Kingdom Of Saudi Arabia
Ministry Of Education
Onaizah Colleges
College of Engineering and IT
Computer Science Dep.

المملكة العربية السعودية
وزارة
كليات عنيزة الأهلية
كلية الهندسة وتقنية المعلومات
قسم علوم الحاسب

Encryption and Decryption Tool Using Python

Student : Mayar Yousef Alsaeed

Student ID : 461210731

Introduction 3

Cybersecurity 4

Crypotography.....5

History.....6

Algorithms.....7

Selected Algorithms.....8

Program.....9

Results.....10

Conclusion.....11

Refrenses.....12

Genaral Introduction

Encryption is the essential process of converting readable information (plaintext) into a secret, coded form (ciphertext) to protect it from unauthorized access **(Khan Academy, n.d.)**. This process is fundamental to keeping personal and sensitive data secure during both transmission and storage.

Conversely,*Decryption* is the inverse process, which transforms the ciphertext back into readable information using a specific cryptographic key. These two processes are vital in the field of *cybersecurity* because they actively prevent attackers from stealing, altering, or disrupting private data.

This research , *“Encryption and Decryption Tool using Python”,* focuses on creating a simple, functional program that demonstrates how secure cryptographic algorithms operate. Python is a powerful language that offers robust libraries, such as `cryptography`, for this purpose. The main goal is to understand the practical implementation of encryption, explore the core concepts of cybersecurity, and detail the main algorithms used today.

Cybersecurity

Cybersecurity refers to the coordinated practice of protecting computer systems, networks, and digital information from unauthorized access, attacks, or damage **(Palo Alto Networks, n.d.)**. In the contemporary digital environment, where vast amounts of sensitive data—including financial records and corporate secrets—are stored and transmitted electronically, cybersecurity is critically important.

Encryption plays a foundational role in cybersecurity**.** By creating coded messages that are only reversible with the correct key, encryption ensures **data confidentiality** and helps organizations prevent data breaches. Crucially, it also ensures **data integrity** (the ability to detect unauthorized changes) and **authentication** (confirming the data source is legitimate) .

Effective modern cybersecurity strategies integrate encryption with other protective layers, such as multi-factor authentication (MFA), secure network protocols, and routine software patching. These combined measures, guided by frameworks **(NIST Cybersecurity Framework, 2024)**, enable organizations to defend against various sophisticated threats like phishing, ransomware, and Business Email Compromise (BEC)

Cryptography

The most widely supported and accessible library for handling tasks in Python is the `cryptography` library, which supports both *symmetric and asymmetric algorithms* **(PyCA Development Team, n.d.)**. Symmetric encryption uses a single shared key for both operations, whereas asymmetric encryption employs a pair of keys (public and private) for enhanced security.

A practical and recommended tool within the library is Fernet. Fernet specifically combines strong AES-based symmetric encryption with authentication features to guarantee that messages remain confidential and tamper-proof. The library also includes support for the Advanced cryptographic Encryption Standard (AES), a globally adopted standard for securing digital data **(NIST, 2001)**. Utilizing Python's cryptographic libraries allows developers to rapidly implement tools, test them, and gain an understanding of essential concepts like key management and message integrity.

Key Features of the Python `cryptography` Library:

- Provides both symmetric (AES, Fernet) and asymmetric (RSA) encryption capabilities.
- Simplifies complex encryption and decryption processes.
- Includes built-in verification mechanisms to detect message tampering.
- Adheres to modern cryptographic standards, ensuring strong security.

History

The history of cryptography is ancient, spanning millennia. Early civilizations utilized simple substitution methods, such as the *Caesar cipher,* to protect sensitive communications **(NSA, n.d.)**.

A major evolutionary leap occurred in the 20th century with mechanical encryption devices. The *Enigma machine,* used during World War II, generated complex codes, and the concerted effort to break it by early

computer scientists was a critical catalyst for the development of modern digital computing **(NSA, n.d.)**.

With the digital age, cryptography transitioned to sophisticated mathematical algorithms. Standards such as *RSA and AES* emerged to provide robust protection for everything from banking to cloud storage. The **AES** standard, specifically, is universally used to secure digital data **(NIST, 2001)**. The continuous evolution of cryptography, coupled with security protocols and network monitoring, is vital for modern cybersecurity .

Key Milestones in Cryptography History:

- ***Ancient Ciphers*:** Caesar cipher, hieroglyphs, and substitution codes.
- ***Mechanical Encryption*:** Enigma and similar machines in the early 20th century.
- ***Modern Digital Encryption*:** RSA, AES, and authenticated symmetric encryption.
- **I*ntegration with Cybersecurity*:** Protecting networks, personal data, and online communications

Algorithms

Encryption algorithms are primarily categorized into two main types: **symmetric** and **asymmetric (Khan Academy, n.d.)**.

