

Network Security

Laboratory Assignment 2

Group 08

Yanxiang Du

Yongjie Hao

Siwei Zhang

Spring 2024

2024-03-14

Exercise 1. Traffic Interception in Local Area Networks

Question 1: Check the common protocols that we use on the internet (or parts thereof), i.e. HTTP, HTTPS, ftp, telnet, ssh, pop3, imap, smtp Which of these are in clear text and which support some form of encryption. If you check when these protocols were defined/put into use can you see a trend developing over time with respect to cryptography?

In clear text:

HTTP(Hypertext Transfer Protocol) is a stateless protocol. It does not save any information about previous requests. It is sent in plain text without encryption or security mechanisms. HTTP uses a set of standardized messages to facilitate communication between clients and servers.

FTP(File Transfer Protocol) is used for the transfer of computer files from a server to a client on a computer network. Users can authenticate through plain text login protocol, usually using a user username and password, but users can also connect anonymously if the server allows.

Telnet(teletype network) by default, does not encrypt any data sent over the connection (including passwords). Telnet is a client/server application protocol, which can access the virtual terminal of a remote system on LAN or Internet.

POP3(Post Office Protocol version 3) POP is an internet protocol used by email clients to fetch emails from a server. Initially, it supported only unencrypted login or Berkeley .rhosts. Today, it provides multiple authentication methods for email security.

IMAP(Internet Message Access Protocol) is an application layer protocol used to enable an email client to retrieve emails from a remote mail server. By default, it sends login information from the client to the server without encryption, meaning usernames and passwords are transmitted in clear text.

Support some form of encryption:

HTTPS(Hypertext Transfer Protocol Secure) uses encryption for secure communication over a computer network and is widely used on the Internet. HTTPS encrypts the communication protocol using either Transport Layer Security (TLS) or its predecessor, Secure Sockets Layer (SSL).

SSH(Secure Shell Protocol) is a cryptographic network protocol for operating network services securely over an unsecured network. It employs public-key cryptography to authenticate both the remote computer and the user when required.

Both in cleartext and support encryption:

