Mini Project Report Cover Sheet

SRM Institute of Science and Technology College of Engineering and Technology Department of Electronics and Communication Engineering
18ECC303J Computer Communication and Networks **Sixth Semester, 2023-24 (Even semester)**

Experiment based Project

On

Implementation of RC4 Cryptography Algorithm

Topic	**Mark**	Manav Chhattri	Sahil Raj	Anindya Sarkar
		RA2111053010036	RA2111053010038	RA2111053010039
Demonstration	15			
Document Preparation	15			
Viva	10			
Total	**40**			

REPORT VERIFICATION

Date :

Staff Name : **Dr. Diwakar R. Marur**

Signature :

11. Implementation of RC4 Cryptography Algorithm

11.1 AIM

Implementation of RC4 Encryption and Decryption Algorithms Using Python.

11.2 INTRODUCTION

Cryptography is a technique of securing communication by converting plain text into unintelligible ciphertext. It involves various algorithms and protocols to ensure data confidentiality, integrity, authentication, and non-repudiation. Thus, preventing unauthorized access to information. The prefix "crypt" means "hidden" and suffix "graphy" means "writing". In Cryptography the techniques which are used to protect information are obtained from mathematical concepts and a set of rule-based calculations known as algorithms to convert messages in ways that make it hard to decode it.
Techniques used For Cryptography: In today's age of computers cryptography is often associated with the process where an ordinary plain text is converted to cipher text which is the text made such that intended receiver of the text can only decode it and hence this process is known as encryption. The process of conversion of cipher text to plain text this is known as decryption.

Features Of Cryptography are as follows:
Confidentiality: Information can only be accessed by the person for whom it is intended and no other person except him can access it.
Integrity: Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.
Non-repudiation: The creator/sender of information cannot deny his intention to send information at later stage.
Authentication: The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed.

11.2.1 Confidentiality

Confidentiality is the protection of information in the system so that an unauthorized person cannot access it. This type of protection is most important in military and government organizations that need to keep plans and capabilities secret from enemies. However, it can also be useful to businesses that need to protect their proprietary trade secrets from competitors or prevent unauthorized persons from accessing the company's sensitive information (e.g., legal, personal, or medical information). Privacy issues have gained an increasing amount of attention in the past few years, placing the importance of confidentialityon protecting personal information maintained in automated systems by both government agencies and private-sector organizations. Confidentiality must be well-defined, and procedures for maintaining confidentiality must be carefully implemented. A crucial aspect of confidentiality is user identification and authentication. Positive identification of each

system user is essential in order to ensure the effectiveness of policies that specify who is allowed access to which data items.
Threats to Confidentiality: Confidentiality can be compromised in several ways. The following are some of the commonly encountered threats to information confidentiality –

- Hackers
- Masqueraders
- Unauthorized user activity
- Unprotected downloaded files
- Local area networks (LANs)
- Trojan Horses

11.2.2 Uses of Confidentiality:

In the field of information security, confidentiality is used to protect sensitive data and information from unauthorized access and disclosure. Some common uses include:

Encryption: Encrypting sensitive data helps to protect it from unauthorized access and disclosure.
Access control: Confidentiality can be maintained by controlling who has access to sensitive information and limiting access to only those who need it.
Data masking: Data masking is a technique used to obscure sensitive information, such as credit card numbers or social security numbers, to prevent unauthorized access.
Virtual private networks (VPNs): VPNs allow users to securely connect to a network over the internet and protect the confidentiality of their data in transit.

Secure file transfer protocols (SFTPs): SFTPs are used to transfer sensitive data securely over the internet, protecting its confidentiality in transit.

Two-factor authentication: Two-factor authentication helps to ensure that only authorized users have access to sensitive information by requiring a second form of authentication, such as a fingerprint or a one-time code.
Data loss prevention (DLP): DLP is a security measure used to prevent sensitive data from being leaked or lost. It monitors and controls the flow of sensitive data, protecting its confidentiality.

