RANDOM KEY ENCRYPTION USING ONE-TIME PAD IN PYTHON

A MINI PROJECT REPORT

By

AJAY KUMAR (RA2111030010084)
SRIVARSHAN(RA2111030010125)
SUDIKSHAN(RA2111030010116)
ESWAR A (RA2111030010086)
SWADHIK (RA2111030010120)

Under the guidance of

Prabhu Chakkaravarthy A

In partial fulfilment for the Course

of

18CSE381T-CRYPTOGRAPHY

FACULTY OF ENGINEERING AND TECHNOLOGY,
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
Kattankulathur, Chenpalpattu District

OCTOBER 2023

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report about **Random Key Encryptio Using One-Time Pad In Python** is the bonafide work of **AJAY KUMAR (RA2111030010084), SWADHIK (RA2111030010120), SUDIKSHAN S (RA2111030010116), ESWAR A (RA2111030010086) & SRIVARSHAN (RA2111030010125)** whom carried out the project work under my supervision.

Dr.Prabhu Chakkaravarthy A

Assistant Professor
Department of Networking and Communications

Dr.Annapurani Panaiyappan.K

HEAD OF THE DEPARTMENT
Department of Networking and Communications

TABLE OF CONTENTS

CHAPTERS	CONTENTS	PAGES
1.	ABSTRACT	3
2.	LITERATURE REVIEW	4
3.	ALGORITHM	9
4.	ARCHITECTURE DIAGRAM	12
5.	RESULT AND DISCUSSION	13
6.	CONCLUSION	17
8.	REFERENCES	18

(CHAPTER 1)

Abstract

The One-Time Pad (OTP) encryption method stands as the epitome of cryptographic security, boasting unparalleled resilience under specific stipulations. This comprehensive report undertakes an extensive examination of OTP, encompassing its historical genesis, theoretical foundations, practical instantiation through Python, and its contemporary significance within the ever-evolving landscape of cryptography. Delving into the intricacies of OTP, the report meticulously dissects the nuanced processes of key generation, encryption, and decryption, aiming to illuminate the inherent strengths, limitations, and intricate challenges intrinsic to OTP in modern cryptographic discourse.

OTP's inception by Gilbert Vernam and Joseph Mauborgne in 1917 marked a watershed moment in cryptographic theory. Founded on the bedrock principle of perfect secrecy elucidated by Claude Shannon, OTP ensures that resulting ciphertext reveals no discernible information about the plaintext, provided the key remains genuinely random, confidential, and unrepeated. This theoretical elegance positions OTP as impervious to brute-force attacks and statistical scrutiny, signifying its unparalleled status as a bastion of secure communication. However, pragmatic constraints, including key distribution complexities and secure key storage challenges, have curtailed its ubiquitous practical application despite its theoretical perfection.

This report aims to illuminate the multifaceted facets of OTP, elucidating its theoretical prowess and real-world implications. By scrutinizing its historical trajectory, theoretical underpinnings, and pragmatic applications through Python implementation, the report endeavors to provide a comprehensive understanding of OTP's significance and the complexities it introduces within modern cryptographic discourse.

(CHAPTER 2)

Literature Review

In 1917, Gilbert Vernam invented a cipher solution for the teletype machine. The United State (U.S.) Army Captain Joseph Mauborgne realized that the character on the key tape could be completely random. Together, they introduced the first One Time Pad encryption system. Since then, One Time Pad systems have been widely used by governments around the world. Outstanding examples of a One Time Pad system include the 'hotline' between the White House and the Kremlin and the famous Sigsaly speech encryption system .

1.The Mathematical Proof of OTP Security:

According to Alfred Menezes et al. 1997 in their book, Handbook of Applied Cryptography, a system can be called perfectly secret, or unconditionally secure, when observing ciphertext gives an eavesdropper no additional information about the original plaintext string.

If we let L be the number of bits in the plaintext string,

then i ranges from 1 to L in the following definitions:

pi = the ith bit in the plaintext string,

ci = the ith bit in the ciphertext string,

ki = the ith bit in the key string,

P(pi) = the probability that pi was sent

P(pi | ci) = the probability that pi was sent given that ci was observed.

A system can be called perfectly secret when P(pi) = P(pi | ci). This section will prove that a One Time Pad system is perfectly secret. In traditional stream cipher systems, the most common method of mixing plaintext data bits with key bits is by performing the XOR operation on the corresponding bits. XOR is short for exclusive OR. The following is a table that defines XOR (the column a as a bit of plain text and column b as its corresponding key bit):

TABLE I
DEFINES XOR

a	b	an XOR b
b	0	0
0	1	1
1	0	1
1	1	0

The sender makes ciphertext by XOR-ing plain text and key one bit at a time: ci = pi XOR ki ……equation (1)

where ci, pi, and ki are as defined above. Because the One Time Pad key is completely random and unpredictable, two conclusions can be drawn:

First, the probability of observing any One Time Pad key bit is equal to the probability of observing any other One Time Key bit.

