SINGAPORE POLYTECHNIC

Diploma in Infocomm Security Management

ST2504

APPLIED CRYPTOGRAPHY

Authors:
Swaathi Lakshmanan (2227171), DISM/FT/1B/05

Class Lecturer:
Mr Calvin Siak

Due Date: 10 February 2023, 11.59 pm

Content Page

1. Introduction

I will be showcasing my understanding of information security principles and cryptographic concepts through the analysis and implementing enhancements to the automated menu system SPAM2. By demonstrating how these ideas are used in a business setting and suggesting solutions to the system's possible problems, I hope to validate my familiarity with these concepts. To ensure the confidentiality, integrity and nonrepudiation of the data at rest and in transit, I will carry out a risk assessment followed by a proposal to address the identified concerns.

2. Current SPAM2 system and its implementation

After the server starts running, the server starts listening for 2 types of commands from the client: “GET_MENU” and “CLOSING”. After the client starts running, it connects to the server and sends the “GET_MENU” command. When the server receives “GET_MENU”, it opens “menu_today.txt” and sends its content to the client. Client then saves the contents to “menu.csv”. Following this, the client sends “CLOSING” and the contents of “day_end.csv” to the server. When the server receives “CLOSING”, it saves the data sent by the client in a file named as “result-<ip address>-<current date and time>.txt”.

3. Assumptions made

I assumed that data being processed in the machine will always remain safe.

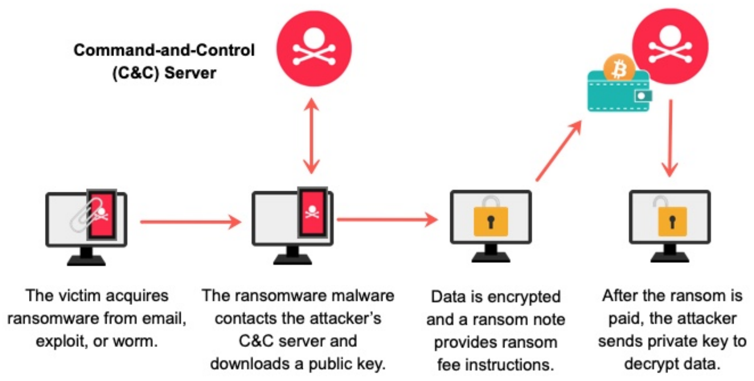
4. Risk assessment

4.1 Assumptions on motivation and capability of attackers

An attacker might aim to steal the data being stored or sent to gain access to trade secrets or for financial gain, modify it and disrupt the operations of the organisation.They may try to exploit the vulnerabilities in the software. They could also have access to sophisticated tools and techniques to intercept and manipulate the data.

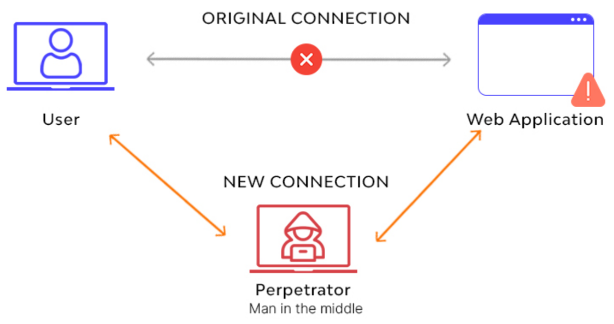
4.2 Processes to be protected

Processes to be protected: Data at rest has to be protected (Storage) and data being sent to the server and data being sent to the client has to be protected. (Communication channel (data in transit))

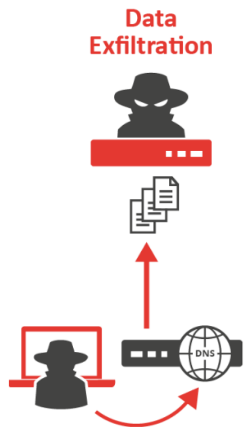

Command-and-Control (C&C) Server
The victim acquires ransomware from email, exploit, or worm.
The ransomware malware contacts the attacker's C&C server and downloads a public key.
Data is encrypted and a ransom note provides ransom fee instructions.
After the ransom is paid, the attacker sends private key to decrypt data.

4.2.1 Confidentiality

Security goals affected: Preserving the confidentiality of the data.
Associated threat of data at rest: Data stored in plaintext form can be easily accessed and read by unauthorised parties.
An example would be the ransomware attack, where an attacker gains unauthorised access to the server that contains unencrypted data, encrypts the data and demands a ransom to decrypt the data.
Associated threat of data in transit: The data can also be vulnerable to eavesdropping, where attackers intercept data travelling between two devices. It can enable other attacks.
An example would be Man-in-the-middle attack, an attacker could intercept the communication between

