Implementation of RC4 Cryptography Algorithm

11.1 AIM

Implement RC4 algorithm for secure encryption and decryption of data by correctly initializing the key and state, and efficiently generating the keystream according to RC4 specifications.

11.2 INTRODUCTION

11.2.1 Cryptography

Cryptography is the discipline of using codes and ciphers to encrypt text and data, securing it from unauthorized access and ensuring its confidentiality. It enables the conversion of standard text into unintelligible text and vice-versa, offering a means of secure communication across various digital platforms. As the digital landscape expands, the role of cryptography grows more significant, underpinning the security frameworks of IT systems, communications networks, and electronic data transactions.

11.2.2 Confidentiality Security Services

Confidentiality, a primary aspect of security services, refers to the protection of information from unauthorized access and disclosure. Techniques such as encryption ensure that sensitive data, when transmitted or stored, is accessible only to authorized parties. Confidentiality is crucial across many domains, including online banking, corporate data protection, and private communications. To maintain confidentiality, cryptographic keys are used to encrypt data, ensuring that only those with the corresponding decryption key or access rights can interpret the data, thereby preventing leakage of information to unintended recipients.

11.2.3 Evolution of Cryptographic Algorithms

The evolution of cryptographic algorithms has been marked by significant milestones, starting with simple ciphers and advancing to complex encryption methods used in modern computing.

11.2.4 Early Cipher Systems

The earliest known cryptographic tool is the Caesar cipher, named after Julius Caesar, who used it to protect military communications. This method involved shifting the letters of the alphabet by a set number of positions. Although elementary by today's standards, the Caesar cipher introduced the basic concept of substitution, forming a foundation for more complex cryptographic systems.

11.2.5 The Advent of Mechanical and Electromechanical Devices

With technological advances, more sophisticated systems such as the Enigma machine used during World War II by Germany came into use. These mechanical devices offered more complex encryption capabilities, utilizing rotating disks to achieve variable

substitution patterns, which were substantially more difficult to decipher without knowledge of the machine settings.

11.2.6 The Digital Age and Modern Cryptography

The advent of computers transformed cryptography with the development of digital algorithms that could process large amounts of data swiftly. The Data Encryption Standard (DES), introduced in the 1970s, became one of the first widely adopted symmetric key algorithms. DES, however, eventually succumbed to the growing computational powers of modern computers, leading to the development of the Advanced Encryption Standard (AES), which offers enhanced security through longer key lengths and complex encryption cycles.

11.2.7 Public Key Cryptography

The introduction of asymmetric cryptography or public key cryptography revolutionized the field by solving key distribution problems. Algorithms such as RSA (Rivest–Shamir–Adleman) allow two parties to communicate securely without having previously exchanged keys. Public key cryptography is fundamental for digital signatures and secure socket layer (SSL) protocols, which secure communications on the Internet.

11.2.8 Stream Ciphers and RC4

In parallel with these developments, stream ciphers, which encrypt digital data one bit at a time, gained prominence. One of the most famous stream ciphers is RC4, widely used in protocols like WEP and TLS for its simplicity and speed. Although vulnerabilities identified in RC4 have led to its decline in favour of more secure algorithms, its impact on the evolution of cryptographic methods remains significant.

Each phase in the development of cryptographic algorithms reflects the ongoing battle between increasing data security demands and the relentless advancement of technology, with each new method building on the knowledge and shortcomings of its predecessors. This constant evolution is crucial in the face of sophisticated cyber threats, ensuring robust security in the digital age.

11.3 SOFTWARE

Jupiter python interpreter

11.4 RC4 ALGORITHM

RC4 operates in two phases: key scheduling and pseudorandom generation. The RC4 algorithm is a widely-used stream cipher for encryption and decryption, known for its simplicity and efficiency in generating a pseudo-random keystream from a secret key.

11.4.1 Key-Scheduling Algorithm (KSA)

The KSA begins with an array "S" containing 256 bytes, which are initially set to the values 0 to 255. The array is then mixed in a complex way based on a variable-length key, typically ranging from 40 to 256 bits.

The introduction of asymmetric cryptography or public key cryptography revolutionized the field by solving key distribution problems. Algorithms such as RSA (Rivest-Shamir-Adleman)

allow two parties to communicate securely without having previously exchanged keys. Public key cryptography is fundamental for digital signatures and secure socket layer (SSL) protocols, which secure communications on the Internet.