1. ***Symmetric Encryption*:** This type utilizes a single, shared secret key for both the encryption and decryption processes. It is generally **faster and more efficient** for processing large volumes of data. Examples of symmetric algorithms include **AES** (Advanced Encryption Standard) and **Fernet**. The primary limitation of this method, however, is the challenge of securely exchanging the secret key between the sender and the receiver.

2. ***Asymmetric Encryption*:** This method, also known as **Public Key Cryptography**, uses a pair of keys: a public key for encryption and a private key for decryption. While **slower** than symmetric encryption, it offers enhanced security for communication between parties that do not initially share a secret key. **RSA** is a common example of an asymmetric algorithm, making it ideal for secure key exchange and digital signatures.

Selected Algorithms : Fernet

Fernet is selected for this project due to its combination of AES-based symmetric encryption and its authentication features, which together ensure both confidentiality and verifiable data integrity **(PyCA Development Team, n.d.)**. Its simple implementation in Python makes it an ideal choice for practical demonstration and educational purposes.

Why Fernet is Selected:

- ***Security*:** It uses AES in CBC mode internally combined with HMAC for verification.
- ***Ease of Use*:** It offers high-level, simple functions for core operations.

- ***Educational Value*:** It clearly illustrates both the encryption and decryption processes

Program

code.py ● cachtheball2.py

PythonVScode > CSC181_PY > code.py > ...

```python
#First, we need to import (bring in) the 'Fernet' tool from the cryptography library to use it.
from cryptography.fernet import Fernet
# This line creates a unique Secret Key that will be used for both locking and unlocking the message.
# Simply put, this is our private password for the encryption process.
key = Fernet.generate_key()
# Now, we create the actual 'cipher' object (the lock/unlock tool) and give it the 'key' it needs to operate.
cipher = Fernet(key)
#--- Encryption (Locking the Message) ---

# This is the original, readable message (Plaintext) that we want to protect.
message = "Hello, this is a secret message."
# We must convert the normal text (string) into 'bytes' because computers encrypt data in byte format Think of it as preparing the message for the l
encoded_message = message.encode()
# This is the main action: we use the 'cipher' tool to lock (encrypt) the message bytes.
encrypted = cipher.encrypt(encoded_message)
# We print the result. The 'b' prefix shows it's a byte string (the unreadable Ciphertext).
print("Encrypted message:", encrypted)

# --- Decryption (Unlocking the Message) ---

# We use the same 'cipher' tool to unlock (decrypt) the ciphertext.
# This only works because the 'cipher' object already holds the correct 'key'.
decrypted_bytes = cipher.decrypt(encrypted)

# Since the result is still in bytes, we must convert it back to readable text (string) using .decode().
decrypted = decrypted_bytes.decode()

# The final step: printing the original, unlocked message!
print("Decrypted message:", decrypted)
```

Results

After running the implementation using Fernet, the output clearly separates the Encrypted Message and the Decrypted Message.

Explanation of Results

1. ***Encrypted Message*:** This output, a long string of unreadable characters, visually confirms **confidentiality**. It demonstrates that without the correct key, the data is protected from any unauthorized party.
2. ***Decrypted Message*:** The successful restoration of the original plaintext confirms the **integrity and reliability** of the cryptographic process, showing that the encryption is fully reversible only when the correct secret key is applied

Conclusion

In conclusion, this project successfully demonstrated the core role of strong encryption in cybersecurity by implementing a practical **Encryption and Decryption Tool using Python and the Fernet algorithm.** The research confirmed that modern, robust algorithms like Fernet are essential for ensuring **confidentiality and data integrity**. Future work should concentrate on advancing the tool by integrating **asymmetric encryption (RSA)** capabilities and developing secure key management protocols to enhance overall application security.

References

1. Khan Academy. (n.d.). *Cryptography Basics*. Khan Academy Computer Science. Retrieved October 29, 2025, https://www.khanacademy.org/computing/computer-science/cryptography
2. National Institute of Standards and Technology (NIST). (2001). *FIPS 197: Advanced Encryption Standard (AES)*. https://doi.org/10.6028/NIST.FIPS.197
3. National Institute of Standards and Technology (NIST). (2024). *Cybersecurity framework*. Retrieved October 29, 2025, from https://www.nist.gov/cyberframework
4. National Security Agency (NSA). (n.d.). *A brief history of cryptography*. Retrieved October 29, 2025 https://www.nsa.gov/what-we-do/research/cryptography/
5. Palo Alto Networks. (n.d.). *What is cybersecurity?* Retrieved October 29, 2025 https://www.paloaltonetworks.com/cyberpedia/what-is-cybersecurity
6. PyCA Development Team. (n.d.). *Cryptography library documentation*. Retrieved October 29, 2025 https://cryptography.io/en/latest/