

# **Comparative Study and Analysis of AES and DES Algorithms on Various Multimedia Platforms.**

## **A PROJECT REPORT**

*Submitted by*

TARUSHI JAIN, 18BCE0859  
SHAURYA SINGH, 18BCE2017  
KARTIKAY GUPTA, 18BCE2199

Course Code: CSE3501  
Course Title: Information Security Analysis and Audit

Under the guidance of  
**Anil Kumar K**  
**Designation, School,**  
**VIT University, Vellore.**



**SCHOOL OF COMPUTER SCIENCE  
AND ENGINEERING**

**November 2020**

## **ACKNOWLEDGEMENT**

We want to acknowledge our respected professor Anil Kumar K who helped us in all research and assisted us in all the way he could have done. He has been the backbone by supporting us in all the possible ways and made us this far to achieve our result. We thank our prestigious university for giving us this wonderful opportunity for proving us whom we are and not limiting theories to just books but also in the involvement of various mini projects. We consider this opportunity as a chance to deliver our token of thanks to VIT for providing us immense facilities and equipment required to make our project a great success. We take this time to express all our gratitude towards all the staff members who helped us during the course of the project.

# **TABLE OF CONTENTS**

## **1. Introduction**

### **1.1 Abstract**

## **2. Background**

### **2.1 Symmetric key cryptosystems**

## **3. Overview and Planning**

### **3.1 Proposed Work**

### **3.2 Working Model**

### **3.3 Design description**

### **3.4 Hardware Requirements**

### **3.5 Software Requirements**

## **4. Literature Summary and Review**

### **4.1 Literature Survey**

## **5. System Implementation**

### **5.1 Description of Modules/Programs**

### **5.2 Source Code**

### **5.3 Snapshot/ Execution of program**

### **5. 4 Results**

## **6 Conclusion**

### **6.1 Conclusion and Future Work**

## **7. References**

# **1. Introduction**

## **1.1 Abstract**

Cryptography is a critical apparatus for the assurance of multimedia content. All the Multimedia files are scrambled before being circulated over the web. Because of the Encryption of the file, it is impossible for everyone to get the keys. So the key for the decoding of the substance ought not to be unveiled to any other person other than the substance supplier. Encryption is an approach to shield data from undesirable assaults by transforming it into a shape that can't be perceived by any aggressors. Information encryption for the most part is changing of the information, for example, content, picture, sound, and so forth with the goal that it is indistinguishable, undetectable or invulnerable amid the transmission. So with a specific end goal to recoup the first information the receiver just inverses information encryption known as data decryption. The problem that we are dealt for this project is to know, understand and get acquainted with the important encryption algorithm related to the security of the documents more specifically the ones that are subject to the transfer between 2 separate entities. For the purpose of this project we need to study the AES, DES and RSA algorithm and their nature, the way in which it works, how it implements over different types of files and gives us an output in different time frames with appropriate results. Moreover, we need to study and use the Eclipse tool which helps us execute the encryption algorithm with subtle variations in the file types and providing a Java platform to execute it.

## 2. Background

Cryptographic techniques play crucial role when users exchange information. When multimedia contents are shared among the users, it faces security threats. Usually multimedia contents take much space. Encryption technique should be time efficient. In this work we consider four encryption techniques: Blowfish, AES, XOR and RSA and four types of media content: text, image, audio and video. Simulation shows that AES is time efficient than others. Comparing between symmetric and asymmetric cryptography, symmetric cryptographic techniques take less time than asymmetric technique.

Cryptography also termed as “secret writing” is a science of concealing information so that only the intended parties can have access to the private information. It protects the privacy and modification of data which may occur due to active and passive attacks in the channel. Cryptography consists of two things – Plain text and Ciphertext .Plain text is the original data which the sender intends to send and Ciphertext is the encrypted format of the plain text. The plain text is converted to the Ciphertext and vice versa with the help of an encryption and decryption algorithm. The encryption- decryption algorithms are mainly classified into two type e.g. symmetric key algorithm and asymmetric key algorithm. In this paper, different encryption algorithms are discussed along with their applications. In traditional end-to-end security solutions, only the end hosts can verify the validity and integrity of the traffic, which leads to several problems. First, they are not effective if the network infrastructure itself is under attack and unable to deliver packets. Second, they are not enough to provide sufficient security by themselves. The large amount of traffic on the current Internet is just unsolicited e-mail or other garbage, and the culprits behind attacks are rarely caught. Therefore, we feel that there is a clear need for security solutions, where the security policies are applied at every hop as the packet travels through the network. If the network infrastructure can verify the validity of the traffic, countermeasures against various attacks could be taken within the network, and not only by the end hosts. This would allow attacks to be stopped quickly and more efficiently, and would increase the chance of catching perpetrators.

Encryption is an approach to shield data from undesirable assaults by transforming it into a shape that can't be perceived by any aggressors. Information encryption for the most part is changing of the information, for example, content, picture, sound, and so forth with the goal that it is indistinguishable, undetectable or invulnerable amid the transmission. So with a specific end goal to recoup the first information the receiver just inverts information encryption known as data decryption.

The encryption process can be described as  $C = E(P, K)$  Where,  $P$  = Original data  $E$  = Encryption Algorithm

$K$  = Encryption Key

$C$  = Cipher message, which is transmitted and can be subject to attack

The decryption procedure can be described as  $P = D(C, K)$  Where,  $C$  = Cipher message;  $D$  = Decryption Algorithm  $K$  = Decryption Key;  $P$  = Recovered data

## **2.1 Symmetric key cryptosystems**

All the classical cryptosystems that were developed before 1970 are an example of symmetric key cryptosystems. Besides that, most of the cryptosystems developed after 1970 are symmetric [3]. Some of the very popular examples of modern symmetric key include:

1. AES(AdvancedEncryptionStandard)
2. DES(DataEncryptionStandard)

All symmetric keys have a common interest, they depend on a secret shared between communicating parties. The secret can be used as both encryption and decryption key. The disadvantage of symmetric key is that it is not able to handle a large network of communication.

On the other hand, the symmetric key requires a smaller size for the same level of security as public key cryptosystems, thus, making the communication faster and memory required smaller.

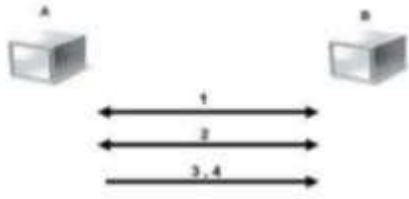


Figure 1. Symmetric Encryption

1. A and B agree on cryptosystems.
2. A and B agree on the key to be used.
3. A encrypts message using the shared key.
4. B decrypts the cipher message using the shared key.

## **3.Overview and Planning**

### **3.1 Proposed Work**

We preprocess the documents to represent them suitable for learning algorithm. Then we will implement an algorithm to learn the training documents to generate the document classifier. In our project, association rule mining algorithm is applied to generate the associative classifier. In testing stage, we put the new documents into the document classifier and get the classified documents. In project implementation, there are three parts: data preprocessing, generate association classifier and validation.