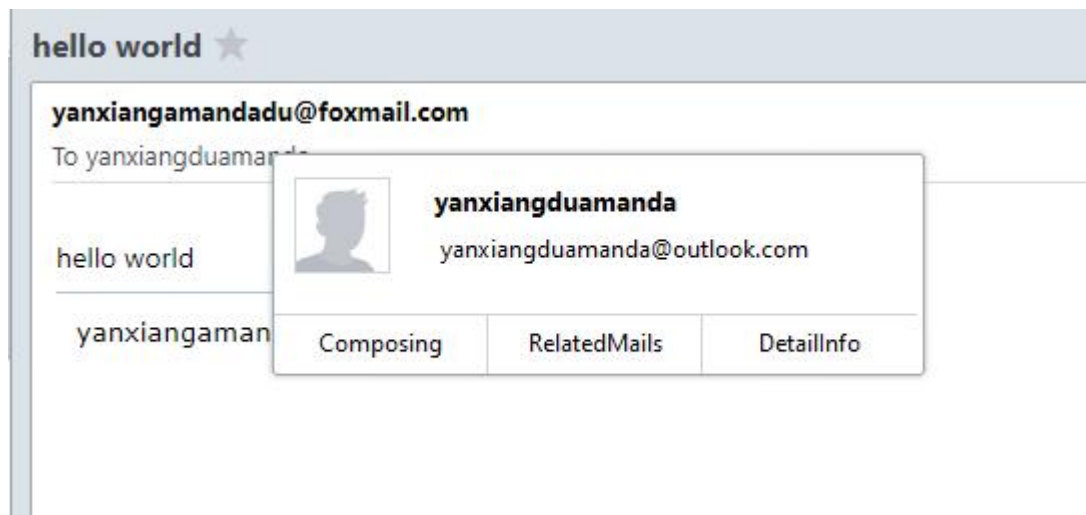
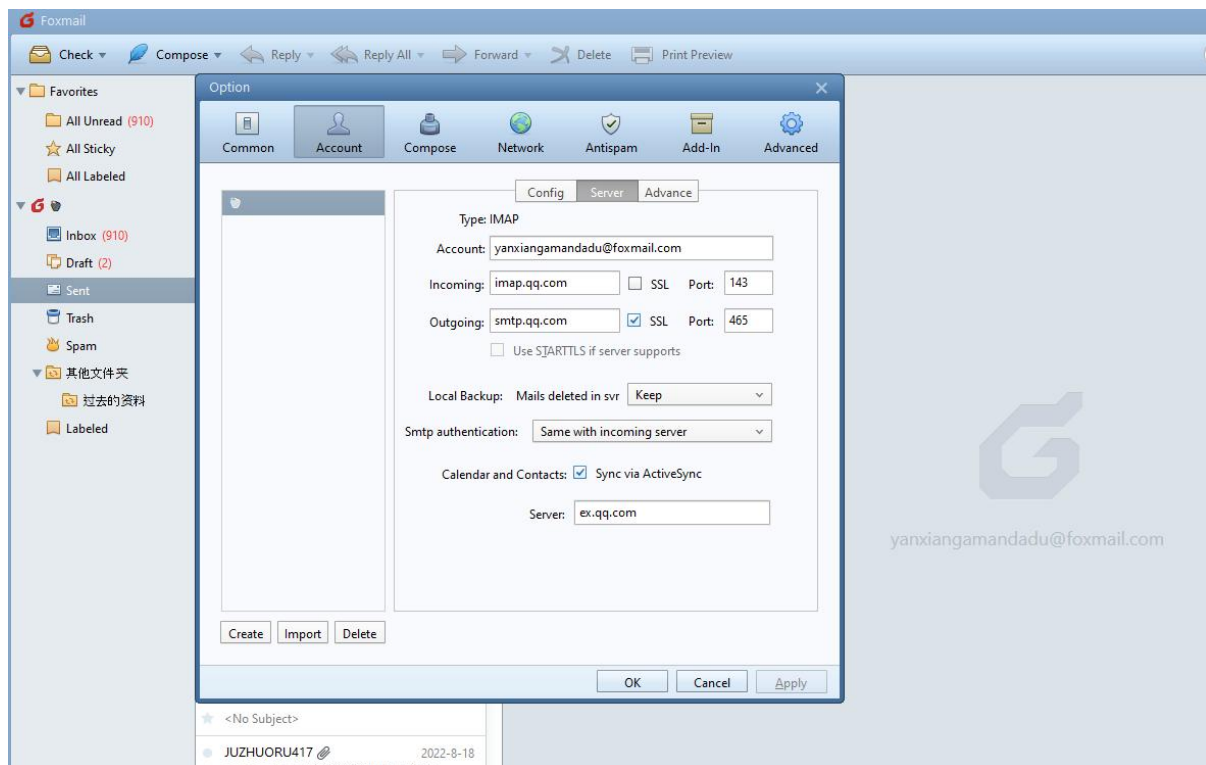
SMTP(Simple Mail Transfer Protocol) is a widely used Internet protocol for sending and receiving email messages. It operates on port 25 for plaintext and 587 for encrypted communication, using the Transmission Control Protocol.

a trend developing over time concerning cryptography:

Protocols that support encryption, such as HTTPS, SSH, and SMTPS, appear later than cleartext protocols. This trend reflects that people tend to use newer protocols/new versions of protocols that support encryption to enhance security.

Exercise 2. Symmetric Encryption – Block Ciphers

Question 2: If you have an email client that connects to an SMTP-server directly (i.e., not a webmail service) try and capture the traffic with the SMTP-server as you're sending an email (using Wireshark). Were you successful, why/why not?