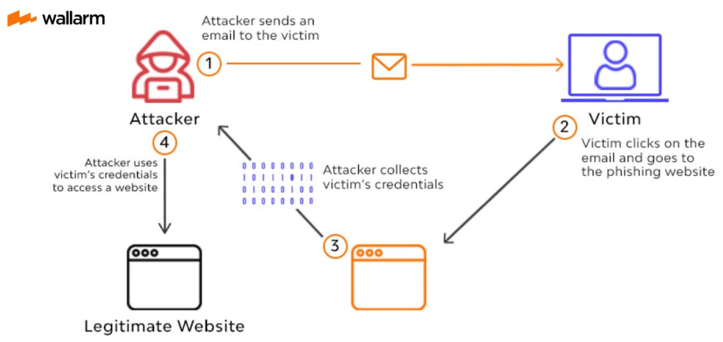


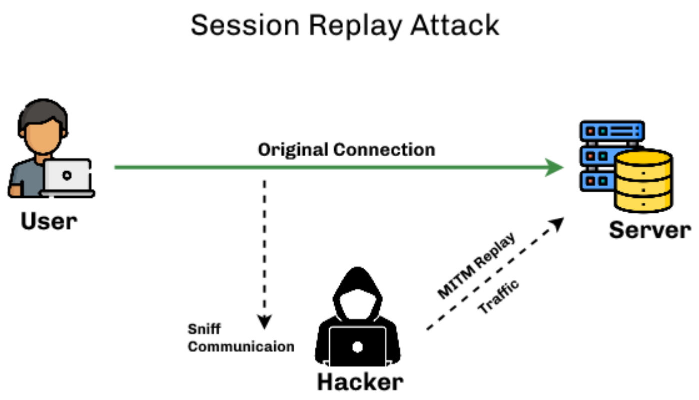
two parties and gain access to the sensitive data being transmitted.
Potential impact: This can lead to the exposure of sensitive information, such as trade secrets.



4.2.2 Integrity

Security goals affected: Preserving the integrity of the data.
Associated threat for data at rest: Without encryption, the data can be tampered with or modified by unauthorised parties without detection.
An example would be data exfiltration. An attacker could gain access to the server that contains unencrypted data, then they could copy, modify or remove the data from the system.
Associated threat for data in transmission: A possible attack would be modification of packets where an attacker alters the contents of a packet in transit.
Potential impact: This can lead to the data becoming corrupted, and the spread of false information. This can also cause significant disruption to the organisation's operations and can result in the loss of important data.


wallarm
Attacker sends an email to the victim
1
Attacker
Victim
2
Victim clicks on the email and goes to the phishing website
4
Attacker uses victim's credentials to access a website
Attacker collects victim's credentials
3
Legitimate Website



4.2.3 Non-repudiation

Security goals affected: Non-repudiation of the data
Associated threat for data in transmission: Non-repudiation of the data could be affected. An attacker could pose as a legitimate sender or receiver. There could also be denial of service, where an attacker can falsely claim that a message was not sent.
An example would be a phishing attack where an attacker could send a message appearing to be from a legitimate sender to trick the organisation into providing sensitive information.
Another example would be replay attack, where an attacker could intercept a valid message and resend it at a later time, bypassing authentication and possibly accessing sensitive information.
Potential impact: This can lead to the inability to identify the true sender or receiver of the data, making it difficult to hold individuals accountable for their actions. Indirectly, through a phishing attack, the confidentiality of the data could be compromised. If a replay attack occurs, the confidentiality and integrity of the data will be compromised.

4.2.4 Counter-measures / controls.

Pycryptodome will be used to implement AES and RSA and encrypt and decrypt data. AES will be used to generate keys to encrypt the data in transit. AES is not vulnerable to attacks because current computing technology cannot use brute force methods to crack AES within one lifetime. RSA- PKCS#1 OAEP will be used to encrypt the AES keys while they are in transit. PKCS#1 OAEP provides protection against chosen-ciphertext attacks, which means without knowing the key used to encrypt the session key, it is almost impossible to decrypt and retrieve the session key. This will ensure the confidentiality of the data in transit.

RSA- PKCS#1 PSS will also be used to generate digital signatures to verify the integrity and non-repudiation of the data. PKCS#1 PSS provides non-repudiation because it uses a random salt during the generation of the signature. This ensures the signature is unique and thus, it cannot be forged or repudiated by the signer. It also ensures that the attacker cannot use precomputed tables to crack the signature. PKCS#1 PSS is also deterministic such that the same private key, salt length and message will always produce the same signature. Therefore, the signer cannot deny having sent the signature. This will ensure the non-repudiation of the data.

PKCS#1 PSS also comes with a cryptographic hash function that compares the message digest from the plain text message, and the message digest from the digital signature. This will allow the admins to detect if any modifications have occurred to the data during the transit and verify the integrity of the data.

I will use the Fernet from the cryptography module to encrypt and decrypt data at rest. Fernet will ensure the data cannot be read or manipulated without its respective key. Ferent also provides authentication guarantees, which means unauthorised changes made to the encrypted data can be detected.This will ensure the confidentiality, and integrity of the data at rest.

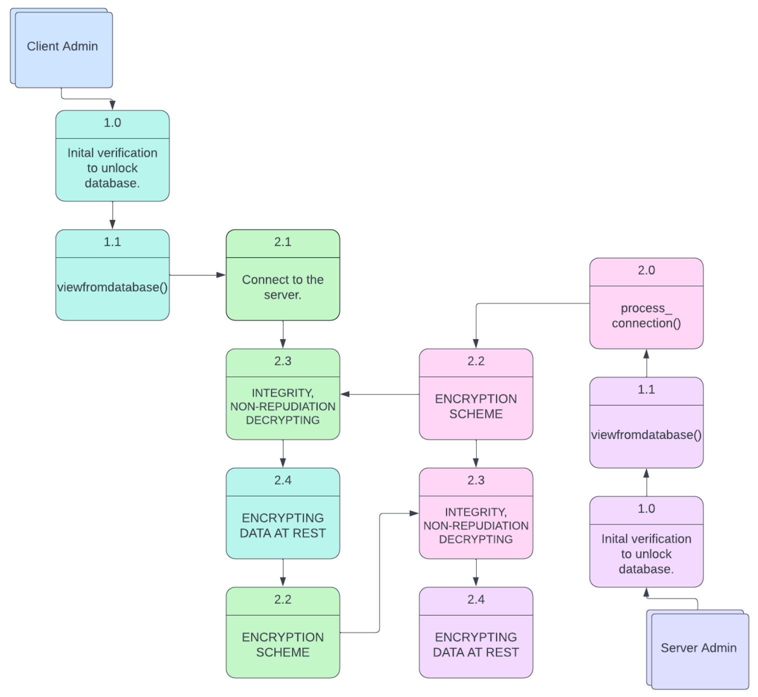
I will use the pysqlitecipher module to store the keys to decrypt the data at rest in an encrypted database.SQLCipher provides a secure AES encryption of SQlite database files. It is also password protected, ensuring that the database can only be accessed by authorised users. This increases the non-repudiation of the data if legible changes are made to it while it is at rest.

