# **ACG Proposal**

A Proposal on Securing SPAM2 with Python

Module Name: Applied Cryptography

Module Code: IT2504

Class: DISM/FT/1B/01

Team Members:

Andrew Dillon Poh Jie Hao (2227168)

Alden Chia Yu Xiang (2205584)

Ng Ye Kai (2227436)

# 

# Table of Contents

[**1. Current SPAM System and Implementation**](#_nzjc51apt92p) **1**

[**2. Assumptions**](#_vbvolt6vrfid) **1**

[**3. Attack Scenarios and Countermeasures**](#_6fbw0cohdseo) **2**

[3.1 Types of attackers](#_mc1wy6x83m4w) 2

[3.2 Types of attacks](#_7ir8zaq716m9) 3

[**4. Proposed System**](#_r59e1z19d1i6) **3**

[4.1 Key management](#_wuupw03k076b) 4

[4.2 Security Implementation](#_ku6toodpk21) 6

[4.3 How the proposed system works](#_bpa9mwajh5a7) 6

[**5. Conclusion and additional considerations**](#_eri57bbfv0ms) **9**

[**6. Task Allocation**](#_yvi29hukmnbe) **10**

[**7. References**](#_b41ditg223d9) **11**

[**Individual Reflections**](#_g4mho9oxedws) **11**

[**Appendix**](#_7o63m9atij9g) **13**

# 1. Current SPAM System and Implementation

The current SPAM system consists of a server and a client. The server sends the client the menu-of-the-day while the client sends the server the day-closing information. However, the connection between the server and the client is currently not protected by any security features. It is stated that the system is using public WiFi such as Wireless@SG, increasing the risk of an attack since anyone can connect to the WiFi and intercept the data. Hence, our team is going to protect the data-in-transit between client and server to prevent any form of interception. This protection is in terms of Confidentiality, Integrity and Non-repudiation.

Figure 1.1 below illustrates the current SPAM2 system without any security features.

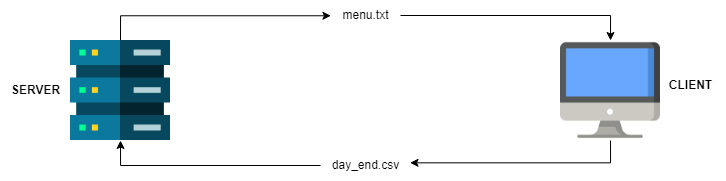


Figure 1.1

# 2. Assumptions

Assumption 1: The public keys for client and server are publicly available. This means that the client possesses the server’s public key and the server possesses the client’s public key

Assumption 2: Attackers are unable to edit or view the scripts created or the scripts created are stored securely.

Assumption 3: The server physical machine is well protected and hence the data files apart from the private key stored there do not need encryption when stored.

# 3. Attack Scenarios and Countermeasures

Solutions will be stated very briefly in this section. For more details, please refer to the Proposed System section.

## 3.1 Types of attackers

There are 2 main types of attackers who would attack the SPAM2 system.

### 

### Competitors

Other food and beverage businesses may be in direct competition with this business. Due to financial motivations, they may try to attack the company in order to gain a competitive advantage. This could be done by altering the menu sent from the server to the client which would cause the client to prepare the wrong dishes for the day, hence resulting in customers being displeased and thus going to the competitor’s outlet to eat instead. Competitors may also try to intercept the day\_end file sent and read the sales for the day so they know the performance of the businesses as well as which items sell well.

### Script Kiddies

Since the client is connected via WiFi, there may be hackers who happen to be looking out on the WiFi for any unprotected connections. These hackers may not have any concrete motivations apart from simply wanting to test their skills or impress their friends. As such, they may try to hack into the client’s connection and read and change data being transmitted and received, or masquerade as the client to send false information to the server vice versa.

## 3.2 Types of attacks

### Man-In-The-Middle attack (MITM)

When data is being transferred over from the client or server, a threat actor is able to position himself in the middle of conversation, which is also known as man-in-the-middle attack (MITM). A MITM is a type of eavesdropping attack in which the threat actor intercepts and relays messages secretly between two parties who believe they are communicating directly with each other. This can result in the attacker controlling the entire conversation, imitating to be the sender throughout. MITM is dangerous as threat actors can manipulate account credentials, steal funds, and in this context, edit the file content or send a completely different file that might contain malware. The company may face severe data loss if a successful MITM attack was launched.