Capturing from Ethernet (port 465)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.11.200.115	43.129.255.54	TCP	66	19250 → 465 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.276419	43.129.255.54	10.11.200.115	TCP	66	465 → 19250 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1424 SACK_PERM WS=128
3	0.000091	10.11.200.115	43.129.255.54	TCP	54	19250 → 465 [ACK] Seq=1 Ack=1 Win=263424 Len=0
4	0.001012	10.11.200.115	43.129.255.54	TLShv1.2	369	Client Hello
5	0.276727	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=1 Ack=316 Win=64128 Len=0
6	0.000614	43.129.255.54	10.11.200.115	TLShv1.2	1478	Server Hello
7	0.000000	43.129.255.54	10.11.200.115	TCP	1478	465 → 19250 [ACK] Seq=1425 Ack=316 Win=64128 Len=1424 [TCP segment of a reassembled PDU]
8	0.000000	43.129.255.54	10.11.200.115	TLShv1.2	1478	Certificate
9	0.000000	43.129.255.54	10.11.200.115	TLShv1.2	217	Server Key Exchange, Server Hello Done
10	0.000073	10.11.200.115	43.129.255.54	TCP	54	19250 → 465 [ACK] Seq=316 Ack=4436 Win=263424 Len=0
11	0.004813	10.11.200.115	43.129.255.54	TLShv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.276601	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4436 Ack=442 Win=64128 Len=0
13	0.000000	43.129.255.54	10.11.200.115	TLShv1.2	105	Change Cipher Spec, Encrypted Handshake Message
14	0.007915	43.129.255.54	10.11.200.115	TLShv1.2	148	Application Data
15	0.000046	10.11.200.115	43.129.255.54	TCP	54	19250 → 465 [ACK] Seq=442 Ack=4581 Win=263168 Len=0
16	0.000979	10.11.200.115	43.129.255.54	TLShv1.2	95	Application Data
17	0.276735	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4581 Ack=483 Win=64128 Len=0
18	0.002273	43.129.255.54	10.11.200.115	TLShv1.2	254	Application Data
19	0.000436	10.11.200.115	43.129.255.54	TLShv1.2	478	Application Data
20	0.317089	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4781 Ack=907 Win=64128 Len=0
21	0.070482	43.129.255.54	10.11.200.115	TLShv1.2	103	Application Data
22	0.006079	10.11.200.115	43.129.255.54	TLShv1.2	136	Application Data
23	0.276579	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4830 Ack=989 Win=64128 Len=0
24	0.037500	43.129.255.54	10.11.200.115	TLShv1.2	91	Application Data
25	0.000184	10.11.200.115	43.129.255.54	TLShv1.2	124	Application Data
26	0.276615	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4867 Ack=1059 Win=64128 Len=0
27	0.059709	43.129.255.54	10.11.200.115	TLShv1.2	91	Application Data
28	0.000602	10.11.200.115	43.129.255.54	TLShv1.2	89	Application Data
29	0.276585	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4904 Ack=1094 Win=64128 Len=0
30	0.002283	43.129.255.54	10.11.200.115	TLShv1.2	121	Application Data
31	0.000214	10.11.200.115	43.129.255.54	TLShv1.2	537	Application Data
32	0.276712	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4971 Ack=1577 Win=64128 Len=0
33	0.000044	10.11.200.115	43.129.255.54	TLShv1.2	1009	Application Data, Application Data
34	0.317637	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=4971 Ack=2592 Win=64128 Len=0
35	0.179804	43.129.255.54	10.11.200.115	TLShv1.2	103	Application Data
36	0.001547	10.11.200.115	43.129.255.54	TLShv1.2	89	Application Data
37	0.276776	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [ACK] Seq=5020 Ack=2627 Win=64128 Len=0
38	0.001520	43.129.255.54	10.11.200.115	TLShv1.2	93	Application Data
39	0.000000	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [FIN, ACK] Seq=5059 Ack=2627 Win=64128 Len=0
40	0.000062	10.11.200.115	43.129.255.54	TCP	54	19250 → 465 [ACK] Seq=2627 Ack=5060 Win=262656 Len=0
41	2.333423	10.11.200.115	43.129.255.54	TLShv1.2	85	Encrypted Alert
42	0.000018	10.11.200.115	43.129.255.54	TCP	54	19250 → 465 [RST, ACK] Seq=2658 Ack=5060 Win=0 Len=0
43	0.277096	43.129.255.54	10.11.200.115	TCP	60	465 → 19250 [RST] Seq=5060 Win=0 Len=0

When attempting to send an email from yanxiangduamanda@foxmail.com to yanxiangduamanda@outlook.com, we identified the SMTP port for yanxiangduamanda@foxmail.com (outgoing Port 465). According to Foxmail's requirements, SSL must be enabled for successful email transmission.

However, during our attempt to track the sending process using Wireshark, we were unable to find the SMTP protocol.