5. Illustration of the Proposed System

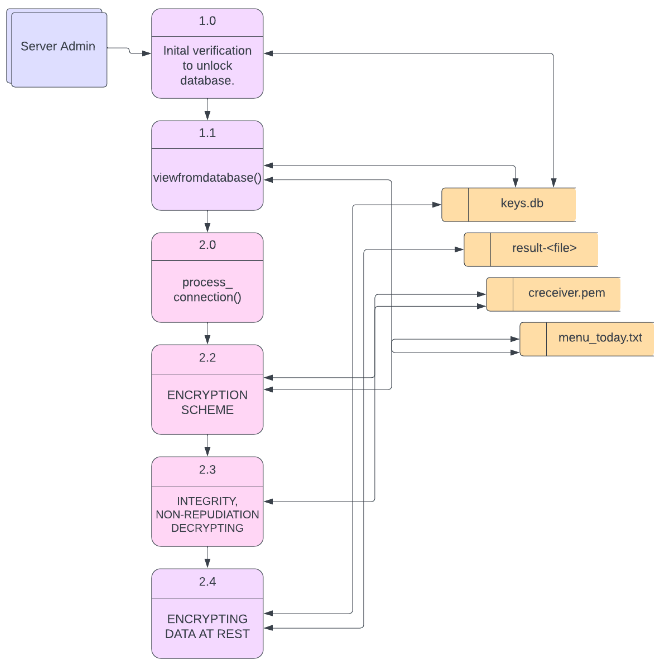
This system is heavily inspired by Pretty Good Privacy and will thus ensure encryption, nonrepudiation, integrity of the data (and authentication of the user), Note that the only text that will consistently be unencrypted in this system will be the public keys of the server and client.

5.1 A Visual representation of the applied system.

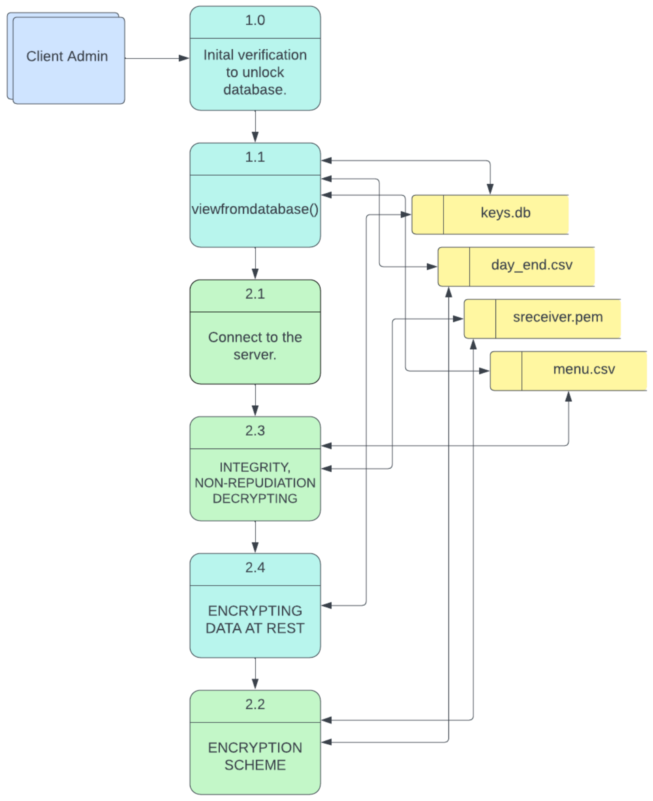
1. How the client script and server script interact