Second, knowing all the previous values of the key in a sequence tells us nothing about the next key bit. By stating another definition, P(ki) = the probability that ki was used to create ci the first conclusion drawn above can be written as;

P(Ki = 1) = P(Ki = 0) = 1/2 for all i…… equation (2) In other words, a bit of One Time Key is just as likely to be a 1 as a 0 at any time. The second conclusion drawn above allows us to consider triples of the key, ciphertext, and plain text for a value of I without regard for other triples.

Equation (1) leads to an important observation: knowing any two of {pi, ci, ki } determines the third. Likewise, given one of {pi, ci, ki }, a second one can be written in terms of the third.

For example, P(ci = 1 | ki = 0) = P(pi=1); in other words, if we know for a fact that the key bit is 0, then plain text and cipher text must be equal. In order to show that P(pi | ci) = P(pi), we first need to show P(ci) = P(ci | pi). Using equation (1), we will do this explicitly by first deriving the distribution of P(ci). Next, we will derive the distribution of P(ci | pi) given that the plain text bit is a 0 and then given that it is a 1.

Distribution of P(Ci)

P(ci = 1) = P(ci = 1 | ki = 1) P(ki = 1) + P(ci = 1 | ki = 0) P(ki = 0) by the definition of conditional probability = P(pi = 0) P(ki = 1) + P(pi = 1) P(ki = 0)

by equation (1) = P(pi = 0) (1/2) + P(pi = 1) (1/2)

by equation (2) = (1/2) [P(pi = 0) + P(pi = 1)]

regrouping = 1/2 since pi can only be 1 or 0 P(ci = 0) = P(ci = 0 | ki = 1) P(ki = 1) + P(ci = 0 | ki = 0) P(ki = 0)

by the definition of conditional probability = P(pi = 1) P(ki = 1) + P(pi = 0) P(ki = 0)

by equation (1) = P(pi =1) (1/2) + P(pi = 0) (1/2)

by equation (2) = (1/2) [P(pi =1) + P(pi = 0)]

regrouping = 1/2 since pi can only be 1 or 0

Distribution of P(ci | pi)

If pi = 0:

P(ci = 0 | pi = 0) = P(ki = 0)

by equation (1) = 1/2

by equation (2) P(ci = 1 | pi = 0) = P(ki = 1)

by equation (1) = ½

by equation (2) If pi =1: P(ci = 0 | pi = 1) = P(ki =1)

by equation (1) = 1/2

by equation (2) P(ci=1 | pi=1) = P(ki=0)

by equation (1) = 1/2

by equation (2) It is clear from the distributions derived above that P(ci | pi) = P(ci). Recall that a system can be called perfectly secret when P(pi) = P(pi | ci).

Using the definition of conditional probability, the joint probability, P(pi and ci), the probability that pi and ci are observed, can be written in the following two (equivalent) forms:

P(pi and ci) = P(ci | pi) P(pi) and P(pi and ci) = P(pi | ci) P(ci). Combining the two equations gives P(pi | ci) P(ci) = P(ci | pi) P(pi).

Since P(ci | pi) = P(ci) as shown above, these two terms cancel, leaving P(pi | ci) = P(pi), which is the condition for perfect secrecy.

Although, the proof has yielded significant result based on a condition for perfect secrecy. However, there is still a practical difficulty of using an OTP and this can be explained further in this paper

2. **Practical difficulty of using an OTP :**

The practical difficulty of using an OTP is that the pad/key bytes cannot be reused. This means that even for a two-way communication, each entity must have a sufficient supply of key material on hand so that they don't run out of keys before new ones can be generated. People are not interested in modifying the algorithm, they are more interested in improving the way the key is generated either by trying to introduce a true random instance or modifying the algorithm that generates the keys to create a lifetime supply of key. Their implementation only tries to solve the problem of getting true randomness but does not solve the distribution/management of key material as different keys still have to be sent for different messages.The proposed algorithm solves the key problem by making it possible to use the same key to encrypt different messages and not reveal any pattern that could be exploited by the attacker.

At the advent of binary systems for computational analysis (computers), memory and processing power was expensive and hard to obtain. This led to brilliant mathematical implementations of encryption that protected data, including communication. Due to the impracticality of OTPs, modern encryption was borne which is based upon limited, finite size keys and produces creative attacks other than a 'brute force' attack. Capable mathematicians and technologists are highly motivated in their attempts to break encryption; they are succeeding. They have devised many attacks such as man-in-the-middle, statistical, side channel attacks, and many more (Belakang, an L. 1991) .

