher:**

The Vigenère cipher we were implementing shifted the key using a predetermined key.

I implemented it using python.

**Plaintext:** LAUNCHTHEATTACKATONEPM

**Key:** 4123

After implementation of the program, I was able to encrypt the plaintext. The key repeats over the length of the plain text and each character are shifted according to its corresponding key value.

**Encrypted message:** PBWOGIXIGBXUEDMBXPRFRN

**S-Box Substitution:**

S-Box Substitution uses a box to map an output. It works by taking in binary input and converting it to something it will use to map to the table. Once it finds what it was looking or in the table, it converts it back to binary.

I implemented the code using python using the following S-Box:

A grid of numbers and letters

AI-generated content may be incorrect.

**Input:** “E4589A34209”

Output: " 11111110110011101101100100101110”

* The input is converted to a binary input and then divided into inputs of 6 bits which are then converted for the table and used for the encryption.
* This lack of a linear pattern helps to improve security.

**RSA Cipher Implementation:**

The algorithm uses mathematics to generate keys. It uses the totient function in its calculation of the keys. It also uses the gcd and mod and provides a private and public key to use in encryption and decryption.

Steps:

* Choose primes and .
* Compute and .
* Select public exponent with .
* Compute private key such that .
* Encrypt message as .
* Decrypt ciphertext as .

I used python to implement the language. The values we used are as follows:

* p = F7E75FDC469067FFDC4E847C51F452DF
* q = E85CED54AF57E53E092113E62F436F4F
* e = 0D88C3

**Output:**

**Message (decimal):** 15691824710153

**Encrypted text:** 89476191203601951464771342265492572724674898646401400930248570943989910571431

**Decrypted text:** E4589A34209

The algorithm works because the decrypted message matches the plaintext exactly. This algorithm relies on the difficulty of factoring n where n=(p\*q). Small prime numbers are easy to decrypt, which is why 128 bits are usually used.

Conclusion:

This lab reinforced the concepts of classical and modern cryptography. Key takeaways:

* Vigenère cipher demonstrates polyalphabetic substitution.
* S-boxes illustrate nonlinear mappings critical for block ciphers.
* RSA highlights key generation, encryption, and decryption in public-key cryptography.

AI Prompts used:

* Why does this calculate the wrong row and column? Help me debug and fix it?
* I want to be able to handle hexadecimal ad binary inputs, where does the error stem from?
* Explain the error: TypeError: not all arguments converted during string formatting, temp = n % 16?
* Why does this not work?

def HexToBin(hexdec):

hex\_map = {

'0': '0000', '1': '0001', '2': '0010', '3': '0011',

'4': '0100', '5': '0101', '6': '0110', '7': '0111',

'8': '1000', '9': '1001', 'A': '1010', 'B': '1011',

'C': '1100', 'D': '1101', 'E': '1110', 'F': '1111'

}

binary\_string = ""

for char in hexdec.upper():

if char in hex\_map:

binary\_string += hex\_map[char]

else:

print(f"\nWarning: Invalid hexadecimal character '{char}' ignored.")

return binary\_string

* Fix this function

def modInverse(e, phi):

    for d in range (2, phi):

        if (e \* d) % phi == 1:

            return d

        else:

            return -1