

Securing Credit Card Transaction

A PROJECT REPORT

by

Name: Ayush Madurwar

Registration number: 20MIA1009

Name: Aman Kumar

Registration number: 20MIA1144

A project report submitted to

DR. Anita X

SCOPE

In partial fulfilment of the requirements for the course of

CSE1029- Network Security and Cryptography



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

VELLORE INSTITUTE OF TECHNOLOGY

CHENNAI – 600127

November 2022

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1	ACKNOWLEDGEMENT	3
2	ABSTRACT	4
3	INTRODUCTION	5-6
4	LITERATURE REVIEW	7-9
5	PROPOSED METHODOLOGY	10-11
6	IMPLEMENTATION	12-25
7	CONCLUSION	26
8	REFERENCES	27

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide,

Dr. ANITA X Professor, School of Computer Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution

AYUSH

AMAN

ABSTRACT

The differences in how payment gateways work generate security difficulties with online card payments. The system fosters a dispersed speculative attack. This attack subverts payment practicality from its planned motive behind card intricacies, so attackers can access all security knowledge fields needed to make online exchanges.

The distributed guessing attack is possible due to heterogeneous payment site security. The security of online card payments can be improved by redesigning it. The mechanism counters the distributed guessing attack by implementing a newly devised algorithm that supports a unified security check at the payment gateway and reduces the system's inability to manage numerous erroneous request. Being the age of supercomputers, transactional knowledge should be encrypted using a high-quality method while breaking down encrypted data.

Keywords:

AES, Decryption, Encryption, Validation.

INTRODUCTION

Cards are the most often used payment option for internet purchases.

However, as the estimation of online deals has grown, so has the measure of online deception. For example, In financial year 2021, over 35 billion digital transactions worth over 60 trillion Indian rupees of transactions across India.

The number of transaction was estimated to rise up to over 214 billion in financial year 2026 in the country. Present the online instalment scene in depth right now. Specifically, we want to highlight the varied behaviours in which online instalment is conducted, as well as the altering security efforts set up by online merchants – investigate Just the card number and expiration date, to fully developed integrated bank roll systems , such as 3D Secure.

The number of enquiries about which we may wish to speak: Is there a security risk as a result of the distinction? If it does, how widespread is the problem and how easily exploitable is it? How much harm should be possible?

Furthermore, how might it be resolved later on? To determine the severity of the problem, we examine the 'online instalment scene,' mapping diverse shipper instalment usage.

The vulnerabilities are now portrayed to cards that don't provide combined checks across exchanges from distinct places. Our investigations were conducted using Visa and MasterCard. While MasterCard's unified system detects the speculative attack after less than ten attempts (at any rate, when those attempts are dispersed over multiple sites), Visa's payment environment does not. Because Visa is the most widely used payment system on the planet, the discovered accountability has a significant impact on the entire global online payment architecture.

Card issuers use a variety of methods to encrypt credit cards. The magnetic strip on the back of a card is typically encrypted and can only be read by a card scanner. Relying solely on the magnetic strip is a less secure method than requiring the use of a PIN-and-chip, because a PIN makes it more difficult for stolen credit cards to be authorized and used. A smart card with an electronic chip may be harder for thieves and hackers to steal information from, compared with other forms of encryption and security put in place to protect credit account information.

For transactions that do not require a card to be scanned, such as an online transaction or in-app purchase, websites require both the credit card number on the front of the card and a CVV number located on the back of the card to be used. Using a CVV prevents an individual from being able to use only a stolen credit card number to conduct a transaction.

- Credit card encryption is a security measure used to reduce the likelihood of credit or debit card information being stolen.
- It makes it impossible to access the credit card information without the corresponding encryption key that lets the merchant and financial institution conduct their transactions.
- Card issuers use many methods to encrypt credit cards including magnetic strips, PIN numbers, electronic chips and a CVV in case of online transactions.

Format-Preserving Encryption

- The name implies, the goal of a Format-Preserving Encryption scheme is to securely encrypt while preserving the original formatting of the plaintext data. That means, in a nutshell, that our 16-digit card number can encrypt to a 16-digit number, without any problem.