11.2.3 Issues of Confidentiality:

Confidentiality in information security can be challenging to maintain, and there are several issues that can arise, including:

Insider threats: Employees and contractors who have access to sensitive information can pose a threat to confidentiality if they intentionally or accidentally disclose it.
Cyberattacks: Hackers and cybercriminals can exploit vulnerabilities in systems and networks to access and steal confidential information.
Social engineering: Social engineers use tactics like phishing and pretexting to trick individuals into revealing sensitive information, compromising its confidentiality.

Human error: Confidential information can be accidentally disclosed through human error, such as sending an email to the wrong recipient or leaving sensitive information in plain sight.
Technical failures: Technical failures, such as hardware failures or data breaches, can result in the loss or exposure of confidential information.
Inadequate security measures: Inadequate security measures, such as weak passwords or outdated encryption algorithms, can make it easier for unauthorized parties to access confidential information.
Legal and regulatory compliance: Confidentiality can be impacted by legal and regulatory requirements, such as data protection laws, that may require the disclosure of sensitive information in certain circumstances.

- The Caesar cipher is a simple encryption technique that was used by Julius Caesar to send secret messages to his allies. It works by shifting the letters in the plaintext message by a certain number of positions, known as the “shift” or “key”.
- The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.
- Thus, to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down. The encryption can be represented using modular arithmetic by first transforming the lettersinto numbers, according to the scheme, A = 0, B = 1…, Z = 25. Encryption of a letter bya shift n can be described mathematically as.
- For example, if the shift is 3, then the letter A would be replaced by the letter D, B would become E, C would become F, and so on. The alphabet is wrapped around so that after Z, it starts back at A.
- Here is an example of how to use the Caesar cipher to encrypt the message “HELLO” with a shift of 3:
  1. Write down the plaintext message: HELLO
  2. Choose a shift value. In this case, we will use a shift of 3.
  3. Replace each letter in the plaintext message with the letter that is three positions to the right in the alphabet.

     H becomes K (shift 3 from H)

     E becomes H (shift 3 from E)

     L becomes O (shift 3 from L)

     O becomes R (shift 3 from O)
  4. The encrypted message is now “KHOOR”.

- To decrypt the message, you simply need to shift each letter back by the same number of positions. In this case, you would shift each letter in “KHOOR” back by 3 positions to get the original message, “HELLO”.

$$E_n(x) = (x + n) mod\ 26$$

(Encryption Phase with shift n)

$$D_n(x) = (x - n) mod\ 26$$

(Decryption Phase with shift n)

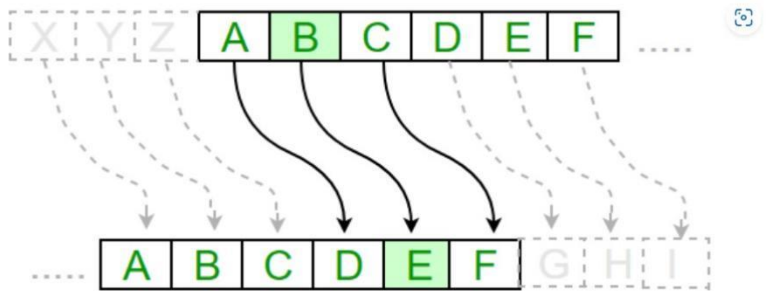


Fig 1. Substitution Cipher

The evolution of cryptographic algorithms from the earliest Caesar Cipher to RC4 demonstrates a progression in complexity and security features:

1. **Caesar Cipher (1st century BCE):** One of the earliest known cryptographic techniques, the Caesar Cipher, involved shifting each letter in the plaintext by a fixed number of positions down the alphabet. It's a simple substitution cipher with a fixed key. Despite its simplicity, it provided some level of confidentiality for messages.

2. **Vigenère Cipher (16th century):** The Vigenère Cipher improved upon the Caesar Cipher byusing a keyword to determine the amount of shift applied to each letter in the plaintext. This made the encryption stronger as it introduced variability based on the keyword. However, it was still susceptible to frequency analysis attacks.