Stream Ciphers and RC4

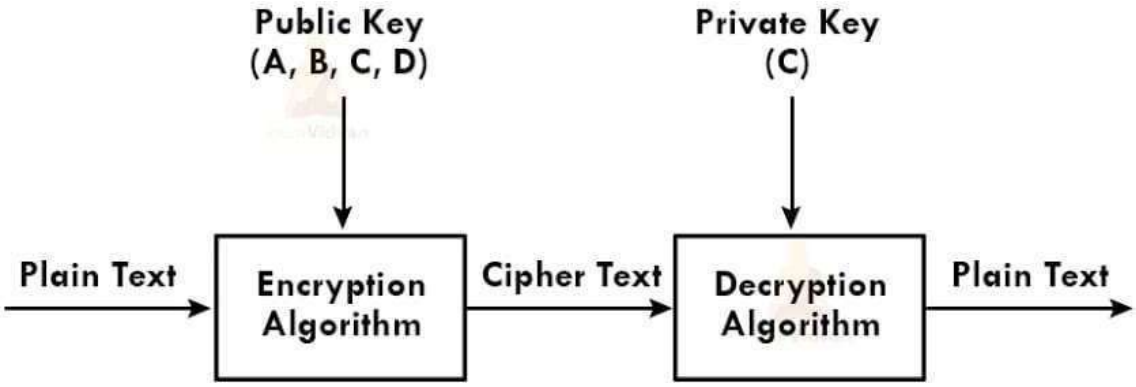

Fig.1 . KSA Block Diagram

In parallel with these developments, stream ciphers, which encrypt digital data one bit at a time, gained prominence. One of the most famous stream ciphers is RC4, widely used in protocols like WEP and TLS for its simplicity and speed. Although vulnerabilities identified in RC4 have led to its decline in favor of more secure algorithms, its impact on the evolution of cryptographic methods remains significant.Each phase in the development of cryptographic algorithms reflects the ongoing battle between increasing data security demands and the relentless advancement of technology, with each new method building on the knowledge and shortcomings of its predecessors. This constant evolution is crucial in the face of sophisticated cyber threats, ensuring robust security in the digital age.

Random Numbers:

Random numbers play an important role in the use of encryption for various network security applications. In this section, we provide a brief overview of the use of random numbers in cryptography and network security and then focus on the principles of pseudorandom number generation. Getting good random numbers is important, but difficult. You don't want someone guessing the key you're using to protect your communications because your "random numbers" weren't (as happened in an early release of Netscape SSL). Traditionally, the concern in the generation of a sequence of allegedly random numbers has been that the sequence of numbers be random in some well-defined statistical sense (with uniform distribution & independent).

In applications such as reciprocal authentication, session key generation, and stream ciphers, the requirement is not just that the sequence of numbers be statistically random but that the successive members of the sequence are unpredictable (so that it is not possible to predict future values having observed previous values), With "true" random sequences, each number is statistically independent of other numbers in the sequence and therefore unpredictable.

However, as is discussed shortly, true random numbers are seldom used; rather, sequences of numbers that appear to be random are generated by some algorithm.

11.4.2 Pseudo-Random Generation Algorithm (PRGA):

Often use deterministic algorithmic techniques to create "random numbers". Although they are not truly random, they can pass many tests of "randomness". It also known as "Pseudorandom Numbers" and created by "Pseudorandom Number Generators (PRNGs)".