The one-time pad system was modified by using the concepts of 10's complement operation. The eavesdropper come across confusion by observing decimal and binary combination with added concept of complements (Patil, S., Patil, A., & Kumar, A. 2012). Another attempt at improving the algorithm was made by using a conventional block cipher and one-way hash algorithm to design the one-time pad algorithm. This algorithm proposed by them totally balance the insufficiencies of the conventional block cipher, and exploit the benefits of the one-way hash algorithm (Tang, S., & Liu, F. 2012) .

Penchalaiah modified the One-Time pad algorithm to work without any secret key overhead while on the transmission (since the key is along as message) by using two algorithms, a Key Exchanging Algorithm, and a Random Bit Generation algorithm (Penchalaiah, P. 2013) .

The problem of key distribution and protection was solved using elliptic curve cryptography. An overview of Koblitz method of encoding was provided and a hybrid security mechanism based on OTP was developed (Katti, J. 2015) . The One-Time Pad was modified with 2's complement approach to introduce more complexity and make the task of cryptanalyzing any ciphertext recovered to be more difficult (Devipriya, M., & Sasikala, G. 2015) .

Another attempt at handling the randomness of the key was to use a simple quantum circuit to generate a truly random OTP using quantum superposition states (Upadhyay, G., & Nene, M. J. 2016) .

A one-time pad cryptographic method was designed using a star network of N Lorenz subsystems, referred to as augmented Lorenz equations, which generates chaotic time series as pseudorandom numbers to be used for masking a plaintext (Miyano, T., & Cho, K. 2016) .

3. Related works on enhanced OTP :

There are publications and extant literature on OTP cipher and its enhancements. Much quantum key distribution (QKD) system has been expanded to quantum network manager using OTP encryption . This outline does not just handle the switch and QKD protocol startup processes but as well handles multiplexing and synchronization of secret key streams. An encryption algorithm based on OTP technique to provide sufficient privacy of images using chaos theory has been stated in (C. Jeyamala et al., 2010) . The investigation results of the study have been evaluated with benchmark images and are compared with different image encryption algorithms reported in the literature. Key sensitivity analysis, key space analysis, and numerical analysis proved that this algorithm proposes better security at minor calculational overhead. ln (M. Borowski et al., 2012) , Borowski and Lesniewicz presented a hardware generation of binary random sequences with the latent output rate of 100 Mbit/s to eliminate the limitation associated with accessibility of lengthy one-time keys.

A new study on OTP encryption enhancement as in (Patil, M. Devare and A. Kumar 2009) , has shown that the random key stream can be employed to generate a lifetime supply of keys for OTPs. Random key generation can easily be created by permutation methods. These methods can be adopted in combination with other procedure such as substitution and encryption function for successful results. The objective of this study is to demonstrate how OTP encryption technique can be accomplished by a combining of these techniques. In (S.G. Srikantaswamy, and H.D. Phaneendra), two new methods of OTP encryption enhancement based on I O'S complement and XOR operations have been presented that do not depend on the original cipher about OTP cipher.

(CHAPTER 3)

Algorithm

Character Assignment and Conversion-Assigns numerical values to characters in a predefined set and converts text (message or key) into corresponding numerical values based on these assigned IDs.

Encryption-Generates a random key as long as the message from a predefined character set.Converts the message and key characters into numerical values.Performs OTP encryption by adding numerical values of message and key characters using modular arithmetic.Converts the resulting encrypted numerical values back to characters.

Decryption-Converts the encrypted message and key into numerical values.Reverses the OTP encryption process by subtracting key values from encrypted message values using modular arithmetic.Converts the decrypted numerical values back to characters to reveal the original message.

Main Function-Orchestrates encryption and decryption by taking user input for the message, generating a key, encrypting the message, and prompting the user to enter the decryption key to decrypt the message.

Python code:

```
import random

CHARACTERS =
"AÄBCDEFGHIJKLMNOPÖQRSßTÜVWXYZaäbcdefghijklmnoöpqrstuvwxyz0123
456789!@#$%&*()-_+=[].{}<>?/\\|:;'\",.,`~ "

def assign_character_ids():
    character_ids = {}
    for i, char in enumerate(CHARACTERS):
        character_ids[char] = i
    return character_ids
```