3. **Enigma Machine (20th century):** Developed during World War II, the Enigma Machine wasa more sophisticated encryption device used by the German military. It employed multiple rotors and plugboards to perform complex permutations and substitutions on letters. The Enigma's cryptographic strength relied on its mechanical complexity and daily-changing settings, making it significantly more secure than earlier ciphers.

4. **Data Encryption Standard (DES) (1970s):** DES was one of the first widely adopted encryption standards. It used a symmetric-key algorithm, meaning the same key was used for both encryption and decryption. DES operated on 64-bit blocks of plaintext and employed a 56-bit key. However, over time, advances in computing power rendered DES vulnerable to brute-force attacks due to its relatively short key length.

5. **Advanced Encryption Standard (AES) (2001):** AES replaced DES as the standard symmetricencryption algorithm. It supports key lengths of 128, 192, or 256 bits, providing much strongersecurity. AES operates on 128-bit blocks of plaintext and uses a substitution-permutation

network to perform encryption and decryption operations. It has become the de facto encryption standard for a wide range of applications due to its robust security and efficiency.

6. RC4 (Rivest Cipher 4) (1987): RC4 is a stream cipher designed by Ron Rivest. Unlike block ciphers like DES and AES, which encrypt fixed-size blocks of data, RC4 generates a pseudorandom stream of bits that is XORed with the plaintext to produce ciphertext. RC4 gained popularity due to its simplicity and speed, making it suitable for applications requiring fast encryption and decryption, such as secure sockets layer (SSL) and Wi-Fi encryption protocols. However, vulnerabilities in RC4 were discovered over time, leading to its deprecation in favor of more secure algorithms like AES.

11.3 SOFTWARE: Python 3.8(Jupyter)

11.4 RC4 ALGORITHM:

RC4 (Rivest Cipher 4) is a symmetric stream cipher algorithm that is widely used in various applications, such as secure socket layer (SSL), wireless network encryption, and Bluetooth. It operates on bytes of data and generates a stream of pseudo-random bytes, which are XOR-ed with the plaintext to produce the ciphertext. RC4 encryption and decryption is achieved by using a secret key of arbitrary length between 40 and 2048 bits. The RC4 algorithm consists of two main stages: key-scheduling and stream generation. In the key-scheduling stage, the algorithm creates a permutation of the 256-byte array, based on the secret key and an initialization vector (IV). The permutation is achieved by performing a series of swaps between elements of the array, depending on the key and IV. In the stream generation stage, the algorithm generates a pseudo-random stream of bytes by repeatedly swapping elements of the array and generating a byte from the array index, based on the current state of the algorithm. This stream is XOR-ed with the plaintext to produce the ciphertext, and vice versa for decryption.

Overall, RC4 is a simple, fast, and widely-used stream cipher that provides a good level of security for various applications. However, it is vulnerable to certain attacks, such as key recovery attacks and related key attacks, and therefore should not be used as the sole security mechanism for critical applications. To encrypt a plaintext message using RC4, the pseudorandom stream is XORed with the plaintext to produce the ciphertext. To decrypt the ciphertext, the same pseudorandom stream is XORed with the ciphertext to recover the original plaintext.

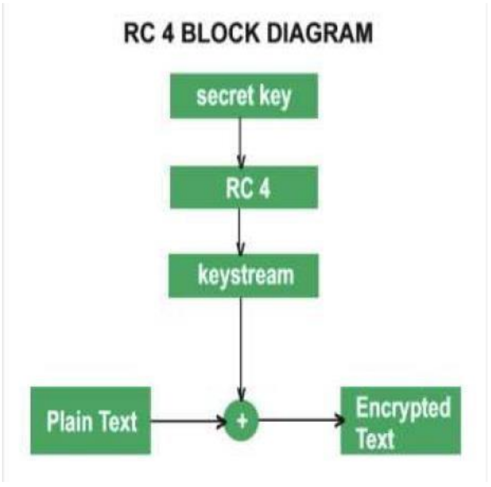


Fig 2. Block Diagram for RC4

11.5 IMPLEMENTATION METHODOLOGY:

RC4 algorithm can be implemented using Python by following these steps:

1. Define a function to perform the key-scheduling stage, which takes the secret key as input and generates the permutation of the 256-byte array.