LITERATURE REVIEW

The vulnerabilities are now portrayed to cards that don't provide combined checks across exchanges from distinct places. Our investigations were conducted using Visa and MasterCard. While MasterCard's unified system detects the speculative attack after less than ten attempts (at any rate, when those attempts are dispersed over multiple sites), Visa's payment environment does not. Because Visa is the most widely used payment system on the planet, the discovered accountability has a significant impact on the entire global online payment architecture.

3.1 Client/CardHolder:

The stated speculating assault depicted right now shipper sites and card instalment framework to collect all the card peculiarities, there isn't much a cardholder can do to avoid it. Simultaneously, the cardholder is substantially impacted by the attack: cash may be lost, cards may need to be blocked, and the result is an exercise in futility and exertion, as well as a weakened belief that all is great and good.

3.2 Online Merchant:

A shopkeeper can't do anything on their own to prevent widespread speculative attacks. To prevent the speculative attack from being coordinated, all shippers would have to agree or be forced to use a similar amount of fields.

Simultaneously, a vendor can avoid being used in the assault by only using cards that use an instalment arrange that isn't vulnerable to the assault, or by utilising 3D Secure advances prescribed by the instalment card organisations

for example, the American Express 'Safe Key,' 'Checked by Visa,' and MasterCard 'Secure Code.'

3.3 Payment Gateway:

It is unrealistic to expect that the full the doors should be able to be properly arranged. Avoid the appropriated speculative attack. Everything is Having said that, instalment entryways can provide propelled attractions for their dealers, and these highlights should include Any event makes it increasingly difficult to abuse a site for the purposes of attack. Entryways, in particular, may make use of IP address speed. mechanisms that are used to distinguish rehashed incorrect data attempts performed during a particular time period from the same IP address. However, because there is no coordination across numerous These speed channels, like portals, may surely be avoided. By switching to a site that uses a different instalment entryway.

3.4 Card Payment Network:

The clearest protection against the mentioned speculative attack would be at the level of the card instalment framework. In any way, we don't know whether instalment arrange suppliers could change their system foundation to recognise instalment needs from several, globally dispersed instalment portals, searching for suspicious activities on a solitary card dispersed over various shipper sites.

3.5 Card Issuing Banks:

The bank may become the most essential factor in the final stage of the payment method, to enable the exchange of assets, but it will not be involved with every individual presumption (unless 3D Secure is used). Banks play an

important role in limiting the harm that could occur if attackers obtain card data. Many lending institutions are currently running astute misrepresentation identification frameworks that spot transactions that are outside their clients' normal money-management procedures. The giving bank then has the option of blocking the payment, approaching the client for confirmation, or accepting the payment in the face of a determined challenge that such exchange may be viewed as untrue later. So, in light of the aforementioned challenges, three adjustments to the online card payment system were developed in this study.

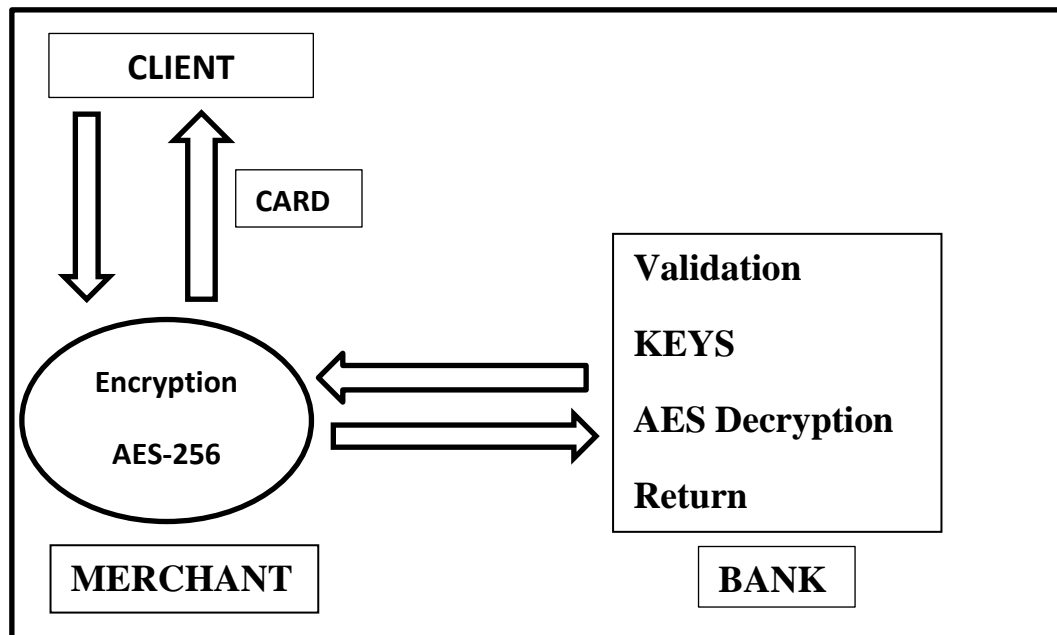
1. Creating an uniform architecture to combat scattered guessing attacks.
2. Limiting the number of invalid card data entered in a single transaction.
3. Encrypting the communication and storing credit/debit card information in encrypted form in the database.