As MITM attacks threaten the data-in-transit in terms of confidentiality, integrity and non-repudiation, we will be implementing encryption using AES, message authentication using HMAC and digital signatures for data-in-transit.

### Insider Threats

An insider refers to anyone within the organization. These insiders may be able to use social engineering skills to gain access to the physical computers. With direct access to the computers, they may be able to view and edit data-in-storage. This means attackers would be able to perform actions on the menu and day\_end files without while they are stored if the hardware is not protected.

As insider threats threaten the data-in-storage, we will be applying encryption for private keys which are sensitive as they are used to decrypt data received.

### Denial-of-Service (DOS)

DOS attacks refer to attacks that send a large amount of requests to prevent a machine from processing legitimate requests. An attacker may spam the server with an overwhelming number of requests which will cause them to be unable to function. This may cause the server to not respond to client machines, which means the clients are unable to receive the menu of the day which would be detrimental as they would not know what dishes to prepare.

In order to prevent DOS attacks, the server can be set up to completely ignore any messages that are not properly authenticated using HMAC or a digital signature.

# 4. Proposed System

With the possible attacks that different threat actors can launch against the SPAM2 system, it is important to implement security features that protect data-in-transit from server to client and vice versa. Sensitive data-in-storage such as private keys are also protected.

In the proposed system, both the server and client will function as per usual. The only differences are the security features that we have implemented such as encryption algorithms, HMAC and digital signatures. We believe that the security of the system will be further enhanced with the implementation of these features in 4 main aspects of security: Confidentiality, Integrity, Non-repudiation and Authenticity. We also ensured that the established connection between both the server and client are secure using the socket service for them to exchange files without any risk of attacks.

We are using AES and RSA for symmetric and asymmetric algorithms, RSA for the digital signature algorithm and lastly, SHA-256 and SHA-1 for the hashing algorithms.

## 4.1 Key management

### Session key

At the start of every session, a new AES session key will be generated. The session key is a randomly generated 16 bytes which is also hashed for certain operations. The advantage of a session key is that the key will be discarded after every session, reducing the likelihood of an attacker finding out the session key since it constantly changes. Furthermore, since session keys are symmetric, they allow for symmetric encryption and decryption which is computationally faster than asymmetric encryption and decryption. In the case of SPAM, this session key will be used in the session later on for AES encryption/decryption and HMAC generation/verification.

### RSA keys

An RSA key pair (private key and public key) has also already been created for both the server and client each. Each of the private keys are kept within their respective owner’s directories while both the public keys are shared to both client and server as we have made the assumption that the public keys are publicly available. Generating the RSA key pair in python is achieved by first generating a new RSA keypair object, then deriving a private and public key from the keypair object. In order to ensure that the private keys of both the client and server are confidential from outsiders in storage, they have been encrypted with a passphrase. The private keys are then decrypted when they need to be used. This RSA key pair is used to encrypt the session key when it is sent from the server to the client for confidentiality reasons. It is also used in the creation and verification of digital signatures for integrity reasons.

## 4.2 Security Implementation

Although we were only required to provide integrity for the menu, as well as confidentiality and non-repudiation for the day\_end file, we decided to implement measures to provide integrity, confidentiality and non-repudiation for both the menu and day\_end files.

### Confidentiality via encryption using AES

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext.

When a message is encrypted using AES, attackers will be unable to get the original message from the ciphertext even if they intercept the transmission unless they have the key that was used to encrypt it. This allows for messages to remain confidential no matter what unless the AES key is leaked. If not implemented, all messages sent would be entirely readable to attackers without proper authorization methods once they intercept the given message.

Thus we coded AES encryption and decryption into the Python script so that whenever a message is to be sent, it is encrypted with AES and a randomly generated session key and will be decrypted when it is received with the same session key. This will help the system maintain confidentiality of information even over insecure networks, preventing important information such as the day\_end and menu files from being leaked.

### Checking message Integrity with HMAC

HMAC, or hash-based message authentication code, involves a cryptographic hash function and a secret key. The message and its HMAC is sent from sender to receiver for the receiver to regenerate the HMAC and verify if the HMAC values match. The main purpose of HMAC is to verify the integrity and authenticity of a message. Without HMAC, if the data is altered in transit, the receiving party would not be aware that the data is incorrect.