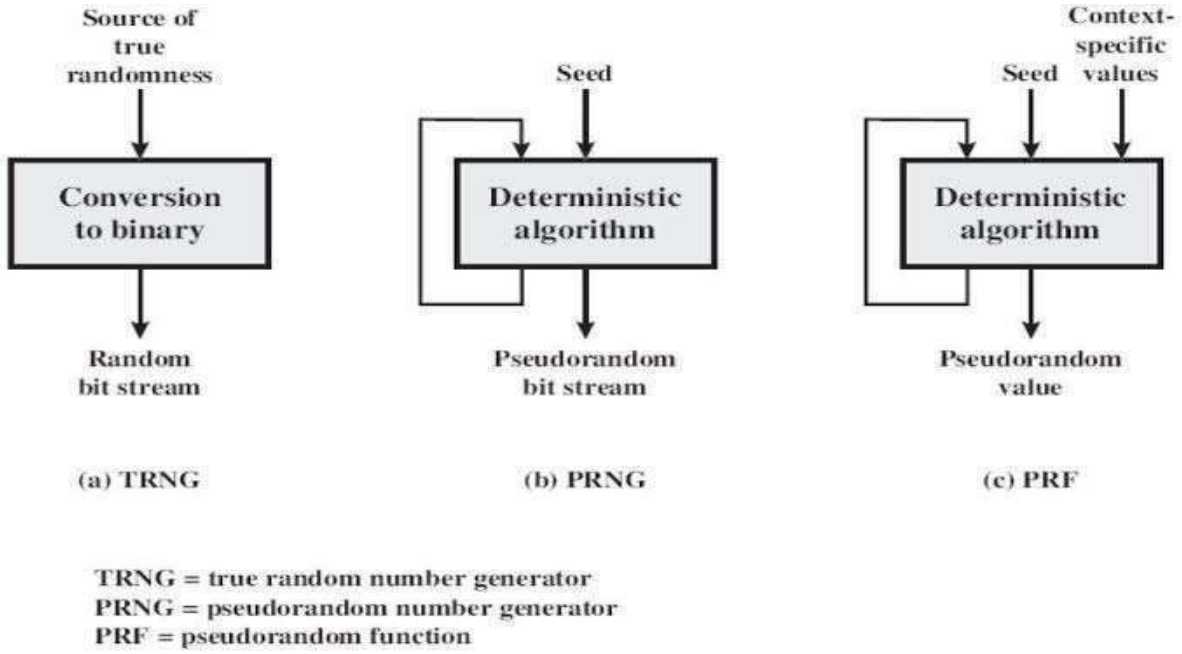

Fig.2 PRGA Block Diagram

During this phase, the algorithm modifies the state of the array "S" and uses its current state to produce the output keystream. For each byte of the plaintext, a byte of the keystream is generated and XORed with the plaintext to produce the ciphertext.

A TRNG takes as input a source that is effectively random; the source is often referred to as an entropy source. In contrast, a PRNG takes as input a fixed value, called the seed, and produces a sequence of output bits using a deterministic algorithm. Typically, as shown, there is some feedback path by which some of the results of the algorithm are fed back as input as additional output bits are produced. The important thing to note is that the output bit stream is determined solely by the input value or values, so that an adversary who knows the algorithm and the seed can reproduce the entire bit stream.

The two different forms of PRNGs, based on application is explained as follows:

Pseudorandom number generator PRNG:

An algorithm that is used to produce an open-ended sequence of bits is referred to as a PRNG. A common application for an open-ended sequence of bits is as input to a symmetric stream cipher.

Pseudorandom function (PRF):

A PRF is used to produce a pseudorandom string of bits of some fixed length. Examples are the symmetric encryption keys and nonces. Typically, the PRF takes as input a seed plus some context specific values, such as a user ID or an application ID.

