CIPHER: ENCRYPTION & DECRYPTION

Punyaslok Sarkar[1], Mrs. Sherly Noel[2]

[1]4th (Final) year, Student, Computer Science & Engineering, CMR Institute of Technology, Karnataka, India
[2]Asst. Professor, Computer Science & Engineering, CMR Institute of Technology, Karnataka, India

---***---

Abstract - *In cryptography, encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot. Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as plaintext, is encrypted using an encryption algorithm – a cipher – generating ciphertext that can be read only if decrypted.*

For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients.

Key Words: *encryption, ciphertext, plaintext, decrypt*

1. INTRODUCTION

Data that can be read and understood without any special measures is called *plaintext* or *cleartext.* The method of disguising plaintext in such a way as to hide its substance is called *encryption.* Encrypting plaintext results in unreadable gibberish called *ciphertext.* We use encryption to ensure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data. The process of reverting *ciphertext* to its original plaintext is called *decryption*.

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient.

A *cryptographic algorithm,* or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a *key* — a word, number, or phrase — to encrypt the plaintext. The same plaintext encrypts to different ciphertext with different keys. The security of encrypted data is entirely dependent on two things: the strength of the cryptographic algorithm and the secrecy of the key.

A cryptographic algorithm, plus all possible keys and all the protocols that make it work comprise a *cryptosystem.* OpenPGP is a cryptosystem.

In conventional cryptography, also called *secret-key* or *symmetric-key* encryption, one key is used both for encryption and decryption. The Data Encryption Standard (DES) is an example of a conventional cryptosystem that is widely employed.

While cryptography is the science of securing data, *cryptanalysis* is the science of analyzing and breaking secure communication. Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. Cryptanalysts are also called *attackers.*

Cryptology embraces both cryptography & cryptanalysis.

2. GLOSSARY

ABE	Attribute-Based Encryption
AES	Advanced Encryption Standard
API	Application Programming Interface
ASN	Abstract Syntax Notation
AVX	Advanced Vector Extensions, a SIMD instruction set extension for Intel architectures
BLISS	Bimodal Lattice Signature Scheme

DES Data Encryption Standard

CERT Computer Emergency Response Team (originally organized by Carnegie Mellon University)

DLP Discrete Logarithm Problem (in finite fields)

DTLS Datagram Transport Layer Security. A datagram mode of TLS.

ECC Elliptic Curve Cryptography

ECDLP Elliptic Curve Discrete Logarithm Problem (in elliptic curve groups)

ECDSA Elliptic Curve Digital Signature Algorithm

FFT Fast Fourier Transform

FPGA Field Programmable Gate Array

GPL GNU General Public License

IBE Identity-Based Encryption

KAT Known Answer Test

KEX Key Exchange (protocol)

LGPL GNU Library or "Lesser" General Public License

MAC Message Authentication Code

NEON SIMD instruction set extension for ARM architectures

NTRU A Lattice-Based Public Key Cryptosystem

NTT Number Theoretic Transform. A finite field analogue of FFT.

3. USER REQUIREMENTS

3.1. FUCTIONAL REQUIREMENTS

The primary function of the software suite is to support the use cases with production-quality implementations of appropriate lattice-based cryptographic algorithms.

A secondary function is to provide a suite of algorithmic implementations that can be integrated into existing applications with relative ease. As such, the new lattice public key algorithms can be used as "drop-in-replacements" to more traditional algorithms such as RSA and ECDSA. The API calling convention of the new primitives should be consistent with current practices in standard C language toolkits (OpenSSL and BoringSSL).

The mechanisms for importing and exporting (portably serialising for transmission or storage) of public and private key information should follow the same outline as that for existing algorithms. One of the main support functionalities provided by the library is integration of new lattice primitives with the OpenSSL certificate processing facilities. All public key operations should be written in a way that supports existing (or future) standards in this field.