In our security implementation, the HMAC function utilizes the SHA-1 hash function and a session key that is exchanged at the start of the session. It takes a message input from the sender and creates an output that has been hashed and secured by the session key via the HMAC function. The sender then sends the message and HMAC to the receiver, who then regenerates the HMAC using the sent message and the session key. The receiver then checks if both HMAC values are the same. If they are the same, the message has not been altered. If they are not the same, the message may have been accidentally or intentionally altered in transit. This is resistant to a man-in-the-middle attack as if the attacker changed the message content, they would need the session key to recreate the HMAC such that the receiver would be able to verify the HMAC.

This will help the SPAM2 system protect the integrity of the day\_end and menu files which is important as if the files are altered, the server or client may receive incorrect data without knowing. This could cause the client to prepare incorrect dishes for the day since the menu is incorrect or affect the data about dishes served in a day which the server will store.

### Ensuring Non-repudiation using Digital Signature

Digital signature is a type of electronic signature that is used to validate the authenticity and integrity of the message. Most importantly, it ensures non-repudiation of the message and sender. To produce a digital signature, the sender will need to hash the message then encrypt the hash using his own private key. He will then send the original message and the digital signature to the receiver. To check the non-repudiation of the message, the receiver will have to compare the digest produced from hashing (using the same algorithm) the original message and the digest after decrypting the digital signature using the sender’s public signature.

Digital signature ensures that the message received originates from the signer (sender) himself and not from another third party. Without digital signature, the receiver might receive a tampered message or even a message that does not originate from the written sender. This could lead to major issues like spreading of malwares and man-in-the-middle attacks.

We have implemented the usage of digital signature in the updated SPAM2 system. For the SPAM2 system, both the server and client will be producing their own digital signature for the files to be sent. The digital signature created will then be transmitted over to the other party to verify the authenticity of the file. This will greatly benefit the SPAM2 system’s security as it will prevent any man-in-the-middle attacks that may tamper with the contents in the file.

### Encryption of sensitive stored data

The client and server private keys are encrypted when stored, hence attackers are unable to see the plaintext keys. Additionally, the day\_end.csv file stored on the client is encrypted. This is because the day\_end file has been highlighted to require confidentiality in the brief and the client machine is more vulnerable to unauthorized physical access by an attacker. Hence the day\_end file requires a password to be decrypted and viewed.

## 4.3 How the proposed system works

The server first creates a socket to connect to the client.

## *From Server to Client*

### Encrypted session key exchange

Upon server startup, a session key is randomly generated by the server.

The session key is encrypted using RSA with the client’s public key. This ensures that only the client is able to decrypt the session key with their client private key.

The session key is then sent to the client, which decrypts the session key using their public key.

Both parties now have the session key.

### Server sends the digital signature

Next, the server generates a digital signature by hashing the menu.txt file and encrypting the hash with the server private key.

This server digital signature is then sent to the client.

### Server sends the encrypted menu

Using AES encryption, the server encrypts the menu.txt file with the session key.

This encrypted menu file is then sent to the client, which decrypts the file using the session key.

The menu has thus been sent with confidentiality.

### Server sends the HMAC of the menu

The server generates a HMAC of the encrypted menu.txt file with the session key and SHA-1 as the hash function.

This HMAC is then sent to the client.

### Client verifies digital signature

The client verifies the digital signature that the server sent using the server public key and menu.txt. This is done by decrypting the digital signature and hashing the menu.txt file, then comparing the outputs. If the outputs are the same, the signature is valid. If the outputs are different, the signature is not valid and the user is prompted with the option to terminate connection.

The menu has thus been sent with non-repudiation as only the server public key would be able to decrypt the signature that has been encrypted with the server private key.

### Client verifies HMAC

The client verifies the HMAC that the server sent by regenerating the HMAC of the file using the menu.txt sent by the server and the session key. If the outputs are the same, the HMAC is valid. If the outputs are different, the user is prompted with the option to terminate connection.

The menu has then been proven to maintain its integrity as a change in the menu in transit would have resulted in a completely different HMAC. The HMAC also cannot be altered due to the use of a session key.

With the above steps, the menu has been successfully sent from the server to client and has been secured in terms of Confidentiality (AES), Integrity (HMAC) and Non-Repudiation and Authentication (Digital Signature).

## *From Client to Server*

### Client sends the digital signature

Now, the client generates a digital signature by hashing the day\_end.csv file and encrypting the hash with the client's private key.

This client digital signature is then sent to the server.

### Client sends the encrypted day\_end file

Using AES encryption, the client encrypts the day\_end.csv file with the session key.

This encrypted file is then sent to the server, which decrypts the file using the session key.