### **3.2 Working Model**

We first of all preprocess the algorithms and the documents to fit accordingly to the algorithms now to put it in the right form we compare the design component of the algorithms to compare and see which algorithm works best. AES and triple DES (TDES OR 3DES) are the most usually utilized block figures. By outline, AES is quicker in term of rounds, i.e., exchanging between the equipment is less complex than exchanging between programming. Notwithstanding, DES encodes information in 64 bit and uses a 56 bit key, because of which it has an estimated probability of 72 quadrillion . Despite the fact that the number is tremendous, because of the computing energy of current technology, it isn't adequate and can be presented to assaults. Along these lines, because of DES not having the capacity to stay aware of the technology progression, it isn't a fitting security. Due to the tremendous utilization of DES, the speediest arrangement was to refresh to 3DES, which is sufficiently secure for current technology. The Rijndael calculation has been supplanted 3DES. The fundamental purpose behind Rijndael to be picked as the cutting edge AES may be: i. Security ii. Software and Hardware Performance iii. Suitability in restricted-space environments iv. Resistance to power analysis and other implementation attacks



Factors	AES	DES
Key length	128, 192 or 256 bits	56 bits
Cipher type	Symmetric block cipher	Symmetric block cipher
Block size	128, 192, or 256 Bits	64 bits
Cryptanalysis resistance	Strong against differential, linear, interpolation and square attacks.	Vulnerable to differential and linear cryptanalysis; weak substitution tables
Security	Considered secure	Proven inadequate
Possible keys	2 <sup>128</sup> , 2 <sup>192</sup> or 2 <sup>256</sup>	2 <sup>56</sup>
Times required checking all possible keys 50 billion Keys per second	For a 128-bit key 5 x 10 <sup>21</sup> years	For a 56-bit 400 days
Rounds	10,12 and 14 for 128, 192 And 256-bits respectively	16

**Table 1 Differences between AES and DES**

### 3.3 Design description

#### 3.3.1 AES

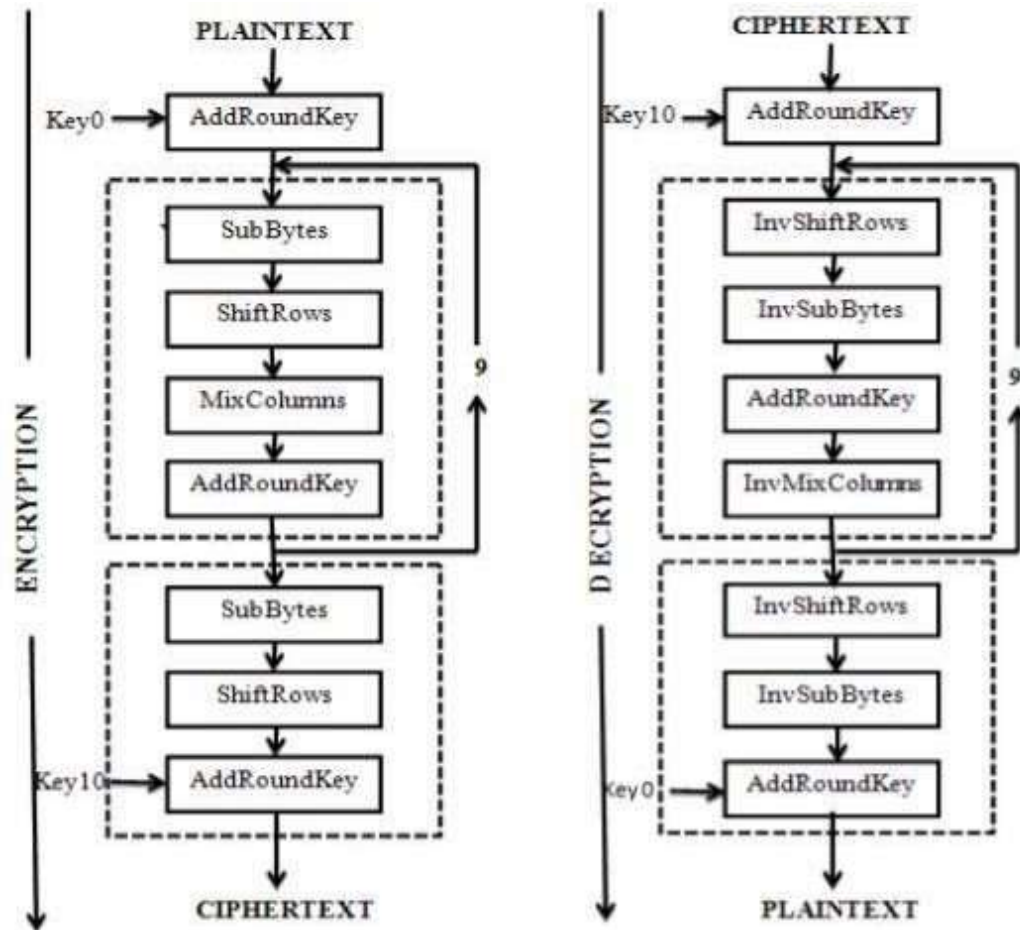
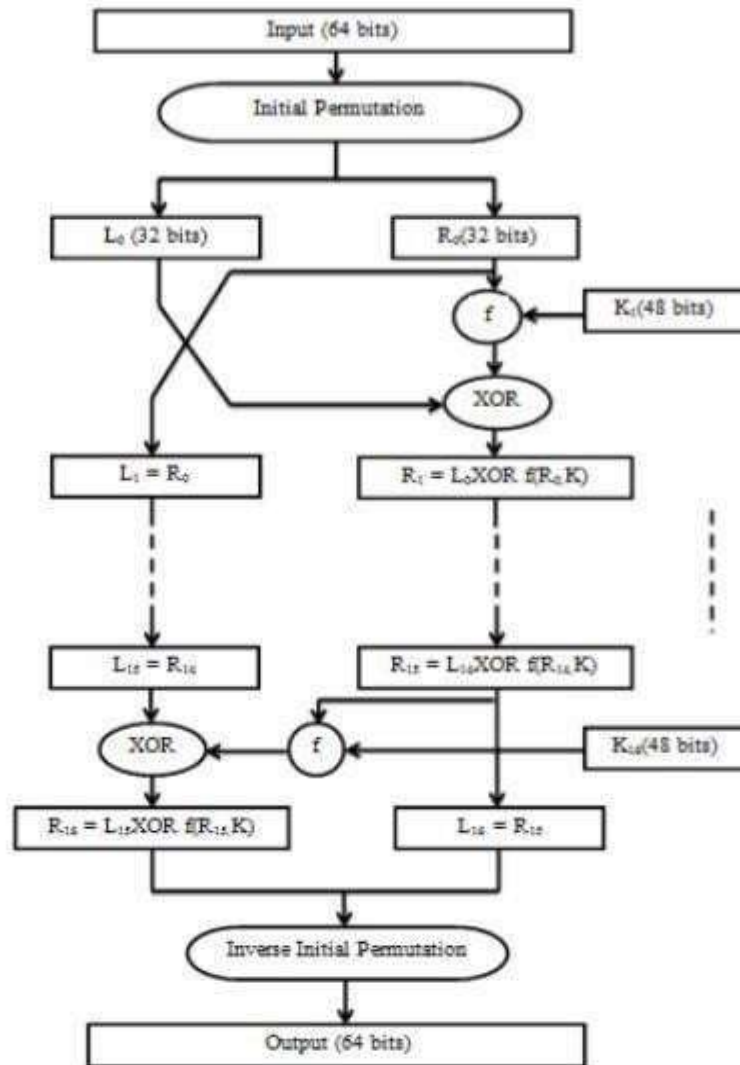


Figure 4. Flowchart of AES Encryption and Decryption

### 3.3.2 DES



## **4.Literature Summary and Review**

### **4.1 Literature Survey**

[1] Performance Evaluation of Cryptographic Algorithms: AES and DES, Divya Sukhija

Student at JCD College of Engineering, India.