Integration with hardware implementations will be done through a generic mechanism that resembles the OpenSSL "engines". Applications should be – as far as possible – able to transparently use the same APIs for hardware cryptography as they would for corresponding software implementations. A functionally equivalent software implementation should be available for all hardware components. The kernel-level driver that interfaces the hardware component to user space may be application and operating-system dependant. On a "bare metal" embedded platform the driver may be incorporated into the suite itself.

3.2. SECURITY EFFICIENCY

Cryptographic software requires special implementation techniques. Some of the basic requirements are:

1. Compare secret strings in constant time

2. Avoid branching controlled by secret data

3. Avoid table look-ups indexed by secret data

4. Avoid secret-dependent loop bounds

5. Prevent compiler interference with security-critical operations

6. Prevent confusion between secure and insecure APIs

7. Avoid mixing security and abstraction levels of cryptographic primitives in the same API layer

8. Use unsigned bytes to represent binary data

9. Use separate types for secret and non-secret information

10. Use separate types for different types of information

11. Clean memory of secret data 12. Use strong randomness

Furthermore, we require explicit validation of most input parameters and all data potentially coming directly from external sources.

4. SYSTEM ARCHITECTURE

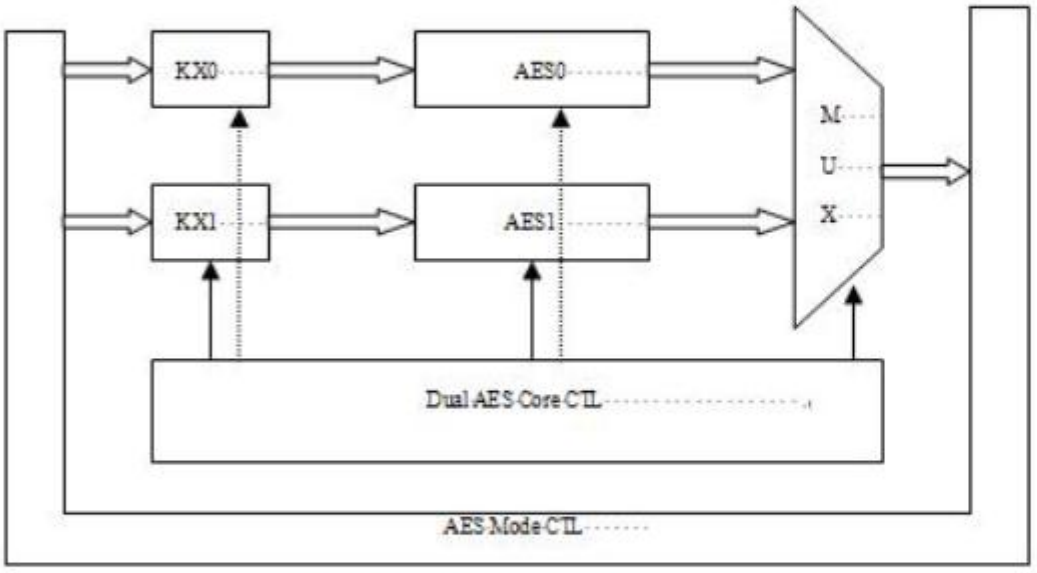
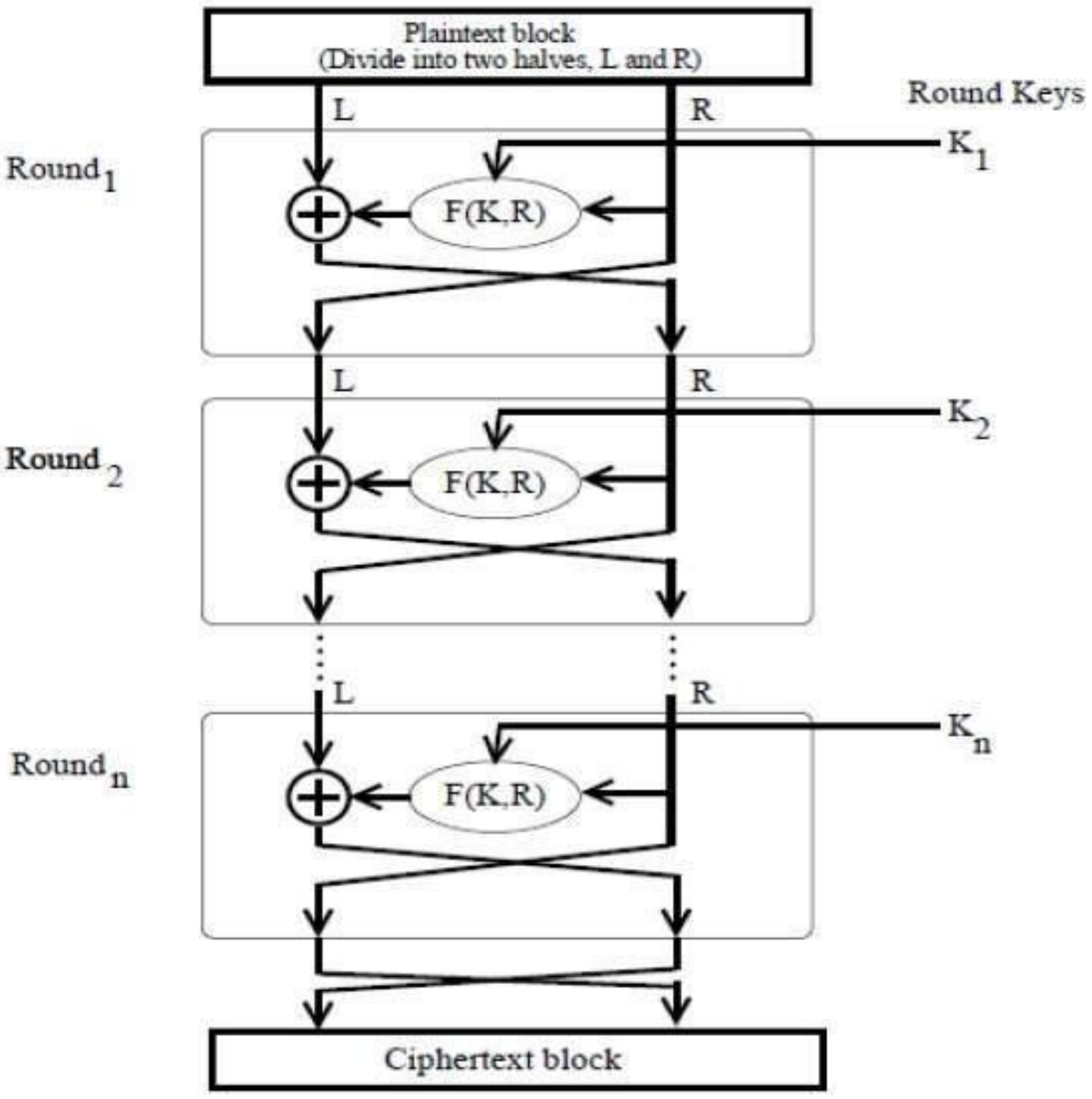


Figure 1. AES Core overview diagram

5. SYSTEM MODEL



6. SYSTEM EVOLUTION

The encryption function converts data from 'plaintext,' or normal text, into 'ciphertext,' which is incomprehensible to the casual observer. The decryption function reverses this process, restoring the data to its original form. In order to perform either of these functions (i.e. to send or receive an encrypted message), the system's user must have a unique 'key,' a sequence of bits. This key is input to the algorithm to successfully perform the desired conversion. The strength of an encryption scheme is dependent both upon the strength of its algorithm and, often, on the length of the keys used for encryption and decryption. Longer key lengths mean more possible keys for an intruder to try and thus imply greater security. Encryption and decryption are generally performed by a computer with the assistance of hardware and/or software cryptographic products.