2. Define a function to generate the pseudo-random stream of bytes, which takes the permutation array as input and generates a stream of bytes to XOR with the plaintext.

3. Define a function to perform the encryption, which takes the plaintext and secret key as input, and generates the ciphertext by XOR-ing the plaintext with the stream of bytes generated in step 2.

4. Define a function to perform the decryption, which takes the ciphertext and secret key as input, and generates the plaintext by XOR-ing the ciphertext with the same stream of bytes generated in step 2.

5. Test the encryption and decryption functions with sample plaintext and secret key, and verify that the output matches the expected ciphertext and plaintext, respectively.

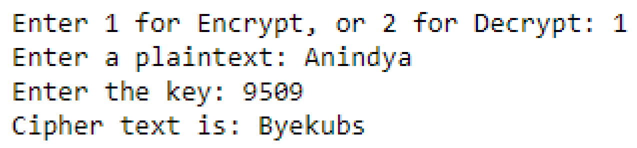
11.6 Result & Discussion:

Multiple binary inputs are now taken as shown in the tables below. The Python code is executed in Jupyter notebook. The plain text and keywords are considered 3 bits at a time.

Encrypted and Decrypted

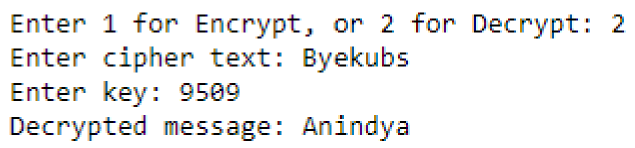
Output:

1. **Encryption:**



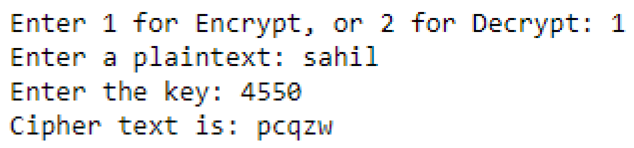
```
Enter 1 for Encrypt, or 2 for Decrypt: 1
Enter a plaintext: Anindya
Enter the key: 9509
Cipher text is: Byekubs
```

Decryption:



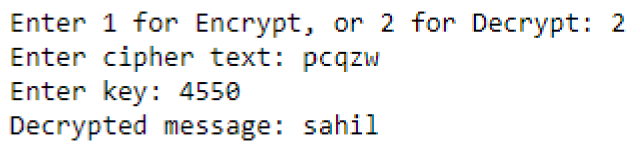
```
Enter 1 for Encrypt, or 2 for Decrypt: 2
Enter cipher text: Byekubs
Enter key: 9509
Decrypted message: Anindya
```

2. **Encryption:**



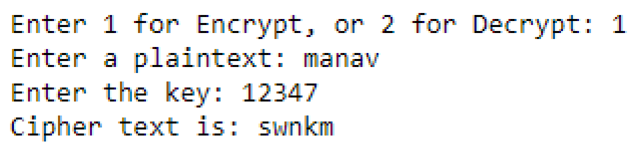
```
Enter 1 for Encrypt, or 2 for Decrypt: 1
Enter a plaintext: sahil
Enter the key: 4550
Cipher text is: pcqzw
```

Decryption:



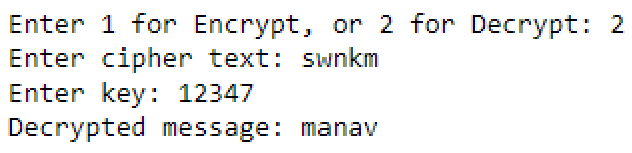
```
Enter 1 for Encrypt, or 2 for Decrypt: 2
Enter cipher text: pcqzw
Enter key: 4550
Decrypted message: sahil
```