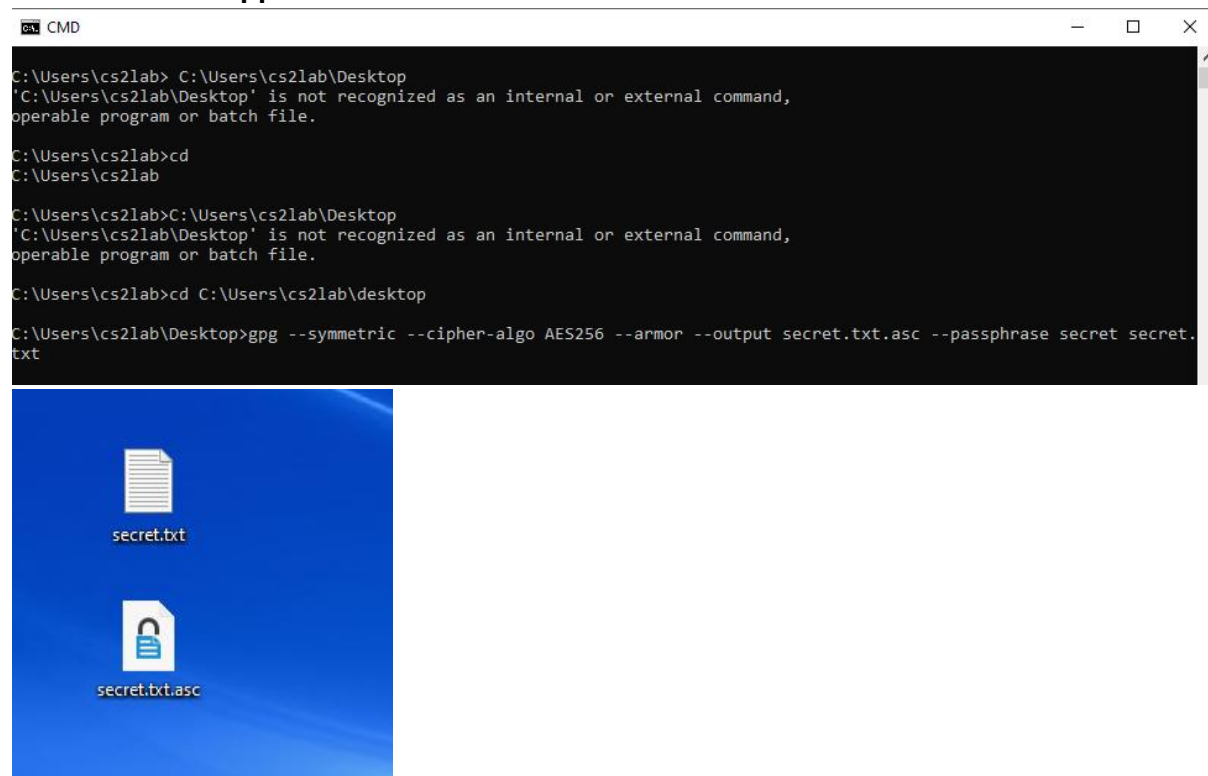
When capturing data on port 465 (Email sending port), we can observe TLS handshake packets, and our emails are successfully sent. Given this, we can exclude the possibility of errors or abnormalities during the connection establishment process. This suggests that the SMTP protocol may be operating over an SSL/TLS encrypted connection.

In an SMTPS connection, the communication content of the SMTP protocol is encrypted using SSL/TLS. Therefore, even though we can see TLS handshake packets in Wireshark, we cannot directly parse and view the plaintext content of the SMTP protocol.

Question 3: What can be considered as the secret key of the ROT13 cipher? Does this cipher use substitution or transposition as an encryption technique? Based only on the intercepted known ciphertext, is the attacker capable of retrieving the plaintext, and if yes, how?

The "key" is simply the number 13, which represents the number of positions each letter is shifted in the alphabet. ROT13 is a substitution cipher. It replaces each letter in the plaintext with the letter 13 positions ahead (or behind) it in the alphabet. For example, 'A' becomes 'N', 'B' becomes 'O', and so on. As for the attacker retrieving the plaintext from the intercepted ciphertext, ROT13 provides very weak security and can be easily broken. Since ROT13 simply shifts each letter by a fixed number of positions, it's essentially equivalent to a Caesar cipher with a fixed key of 13. Therefore, an attacker can easily decrypt the ciphertext by performing the same ROT13 operation again, as ROT13 is its inverse.

Question 4: With the encryption above, what is the size of the generated key? (Both to the algorithm and the effective key length?) Can you identify any security problems with the above approach?



On the desktop, there exists a file named "secret.txt". To encrypt this file, the following command line is used: `gpg --symmetric --cipher-algo AES256 --armor --output secret.txt.asc --passphrase secret secret.txt`

After executing this command, a new file named "secret.txt.asc" is generated. This file contains the encrypted version of the original "secret.txt" file using AES256 symmetric encryption, with the passphrase "secret".

The size of the generated key is 256 bits (32 bytes) for the AES256 algorithm. However, the effective key length is determined by the passphrase provided by the user. In this case, the passphrase is "secret," which is 6 characters long.