2. How data in the server script interacts with files containing data at rest.

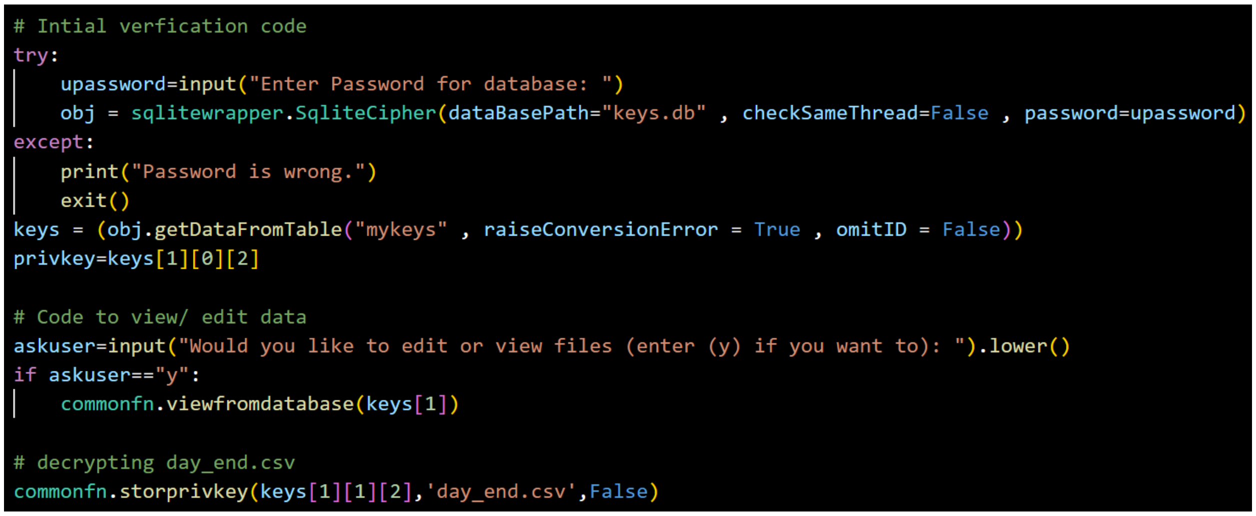


3. How data in the client script interacts with files containing data at rest.



5.2 Explanation of the processes/ functions.

Please refer to the diagrams in 5.1 to understand the headings in this section.



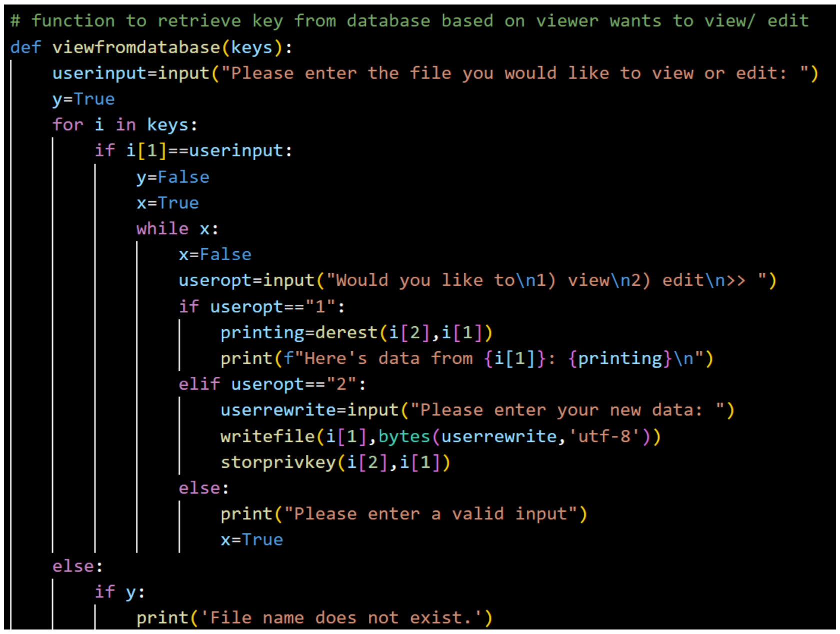
```python
# Intial verfication code
try:
    upassword=input("Enter Password for database: ")
    obj = sqlitewrapper.SqliteCipher(dataBasePath="keys.db" , checkSameThread=False , password=upassword)
except:
    print("Password is wrong.")
    exit()
keys = (obj.getDataFromTable("mykeys" , raiseConversionError = True , omitID = False))
privkey=keys[1][0][2]

# Code to view/ edit data
askuser=input("Would you like to edit or view files (enter (y) if you want to): ").lower()
if askuser=="y":
    commonfn.viewfromdatabase(keys[1])

# decrypting day_end.csv
commonfn.storprivkey(keys[1][1][2],'day_end.csv',False)
```

1.0 Initial verification to unlock the database.

(For both client and server) The program begins by asking for the password to unlock the encrypted pysqlitecipher database. This authenticates the user. If the password is wrong, the program stops running.

Purpose of 1.0

Initial authentication helps protect the confidentiality and integrity of the data because it ensures that only authorised users are viewing and editing the data.



```python
# function to retrieve key from database based on viewer wants to view/ edit
def viewfromdatabase(keys):
    userinput=input("Please enter the file you would like to view or edit: ")
    y=True
    for i in keys:
        if i[1]==userinput:
            y=False
            x=True
            while x:
                x=False
                useropt=input("Would you like to\n1) view\n2) edit\n>> ")
                if useropt=="1":
                    printing=derest(i[2],i[1])
                    print(f"Here's data from {i[1]}: {printing}\n")
                elif useropt=="2":
                    userrewrite=input("Please enter your new data: ")
                    writefile(i[1],bytes(userrewrite,'utf-8'))
                    storprivkey(i[2],i[1])
                else:
                    print("Please enter a valid input")
                    x=True
    else:
        if y:
            print('File name does not exist.')
```

1.1 viewfromdatabase()