PROPOSED METHODOLOGY

ARCHITECTURE:

We see that different sites have different input fields in their payment processing page like some have Card no., Exp, CVV; and some have an additional field as card holder name; some sites don't have cvv, instead they have 3D secure pin. This difference in the input fields and number of fields makes it easy for the hacker to steal the data. Some sites fixed their checkout framework by including a location confirmation field. Be that as it may, this is certifiably not a smart thought since it won't give extra security, yet rather opens up another road for speculating. So it is recommended for all the merchant sites to use the same architecture for the payment process. Using the unified architecture can

1. Eradicate distributed guessing attack to most extent,
2. Can be implemented with the latest security algorithms,
3. Can be easily updated later in the future.



Step 1(Validation): The user will enter credit card details on the merchant site. On submission the data is sent to the merchant server and check whether the card number is valid or not.

Step 2: In this step will take password/private key for decryption.

Step 3: AES key and AES Initialization Vector for 256 bit AES encryption are generated.

Step 4: The above generated AES key and AES are used to encrypt the data acquired from the merchant site with AES 256 bit.

Step 5: After the encryption, merchant server sends a request to the bank server for password which is required to encrypt the file

Step 6: AES key and AES encrypted card details are sent to the bank server .

Step 7: After the data is received at the bank server, AES key and AES IV are decrypted using private key/password given in **STEP 2**.

Implementation

Importing libraries:

```
from Crypto import Random
from Crypto.Cipher import AES
import os
import os.path
from os import listdir
from os.path import isfile, join
import time
```

```
from Crypto import Random
from Crypto.Cipher import AES
import os
import os.path
from os import listdir
from os.path import isfile, join
import time
```

Validation of Credit Card number:

```
##Check the Credit card number is valid or not

class CreditCard:
    def main(args):
        number = int(input("Enter Credit Card Number: "))
        if (CreditCard.isValid(number)):
            f = open("Credit Card Number.txt", "w+")
            f.write(str(number))
            f.close()

            print(str(number) + " Credit Card number is " + ("valid" if CreditCard.isValid(number) else "invalid"))

        def isValid(number):
            return (CreditCard.getSize(number) >= 13 and CreditCard.getSize(number) <= 16) and (CreditCard.prefixMatched(number, 4) or CreditCard.prefixMatched(number, 5)
            or CreditCard.prefixMatched(number, 37) or CreditCard.prefixMatched(number, 6)) and ((CreditCard.sumOfDoubleEvenPlace(number) + CreditCard.sumOfOddPlace(number)) % 10 == 0)

        def sumOfDoubleEvenPlace(number):
            sum = 0
            num = str(number) + ""
            i = CreditCard.getSize(number) - 2
            while (i >= 0):
                sum += CreditCard.getDigit(int(str(num[i]) * 2))
                i -= 2
            return sum

        def getDigit(number):
            if (number < 9):
                return number
            return int(number / 10) + number % 10
```



```
def sumOfOddPlace(number):
    sum = 0
    num = str(number) + ""
    i = CreditCard.getSize(number) - 1
    while (i >= 0):
        sum += int(str(num[i]) + "")
        i -= 2
    return sum

def prefixMatched(number, d):
    return CreditCard.getPrefix(number, CreditCard.getSize(d)) == d

def getSize(d):
    num = str(d) + ""
    return len(num)

def getPrefix(number, k):
    if (CreditCard.getSize(number) > k):
        num = str(number) + ""
        return int(num[0:k])
    return number

if __name__ == "__main__":
    CreditCard.main([])
```