The day\_end information has thus been sent with confidentiality.

### Client sends the HMAC of the day\_end file

The client generates a HMAC of the encrypted day\_end.csv file with the session key and SHA-1 as the hash function.

This HMAC is then sent to the server.

### Server verifies digital signature

The server verifies the digital signature that the client sent using the client public key and day\_end.csv. This is done by decrypting the digital signature and hashing the day\_end.csv file, then comparing the outputs. If the outputs are the same, the signature is valid. If the outputs are different, the signature is not valid and the connection is terminated.

The day\_end.csv file has thus been sent with non-repudiation as only the client public key would be able to decrypt the signature that has been encrypted with the client private key.

### Server verifies HMAC

The server verifies the HMAC that the client sent by regenerating the HMAC of the file using the day\_end.csv sent by the client and the session key. If the outputs are the same, the HMAC is valid. If the outputs are different, HMAC is not valid and the connection is terminated.

The day\_end file has thus been proven to maintain its integrity as a change in the day\_end file in transit would have resulted in a completely different HMAC. The HMAC also cannot be altered due to the use of a session key. Connection between client and server ends.

With the above steps, the day\_end file has been successfully sent from the client to server and been secured in terms of Confidentiality (AES), Integrity (HMAC) and Non-Repudiation and Authentication (Digital Signature)

Figure 3.1 below illustrates the proposed solution

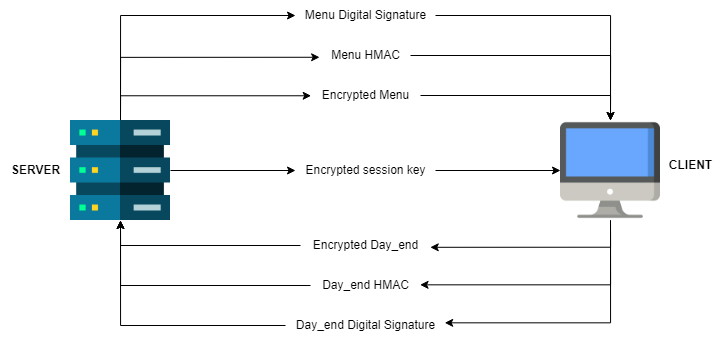


Figure 3.1

# 5. Conclusion and additional considerations

**Conclusion**

In conclusion, the SPAM2 system is used for data transmission between the server and the client. The server will receive the day end sales data from the client while the client will receive the menu of the day from the server. Without the security features implemented, SPAM2 will be vulnerable to many different types of cyber attacks, especially man-in-the-middle attack. Therefore, we have implemented these security features to combat these cyber threats. We believe that by using hashing and encryption algorithms, digital signatures, and HMAC, SPAM2’s vulnerability to attacks will be greatly reduced. All these implementations will ensure that the data of the files are not tampered with by unauthorized users while it is being transferred. All these features help provide confidentiality, integrity and non-repudiation for the SPAM2 system. With such security features enforced on the SPAM2 system, we believe that it will strengthen the system and the likelihood of successful attacks on the system will be reduced greatly.

**Additional Considerations**

Back-up

In the event that the main server fails, there will no longer be any server for the client to make connections. This will result in the client not receiving the menu of the day file and the server not receiving the day end file. We recommend the SPAM2 system to have a backup server for disaster recovery. This backup server will be the exact same as the original server, with all the security features implemented. Even though this might seem redundant, it is definitely worth the redundancy to ensure the continuity of the business as a whole.

Use of Wi-Fi

The use of public Wi-Fi, like Wireless@SG is a huge security concern. Using public Wi-Fi to transmit data is prone to many types of eavesdropping cyber attack such as replay attack and man-in-the-middle attacks. The user may also not know if the public Wi-Fi is safe as it might be an evil twin stealing data. Thus, the management should consider using private networks instead. Private networks will usually be configured with security configurations like WPA2 for encryption, have an authentication process and MAC filtering. These configurations will definitely help out in the security aspect for the SPAM2 system when they are transferring data.

# 6. Task Allocation

Alden:

1. Current Spam Implementation

2. Assumptions

4.2 Security Implementation

Andrew:

3.1 Attack Scenarios and countermeasures

4.3 How the proposed system works

7. References

Ye Kai:

4.1 Key Management

5. Conclusion

Risk Assessment form

Check contributions.txt for information on code allocation

# 7. References

Fruhlinger, J. (2020) *What is PKI? and how it secures just about everything online*, *CSO Online*. CSO. Available at: https://www.csoonline.com/article/3400836/what-is-pki-and-how-it-secures-just-about-everything-online.html (Accessed: February 10, 2023).