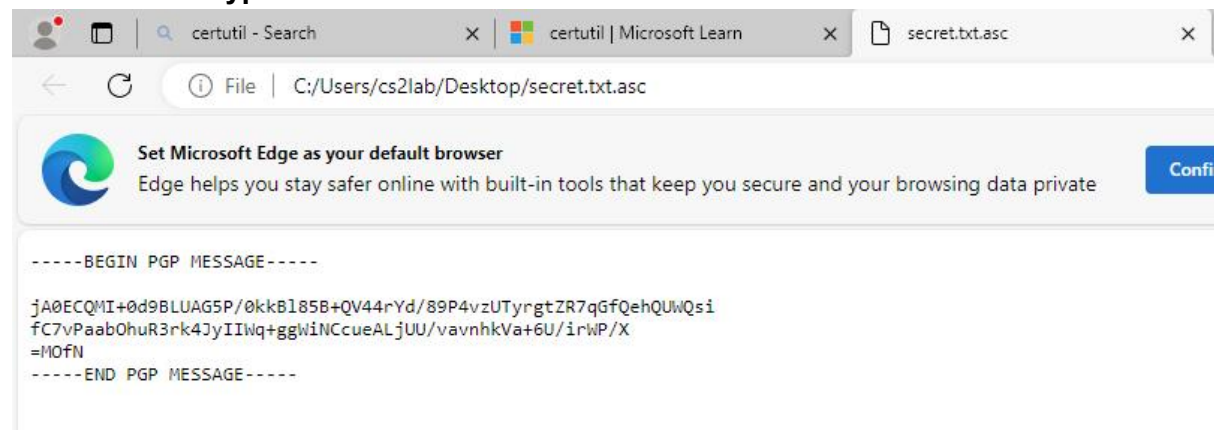
Here are some security problems with the approach described:

Using the key "secret" for AES256 encryption poses a security vulnerability due to its weakness. This key is short, easily guessable, and lacks entropy, making it susceptible to brute-force attacks.

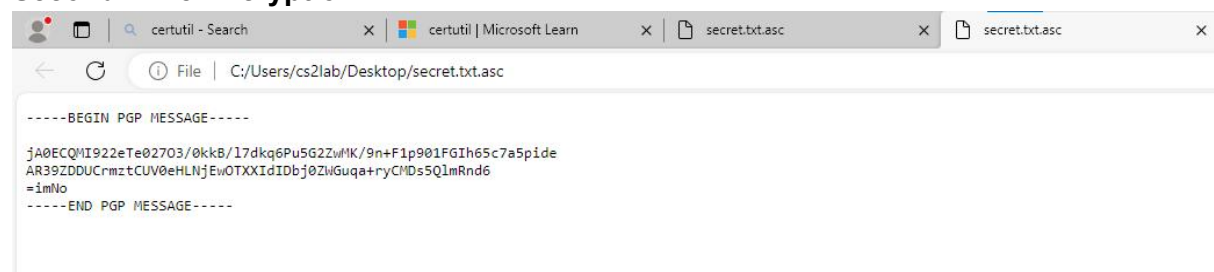
Storing the encryption key directly within the command line or script presents an insecure key management practice. If an unauthorized party gains access to the script or command history, they could easily obtain the key, compromising the security of the encrypted data.

Symmetric encryption lacks forward secrecy, posing a significant security risk. If the key "secret" is compromised at any point, all past and future encrypted messages using that key are vulnerable to decryption.