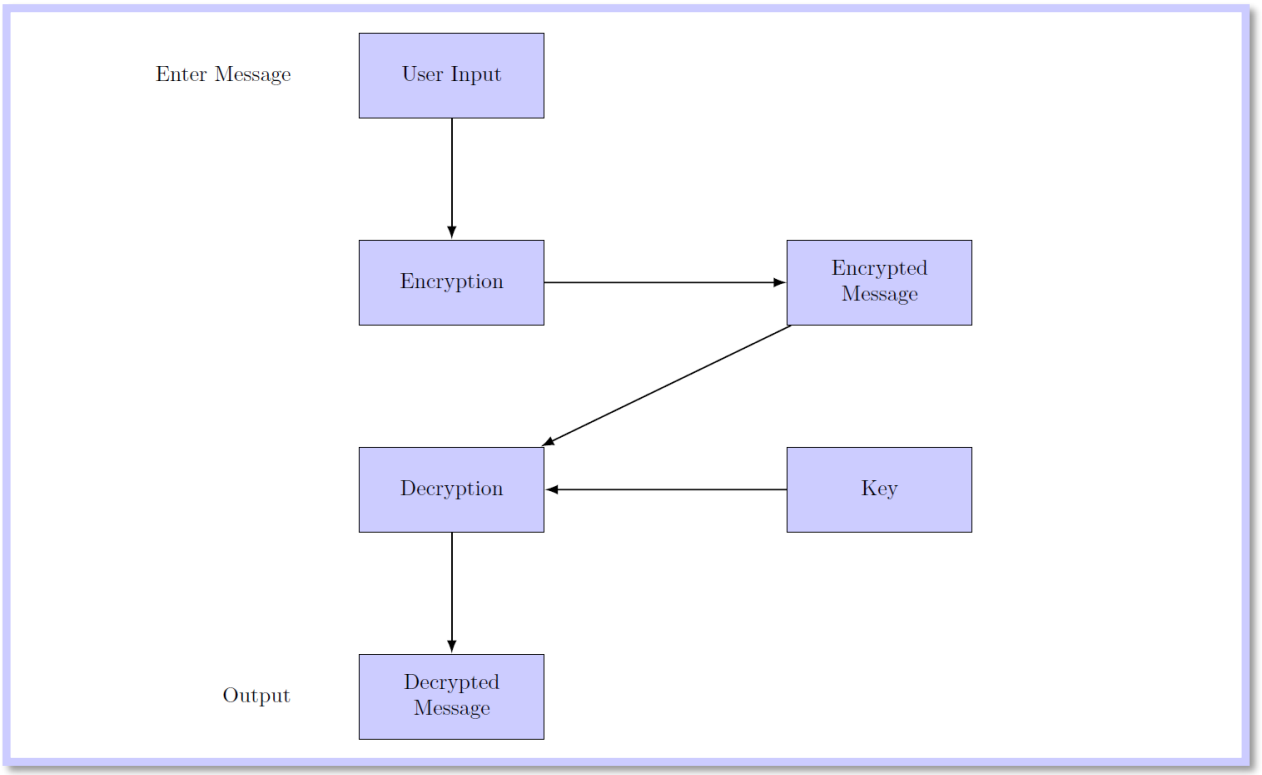
```
def convert_to_numbers(text, character_ids):
    converted_chars = []
    for c in text:
        value = character_ids[c]
        converted_chars.append(value)
    return converted_chars

def encrypt(message, character_ids):
    random_bytes = [random.randint(0, 255) for _ in range(len(message))]

    key_builder = []
    for i in range(len(message)):
        index = random_bytes[i] % len(CHARACTERS)
        key_builder.append(CHARACTERS[index])
    key = ''.join(key_builder)

    converted_message_chars = convert_to_numbers(message, character_ids)
    converted_key_chars = convert_to_numbers(key, character_ids)

    encrypted_message = []
    for i in range(len(converted_message_chars)):
        message_char = converted_message_chars[i]
        key_char = converted_key_chars[i]
        encrypted_char = (message_char + key_char) % len(CHARACTERS)
        encrypted_message.append(encrypted_char)

    encrypted_text = ''.join([CHARACTERS[i] for i in encrypted_message])
    return encrypted_text, key

def decrypt(message, key, character_ids):
    converted_message_to_numbers = convert_to_numbers(message, character_ids)
    converted_key_to_numbers = convert_to_numbers(key, character_ids)

    decrypted_message = []
    for i in range(len(converted_message_to_numbers)):
        message_char_number = converted_message_to_numbers[i]
        key_char_number = converted_key_to_numbers[i]
        decrypted_char = (message_char_number - key_char_number) %
len(CHARACTERS)
```

```
        if decrypted_char < 0:
            decrypted_char += len(CHARACTERS)
        decrypted_message.append(decrypted_char)

    decrypted_text = ''.join([CHARACTERS[i] for i in decrypted_message])
    return decrypted_text

def main():
    character_ids = assign_character_ids()

    user_message = input("Enter the message to encrypt: ")

    encrypted_msg, key = encrypt(user_message, character_ids)
    print(f"Encrypted: {encrypted_msg}")
    print(f"Single-use Key: {key}")

    decryption_key = input("Enter the decryption key: ")
    decrypted_msg = decrypt(encrypted_msg, decryption_key, character_ids)
    print(f"Decrypted: {decrypted_msg}")

if __name__ == "__main__":
    main()
```