*HMAC - keyed-hashing for message authentication* (no date) *Python documentation*. Available at: https://docs.python.org/3/library/hmac.html (Accessed: February 10, 2023).

*PKCS#1 v1.5 (RSA)¶* (no date) *PKCS#1 v1.5 (RSA) - PyCryptodome 3.17.0 documentation*. Available at: https://pycryptodome.readthedocs.io/en/latest/src/signature/pkcs1\_v1\_5.html (Accessed: February 10, 2023).

*PYMOTW* (no date) *hmac – Cryptographic signature and verification of messages. - Python Module of the Week*. Available at: http://pymotw.com/2/hmac/ (Accessed: February 10, 2023).

*Quick programming tips* (no date) *AES 256 Encryption and Decryption in Python*. Available at: https://www.quickprogrammingtips.com/python/aes-256-encryption-and-decryption-in-python.html (Accessed: February 10, 2023).

Venkatesh PanabakaVenkatesh Panabaka 2 *et al.* (1963) *How to use SHA256-HMAC in python code?*, *Stack Overflow*. Available at: https://stackoverflow.com/questions/39767297/how-to-use-sha256-hmac-in-python-code (Accessed: February 10, 2023).

# Individual Reflections

Andrew:

This assignment presented many challenges. First off, while I had learnt the various cryptography concepts before, I was not confident in my knowledge and translating it to code was tricky. Some of us were weaker than others in coding and hence debugging was a frequent problem. At times the errors seemed impossible to fix which seemed daunting to us as many of these errors were unfamiliar to us. This was the first time I actually had to create sockets to send data from a client to server, which was a steep learning curve and resulted in many of the errors I faced. Furthermore, even when the code worked on our individual computers, combining them all into one machine created a whole new set of errors. I spent entire days looking at the code and trying to fix the various errors. The entire coding process was very tiring and I felt exhausted by the end of it.

Despite the challenges, I must say I really enjoyed this assignment. The challenge of having to learn completely new concepts and translating it into practice was fun as I have not been challenged this way in a long time. The constant debugging was tiring, yet rewarding when the solution was finally discovered. Even though the class was split into groups, we found ourselves all working together to try and help fix one another’s code. I found myself going around teaching others the cryptography concepts and debugging other groups’ programs while I also received advice from friends who had faced similar errors. It was also fun discussing the various security features we thought were necessary. One interesting discussion we had was about key exchange and how we should settle the public keys. Eventually we settled on publicly available public keys, but the discussion was still insightful and helped strengthen my knowledge of cryptography.

While the assignment was stressful, I definitely think it was one of the more challenging and fun assignments I’ve had to do. Working together with others and struggling until we found a viable solution was a good experience. I also definitely fortified my understanding of digital signatures, AES encryption, HMAC and symmetric/asymmetric encryption along with key management.

Ye Kai:

This assignment was very fun and interesting. I learnt a lot from doing it such as how to use AES encryption and decryption in Python, the importance of properly securing keys for security purposes, how RSA encryption and digital signing works and more. I also learned about how most secure systems work as well as the difference between AES and RSA encryption for real-life purposes. However I had to suffer a lot for days to learn these things which was very tiring. I hope that the future projects I do will be easier and just as rewarding.

Alden:

This assignment seems pretty manageable at first but as I dive deeper, it is actually harder than I thought. Even though I am pretty good at normal coding, this assignment has broadened my horizon in terms of python libraries. I was actually shocked when I first look at the code provided by the assignment as I do not understand a single thing. After countless hour of researching and having my friends explain the codes to me, I eventually knew how it works.

My contribution for the coding aspect is the digital signature. I thought that it would be really hard and time consuming to code something as complicated as digital signature. But after reading the lecture notes to understand how digital signatures work and watching some youtube videos, I found it pretty manageable. There are also countless online materials that guided and helped me to complete my code. Even though I only did the digital signature aspect, I also learnt more about AES and RSA encryption and HMAC from my other teammate. They were very enthusiastic about the code and helped me alot too. Without the help of my groupmate and classmates, I doubt I would even finish the code.

Even though this assignment is pretty tough, I definitely enjoy the challenge and the fact that our classmates helped one another. This assignment also definitely did force me to revise for digital signature, HMAC, and AES, RSA encryption before the final examinations.

# Appendix