Result Output:

```
Enter Credit Card Number: 379354508162306
379354508162306 Credit Card number is valid
```

+ Code + Text

3m Enter Credit Card Number: 379354508162306
379354508162306 Credit Card number is valid

RAM Disk

Editing ^

Credit Card Number.txt X ...

1 379354508162306

Code:

```
##Check the Credit card number is valid or not

class CreditCard:
    def main(args):
        number = int(input("Enter Credit Card Number: "))
        if (CreditCard.isValid(number)):

            f = open("Credit Card Number.txt", "w+")
            f.write(str(number))
            f.close()
```

```

    print(str(number) + " Credit Card number is " + ("valid" if CreditCard
.isValid(number) else "invalid"))

def isValid(number):
    return (CreditCard.getSize(number) >= 13 and CreditCard.getSize(number
) <= 16) and (CreditCard.prefixMatched(number, 4) or CreditCard.prefixMatc
hed(number, 5) or CreditCard.prefixMatched(number, 37) or CreditCard.prefi
xMatched(number, 6)) and ((CreditCard.sumOfDoubleEvenPlace(number) + Credi
tCard.sumOfOddPlace(number)) % 10 == 0)

def sumOfDoubleEvenPlace(number):
    sum = 0
    num = str(number) + ""
    i = CreditCard.getSize(number) - 2
    while (i >= 0):
        sum += CreditCard.getDigit(int(str(num[i]) + "")) * 2
        i -= 2
    return sum

def getDigit(number):
    if (number < 9):
        return number
    return int(number / 10) + number % 10
def sumOfOddPlace(number):
    sum = 0
    num = str(number) + ""
    i = CreditCard.getSize(number) - 1
    while (i >= 0):
        sum += int(str(num[i]) + "")
        i -= 2
    return sum

def prefixMatched(number, d):
    return CreditCard.getPrefix(number, CreditCard.getSize(d)) == d
def getSize(d):
    num = str(d) + ""
    return len(num)

def getPrefix(number, k):
    if (CreditCard.getSize(number) > k):
        num = str(number) + ""
        return int(num[0:k])
    return number
if __name__ == "__main__":
    CreditCard.main([])