(CHAPTER 4)

Architecture Diagram

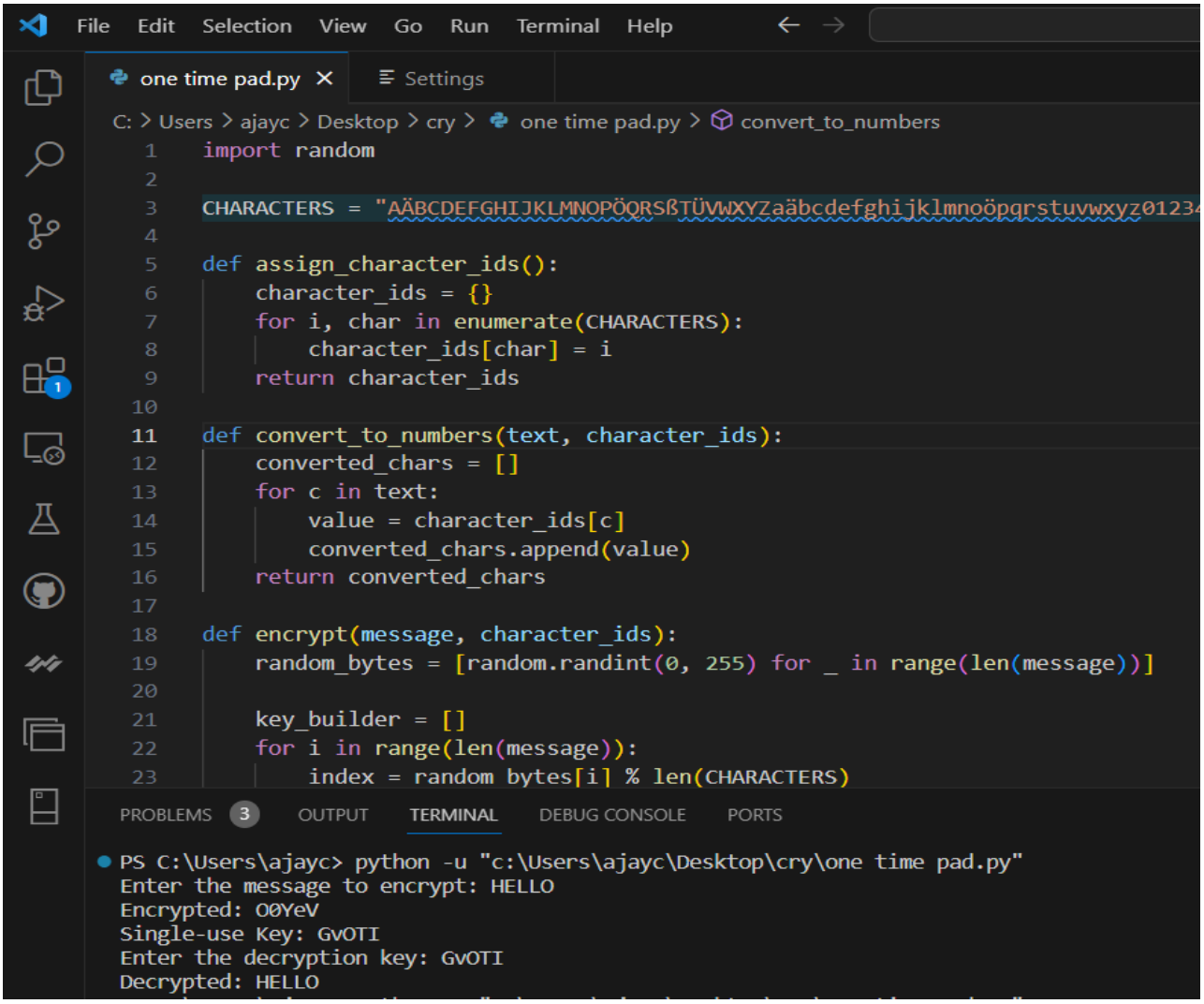
The architectural blueprint of the OTP encryption methodology delineates a structured framework encompassing key generation, encryption, and decryption stages. This linear trajectory entails the generation of a random key, its utilization in the encryption process to yield ciphertext, and the subsequent decryption employing the same key to meticulously reconstruct the original plaintext. This systematic approach ensures a secure communication channel, contingent upon the unassailable security and confidentiality of the key.