Commercial encryption technology has evolved since the popular 'Data Encryption Standard' (DES) was released to the public in 1977 and will continue to do so during the foreseeable future. From a situation then when only private key systems were generally in use, public key systems have become increasingly popular, especially for authentication. Under a more traditional single key system, the same key is used both for encrypting and decrypting the message. Although this is reasonably secure, there is a risk that this key will be intercepted when the parties involved exchange keys. A public key system, however, does not necessitate the exchange of a secret key in the transmission of messages. The sender encrypts the message with the recipient's freely-disclosed, unique public key. The recipient, in turn, uses their unique private key to decrypt the message. It is also possible to encrypt messages with the sender's private key, allowing anyone who knows the sender's public key to decrypt the message.

7. SYSTEM REQUIREMENTS :

JAVA 5.0

MEMORY: 1 GB

FREE DISK SPACE : 1 GB

PROCESSOR SPEED : 1.5 Ghz

Eclipse

8. CODE

```
public class MayankEnigmaEncryption {
 public static void main(String...s){
 String message, encryptedMessage = "";
 int key;
 char ch;
 Scanner sc = new Scanner(System.in);
  System.out.println("Enter a message to be encrypt: ");
 message = sc.nextLine();
  System.out.println("Enter key: ");
 key = sc.nextInt();
  for(int i = 0; i < message.length(); ++i){
 ch = message.charAt(i);
  if(ch >= 'a' && ch <= 'z'){
 ch = (char)(ch + key);
  if(ch > 'z'){
 ch = (char)(ch - 'z' + 'a' - 1);
 }
  encryptedMessage += ch;
 }
 else if(ch >= 'A' && ch <= 'Z'){
 ch = (char)(ch + key);
  if(ch > 'Z'){
 ch = (char)(ch - 'Z' + 'A' - 1);
 }
```

```
encryptedMessage += ch;
}
else {
encryptedMessage += ch;
}
}
 System.out.println("Encrypted Message = " + encryptedMessage);
}
}
import java.util.Scanner;
public class MayankEnigmaDecryption {
public static void main(String...s){
String message, decryptedMessage = "";
int key;
char ch;
Scanner sc = new Scanner(System.in);
 System.out.println("Enter a message to be decrypt: ");
message = sc.nextLine();
 System.out.println("Enter key: ");
key = sc.nextInt();
 for(int i = 0; i < message.length(); ++i){
ch = message.charAt(i);
 if(ch >= 'a' && ch <= 'z'){
ch = (char)(ch - key);
 if(ch < 'a'){
ch = (char)(ch + 'z' - 'a' + 1);
}
 decryptedMessage += ch;
}
else if(ch >= 'A' && ch <= 'Z'){
ch = (char)(ch - key);
```

```
if(ch < 'A'){
ch = (char)(ch + 'Z' - 'A' + 1);
}
 decryptedMessage += ch;
}
else {
decryptedMessage += ch;
}
}
 System.out.println("Decrypted Message = " + decryptedMessage);
}
}
```

9. RESULT AND DISCUSSION

When any word is given it will be encrypted in a form of different alphabets or numbers or both. Then if its required it will be decrypted back to the word that was given initially. In this way, cryptographic ciphers are used to convert ciphertext to plaintext and back. Symmetric cryptography uses the same key to encrypt and decrypt data, while asymmetric cryptography, also known as public key cryptography, uses public and private keys to encrypt and decrypt data.

10. REFERENCES

- Abraham Sinkov, Elementary Cryptanalysis: A Mathematical Approach, Mathematical Association of America, 1966.
- William Stallings, Cryptography and Network Security, principles and practices, 4th Edition
- Stinson, Douglas R. (1995), Cryptogtaphy / Theory and Practice, CRC Press