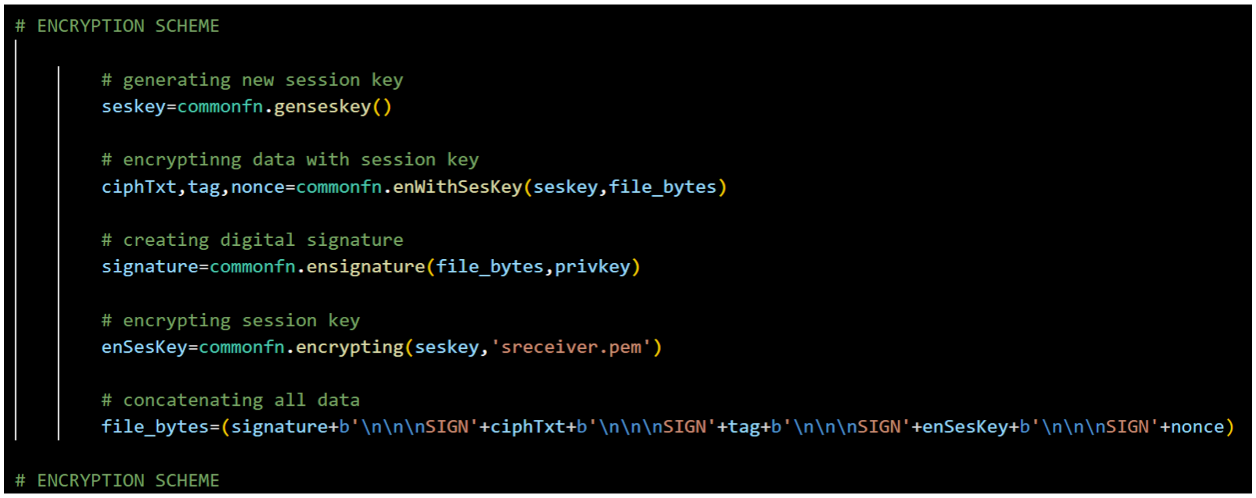
Before the server starts or the client connects with the server, the admin is given the opportunity to view or change the data of the existing files. This increases the functionality of the program. It should be noted that the actual files are never decrypted at this point. The encrypted data is read from the files and the relevant key from the database is retrieved and the unencrypted data is printed if the user chooses to view the data. Otherwise the previous data is overwritten by the new data encrypted with the relevant key if the user chooses to edit the data. After this function, the user's private key will be retrieved from the database.

2.0 process_connection()

The server starts. This function listens for commands from the client. If it receives "GET_MENU", it will unencrypt menu_today.txt and send its content (after being processed in 2.2) and send it to the client. Menu_today.txt will be encrypted again after the contents are sent. After that it will await the "CLOSING" command. When it receives the command, it will retrieve, check and store (process will be further explained in 2.2, 2.3 and 2.4) the data it receives from the client.

2.1 Connect to the server

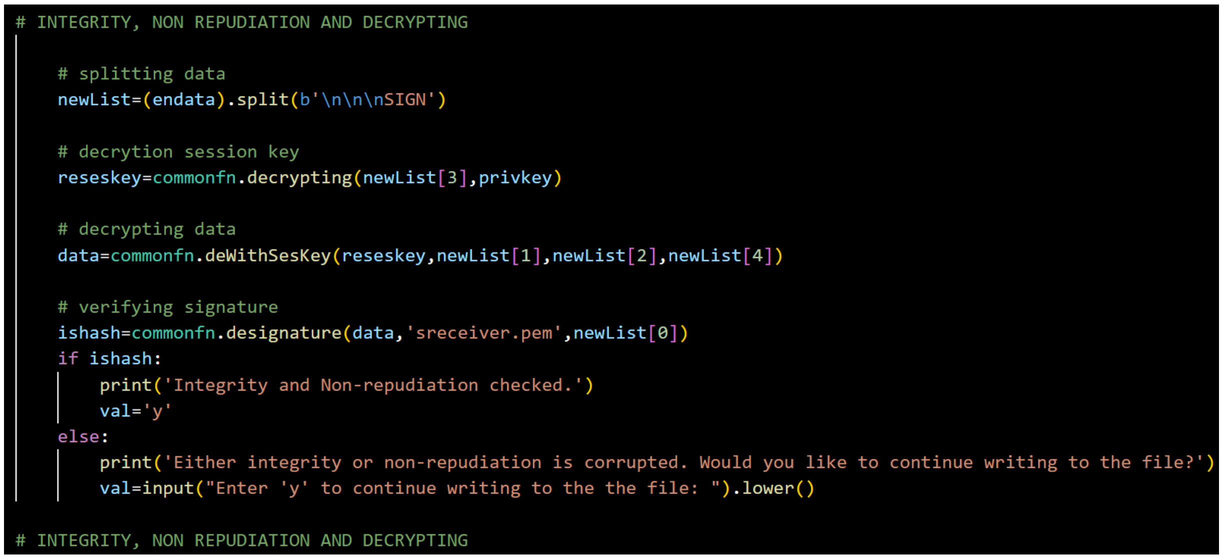
The day_end.csv file is unencrypted after the key is retrieved from the database. The client connects to the server and sends the "GET_MENU" command. After retrieving and checking and storing (process will be further explained in 2.2, 2.3 and 2.4) the data it receives from the server, it will send the "CLOSING" command and send the contents of day_end.csv (after being processed in 2.2 to be encrypted) to the server.



```
# ENCRYPTION SCHEME

        # generating new session key
        seskey=commonfn.genseskey()

        # encryptinng data with session key
        ciphTxt,tag,nonce=commonfn.enWithSesKey(seskey,file_bytes)

        # creating digital signature
        signature=commonfn.ensignature(file_bytes,privkey)

        # encrypting session key
        enSesKey=commonfn.encrypting(seskey,'sreceiver.pem')

        # concatenating all data
        file_bytes=(signature+b'\n\n\nSIGN'+ciphTxt+b'\n\n\nSIGN'+tag+b'\n\n\nSIGN'+enSesKey+b'\n\n\nSIGN'+nonce)

# ENCRYPTION SCHEME
```