In this paper author discuss about strength and weakness of these algorithms by using various sources and explains about cryptography. Cryptographic algorithms are also known as encryption algorithms. It is a mathematical procedure for performing encryption of data. Through the use of an algorithm, information is made into meaningless cipher text and require the use of a key to transform the data back into its original form. There are a number of encryption algorithms available to encrypt the data. Their strengths depend upon the cryptographic system. Any computer system which involves cryptography is known as cryptographic system, the strength of encryption algorithm heavily relay on the computer system used for the generation of keys

Pros and cons:

Advanced Encryption Standard (AES): • AES is highly efficient, secure and it is not complex. • It needs more processing. • It requires more rounds of communication as compared to DES.

Data Encryption Standard (DES): • DES has been around a long time since 1978. and has been studied to death. even now no real weakness have been found. • The most efficient attack is still brute force. The 56 bit key size is the biggest defect. • Hardware implementations of DES are very fast; DES was not designed for software and hence runs relatively slowly.

[2] IITM Journal of Management and ITSOUVENIR National Conference on Emerging Trends in Information Technology-Cyber Security.

In this paper author discuss about how aes and des works in ATM. This paper based on AES and DES cryptographic algorithm technique, how DES at some place used in ATM and AES is more secure than DES so at everywhere in ATM AES algorithm should be used.

ATM uses secret key ,called the PIN key ,to derive the PIN from the account number in terms of algorithm known as DES. The result is natural PIN ,an offset can be added to it and then final PIN which the customer enter. The offset has no cryptographic function, it just used for customer to choose their own PIN

ATM using DES has been breached 24 hours. Advanced encryption standard (AES) is recent and new encryption algorithm. AES support AES with CBC (cipher block chaining) mode to IP security . Usually DES is use to encrypt the ATM transaction but most of time need more secure triple

DES There are many illegal withdrawals take place from ATM. Ross Anderson, a researcher investigated various cases of illegal withdrawals and exposing errors in bank security. There have many cases in which criminals used fake machines, attached keypads or card readers to real machines, and record customer's PIN and bank account details to access the accounts illegally.

[3] An Advanced Security Analysis by Using Blowfish Algorithm R. Vasantha<sup>1</sup>, Dr. R. Satya Prasad<sup>2</sup>  
<sup>1</sup>Research Scholar, Department of CSE, Acharya Nagarjuna University, Guntur, India  
<sup>2</sup>Associate Professor, Department of CSE, Acharya Nagarjuna University, Guntur, India

In this paper author discuss about security of information in cloud. Cloud enrolling appears to be to a great degree accommodating organization for a few people; every third individual is using cloud in different ways. As a result of its versatility, various individuals are trading their data to cloud. Disseminated registering exhibit a greatly viable application for affiliations. Since affiliations have broad measure of data to store and cloud gives that space to its customer and moreover empowers its customer to get to their data from wherever at whatever point easily. As people are saving their own and basic data to fogs, so it transforms into a vital issue to store that data securely. Numerous calculations exist for the information security like DES, AES, and Triple DES. These are symmetric key calculations in which a solitary key is utilized for encryption and decoding.

Each message comprises of various hash esteem, yet the hashing has one downside i.e. once the information is scrambled, it can't be unscrambled. This confinement of hashing was evacuated by symmetric and awry calculations. Symmetric calculation is otherwise called "Mystery Key Encryption Algorithm" in symmetric key calculation, just a single key is utilized for encryption and unscrambling i.e. private key, where as in hilter kilter calculation both open and private keys are utilized for encryption and decoding, uneven calculation is otherwise called "Open Key Encryption Algorithm". AES works quick on both programming and equipment.

[4] Simulation of Image Encryption using AES Algorithm. P.Karthigaikumar Asst.Professor (SG) Department of Electronics and Communication Karunya University,Coimbatore.

In this paper comparative study of these existing techniques has been presented also present types of images and different techniques of image processing with steps used to process an image. Security in transmission of computerized pictures has its significance in today's picture interchanges, because of the expanding utilization of pictures in modern process, it is fundamental to shield the private picture information from unapproved get to, Image security has turned into a basic issue. The troubles in guaranteeing people security turn out to be progressively testing. Different techniques have been explored and created to secure information and individual protection. Encryption is likely the clearest one. With a specific end goal to shield significant data from undesirable readers, picture encryption is basic. In these section different techniques for image processing to provide secure image processing proposed by various researchers has been reviewed.

### **DES Pros and Cons:**

It was a secure algorithm till the 1970's. It was based on the hardware implementation, hence it runs fast.

It was easily cracked by the Brute Force attack and described as a weak algorithm in terms of security. Its software implementation cannot be described.

**AES Pros and Cons:**

It provides the better security as compared to the DES, TDES, IDEA, BLOWFISH, etc. algorithms.

It is free of cost.

It provides the better security as compared to the DES, TDES, IDEA, BLOWFISH, etc. algorithms.

It is free of cost.

[5] “Design and Implementation of a Private and Public Key Crypto Processor and Its Application to a Security System” HoWon Kim, Member, IEEE, and Sunggu Lee, Member, IEEE.

This paper entirely talks about the design and implementation of a crypto processor, a special-purpose microprocessor optimized for the execution of cryptography algorithms. This crypto processor can be used for various security applications such as storage devices, embedded systems, network routers, security gateways using IPsec and SSL protocol, etc. The crypto processor consists of a 32-bit RISC processor block and coprocessor blocks dedicated to the AES, KASUMI, SEED, triple-DES private key crypto algorithms and ECC and RSA public key crypto algorithm. The dedicated coprocessor block permits fast execution of encryption, decryption, and key scheduling operations.

Here the high performance and high flexibility of the crypto processor design makes it applicable to various security applications such as storage devices, embedded systems, network routers, security gateways for IPsec and SSL protocol processing, etc.

It can also be implemented to develop additional high performance public key crypto blocks. Also, to enhance the security of the crypto processor, it can be devised to side channel attack resistant techniques in the private and public key crypto blocks.

[6] Comparative Analysis of AES and RC4 Algorithms for Better Utilization Nidhi Singhal<sup>1</sup>, J.P.S.Raina<sup>2</sup> Department of Electronics & Communication, BBSB engineering college, Fatehgarh Sahib, Punjab, India.

In this paper they compared the AES algorithm with different modes of operation (block cipher) and RC4 algorithm (stream cipher) in terms of CPU time, encryption time, memory utilization and throughput at different settings like variable key size and variable data packet size. And we can conclude from this paper that it is better to use symmetric crypto algorithms rather than asymmetric ones for better efficiency as they are quick and fast to response while getting executed in a system.

## Results:

- Here they compared AES and RC4 algorithms for encryption time over different packet sizes and it is resulted that RC4 took less time for encryption beating AES in efficiency. Also, when compared the decryption time, RC4 takes less time with respect to AES and when tested against varying key sizes from 128 bits to 192 bits to 256 bits, encryption time for RC4 is almost constant and is less than AES. Hence it consumes less power w.r.t AES. But for different modes of AES, encryption time increases as key size increases and the results hold same with the decryption one.
- The second result was about the Throughput for AES and RC4 with different key size. The result shows the superiority of RC4 over AES. With different key sizes RC4 gives almost the same result. But for different modes of AES, throughput decreases as key size increases because of more usage of computational power and encryption characteristics. Thus, RC4 is fast in nature and consume less power w.r.t its counterparts. Better results were obtaining in decryption w.r.t encryption.
- In the result we talk about the memory utilization for AES and RC4 with different file size. As per the result AES consume more memory w.r.t.RC4 because of its characteristics. And as the file size increases memory size is drastically increased in AES means for extra-large files, we need a system with good memory and more CPU.

[7] An FPGA-Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists AJ Elbirt<sup>1</sup>, W Yip<sup>1</sup>, B Chetwynd<sup>2</sup>, C Paar<sup>1</sup> ECE Department, Worcester Polytechnic Institute 100 Institute Road Worcester, MA 01609, USA.