11.4.3 BLOCK DIAGRAM

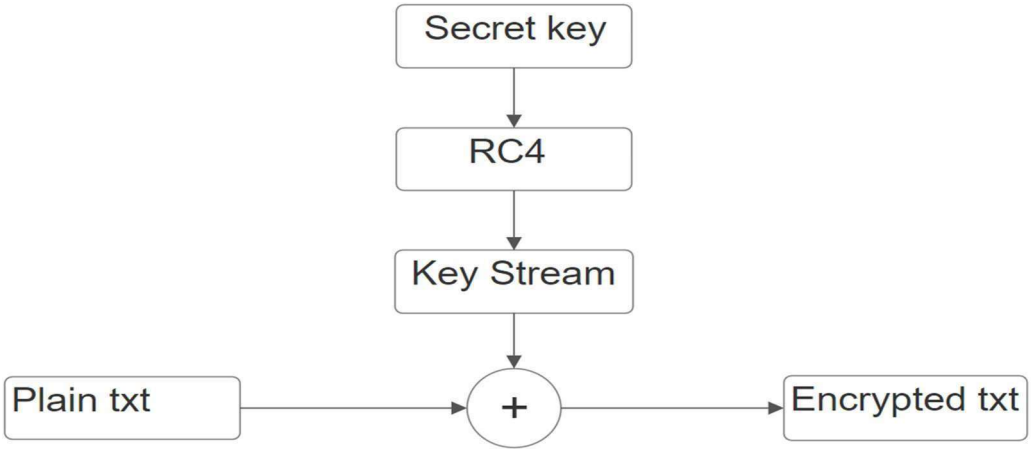

Fig.3 . RC4 Block Diagram

11.5 IMPLEMENTATION METHODOLOGY

RC4 encryption isn't omnipresent. In fact, the Internet Engineering Task Force has explicitly banned RC4's use in some spaces. But knowing how the tool works could be you improve upon it as you look for ways to protect your data.

To explain things simply, RC4 relies on this step-by-step model:

1) Initiate: You input a secret key and the text you'd like to protect.
2) Encrypt: The cipher scrambles your text via encryption. The work happens byte by byte rather than in chunks.
3) Send: Your scrambled text heads to the recipient. That person should have a copy of the secret key you used to protect the data.
4) Decryption: The recipient walks back through these steps to uncover your original text.
5) For the math-minded among us, let's dig a little deeper. RC4 relies on two mathematical concepts:
6) KSA: A key-scheduling algorithm initializes the process in an array typically referred to as "S." That "S" is processed 256 times, and bytes from the key are mixed in too.
7) PRGA: Data is fed in byte by byte, and a mathematical model modifies it. The model looks up values, add them to 256, and uses the sum as the byte within the keystream. It swaps each element with another at least once every 256 rounds.

RC4 relies on random number generators. But unlike other stream ciphers, RC4 doesn't need linear- feedback shift registers.RC4's encryption tools are sophisticated. They typically contain 256 bytes, and the text passes through mathematical rules multiple times before it's considered complete. If you intercept data encrypted with RC4, you'll see only a series of zeroes and ones. But if you have the proper key, you can transform that data into legible information.

Despite its complexity, RC4 is remarkably fast. In fact, it's one of the fastest tools on the market. For people who don't want to spend long minutes on both encryption and decryption, that speed is ideal.

11.6 RESULT AND DISCUSSION

The provided code implements a basic stream cipher for encryption and decryption. It generates a pseudo-random key stream based on a given key and uses it to perform bitwise XOR operations on the plaintext to produce ciphertext, and vice versa for decryption. The main function prompts the user to choose between encryption and decryption, takes input accordingly, and displays the result. However, the code lacks robustness and security features found in modern encryption algorithms and should be used for educational purposes only.

11.6.1 Tabulation for Encryption

S. No	Plain text	Keyword	Cipher Text
1	SRM UNIVERSITY	2030	e3fb20648116864f4b35135949e2
2	ECE	65	f5ea28
3	Computer Communication Network	23	90ea0229a42dbb7c7c47037f70d6e2ac42356809e192937ea09c473a1ec3c95b
4	Professor	Ab	e0db0222b12bbc767c47
5	Mini Project	ccn	fdc0032df408bd76640223643d

11.6.2 Tabulation for Decryption

S. No	Cipher Text	Keyword	Plain Text
1	3f6064e7f934a95c146255df9baa	2030	SRM UNIVERSITY
2	e749df	65	ECE
3	35cc9c737ca2ffd0088fd0bb26da1b0933e408a81511d27bb2b98f5fc3dc	23	Computer Communication Network
4	7e0eb71570aa5cdcb7	Ab	Professor
5	24787101704ed7139a6174	Ccn	Mini Project

11.7 CONCLUSION

This mini project demonstrates a basic implementation of the RC4 algorithm for educational purposes, showcasing key scheduling, stream generation, encryption, and decryption functionalities. The main code block prompts the user to choose between encryption and decryption, takes input accordingly, and displays the result.

11.8 REFERENCES

[1] S. Jarkas (2023, April, 24). RC4 Encryption Algorithm [Online]. Available: https://www.geeksforgeeks.org/rc4-encryption-algorithm/

[2] A. Jaiswal (2022, February,16). What Is RC4 Algorithm & How It Work?[Online]. Available: https://cybermeteoroid.com/wp-content/uploads/2022/02/RC4-algorithm-working.jpg1