2.2 Encryption scheme

Firstly a random session key will be generated. The bytes read from the file the machine intends to send (we will call this message) will be encrypted with the session key using AES in EAX mode. This will generate a nonce and message authentication tag along with the encrypted message. The session key is then encrypted with the receiver's public key (creceiver.pem - client's public key/ sreceiver.pem- server's public key) using PKCS#1 OAEP encryption scheme. Following this, a digital signature will be generated with PKCS#1 PSS, using the message and admin's private key. Then the digital signature, encrypted message, message authentication tag, encrypted session key and nonce will be concatenated and sent to the receiver. After this, the file which bytes were read will be encrypted again.



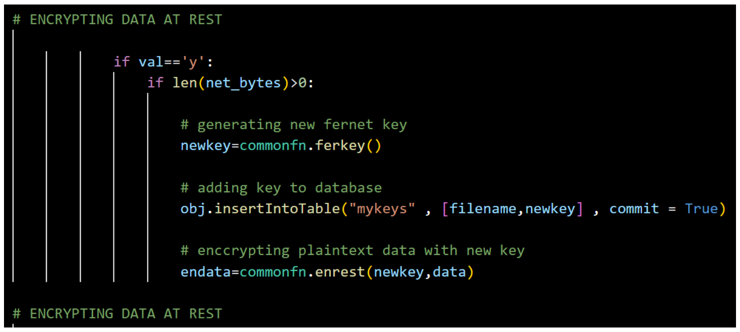
```
# INTEGRITY, NON REPUDIATION AND DECRYPTING

    # splitting data
    newList=(endata).split(b'\n\n\nSIGN')

    # decrytion session key
    reseskey=commonfn.decrypting(newList[3],privkey)

    # decrypting data
    data=commonfn.deWithSesKey(reseskey,newList[1],newList[2],newList[4])

    # verifying signature
    ishash=commonfn.designature(data,'sreceiver.pem',newList[0])
    if ishash:
        print('Integrity and Non-repudiation checked.')
        val='y'
    else:
        print('Either integrity or non-repudiation is corrupted. Would you like to continue writing to the file?')
        val=input("Enter 'y' to continue writing to the the file: ").lower()

# INTEGRITY, NON REPUDIATION AND DECRYPTING
```

2.3 Integrity, Non-repudiation and Decrypting

When the data from the sender is received, the data split again into the digital signature, encrypted message, message authentication tag, encrypted session key and nonce. The encrypted session key is first decrypted and the session key used to decrypt the message. Following this, the message is hashed and the digital signature is decrypted using the sender's public key and the signature's hash is retrieved. Both hashes are compared and if it passes the integrity check, the unencrypted data is written to the relevant file, otherwise, the user is prompted as to whether they continue writing the corrupted data to the file.

Purpose of 2.2 and 2.3

The confidentiality of the data is ensured through encryption because it prevents unauthorised access. The encrypted data can only be read by someone who has the correct session key which can be decrypted by the right decryption key (the receiver's private key).
The digital signature provides a mechanism to check the integrity of the data. The digital signature will no longer match the content if it is modified in any way making any manipulation detectable.
A digital signature can also be used to check the non-repudiation of the data. The digital signature cannot be forged, giving evidence that the data belongs to the sender. This makes it difficult for the part to dispute having originated or sent the data.



```
# ENCRYPTING DATA AT REST

                if val=='y':
                    if len(net_bytes)>0:

                        # generating new fernet key
                        newkey=commonfn.ferkey()

                        # adding key to database
                        obj.insertIntoTable("mykeys" , [filename,newkey] , commit = True)

                        # enccrypting plaintext data with new key
                        endata=commonfn.enrest(newkey,data)

# ENCRYPTING DATA AT REST
```

2.4 Encrypting Data at rest

A fernet key is generated and stored in the encrypted database. The key is used to encrypt the plain text data in the file. A new fernet key is generated for every new/ overwritten file.

Purpose of 2.4

Since each key is unique and can only be used to decode one particular file, this helps prevent unauthorised access to critical information, preserving the confidentiality of the data.
Fernet encryption also includes an integrity check value, which can be used to detect any changes made to the s data. If the integrity check fails, there will be an error message alerting the recipient the data has been tampered with. This ensures the integrity of the data.
By combining both of these aspects, fernet encryption protects the data from unauthorised modification and access.

6. Conclusion and Additional considerations

In conclusion, the risk analysis of the SPAM2 system highlighted several security threats that could threaten confidentiality, integrity and non-repudiation. These risks include unauthorised viewing of data, unauthorised modification of data and repudiating the data sent. Cryptographic solutions like encryption and digital signatures can help effectively manage these risks. Encryption protects the confidentiality of the data and digital signatures provide integrity and non-repudiation by verifying the authenticity of the sender and ensuring the data was not tampered with during transit. If the counter measures and proposed system is implemented the SPAM2 system can be fortified against possible attacks.

It is recommended to regularly review and update the security measures in place and make certain they defend against the risks mentioned above. An additional consideration taken for the proposed implementation was to apply confidentiality, integrity and non-repudiation to all the data (both menu_today.txt and day_end.csv) in the system. This is to enhance the security of the system.