This paper explores about the hardware implementations of encryption algorithms as they provide cryptographic algorithm agility, physical security, and potentially much higher performance than software solutions. This contribution investigates the significance of FPGA implementations of the Advanced Encryption Standard candidate algorithms. Multiple architectural implementation options are explored for each algorithm.



- For each of the AES finalists, an implementation analysis for each architecture option when optimized for both area and speed was performed to determine the suitability for hardware implementation of each finalist.
- Upon comparison, it was determined that the Serpent algorithm yielded the best throughput results in both modes of operation.
- When evaluating throughput per slice, the Serpent algorithm was also found to yield the best results when operating in non-feedback mode while the Rijndael algorithm was found to yield the best results when operating in feedback mode.
- The Serpent algorithm is clearly the best choice when throughput is the key evaluation characteristic regardless of the mode of operation.

[8] A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security by Gurpreet Singh M.Tech Research Scholar, Department of Computer Science and Engineering Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India.

Encryption is the process of scrambling a message so that only the intended recipient can read it. Encryption can provide a means of securing information. As more and more information is stored on computers or communicated via computers, the need to insure that this information is invulnerable to snooping and/or tampering becomes more relevant. With the fast progression of digital data exchange in electronic way, Information Security is becoming much more important in data storage and transmission. Information Confidentiality has a prominent significance in the study of ethics, law and most recently in Information Systems. With the evolution of human intelligence, the art of cryptography has become more complex in order to make information more secure. Arrays of Encryption systems are being deployed in the world of Information Systems by various organizations. In this paper, a survey of various Encryption Algorithms is presented.

This paper presents a detailed study of the popular Encryption Algorithms such as RSA, DES, 3DES and AES. The use of internet and network is growing rapidly. So there are more requirements to secure the data transmitted over different networks using different services. To provide the security to the network and data different encryption methods are used. In this paper, a survey on the existing works on the Encryption techniques has been done. To sum up, all the techniques are useful for real-time Encryption. Each technique is unique in its own way, which might be suitable for different applications and has its own pro's and con's. According to research done and literature survey it can be found that AES algorithm is most efficient in terms of speed, time, throughput and avalanche effect. The Security provided by these algorithms can be enhanced further, if more than one algorithm is applied to data. Our future work will explore this concept and a combination of algorithms will be applied either sequentially or parallel, to setup a more secure environment for data storage and retrieval.

[9] Design and Implementation of a Private and Public Key Crypto Processor and Its Application to a Security System HoWon Kim, Member, IEEE, and Sunggu Lee, Member, IEEE.

This paper presents the design and implementation of a crypto processor, a special-purpose microprocessor optimized for the execution of cryptography algorithms. This crypto processor can be used for various security applications such as storage devices, embedded systems, network routers, security gateways using IPSec and SSL protocol, etc. The crypto processor consists of a 32-bit RISC processor block and coprocessor blocks dedicated to the AES, KASUMI, SEED, triple-DES private key crypto algorithms and ECC and RSA public key crypto algorithm. The dedicated coprocessor block permits fast execution of encryption, decryption, and key scheduling operations. The 32-bit RISC processor block can be used to execute various crypto algorithms such as Hash and other application programs such as user authentication and IC card interface. The crypto processor has been designed and implemented using an FPGA, and some parts of crypto algorithms have been fabricated as a single VLSI chip using 0.5 $\mu$ m CMOS technology. To test and demonstrate the capabilities of this chip, a custom board providing real-time data security for a data storage device has been developed.

For future work, they plan to develop additional high performance public key crypto blocks. Also, to enhance the security of our crypto processor, we will devise side channel attack resistant techniques in the private and public key crypto blocks.

## 5. System Implementation

### 5.1 Description of the code/program

In this program we run 2 different algorithm namely AES and DES then we find the file that can be any file type example PDF, doc, txt, xlsx, etc then we calculate the time of finding the file by different algorithms and thus conclude which is the best algorithm.

### 5.2 Source Code

```
package cyber; import java.io.File; import java.io.FileInputStream; import
java.io.FileOutputStream; import java.security.Key; import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec; public class cryptog { private static long
startTime = System.currentTimeMillis(); static void fileProcessor(int cipherMode
,String key,File inputFile,File outputFile){ try {
Key secretKey = new SecretKeySpec(key.getBytes(),"AES");
Cipher cipher = Cipher.getInstance("AES"); cipher.init(cipherMode,
secretKey);
FileInputStream inputStream = new FileInputStream(inputFile); byte[]
inputBytes = new byte[(int) inputFile.length()]; inputStream.read(inputBytes);
byte[] outputBytes = cipher.doFinal(inputBytes);
FileOutputStream outputStream = new FileOutputStream(outputFile);
outputStream.write(outputBytes); inputStream.close(); outputStream.close();
} catch (Exception e) { e.printStackTrace();
}
}
```

```
public static void main(String[] args) {
String key = "This is a secret";
```

```
File inputFile = new File("C:\\Users\\Resham B\\Documents\\DSC_0258.jpg");  
  
File encryptedFile = new File("text.encrypted"); File decryptedFile = new File(  
"decrypted-text.jpg"); try { cryptog.fileProcessor(Cipher.ENCRYPT_MODE,key,  
inputFile,encryptedFile); cryptog.fileProcessor(Cipher.DECRYPT_MODE,key,  
encryptedFile,decryptedFile);  
  
System.out.println("Success");  
  
} catch (Exception ex) {  
  
System.out.println(ex.getMessage()); ex.printStackTrace();  
  
}  
  
long endTime = System.currentTimeMillis();  
  
System.out.println("It took " + (endTime - startTime) + " milliseconds");  
  
}  
}
```

## 5.3 Snapshot/Execution of the Program

```
cryptog.java
1  e crypto;
2* java.io.File;
8  class cryptog {
9  private static long
10 startime = System.currentTimeMillis();
11 public void fileProcessor(int cipherMode ,String key,File inputFile,File outputFile)
12
13     try {
14         Key secretKey = new SecretKeySpec(key.getBytes(),"DES");
15         Cipher cipher = Cipher.getInstance("DES");
16         cipher.init(cipherMode,secretKey);
17         FileInputStream inputStream = new FileInputStream(inputFile);
18         byte[]
19         inputBytes = new byte[(int) inputFile.length()];
20         inputStream.read(inputBytes);
21         byte[] outputBytes = cipher.doFinal(inputBytes);
22         FileOutputStream outputStream = new FileOutputStream(outputFile);
23         outputStream.write(outputBytes);
24         inputStream.close();
25         outputStream.close();
26     } catch (Exception e)
27     {
28         e.printStackTrace();
29     }

```

Problems Javadoc Declaration Console Coverage

<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 7:09:00 pm – 7:09:01 pm)

**java.security.InvalidKeyException: Wrong key size**

- at java.base/com.sun.crypto.provider.DESCrypt.init(DESCrypt.java:536)
- at java.base/com.sun.crypto.provider.ElectronicCodeBook.init(ElectronicCodeBook.java:97)
- at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:593)
- at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:469)
- at java.base/com.sun.crypto.provider.DESCipher.engineInit(DESCipher.java:186)
- at java.base/javax.crypto.Cipher.implInit(Cipher.java:869)
- at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:931)
- at java.base/javax.crypto.Cipher.init(Cipher.java:1301)
- at java.base/javax.crypto.Cipher.init(Cipher.java:1238)
- at crypto.cryptog.fileProcessor(cryptog.java:16)
- at crypto.cryptog.main(cryptog.java:38)