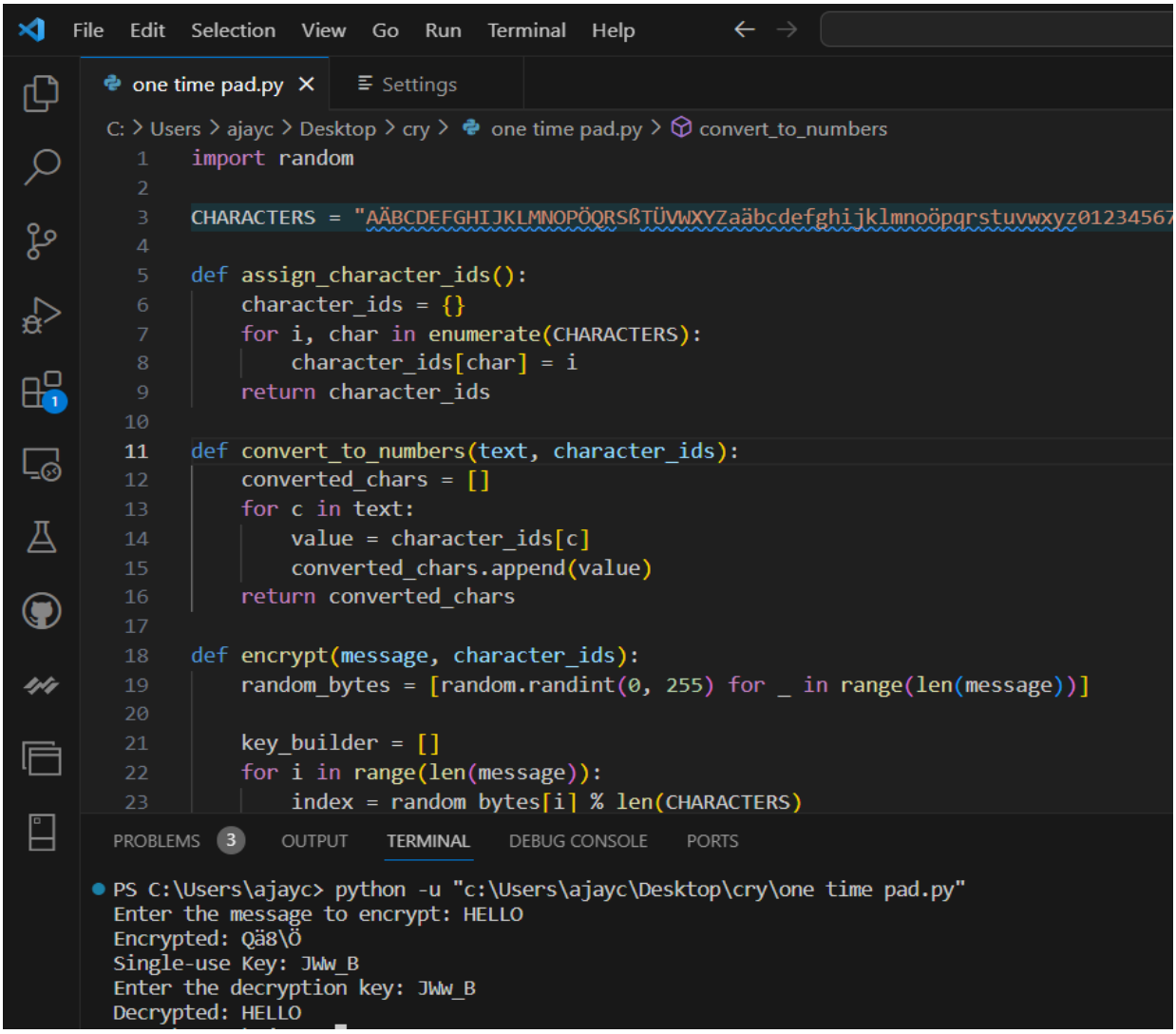
(CHAPTER 5)