7. Task Allocation

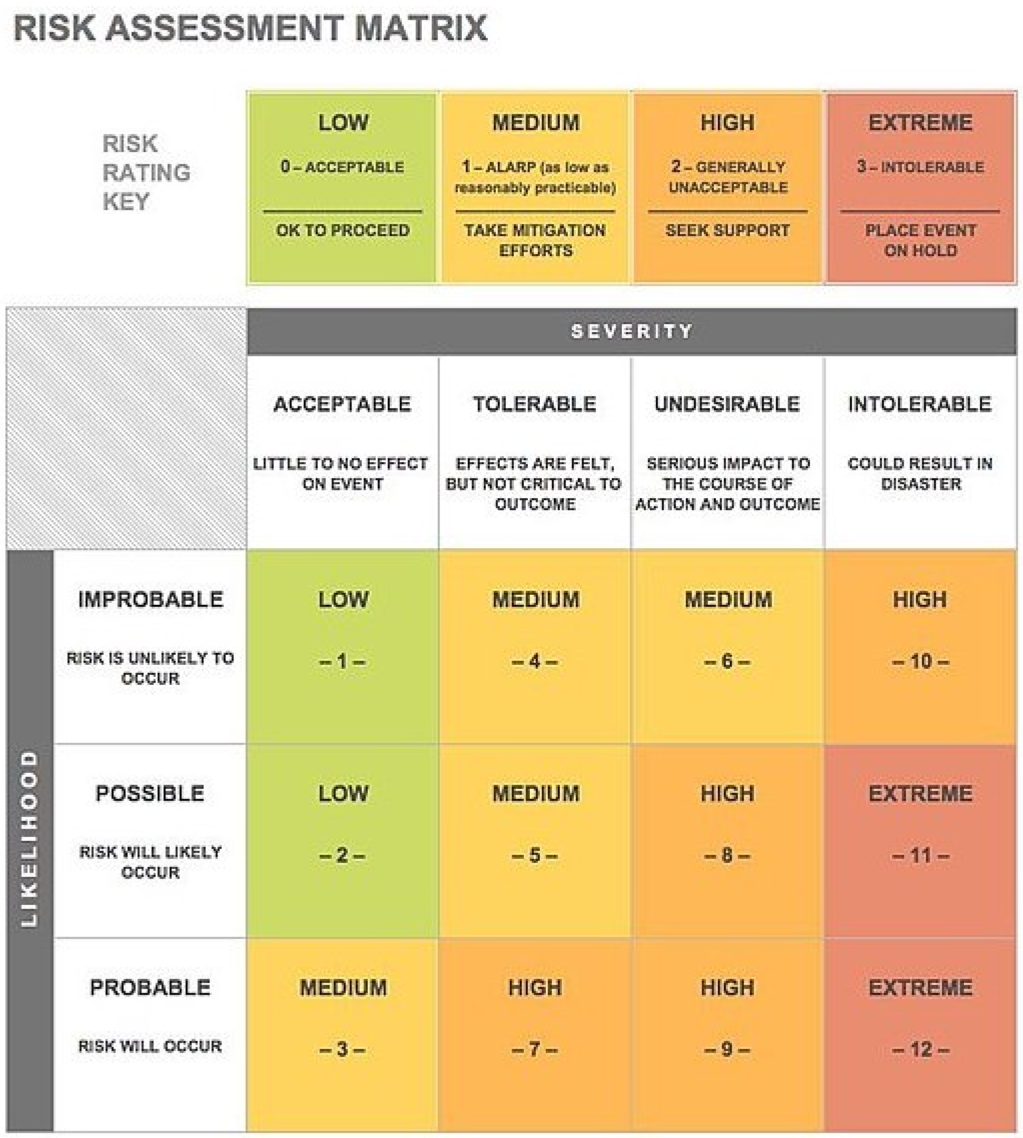
This entire assignment was done by Swaathi Lakshmanan.

8. Reflection

This assignment was daunting because I have never written large pieces of code that maintains the integrity, nonrepudiation and confidentiality of data. Furthermore, I was tackling this assignment alone and that made sure I can only rely on myself to complete this assignment. I started by securing the data in transit because I was familiar with PGP and intended to use its concepts in this assignment. Afterwards I had to solve the problem of data at rest. I did not think it would be reasonable to leave the keys of the encrypted data at rest in the client and server files without any protection. I tried consulting people in the IT industry and discovered the concept of using databases to store keys. So I started exploring cloud services and online databases. Eventually, I realised it was close to impossible to find free solutions to this problem. It was by incredible luck that I came across an article about SQLite and its python module when I was doing research. With SQLite, I was able to add an encrypted database to store the keys for the data at rest.

Due to the nature of the assignment, I had to force myself to leave my comfort zone and familiarise myself with coding cryptography scripts using python. I learnt many things through this assignment, like PKCS and how it differentiates from RSA, and how it can be more secure compared to RSA and how to create SQLite databases on python and encrypt them. This assignment gave me a glimpse into how professional applications may manage the security of data.