Success

It took 201 milliseconds

## 5.4 Results

### 5.4.1 Statistics of the run time execution of AES:

#### 1. XLSX file

Runtime 93ms

Used memory is bytes: 1675464

Used memory is megabytes: 1

**Code:**

```
11= static void fileProcessor(int cipherMode ,String key,File inputFile,File outputFile)
12 {
13     try {
14         Key secretKey = new SecretKeySpec(key.getBytes(),"AES");
15         Cipher cipher = Cipher.getInstance("AES");
16         cipher.init(cipherMode,secretKey);
17         FileInputStream inputStream = new FileInputStream(inputFile);
18         byte[]
19         inputBytes = new byte[(int) inputFile.length()];
20         inputStream.read(inputBytes);
21         byte[] outputBytes = cipher.doFinal(inputBytes);
22         FileOutputStream outputStream = new FileOutputStream(outputFile);
23         outputStream.write(outputBytes);
24         inputStream.close();
25         outputStream.close();
26     } catch (Exception e)
27     {
28         e.printStackTrace();
29     }
30 }
31= public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\5th SEM TOTAL.xlsx");
34     File encryptedFile = new File("text.encrypted");
```

Problems Javadoc Declaration Console Coverage

<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:22:02 pm – 9:22:03 pm)

Success

It took 93 milliseconds

## Output:



## 2. Word file

Runtime 113ms

Used memory is bytes: 1357464

Used memory is megabytes: 1

## Code:

```
cryptog.java
12 {
13     try {
14         Key secretKey = new SecretKeySpec(key.getBytes(), "AES");
15         Cipher cipher = Cipher.getInstance("AES");
16         cipher.init(cipherMode, secretKey);
17         FileInputStream inputStream = new FileInputStream(inputFile);
18         byte[]
19         inputBytes = new byte[(int) inputFile.length()];
20         inputStream.read(inputBytes);
21         byte[] outputBytes = cipher.doFinal(inputBytes);
22         FileOutputStream outputStream = new FileOutputStream(outputFile);
23         outputStream.write(outputBytes);
24         inputStream.close();
25         outputStream.close();
26     } catch (Exception e)
27     {
28         e.printStackTrace();
29     }
30 }
31 public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\FFCS 7TH SEM.docx");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.docx");
36 }
37 }

Problems Javadoc Declaration Console Coverage
<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:24:54 pm - 9:24:55 pm)
Success
It took 113 milliseconds
```

[illegible]



### 3. Pdf file

RUNTIME 107ms

Used memory is bytes: 1310392

Used memory is megabytes: 1

Code:



```
14 Key secretKey = new SecretKeySpec(key.getBytes(), "AES");
15 Cipher cipher = Cipher.getInstance("AES");
16 cipher.init(cipherMode, secretKey);
17 FileInputStream inputStream = new FileInputStream(inputFile);
18 byte[]
19 inputBytes = new byte[(int) inputFile.length()];
20 inputStream.read(inputBytes);
21 byte[] outputBytes = cipher.doFinal(inputBytes);
22 FileOutputStream outputStream = new FileOutputStream(outputFile);
23 outputStream.write(outputBytes);
24 inputStream.close();
25 outputStream.close();
26 } catch (Exception e)
27 {
28     e.printStackTrace();
29 }
30
31 static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\CSE3002_INTERNET-AND-WEB-PROGRAMMING_ETH_1.1_47_CSE3002.pdf");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.pdf");
36 }
37 try {
38     encryptFile(inputFile, encryptedFile, key);
39 } catch (Exception e) {
40     e.printStackTrace();
41 }
```

Problems Javadoc Declaration Console Coverage

<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:13:12 pm - 9:13:14 pm)

Success

It took 107 milliseconds

Output:



#### 4. Text file

RUNTIME 107ms

Used memory is bytes: 1352608

Used memory is megabytes: 1

#### Code:

```
cryptog.java
13     try {
14         Key secretKey = new SecretKeySpec(key.getBytes(), "AES");
15         Cipher cipher = Cipher.getInstance("AES");
16         cipher.init(cipherMode, secretKey);
17         FileInputStream inputStream = new FileInputStream(inputFile);
18         byte[]
19         inputBytes = new byte[(int) inputFile.length()];
20         inputStream.read(inputBytes);
21         byte[] outputBytes = cipher.doFinal(inputBytes);
22         FileOutputStream outputStream = new FileOutputStream(outputFile);
23         outputStream.write(outputBytes);
24         inputStream.close();
25         outputStream.close();
26     } catch (Exception e)
27     {
28         e.printStackTrace();
29     }
30 }
31 public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\s\\online compiler code.txt");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.txt");
36     try {
37
38     }
39 }
```

Problems @ Javadoc Declaration Console Coverage

<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:13:12 pm – 9:13:14 pm)

Success

It took 107 milliseconds

#### Output:



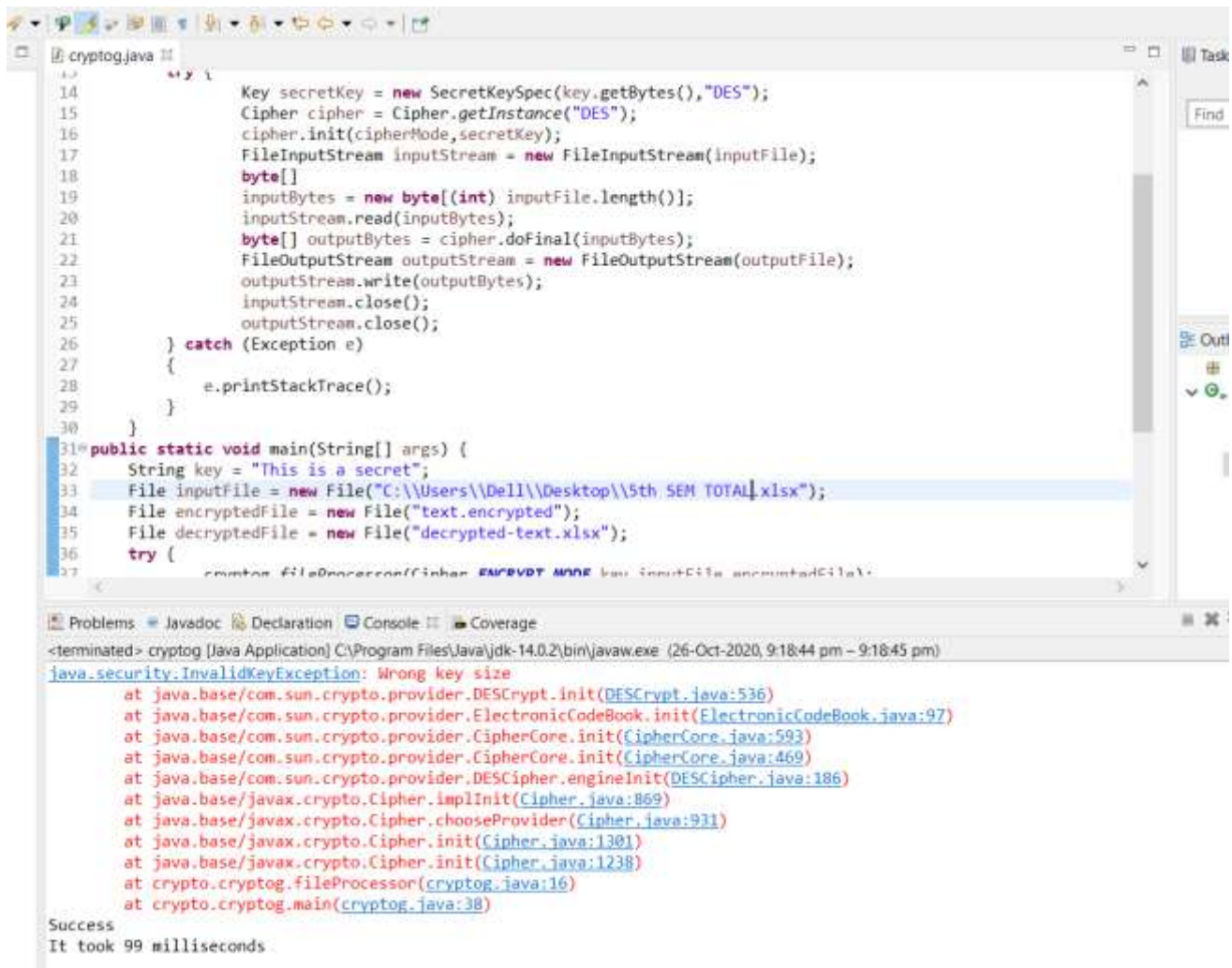
## 5.4.2 Statistics of the run time execution of DES:

### 1. XLSX file

Runtime 99ms

Used memory is bytes: 1895464

Used memory is megabytes: 3



```
14      Key secretKey = new SecretKeySpec(key.getBytes(), "DES");
15      Cipher cipher = Cipher.getInstance("DES");
16      cipher.init(cipherMode, secretKey);
17      FileInputStream inputStream = new FileInputStream(inputFile);
18      byte[]
19      inputBytes = new byte[(int) inputFile.length()];
20      inputStream.read(inputBytes);
21      byte[] outputBytes = cipher.doFinal(inputBytes);
22      FileOutputStream outputStream = new FileOutputStream(outputFile);
23      outputStream.write(outputBytes);
24      inputStream.close();
25      outputStream.close();
26  } catch (Exception e)
27  {
28      e.printStackTrace();
29  }
30  }
31  public static void main(String[] args) {
32      String key = "This is a secret";
33      File inputFile = new File("C:\\Users\\Dell\\Desktop\\5th SEM TOTAL.xlsx");
34      File encryptedFile = new File("text.encrypted");
35      File decryptedFile = new File("decrypted-text.xlsx");
36      try {
37          cryptoProcessor(cipher, key, inputFile, encryptedFile);
38      } catch (Exception e) {
39          e.printStackTrace();
40      }
41  }
```

Problems Javadoc Declaration Console Coverage

<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:18:44 pm - 9:18:45 pm)

java.security.InvalidKeyException: Wrong key size

at java.base/com.sun.crypto.provider.DESCrypt.init(DESCrypt.java:536)

at java.base/com.sun.crypto.provider.ElectronicCodeBook.init(ElectronicCodeBook.java:97)

at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:593)

at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:469)

at java.base/com.sun.crypto.provider.DESCipher.engineInit(DESCipher.java:186)

at java.base/com.sun.crypto.provider.CipherCore.implInit(CipherCore.java:869)

at java.base/com.sun.crypto.provider.CipherCore.chooseProvider(CipherCore.java:931)

at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:1301)

at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:1238)

at crypto.cryptog.fileProcessor(cryptog.java:16)

at crypto.cryptog.main(cryptog.java:38)

Success

It took 99 milliseconds.

Output:



## 2. Word file

Runtime 127ms

Used memory is bytes: 2337465

Used memory is megabytes: 3

### Code:

```
cryptog.java
12 {
13     try {
14         Key secretKey = new SecretKeySpec(key.getBytes(), "DES");
15         Cipher cipher = Cipher.getInstance("DES");
16         cipher.init(cipherMode, secretKey);
17         FileInputStream inputStream = new FileInputStream(inputFile);
18         byte[]
19         inputBytes = new byte[(int) inputFile.length()];
20         inputStream.read(inputBytes);
21         byte[] outputBytes = cipher.doFinal(inputBytes);
22         FileOutputStream outputStream = new FileOutputStream(outputFile);
23         outputStream.write(outputBytes);
24         inputStream.close();
25         outputStream.close();
26     } catch (Exception e)
27     {
28         e.printStackTrace();
29     }
30 }
31 public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\FFCS 7TH SEM.docx");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.docx");
36 }

Problems Javadoc Declaration Console Coverage
<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:27:02 pm – 9:27:02 pm)
java.security.InvalidKeyException: Wrong key size
    at java.base/com.sun.crypto.provider.DESCrypt.init(DESCrypt.java:536)
    at java.base/com.sun.crypto.provider.ElectronicCodeBook.init(ElectronicCodeBook.java:97)
    at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:593)
    at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:469)
    at java.base/com.sun.crypto.provider.DESCipher.engineInit(DESCipher.java:186)
    at java.base/javax.crypto.Cipher.implInit(Cipher.java:869)
    at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:931)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1301)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1238)
    at crypto.cryptog.fileProcessor(cryptog.java:16)
    at crypto.cryptog.main(cryptog.java:38)

Success
It took 127 milliseconds
```

Output:

```

[{"id": "1", "name": "John", "age": 30, "gender": "Male", "height": 1.8, "weight": 75, "eye_color": "Blue", "hair_color": "Brown", "skin_color": "Fair", "blood_type": "A+", "marital_status": "Single", "education": "Bachelor's", "occupation": "Software Engineer", "hobbies": ["Reading", "Golfing", "Traveling"], "pets": ["Dog", "Cat"], "allergies": ["Peanut Allergy"], "medical_history": ["Hypertension", "Asthma"], "current_medications": ["Lisinopril", "Albuterol"], "last_visit": "2023-10-26", "next_visit": "2023-11-23", "referral": "Primary Care", "insurance": "Blue Cross", "notes": "Patient is healthy, no significant changes noted. Continue with current treatment plan."}]
```



### 3. Pdf file

RUNTIME 201 ms

Used memory is bytes: 2336292

Used memory is megabytes: 3

Code:

```
cryptog.java
1  * crypto;
2  * java.io.File;
3  *
4  * class cryptog {
5  *     private static long
6  *     artTime = System.currentTimeMillis();
7  *     static void fileProcessor(int cipherMode ,String key,File inputFile,File outputFile)
8  *     {
9  *         try {
10 *             Key secretKey = new SecretKeySpec(key.getBytes(),"DES");
11 *             Cipher cipher = Cipher.getInstance("DES");
12 *             cipher.init(cipherMode,secretKey);
13 *             FileInputStream inputStream = new FileInputStream(inputFile);
14 *             byte[]
15 *             inputBytes = new byte[(int) inputFile.length()];
16 *             inputStream.read(inputBytes);
17 *             byte[] outputBytes = cipher.doFinal(inputBytes);
18 *             FileOutputStream outputStream = new FileOutputStream(outputFile);
19 *             outputStream.write(outputBytes);
20 *             inputStream.close();
21 *             outputStream.close();
22 *         } catch (Exception e)
23 *         {
24 *             e.printStackTrace();
25 *         }
26 *     }
27 * }
28 *
29 *
Problems Javadoc Declaration Console Coverage
terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 7:09:00 pm - 7:09:01 pm)
java.security.InvalidKeyException: Wrong key size
    at java.base/com.sun.crypto.provider.DESCrypt.init(DESCrypt.java:536)
    at java.base/com.sun.crypto.provider.ElectronicCodeBook.init(ElectronicCodeBook.java:97)
    at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:593)
    at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:469)
    at java.base/com.sun.crypto.provider.DESCipher.engineInit(DESCipher.java:186)
    at java.base/javax.crypto.Cipher.implInit(Cipher.java:869)
    at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:931)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1301)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1238)
    at crypto.cryptog.fileProcessor(cryptog.java:16)
    at crypto.cryptog.main(cryptog.java:38)
Success
It took 201 milliseconds
```