Result and Discussion

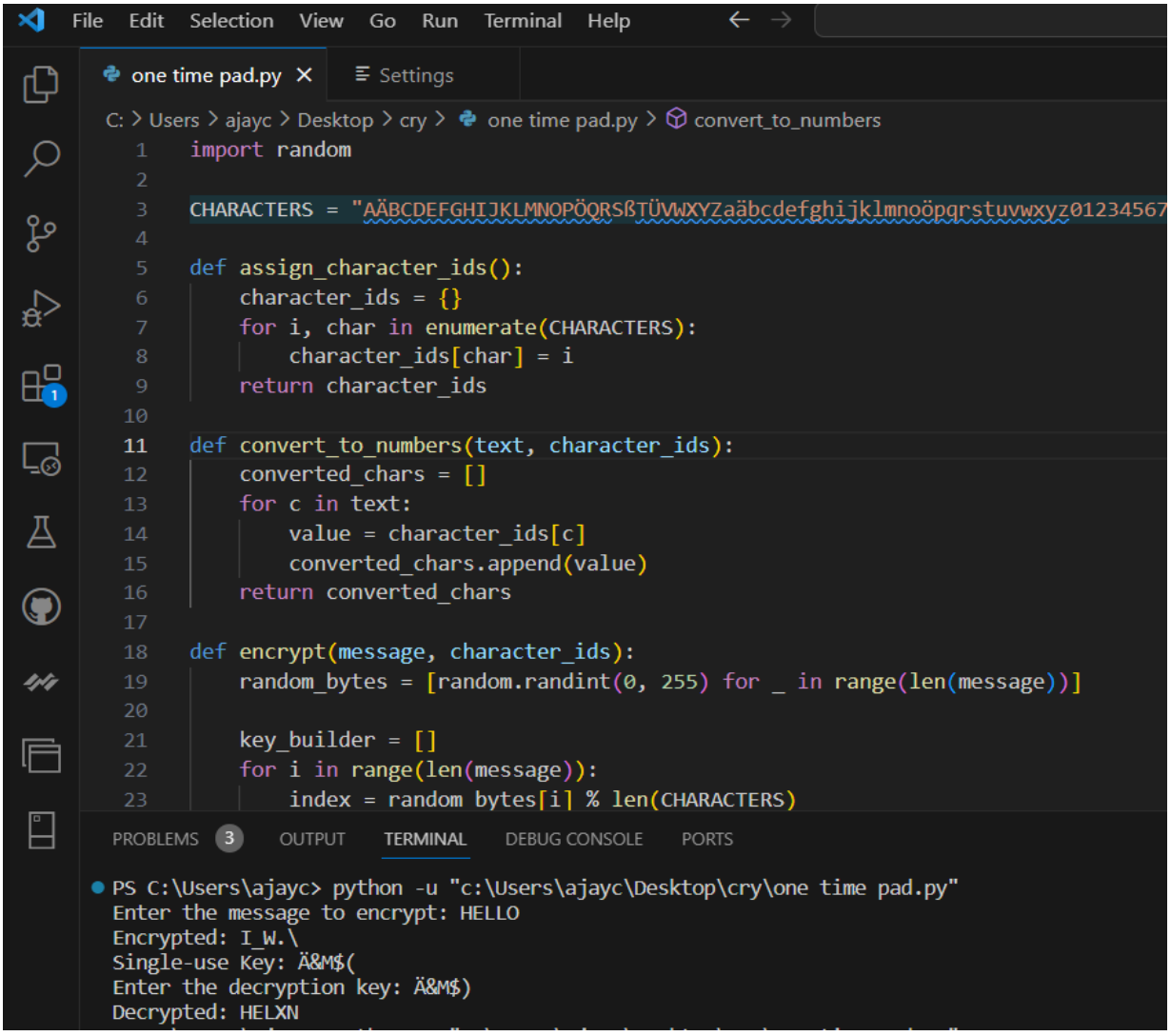
Scenario 1:



Scenario 2:


File Edit Selection View Go Run Terminal Help
one time pad.py
Settings
C: > Users > ajayc > Desktop > cry > one time pad.py > convert_to_numbers
1 import random
2
3 CHARACTERS = "AÄBCDEFGHIJKLMNOPÖQRSßTÜVWXYZaäbcdefghijklmnoöpqrstuvwxyz01234567
4
5 def assign_character_ids():
6 character_ids = {}
7 for i, char in enumerate(CHARACTERS):
8 character_ids[char] = i
9 return character_ids
10
11 def convert_to_numbers(text, character_ids):
12 converted_chars = []
13 for c in text:
14 value = character_ids[c]
15 converted_chars.append(value)
16 return converted_chars
17
18 def encrypt(message, character_ids):
19 random_bytes = [random.randint(0, 255) for _ in range(len(message))]
20
21 key_builder = []
22 for i in range(len(message)):
23 index = random bytes[i] % len(CHARACTERS)
PROBLEMS 3 OUTPUT TERMINAL DEBUG CONSOLE PORTS
PS C:\Users\ajayc> python -u "c:\Users\ajayc\Desktop\cry\one time pad.py"
Enter the message to encrypt: HELLO
Encrypted: Qä8\Ö
Single-use Key: JWW_B
Enter the decryption key: JWW_B
Decrypted: HELLO

Scenario 3:


File Edit Selection View Go Run Terminal Help
one time pad.py
Settings
C: > Users > ajayc > Desktop > cry > one time pad.py > convert_to_numbers
import random
CHARACTERS = "AÄBCDEFGHIJKLMNOPÖQRSßTÜVWXYZaäbcdefghijklmnoöpqrstuvwxyz01234567
def assign_character_ids():
character_ids = {}
for i, char in enumerate(CHARACTERS):
character_ids[char] = i
return character_ids
def convert_to_numbers(text, character_ids):
converted_chars = []
for c in text:
value = character_ids[c]
converted_chars.append(value)
return converted_chars
def encrypt(message, character_ids):
random_bytes = [random.randint(0, 255) for _ in range(len(message))]
key_builder = []
for i in range(len(message)):
index = random_bytes[i] % len(CHARACTERS)
PROBLEMS 3 OUTPUT TERMINAL DEBUG CONSOLE PORTS
PS C:\Users\ajayc> python -u "c:\Users\ajayc\Desktop\cry\one time pad.py"
Enter the message to encrypt: HELLO
Encrypted: I_W.\
Single-use Key: Ä&M$(
Enter the decryption key: Ä&M$)
Decrypted: HELXN

Scenario 1: Encrypting "hello" with a Generated Key and Decryption

- **Message:** "HELLO"
- **Key:** GvOT1
- **Encryption:** The code takes "hello" as the input message and generates a key of the same length as the message. It then encrypts the message using this generated key.
- **Decryption**: Upon entering the correct key during decryption, the code retrieves the original "hello" message successfully.

Scenario 2: Using Different Keys for Encrypting and Decryption

- **Message:** "HELLO"
- **Key :** JWw_B
- **Encryption with Key** : The same "hello" message is encrypted using a different randomly generated key from the first scenario.

Scenario 3: Incorrect Key Used for Decryption

- **Message:** “HELLO”
- **Correct Key:** Ä&M$(
- **Incorrect Key:** Ä&M$)
- **Decryption Attempt with Incorrect Key:** When providing an incorrect key during decryption, the code fails to retrieve the original message. Instead, it produces a completely different and incorrect decrypted output.

These scenarios highlight the critical aspect of OTP encryption and decryption. The correct key, which needs to be the same as the one used for encryption, is paramount. Any deviation or incorrect key results in an entirely different decrypted message, emphasizing the principle of perfect secrecy and the importance of the key's secrecy and accuracy in OTP decryption.

(CHAPTER 6)

Conclusion

The One-Time Pad encryption method stands as an enduring symbol of cryptographic invincibility, revered for its unparalleled theoretical strength. Its foundation rests upon the bedrock of perfect secrecy—a rare and coveted attribute in the realm of cryptography. The fundamental premise that underpins its impregnability lies in the sheer randomness and unpredictability of the key stream, perfectly matching the length of the plaintext, rendering deciphering attempts futile without access to the precise key.

In its pristine form, the OTP remains an impregnable bastion against cryptanalysis. Its theoretical underpinnings, rooted in the principles of information theory, bolster the belief that a cipher system can achieve absolute secrecy, provided the key material adheres strictly to the conditions of randomness, uniqueness per use, and remains securely shared between the communicating parties.

The crux of OTP's security hinges on the flawless generation, distribution, and safeguarding of keys—each key must be as long as the message, never reused, and shared securely between the sender and receiver without falling into the hands of adversaries. Achieving this level of logistical perfection remains a daunting task in real-world scenarios, where the sheer logistics of managing and distributing keys become exponentially complex with increasing message lengths and the number of communicating parties.

Yet, despite these practical limitations, the One-Time Pad stands tall as a theoretical cornerstone in the realm of cryptography. Its existence and the pursuit of its perfection have served as a lodestar, inspiring the evolution of modern encryption paradigms. The quest for encryption methods that emulate the unbreakable security of OTP while mitigating its logistical challenges has spurred the development of innovative cryptographic techniques, such as public-key cryptography and symmetric encryption algorithms, each striving to strike a delicate balance between security and practicality.

In essence, the One-Time Pad, with its theoretical invincibility and practical limitations, remains a beacon illuminating the path toward cryptographic innovation. Its existence serves as a constant reminder of the delicate interplay between theory and practice in the realm of secure communication.

(CHAPTER 7)

References

Cryptography_Engineering:https://cpuu.postype.com/post/217781: https://cpuu.postype.com/post/217781

TechTarget:https://www.techtarget.com/searchsecurity/definition/one-time-pad: https://www.techtarget.com/searchsecurity/definition/one-time-pad

Claude Shannon's 1949 paper "Communication Theory of Secrecy Systems":https://ieeexplore.ieee.org/document/6769090: https://ieeexplore.ieee.org/document/6769090

Gilbert Vernam's 1949 patent "Secret Communication":https://patents.google.com/: https://patents.google.com/

Katz, Jonathan, and Yehuda Lindell. Introduction to Modern Cryptography. CRC Press, 2014.

Goldwasser, Shafi, and Mihir Bellare. "Lecture Notes on Cryptography." Massachusetts Institute of Technology, 2019.

Diffie, Whitfield, and Martin E. Hellman. "New directions in cryptography." IEEE Transactions on Information Theory 22.6 (1976): 644-654.