# Cybersecurity Internship Report 2025

Prepared by: Marie Claude Hayfa

Internship Role: Cybersecurity Intern

Advisor: Ibrahim Fleifel

Company: BabiEat

Date: 2/25/2025

## Introduction

BabiEat company is a premium food delivery and restaurant services platform connecting customers with local dining establishments. BabiEat offers real-time ordering, diverse cuisine options, reliable delivery services, and restaurant discovery features.

## 1. CIA Triad

The CIA Triad is a core model in information security:

• Confidentiality: Only authorized users can access data (e.g., phone passwords).

• Integrity: Data remains accurate and unaltered (e.g., patient medical records).

• Availability: Systems and data are accessible when needed (e.g., online banking).

## 2. Symmetric Encryption

Uses a single shared key for both encryption and decryption. Efficient for large data transfers.

Example: AES or DES encrypting a message between two parties who share a secret key.

## Symmetric Encryption Demo

```
from cryptography.fernet import Fernet
def generate_key():
    key = Fernet.generate_key()
    with open("key.key", "wb") as key_file:
        key_file.write(key)
    print("Key is generated and saved to key.key")
```

```
# Function to load the key from the file
def load_key():
    return open("key.key", "rb").read()

# Function to encrypt a message using the loaded key
def encrypt_message(message):
    key = load_key()
    f = Fernet(key)
    encrypted_message = f.encrypt(message.encode())
    return encrypted_message
# Function to Decrypt a message using the loaded key

def decrypt_message(encrypted_message):
    key = load_key()
    f = Fernet(key)
    return f.decrypt(encrypted_message).decode()

# Generate a key (run once)
generate_key()

# Encrypt a message
encrypted_message = encrypt_message("Hello, World!")
print(f"Encrypted Message: {encrypted_message}")
# Decrypt a message
decrypted = decrypt_message(encrypted_message)
print(f"Decrypted: {decrypted}")
```

### 2.1 Symmetric Demo Output

Key is generated and saved to key.key

Encrypted Message: b'gAAAAABonhl4wqo-A93ZpDh60BJaiFeJbF4TfdGPsWERouvf9yTN36VzPZEJyjbqYkaAxo1QDydmwqf5GXNuaJk6ZoB_rCoLzQ=='
Decrypted: Hello, World!

## 3. Asymmetric Encryption

Uses two keys – a public key to encrypt and a private key to decrypt. Ensures secure communication without sharing a secret key.
Example: RSA encrypting a message with a public key, decrypted only with the recipient's private key.

### Asymmetric Encryption Demo

```
import rsa
public_key,private_key=rsa.newkeys(1024)
message = "Hello, this is an RSA test!encrypt and decrypt message"
print("Original Message:", message)
encrypted_message = rsa.encrypt(message.encode(), public_key)
print("Encrypted Message:", encrypted_message)
decrypted_message = rsa.decrypt(encrypted_message, private_key).decode()
print("Decrypted Message:", decrypted_message)
```

### 3.1 Asymmetric Demo Output

Original Message: Hello, this is an RSA test!encrypt and decrypt message

Encrypted Message:
b'\x04\x05\x88b^\x19\x7f\xa5,\xfai\xca\x1b\x06F\xe34\x89\xa9\xf9\x8e\x1c>\xc8\xe2\xc3\xdfe\xe9+rD\xb8\xf7\xd6\xef\xbc3c\xfe\x10\x8d\xe7\xd6f\x15Ase\xa0\xe5\x7fo\xedt\x96\xb7\xce\xe8\xb7\x91\xd0\x04\xc9\x03b\x15U\xae"\xa6\xac\xd8\xab\xbb\x18\xee\x15O\xf4\xcf\xfc\xf0\x04\x8d\x98\xc7!\xfe\xdc\xb73\x11\xace\t\xd73\x1f\xb2\xecO\xf0\xcd\xffA\xd4\xd5tr\x00\xf0\xb4\x0f=\xfb=\xbc\xd7j|\x92\xf5\xf0\xdam&,'

Decrypted Message: Hello, this is an RSA test!encrypt and decrypt message

### 4. Digital Signatures

Ensures data integrity and authentication. The sender signs a message, and the receiver can verify it hasn’t been altered and came from the correct sender.
Example: Digitally signing a document to confirm authenticity.

### Digital Signatures Demo

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
```

```
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
    backend=default_backend()
)

public_key = private_key.public_key()

data = b"This is some data to be signed."
signature = private_key.sign(
    data,
    padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),
        salt_length=padding.PSS.MAX_LENGTH
    ),
    hashes.SHA256()
)


try:
    public_key.verify(
        signature,
        data,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH

        ),
        hashes.SHA256()
    )

    print("Signature is valid.")
except Exception as e:
    print("Signature verification failed:", str(e))
```

### 5. Network Security Basics

Key measures to protect systems and data:

• Firewalls: Monitor and control incoming/outgoing traffic.

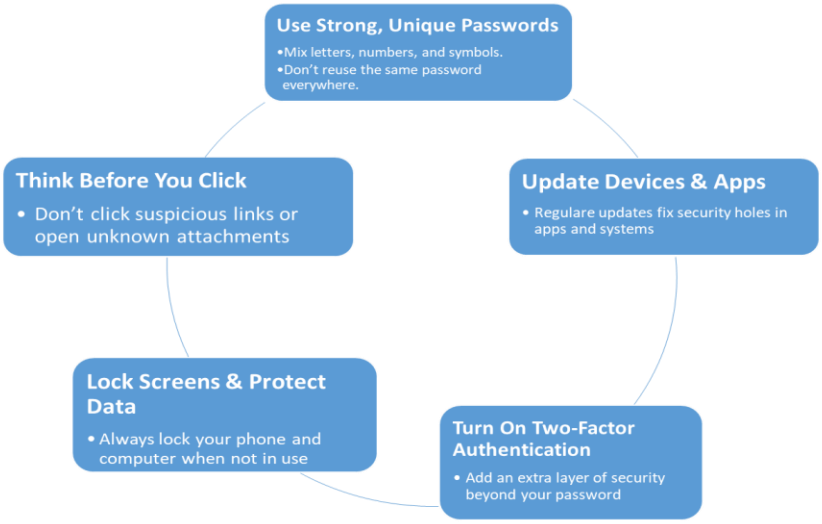
• VPNs: Encrypt network connections for secure communication.

• HTTPS: Secure web communication using encryption.

• Port Scanning & Tools (e.g., Nmap): Identify open ports and potential vulnerabilities.

## Cyber Hygiene Poster

5 tips for staying safe online




## Summary

During this internship at **BabiEat**, I explored key cybersecurity concepts and applied them through practical examples:

1. **CIA Triad:** Learned the core principles of Confidentiality**,** Integrity**,** and Availability, which are essential for protecting sensitive data in any system.
2. **Symmetric Encryption:** Practiced using a single shared key for secure communication and implemented a Python demo with AES encryption.
3. **Asymmetric Encryption:** Understood public/private key encryption and its role in secure communication without sharing secret keys.
4. **Digital Signatures:** Learned how to authenticate data and ensure integrity using digital signatures, verifying that information has not been tampered with.
5. **Network Security Basics:** Studied firewalls, VPNs, HTTPS, and port scanning to understand how networks can be secured and monitored for vulnerabilities.

**Overall Learning:**
This internship allowed me to connect theoretical cybersecurity concepts with practical implementation, reinforcing my understanding of how organizations like BabiEat protect user data, secure communications, and maintain trust in their systems.