3. **Encryption**



```
Enter 1 for Encrypt, or 2 for Decrypt: 1
Enter a plaintext: manav
Enter the key: 12347
Cipher text is: swnkm
```

Decryption:



```
Enter 1 for Encrypt, or 2 for Decrypt: 2
Enter cipher text: swnkm
Enter key: 12347
Decrypted message: manav
```

Tabulation for Encryption:

S.No	Plain text	Keyword	Cipher Text
1	Anindya	9509	Byekubs
2	sahil	4550	pcqzw
3	manav	12347	swnkm
4	ECE-DS	4567	QAQ-JK
5	SRM	DS	XPZ

Tabulation for Decryption:

S.No	Cipher Text	Keyword	Plain text
1	Byekubs	9509	Anindya
2	pcqzw	4550	sahil
3	swnkm	12347	manav
4	QAQ-JK	4567	ECE-DS
5	XPZ	DS	SRM

11.7 CONCLUSION:

The RC4 algorithm is a widely used stream cipher for encryption and decryption of data. It is known for its simplicity and speed. In this report, we have explained the theory behind the RC4algorithm, its implementation using Python and the software used. And the functions that provides a convenient and easy-to-use interface for implementing the RC4 algorithm in Python.

11.8 REFERENCES:

- Data communications and networking I Behrouz A Forouza

11.7 APPENDIX

A. PYTHON CODE

```
def Key_Scheduling(key):
    key_length = len(key)
    if key_length > 256:
        raise ValueError("Key too long (max length = 256)")
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + key[i % key_length]) % 256
        S[i], S[j] = S[j], S[i]
    return S

def pad_key(key):
    padded_key = bytearray(256)
    key_len = len(key)
    if key_len > 256:
        raise ValueError("Key too long (max length = 256)")
    padded_key[:key_len] = bytearray(key.encode())
    padded_key[key_len:] = bytearray(256 - key_len)
    return padded_key

def stream_generation(S):
    i = 0
    j = 0
    while True:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        K = S[(S[i] + S[j]) % 256]
        yield K

def encrypt(plaintext, key):
    key = pad_key(key)
    S = Key_Scheduling(key)
    keystream = stream_generation(S)
    encrypted_text = ""
    for c in plaintext:
        if c.isalpha():
            if c.islower():
                c = chr((ord(c) + next(keystream) - 97) % 26 + 97)
            elif c.isupper():
                c = chr((ord(c) + next(keystream) - 65) % 26 + 65)
        elif c.isnumeric():
            c = str((int(c) + next(keystream)) % 10)
        encrypted_text += c
    return encrypted_text

def decrypt(ciphertext, key):
    key = pad_key(key)
    S = Key_Scheduling(key)
    keystream = stream_generation(S)
    decrypted_text = ""
    for c in ciphertext:
        if c.isalpha():
            if c.islower():
                c = chr((ord(c) - next(keystream) - 97) % 26 + 97)
            elif c.isupper():
                c = chr((ord(c) - next(keystream) - 65) % 26 + 65)
        elif c.isnumeric():
            c = str((int(c) - next(keystream)) % 10)
        decrypted_text += c
    return decrypted_text

ed = input('Enter 1 for Encrypt, or 2 for Decrypt: ').upper()
if ed == '1':
    text = input("Enter a plaintext: ")
    key = input("Enter the key: ")
    encrypted_text = encrypt(text, key)
    print("Cipher text is:", encrypted_text)
elif ed == '2':
    TEXT = input("Enter cipher text: ")
    KEY = input("Enter key: ")
    decrypted_text = decrypt(TEXT, KEY)
    print("Decrypted message:", decrypted_text)
else:
    print('Error in input - try again.')
```