```

Password for Decryption:


```
[ ] class Encryptor:
    def __init__(self, key):
        self.key = key

    def pad(self, s):
        return s + b"\0" * (AES.block_size - len(s) % AES.block_size)

    def encrypt(self, message, key, key_size=256):
        message = self.pad(message)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(key, AES.MODE_CBC, iv)
        return iv + cipher.encrypt(message)

    def encrypt_file(self, file_name):
        with open(file_name, 'rb') as fo:
            plaintext = fo.read()
            enc = self.encrypt(plaintext, self.key)
            with open(file_name + ".enc", 'wb') as fo:
                fo.write(enc)
            os.remove(file_name)

    def decrypt(self, ciphertext, key):
        iv = ciphertext[:AES.block_size]
        cipher = AES.new(key, AES.MODE_CBC, iv)
        plaintext = cipher.decrypt(ciphertext[AES.block_size:])
        return plaintext.rstrip(b"\0")
```

```
 def decrypt_file(self, file_name):
    with open(file_name, 'rb') as fo:
        ciphertext = fo.read()
    dec = self.decrypt(ciphertext, self.key)
    with open(file_name[:-4], 'wb') as fo:
        fo.write(dec)
    os.remove(file_name)

def getAllFiles(self):
    dir_path = os.path.dirname(os.path.realpath(__file__))
    dirs = []
    for dirName, subdirList, fileList in os.walk(dir_path):
        for fname in fileList:
            if (fname != 'script.py' and fname != 'data.txt.enc'):
                dirs.append(dirName + "\\\" + fname)
    return dirs

def encrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.encrypt_file(file_name)
```

```

[ ] def decrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.decrypt_file(file_name)

key = b'[EX\xc8\xd5\xbfI{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02j\xdf\xcb\xc4\x94\x9d(\x9e'
enc = Encryptor(key)
clear = lambda: os.system('cls')

if os.path.isfile('data.txt.enc'):
    while True:
        password = str(input("Enter password: "))
        enc.decrypt_file("data.txt.enc")
        p = ''
        with open("data.txt", "r") as f:
            p = f.readlines()
        if p[0] == password:
            enc.encrypt_file("data.txt")
            break

while True:
    clear()
    choice = int(input("1. Press '1' to encrypt file.\n2. Press '2' to decrypt file.\n3. Press '3' to Encrypt all files in the directory.\n4. Press '4' to setup decryption password.\n5. Press '5' to exit.\n"))
    clear()
    if choice == 1:
        enc.encrypt_file(str(input("Enter name of file to encrypt: ")))
    elif choice == 2:
        enc.decrypt_file(str(input("Enter name of file to decrypt: ")))
    elif choice == 3:
        enc.encrypt_all_files()
    elif choice == 4:
        enc.decrypt_all_files()
    elif choice == 5:
        exit()
    else:
        print("Please select a valid option!")

else:
    while True:
        clear()
        password = str(input("Setting up stuff. Enter a password that will be used for decryption: "))
        repassword = str(input("Confirm password: "))
        if password == repassword:
            break
        else:
            print("Passwords Mismatched!")
    f = open("data.txt", "w+")
    f.write(password)
    f.close()
    enc.encrypt_file("data.txt")
    print("Please restart the program to complete the setup")
    time.sleep(15)

```

Result Output:

```

Setting up stuff. Enter a password that will be used for decryption: Ayush
Confirm password: Ayush

```


Code:

```
class Encryptor:
    def __init__(self, key):
        self.key = key

    def pad(self, s):
        return s + b"\0" * (AES.block_size - len(s) % AES.block_size)

    def encrypt(self, message, key, key_size=256):
        message = self.pad(message)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(key, AES.MODE_CBC, iv)
        return iv + cipher.encrypt(message)

    def encrypt_file(self, file_name):
        with open(file_name, 'rb') as fo:
            plaintext = fo.read()
        enc = self.encrypt(plaintext, self.key)
        with open(file_name + ".enc", 'wb') as fo:
            fo.write(enc)
        os.remove(file_name)

    def decrypt(self, ciphertext, key):
        iv = ciphertext[:AES.block_size]
        cipher = AES.new(key, AES.MODE_CBC, iv)
        plaintext = cipher.decrypt(ciphertext[AES.block_size:])
        return plaintext.rstrip(b"\0")

    def decrypt_file(self, file_name):
        with open(file_name, 'rb') as fo:
            ciphertext = fo.read()
        dec = self.decrypt(ciphertext, self.key)
        with open(file_name[:-4], 'wb') as fo:
            fo.write(dec)
        os.remove(file_name)

    def getAllFiles(self):
        dir_path = os.path.dirname(os.path.realpath(__file__))
        dirs = []
        for dirName, subdirList, fileList in os.walk(dir_path):
            for fname in fileList:
                if (fname != 'script.py' and fname != 'data.txt.enc'):
                    dirs.append(dirName + "\\\" + fname)
        return dirs
```

```

def encrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.encrypt_file(file_name)

def decrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.decrypt_file(file_name)

key = b'[EX\xc8\xd5\xbfI{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02)j\xdf\xcb\xc4\x94\x9d(\x9e'
enc = Encryptor(key)
clear = lambda: os.system('cls')

if os.path.isfile('data.txt.enc'):
    while True:
        password = str(input("Enter password: "))
        enc.decrypt_file("data.txt.enc")
        p = ''
        with open("data.txt", "r") as f:
            p = f.readlines()
        if p[0] == password:
            enc.encrypt_file("data.txt")
            break

    while True:
        clear()
        choice = int(input("1. Press '1' to encrypt file.\n2. Press '2' to
        decrypt file.\n3. Press '3' to Encrypt all files in the directory.\n4. Pr
        ess '4' to decrypt all files in the directory.\n5. Press '5' to exit.\n"))
        clear()
        if choice == 1:
            enc.encrypt_file(str(input("Enter name of file to encrypt: "))
)
        elif choice == 2:
            enc.decrypt_file(str(input("Enter name of file to decrypt: "))
)
        elif choice == 3:
            enc.encrypt_all_files()
        elif choice == 4:
            enc.decrypt_all_files()
        elif choice == 5:

```

```

        exit()
    else:
        print("Please select a valid option!")

else:
    while True:
        clear()
        password = str(input("Setting up stuff. Enter a password that will
be used for decryption: "))
        repassword = str(input("Confirm password: "))
        if password == repassword:
            break
        else:
            print("Passwords Mismatched!")
    f = open("data.txt", "w+")
    f.write(password)
    f.close()
    enc.encrypt_file("data.txt")
    print("Please restart the program to complete the setup")
    time.sleep(15)

```

Encryption of Credit Card Number file:

It will encrypt the file first then go back to password code one for password then it will take the password and check the password if it is true then it will give options to choose like

1. Press '1' to encrypt file.
2. Press '2' to decrypt file.
3. Press '3' to Encrypt all files in the directory.
4. Press '4' to decrypt all files in the directory.
5. Press '5' to exit.

```

▶ class Encryptor:
    def __init__(self, key):
        self.key = key

    def pad(self, s):
        return s + b"\0" * (AES.block_size - len(s) % AES.block_size)

    def encrypt(self, message, key, key_size=256):
        message = self.pad(message)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(key, AES.MODE_CBC, iv)
        return iv + cipher.encrypt(message)

    def encrypt_file(self, file_name):
        with open(file_name, 'rb') as fo:
            plaintext = fo.read()
        enc = self.encrypt(plaintext, self.key)
        with open(file_name + ".enc", 'wb') as fo:
            fo.write(enc)
        os.remove(file_name)

```

```

▶ def decrypt(self, ciphertext, key):
    iv = ciphertext[:AES.block_size]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = cipher.decrypt(ciphertext[AES.block_size:])
    return plaintext.rstrip(b"\0")

def decrypt_file(self, file_name):
    with open(file_name, 'rb') as fo:
        ciphertext = fo.read()
    dec = self.decrypt(ciphertext, self.key)
    with open(file_name[:-4], 'wb') as fo:
        fo.write(dec)
    os.remove(file_name)

def getAllFiles(self):
    dir_path = os.path.dirname(os.path.realpath(__file__))
    dirs = []
    for dirName, subdirList, fileList in os.walk(dir_path):
        for fname in fileList:
            if (fname != 'script.py' and fname != 'data.txt.enc'):
                dirs.append(dirName + "\\\" + fname)
    return dirs

```

```

def encrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.encrypt_file(file_name)

def decrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.decrypt_file(file_name)

key = b'[EX\xc8\xd5\xbfI{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02]j\xdf\xcb\xc4\x94\x9d(\x9e'
enc = Encryptor(key)
clear = lambda: os.system('cls')

if os.path.isfile('data.txt.enc'):
    while True:
        password = str(input("Enter password: "))
        enc.decrypt_file("data.txt.enc")
        p = ''
        with open("data.txt", "r") as f:
            p = f.readlines()
        if p[0] == password:
            enc.encrypt_file("data.txt")
            break

```

```

while True:
    clear()
    choice = int(input(
        "1. Press '1' to encrypt file.\n2. Press '2' to decrypt file.\n3. Press '3' to Encrypt all files in the directory.
    ))
    clear()
    if choice == 1:
        enc.encrypt_file(str(input("Enter name of file to encrypt: ")))
    elif choice == 2:
        enc.decrypt_file(str(input("Enter name of file to decrypt: ")))
    elif choice == 3:
        enc.encrypt_all_files()
    elif choice == 4:
        enc.decrypt_all_files()
    elif choice == 5:
        exit()
    else:
        print("Please select a valid option!")