Encrypted file

1、編碼與解碼：編碼是指將明文轉換成密文的过程，而解碼則是將密文還原成明文的过程。在加密过程中，明文（即需要保護的信息）通過一個稱為“加密函數”的算法，與一個稱為“密鑰”的參數進行計算，生成密文。相反，在解碼過程中，密文通過一個稱為“解密函數”的算法，與相同的密鑰進行計算，還原成明文。2、加密函數：加密函數是實現加密過程的核心。它通常由一個算法和一個密鑰組成。算法定義了如何對明文進行數學運算，而密鑰則用於控制這些運算的具體方式。3、密鑰：密鑰是加密和解密過程中的關鍵參數。它是一個隨機生成的字符串，用於控制加密函數和解密函數的運算。密鑰的長度通常與明文的長度相同，或者是一個較小的固定長度。4、加密算法：加密算法是指用於實現加密函數的具體算法。常見的加密算法包括對稱加密（如DES、AES）和非對稱加密（如RSA、ECC）。對稱加密使用相同的密鑰進行加密和解密，而非對稱加密則使用一對公鑰和私鑰。5、解密函數：解密函數是實現解密過程的核心。它通常由一個算法和一個密鑰組成。算法定義了如何對密文進行數學運算，而密鑰則用於控制這些運算的具體方式。6、解密算法：解密算法是指用於實現解密函數的具體算法。常見的解密算法包括對稱解密（如DES、AES）和非對稱解密（如RSA、ECC）。對稱解密使用相同的密鑰進行解密，而非對稱解密則使用一對公鑰和私鑰。7、加密過程：加密過程是指將明文轉換成密文的整個過程。它包括選擇加密算法、生成密鑰、將明文輸入加密函數、計算密文等步驟。8、解密過程：解密過程是指將密文還原成明文的整個過程。它包括選擇解密算法、獲取密鑰、將密文輸入解密函數、計算明文等步驟。9、加密與解密的關係：加密和解密是互為逆過程的。加密過程將明文轉換成密文，而解密過程則將密文還原成明文。只有使用正確的密鑰，才能成功地解密密文。10、加密與解密的應用：加密和解密技術廣泛應用於各種場景，如數據傳輸、數據存儲、身份驗證等。在數據傳輸中，加密可以防止數據在傳輸過程中被竊聽或篡改。在數據存儲中，加密可以防止數據被未經授權的人員訪問。在身份驗證中，加密可以用於生成數字簽名，以證明數據的真實性和完整性。11、加密與解密的挑戰：加密和解密技術面臨著許多挑戰，如密鑰管理、算法安全、性能優化等。密鑰管理是加密技術中的一個重要問題，因為密鑰的丟失或洩露將導致數據的丟失或洩露。算法安全是加密技術中的另一個重要問題，因為如果加密算法被破解，那麼加密的數據將不再安全。性能優化是加密技術中的另一個重要問題，因為加密和解密過程通常會消耗大量的計算資源。12、加密與解密的未來：隨著技術的不斷發展，加密和解密技術將不斷進步。未來，我們將看到更多更強大、更安全、更高效的加密和解密算法的出現。同時，我們也將看到更多更完善的密鑰管理方案的出現。總之，加密和解密技術將繼續在保護數據安全方面發揮著至關重要的作用。

#### 4. Text file

RUNTIME 97ms

Used memory is bytes: 1352574

Used memory is megabytes: 3

#### Code:

```
cryptog.java
13     try {
14         Key secretKey = new SecretKeySpec(key.getBytes(), "DES");
15         Cipher cipher = Cipher.getInstance("DES");
16         cipher.init(cipherMode, secretKey);
17         FileInputStream inputStream = new FileInputStream(inputFile);
18         byte[]
19         inputBytes = new byte[(int) inputFile.length()];
20         inputStream.read(inputBytes);
21         byte[] outputBytes = cipher.doFinal(inputBytes);
22         FileOutputStream outputStream = new FileOutputStream(outputFile);
23         outputStream.write(outputBytes);
24         inputStream.close();
25         outputStream.close();
26     } catch (Exception e)
27     {
28         e.printStackTrace();
29     }
30 }
31 public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\s\\online compiler code.txt");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.txt");
36     try {
37
38     }
39 }

Problems  @ Javadoc  Declaration  Console  Coverage
<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:10:12 pm – 9:10:14 pm)
java.security.InvalidKeyException: Wrong key size
    at java.base/com.sun.crypto.provider.DESCrypt.init(DESCrypt.java:536)
    at java.base/com.sun.crypto.provider.ElectronicCodeBook.init(ElectronicCodeBook.java:97)
    at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:593)
    at java.base/com.sun.crypto.provider.CipherCore.init(CipherCore.java:469)
    at java.base/com.sun.crypto.provider.DESCipher.engineInit(DESCipher.java:186)
    at java.base/javax.crypto.Cipher.implInit(Cipher.java:869)
    at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:931)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1301)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1238)
    at crypto.cryptog.fileProcessor(cryptog.java:16)
    at crypto.cryptog.main(cryptog.java:38)

Success
It took 97 milliseconds
```





### 5.4.3 Statistics of the run time execution of RSA:

## 1. Excel file

RUNTIME 108ms

Used memory is bytes: 1352574

Used memory is megabytes: 3

**Code:**

```

12     {
13         try {
14             Key secretKey = new SecretKeySpec(key.getBytes(), "RSA");
15             Cipher cipher = Cipher.getInstance("RSA");
16             cipher.init(cipherMode, secretKey);
17             FileInputStream inputStream = new FileInputStream(inputFile);
18             byte[]
19             inputBytes = new byte[(int) inputFile.length()];
20             inputStream.read(inputBytes);
21             byte[] outputBytes = cipher.doFinal(inputBytes);
22             FileOutputStream outputStream = new FileOutputStream(outputFile);
23             outputStream.write(outputBytes);
24             inputStream.close();
25             outputStream.close();
26         } catch (Exception e)
27         {
28             e.printStackTrace();
29         }
30     }
31 public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\5th SEM TOTAL.xlsx");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.xlsx");

```

Problems Javadoc Declaration Console Coverage

<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:36:51 pm – 9:36:51 pm)

[java.security.InvalidKeyException](#): No installed provider supports this key: javax.crypto.spec.SecretKeySpec  
 at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:960)  
 at java.base/javax.crypto.Cipher.init(Cipher.java:1301)  
 at java.base/javax.crypto.Cipher.init(Cipher.java:1238)  
 at crypto.cryptog.fileProcessor(cryptog.java:16)  
 at crypto.cryptog.main(cryptog.java:37)

[java.security.InvalidKeyException](#): No installed provider supports this key: javax.crypto.spec.SecretKeySpec  
 at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:960)  
 at java.base/javax.crypto.Cipher.init(Cipher.java:1301)  
 at java.base/javax.crypto.Cipher.init(Cipher.java:1238)  
 at crypto.cryptog.fileProcessor(cryptog.java:16)  
 at crypto.cryptog.main(cryptog.java:38)

Success  
 It took 108 milliseconds

**Output:**

## 2. PDF file

RUNTIME 105ms

Used memory is bytes: 1352574

Used memory is megabytes: 3

Code:

```
13 try {
14     Key secretKey = new SecretKeySpec(key.getBytes(), "RSA");
15     Cipher cipher = Cipher.getInstance("RSA");
16     cipher.init(cipherMode, secretKey);
17     FileInputStream inputStream = new FileInputStream(inputFile);
18     byte[]
19     inputBytes = new byte[(int) inputFile.length()];
20     inputStream.read(inputBytes);
21     byte[] outputBytes = cipher.doFinal(inputBytes);
22     FileOutputStream outputStream = new FileOutputStream(outputFile);
23     outputStream.write(outputBytes);
24     inputStream.close();
25     outputStream.close();
26 } catch (Exception e) {
27
28     e.printStackTrace();
29
30
31 static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\CSE3002_INTERNET-AND-WEB-PROGRAMMING_ETH_1.1_47_CSE3002.pdf");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.pdf");
36
37 }
38
39 }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617