Appendix A- Simplified Application Risk Assessment Form



RISK ASSESSMENT MATRIX

RISK RATING KEY	LOW	MEDIUM	HIGH	EXTREME
	0 – ACCEPTABLE	1 – ALARP (as low as reasonably practicable)	2 – GENERALLY UNACCEPTABLE	3 – INTOLERABLE
	OK TO PROCEED	TAKE MITIGATION EFFORTS	SEEK SUPPORT	PLACE EVENT ON HOLD

		SEVERITY			
		ACCEPTABLE LITTLE TO NO EFFECT ON EVENT	TOLERABLE EFFECTS ARE FELT, BUT NOT CRITICAL TO OUTCOME	UNDESIRABLE SERIOUS IMPACT TO THE COURSE OF ACTION AND OUTCOME	INTOLERABLE COULD RESULT IN DISASTER
LIKELIHOOD	IMPROBABLE RISK IS UNLIKELY TO OCCUR	LOW – 1 –	MEDIUM – 4 –	MEDIUM – 6 –	HIGH – 10 –
	POSSIBLE RISK WILL LIKELY OCCUR	LOW – 2 –	MEDIUM – 5 –	HIGH – 8 –	EXTREME – 11 –
	PROBABLE RISK WILL OCCUR	MEDIUM – 3 –	HIGH – 7 –	HIGH – 9 –	EXTREME – 12 –

The image above is my reference for my evaluations in the following risk assessment form.

Risk	Which system does it apply to (Server or client)	Likelihood (e.g. High, Medium, Low)	Severity (e.g. High, Medium, Low)	Response Strategy (e.g. Avoid, Transfer, Mitigate or Accept)	Actions required
(Integrity, Confidentiality) Data at rest is not protected: ● Data vulnerable to unauthorised	Both	Probable	Intolerable	Mitigate	Data at rest should be encrypted.
		Risk: Extreme			

access and manipulation.					
(confidentiality) The data is not encrypted, so anyone can see the data: ● Data can be intercepted and viewed.	Both	Probable	Intolerable	Mitigate	Data should be encrypted before it is sent.
		Risk: Extreme			
(non-repudiation) There is no way to confirm who sent the data: ● You cannot verify whether the data is valid and the person who sent it will not be held accountable	Both	Probable	Undesirabl e	Mitigate	You can add a hash function to the system.
		Risk: High			
(integrity) You cannot confirm whether the data remains unchanged when it is received: ● Data can be intercepted and modified or tampered with.	Both	Probable	Intolerable	Mitigate	You can run the received data through the same hash function used in the digital signature and check whether the hash value is the same.
		Risk: Extreme			

References

Andrada Coos (n.d.). *How to Protect Your Data at Rest*. [online] Endpoint Protector Blog. Available at: https://www.endpointprotector.com/blog/how-to-protect-your-data-at-rest/#:~:text=Data%20at%20rest%20is%20at.

Katie (2018). *The CIA Triad: The key to Improving Your Information Security*. [online] Commissum. Available at: https://commissum.com/blog-articles/the-cia-triad-the-key-to-improving-your-information-security/.
Kaliski, B., Jonsson, J. and Rusch, A. (2016). PKCS #1: RSA Cryptography Specifications Version 2.2. doi:https://doi.org/10.17487/rfc8017.

Cryptography Stack Exchange. (n.d.). *Should I be using PKCS1 v1.5 or PSS for RSA signatures?* [online] Available at: https://crypto.stackexchange.com/questions/48407/should-i-be-using-pkcs1-v1-5-or-pss-for-rsa-signatures [Accessed 9 Feb. 2023].
Housley, R., Schaad, J. and Kaliski, B. (2005). *RFC 4055: Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. [online] IETF Datatracker. Available at: https://datatracker.ietf.org/doc/rfc4055/ [Accessed 9 Feb. 2023].
Ireland, D. (2021). *RSA signature and encryption schemes: RSA-PSS and RSA-OAEP*. [online] Cryptosys.net. Available at: https://www.cryptosys.net/pki/manpki/pki_rsaschemes.html.
N-able (2019). *Advanced Encryption Standard: Understanding AES 256*. [online] N-able. Available at: https://www.n-able.com/blog/aes-256-encryption-algorithm.
Cryptography.io. (2014). *Fernet (symmetric encryption) — Cryptography 2.9.dev1 documentation*. [online] Available at: https://cryptography.io/en/latest/fernet/.
event-organisation-guide.readthedocs.io. (n.d.). *Risk Assessment — Event Organisation Guide (SSI-EOG) 1.1.3 beta 1 documentation*. [online] Available at: https://event-organisation-guide.readthedocs.io/en/latest/eog/eps-risk-management-assessment.html [Accessed 9 Feb. 2023].
Data Exfiltration Via DNS And How to Combat It Webinar. (2020). Available at: https://genesis.swiss/wp-content/uploads/2020/04/DE_Webinar-Data-Exfiltration-200410-FINAL.pdf [Accessed 10 Feb. 2023].
www.wallarm.com. (n.d.). *What Is MITM (Man-in-the Middle) Attack Types and detection methods | Wallarm*. [online] Available at: https://www.wallarm.com/what/what-is-mitm-man-in-the-middle-attack.
www.extrahop.com. (n.d.). *How Ransomware Works and How to Prevent It | ExtraHop*. [online] Available at: https://www.extrahop.com/company/blog/2020/ransomware-explanation-and-prevention/.
www.wallarm.com. (n.d.). *What is phishing attacks Types and business impact | Wallarm*. [online] Available at: https://www.wallarm.com/WHAT/TYPES-OF-PHISHING-ATTACKS-AND-BUSINESS-IMPACT.
GeeksforGeeks. (2022). *What are Session Replay Attacks?* [online] Available at: https://www.geeksforgeeks.org/what-are-session-replay-attacks/ [Accessed 10 Feb. 2023].