```

```

else:
    while True:
        clear()
        password = str(input("Setting up stuff. Enter a password that will be used for decryption: "))
        repassword = str(input("Confirm password: "))
        if password == repassword:
            break
        else:
            print("Passwords Mismatched!")
    f = open("data.txt", "w+")
    f.write(password)
    f.close()
    enc.encrypt_file("data.txt")
    print("Please restart the program to complete the setup")
    time.sleep(15)

```

Code:

```
class Encryptor:
    def __init__(self, key):
        self.key = key

    def pad(self, s):
        return s + b"\0" * (AES.block_size - len(s) % AES.block_size)

    def encrypt(self, message, key, key_size=256):
        message = self.pad(message)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(key, AES.MODE_CBC, iv)
        return iv + cipher.encrypt(message)

    def encrypt_file(self, file_name):
        with open(file_name, 'rb') as fo:
            plaintext = fo.read()
        enc = self.encrypt(plaintext, self.key)
        with open(file_name + ".enc", 'wb') as fo:
            fo.write(enc)
        os.remove(file_name)

    def decrypt(self, ciphertext, key):
        iv = ciphertext[:AES.block_size]
        cipher = AES.new(key, AES.MODE_CBC, iv)
        plaintext = cipher.decrypt(ciphertext[AES.block_size:])
        return plaintext.rstrip(b"\0")

    def decrypt_file(self, file_name):
        with open(file_name, 'rb') as fo:
            ciphertext = fo.read()
        dec = self.decrypt(ciphertext, self.key)
        with open(file_name[:-4], 'wb') as fo:
            fo.write(dec)
        os.remove(file_name)

    def getAllFiles(self):
        dir_path = os.path.dirname(os.path.realpath(__file__))
        dirs = []
        for dirName, subdirList, fileList in os.walk(dir_path):
            for fname in fileList:
                if (fname != 'script.py' and fname != 'data.txt.enc'):
                    dirs.append(dirName + "\\\" + fname)
        return dirs
```

```

def encrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.encrypt_file(file_name)

def decrypt_all_files(self):
    dirs = self.getAllFiles()
    for file_name in dirs:
        self.decrypt_file(file_name)

key = b'[EX\xc8\xd5\xbfI{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02)j\xdf\xcb\xc4\x94\x9d(\x9e'
enc = Encryptor(key)
clear = lambda: os.system('cls')

if os.path.isfile('data.txt.enc'):
    while True:
        password = str(input("Enter password: "))
        enc.decrypt_file("data.txt.enc")
        p = ''
        with open("data.txt", "r") as f:
            p = f.readlines()
        if p[0] == password:
            enc.encrypt_file("data.txt")
            break

    while True:
        clear()
        choice = int(input(
            "1. Press '1' to encrypt file.\n2. Press '2' to decrypt file.\n3. Press '3' to Encrypt all files in the directory.\n4. Press '4' to decrypt all files in the directory.\n5. Press '5' to exit.\n"))
        clear()
        if choice == 1:
            enc.encrypt_file(str(input("Enter name of file to encrypt: ")))
        )
        elif choice == 2:
            enc.decrypt_file(str(input("Enter name of file to decrypt: ")))
        )
        elif choice == 3:
            enc.encrypt_all_files()
        elif choice == 4:
            enc.decrypt_all_files()

```

```

        elif choice == 5:
            exit()
        else:
            print("Please select a valid option!")

else:
    while True:
        clear()
        password = str(input("Setting up stuff. Enter a password that will
be used for decryption: "))
        repassword = str(input("Confirm password: "))
        if password == repassword:
            break
        else:
            print("Passwords Mismatched!")
    f = open("data.txt", "w+")
    f.write(password)
    f.close()
    enc.encrypt_file("data.txt")
    print("Please restart the program to complete the setup")
    time.sleep(15)

```

Result And output:

... Enter password:

... Enter password: Ayush

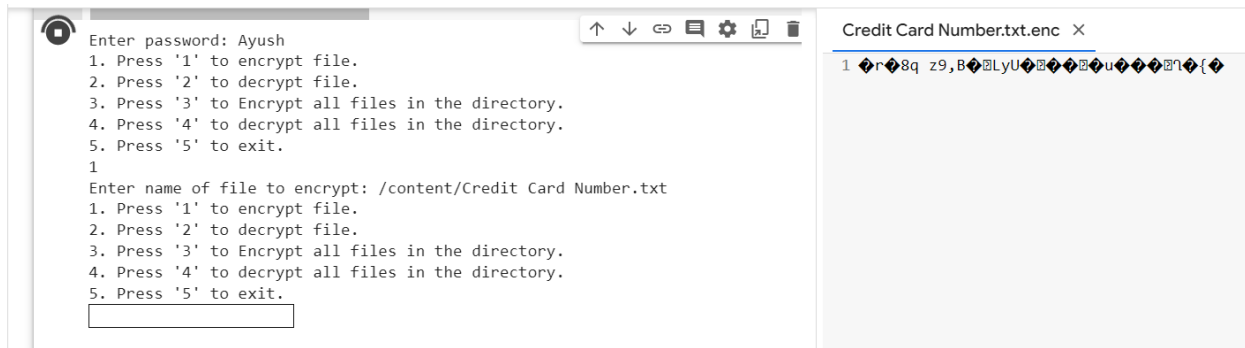
1. Press '1' to encrypt file.
2. Press '2' to decrypt file.
3. Press '3' to Encrypt all files in the directory.
4. Press '4' to decrypt all files in the directory.
5. Press '5' to exit.

For Encryption:

- ... Enter password: Ayush
- 1. Press '1' to encrypt file.
- 2. Press '2' to decrypt file.
- 3. Press '3' to Encrypt all files in the directory.
- 4. Press '4' to decrypt all files in the directory.
- 5. Press '5' to exit.

1

Enter name of file to encrypt: tent/Credit Card Number.txt

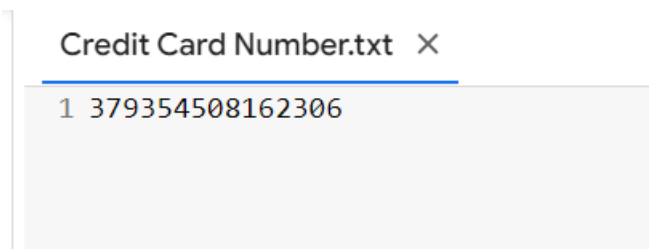


For Decryption:

- Enter name of file to encrypt: /content/Credit Card Number.txt
- 1. Press '1' to encrypt file.
- 2. Press '2' to decrypt file.
- 3. Press '3' to Encrypt all files in the directory.
- 4. Press '4' to decrypt all files in the directory.
- 5. Press '5' to exit.

2

Enter name of file to decrypt: Credit Card Number.txt.enc



Conclusion and Future works

The severity of the key drawbacks of online payment systems, namely Distributed Guessing Attack and Data Breach, has been decreased in this study. We have successfully eliminated the major weaknesses that render databases vulnerable to attack in an online or digital payment system. We may use this architecture not only in payment systems, but also in other situations where privacy and data security are more important CRM (Customer Relationship Management) and secure government operations.

In actuality, nothing is 100% secure in the internet age, therefore a hacker may still breach the database containing this architecture, however the unique part about this architecture is that it takes 1.9×10^{33} years (about 1 billion billion years) using today's supercomputers. However, as hackers create new strategies for breaking security algorithms, there may be a methodology in the future that breaks this architecture faster than it takes to breach currently. However, because every merchant would be using this design, it may be simply changed if new encrypting algorithms that are stronger and better than these methods emerge.

References

1. A. Aruna, Devansh Sharma, Manikanta Elluru, Subha Sarkar, 'Securing Online Transactions with Cryptography And Secured Authentication Methods', IJRTE, Volume-8, Issue-1, 2019, pp: 1424-1427.
2. Irma T. Plata, Edward B. Panganiban and Bryan B. Bartolome, 2019, 'A Security Approach for File Management System using Data Encryption Standard (DES) algorithm', IJATCSE-Volume 8 No. 5, pp:2042 – 2048.
3. <https://github.com/giorgos314/fpecredit>
4. <https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard>
5. <https://blog.cryptographyengineering.com/2011/11/10/format-preserving-encryption-or-how-to/>