```

### 3. Text file

RUNTIME 98ms

Used memory is bytes: 1352574

Used memory is megabytes: 3

Code:

```
13 try {
14     Key secretKey = new SecretKeySpec(key.getBytes(), "RSA");
15     Cipher cipher = Cipher.getInstance("RSA");
16     cipher.init(cipherMode, secretKey);
17     FileInputStream inputStream = new FileInputStream(inputFile);
18     byte[]
19     inputBytes = new byte[(int) inputFile.length()];
20     inputStream.read(inputBytes);
21     byte[] outputBytes = cipher.doFinal(inputBytes);
22     FileOutputStream outputStream = new FileOutputStream(outputFile);
23     outputStream.write(outputBytes);
24     inputStream.close();
25     outputStream.close();
26 } catch (Exception e)
27 {
28     e.printStackTrace();
29 }
30 }
31 public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\s\\online compiler code.txt");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.txt");
36     try {
37         cryptog.fileProcessor(cipher, ENCRYPT, MODE, key, inputFile, encryptedFile);
38     }
39 }
```

Problems Javadoc Declaration Console Coverage

<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:40:15 pm – 9:40:16 pm)

java.security.InvalidKeyException: No installed provider supports this key: javax.crypto.spec.SecretKeySpec  
at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:960)  
at java.base/javax.crypto.Cipher.init(Cipher.java:1301)  
at java.base/javax.crypto.Cipher.init(Cipher.java:1238)  
at cryptog.cryptog.fileProcessor(cryptog.java:16)  
at cryptog.cryptog.main(cryptog.java:37)

java.security.InvalidKeyException: No installed provider supports this key: javax.crypto.spec.SecretKeySpec  
at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:960)  
at java.base/javax.crypto.Cipher.init(Cipher.java:1301)  
at java.base/javax.crypto.Cipher.init(Cipher.java:1238)  
at cryptog.cryptog.fileProcessor(cryptog.java:16)  
at cryptog.cryptog.main(cryptog.java:37)

Success  
It took 98 milliseconds

Output:





#### 4. Word file

RUNTIME 116ms

Used memory is bytes: 1352574

Used memory is megabytes: 3

#### Code:

```
cryptog.java
12 {
13     try {
14         Key secretKey = new SecretKeySpec(key.getBytes(), "RSA");
15         Cipher cipher = Cipher.getInstance("RSA");
16         cipher.init(cipherMode, secretKey);
17         FileInputStream inputStream = new FileInputStream(inputFile);
18         byte[]
19         inputBytes = new byte[(int) inputFile.length()];
20         inputStream.read(inputBytes);
21         byte[] outputBytes = cipher.doFinal(inputBytes);
22         FileOutputStream outputStream = new FileOutputStream(outputFile);
23         outputStream.write(outputBytes);
24         inputStream.close();
25         outputStream.close();
26     } catch (Exception e)
27     {
28         e.printStackTrace();
29     }
30 }
31 public static void main(String[] args) {
32     String key = "This is a secret";
33     File inputFile = new File("C:\\Users\\Dell\\Desktop\\FFCS 7TH SEM.docx");
34     File encryptedFile = new File("text.encrypted");
35     File decryptedFile = new File("decrypted-text.docx");
36 }

Problems @ Javadoc Declaration Console Coverage
<terminated> cryptog [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (26-Oct-2020, 9:34:11 pm – 9:34:12 pm)
java.security.InvalidKeyException: No installed provider supports this key: javax.crypto.spec.SecretKeySpec
    at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:960)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1301)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1238)
    at crypto.cryptog.fileProcessor(cryptog.java:16)
    at crypto.cryptog.main(cryptog.java:37)
java.security.InvalidKeyException: No installed provider supports this key: javax.crypto.spec.SecretKeySpec
    at java.base/javax.crypto.Cipher.chooseProvider(Cipher.java:960)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1301)
    at java.base/javax.crypto.Cipher.init(Cipher.java:1238)
    at crypto.cryptog.fileProcessor(cryptog.java:16)
    at crypto.cryptog.main(cryptog.java:38)
Success
It took 116 milliseconds
```

[illegible]

## **6. Conclusion and Future Work**

There are different cryptography strategies and sub-procedures. A few systems perform incomplete encryption while others perform full encryption. A portion of the strategies complete image compressions however others don't. Contingent upon the kind of application, speed, bandwidth, privacy, security and legitimacy degree, one may choose a specific sort of encryption strategy. Every single strategy has its own benefits and faults. One must think in choosing an appropriate figure since now cryptanalysis methods investigate is under core interest. Once a figure is created, one must do different security investigations specified in the project. There is enormous potential in this field for future research and arrangement. With a probability of growing such encryption plans for movement remuneration and execution on installed gadget models.

Our future work will center on looking at and breaking down existing cryptographic algorithmic blocks of encryption techniques like AES, DES and RSA. It will incorporate investigations on image and sound information and focus will be to enhance encryption time and decryption time.

## 7. References

- [1] Borka Jerman-Blazic, Tomaz Klobucar,. Advanced Communications and Multimedia Security. New York: Springer Science+Business Media New York. 2002
- [2]Ching-Yung Lin,. Topics in Signal Processing -- Multimedia Security Systems. 2006
- [3]Emanuil Rednic; Andrei Toma, n.d. Software Analysis. SECURITY MANAGEMENT IN A MULTIMEDIA SYSTEM, 4(2), pp. 237-247.
- [4] Saha Arunabh n.d. Overview of Multimedia Security,
- [5] Amit Pande; Prasant Mohapatra; Joseph Zambreno, n.d. Securing Multimedia Content using Joint Compression and Encryption,
- [6] Sonal Guleria, Sonia Vatta, TO ENHANCE MULTIMEDIA SECURITY IN CLOUD COMPUTING ENVIRONMENT USING CROSSBREED ALGORITHM, 2(6), pp. 562-568. 2013.
- [7] A.A.Zaidan, B.B.Zaidan, Anas Majeed, "High Securing Cover-File of Hidden Data Using Statistical Technique and AES Encryption Algorithm", World Academy of Science Engineering and Technology (WASET), Vol.54, ISSN: 2070-3724, P.P 468-479.
- [8] A.A.Zaidan, B.B.Zaidan, "Novel Approach for High Secure Data Hidden in MPEG Video Using Public Key Infrastructure", International Journal of Computer and Network Security, Vol.1, No.1, ISSN: 1985-1553, P.P 71-76, 2009.



- [9] A.W.Naji, A.A.Zaidan, B.B.Zaidan, Shihab A, Othman O. Khalifa, “ Novel Approach of Hidden Data in the (Unused Area 2 within EXE File) Using Computation Between Cryptography and Steganography ”, International Journal of Computer Science and Network Security (IJCSNS) , Vol.9, No.5 , ISSN : 1738- 7906, pp. 294-300.
- [10] Anas Majed Hamid, Miss Laiha Mat Kiah, Hayan .T. Madhloom, B.B Zaidan, A.A Zaidan,” Novel Approach for High Secure and High Rate Data Hidden in the Image Using Image Texture Analysis”, International Journal of Engineering and Technology (IJET) , Published by: Engg Journals Publications, ISSN:0975-4042, Vol.1,NO.2,P .P 63-69.