First Time Encryption



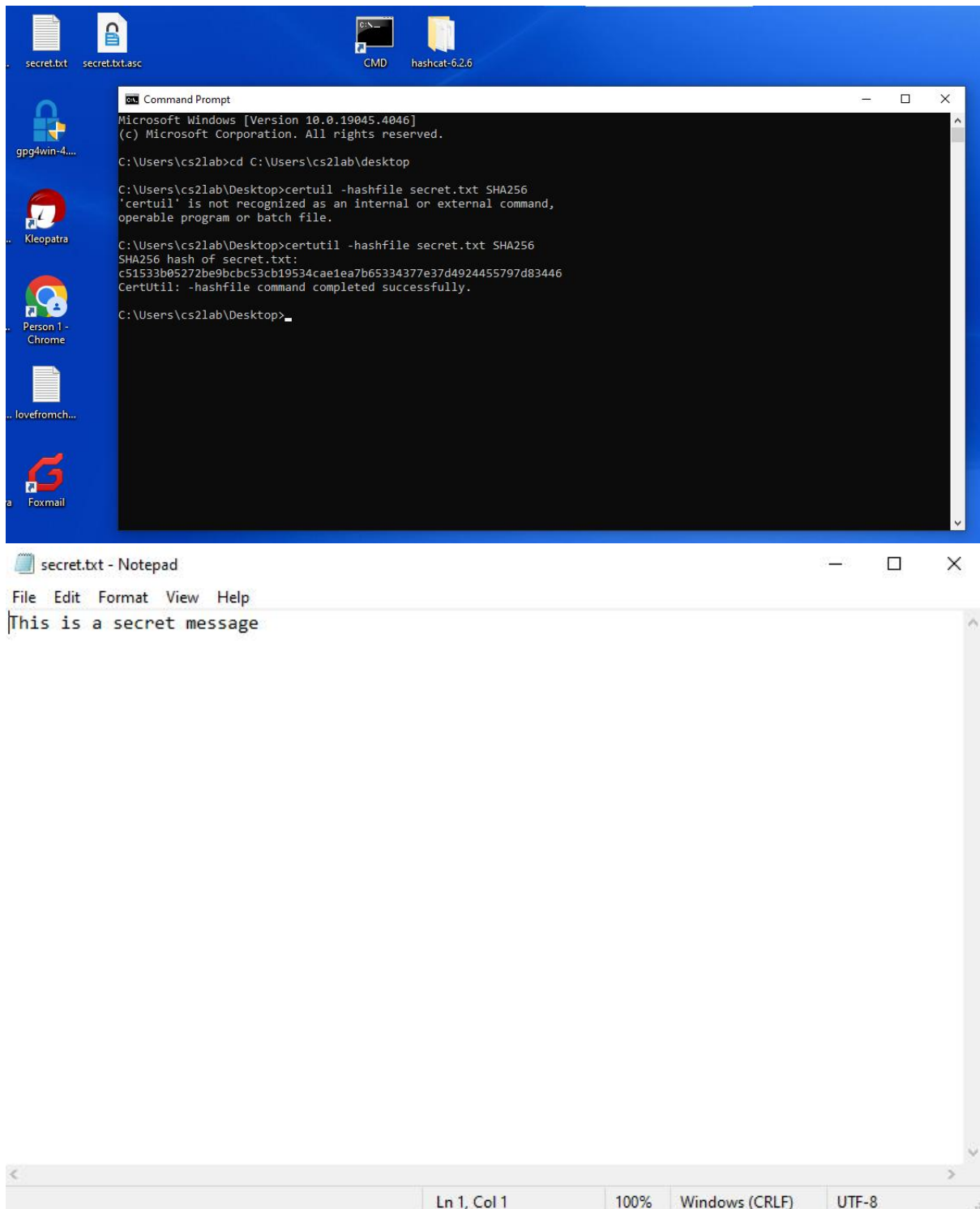
Second Time Encryption



It is shown that the results of the encrypted files with the same key twice are different.

Exercise 3. Message Integrity

Question 5: What are the hash values of the file? What are the hash algorithms' output sizes? Briefly explain why a larger output size of a hash algorithm is preferable." Note that most/all editors/operating systems add line-feed characters to a file. And, these are different for different operating systems The file I encrypted contained a single carriage return character (ASCII CR) at the end.



The screenshot shows a Windows desktop with a blue background. On the desktop are icons for 'secret.txt', 'secret.txt.asc', 'CMD', and 'hashcat-6.2.6'. A taskbar at the bottom shows icons for 'gpg4win-4...', 'Kleopatra', 'Person 1 - Chrome', 'lovefromch...', and 'Foxmail'. Two windows are open:

- Command Prompt:** The title bar reads 'Command Prompt'. The text inside shows the following commands and output:

```
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\cs2lab>cd C:\Users\cs2lab\desktop

C:\Users\cs2lab\Desktop>certuill -hashfile secret.txt SHA256
'certuill' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\cs2lab\Desktop>certutil -hashfile secret.txt SHA256
SHA256 hash of secret.txt:
c51533b05272be9bcb53cb19534cae1ea7b65334377e37d4924455797d83446
CertUtil: -hashfile command completed successfully.

C:\Users\cs2lab\Desktop>
```
- secret.txt - Notepad:** The title bar reads 'secret.txt - Notepad'. The menu bar includes 'File', 'Edit', 'Format', 'View', and 'Help'. The text area contains the message: 'This is a secret message'.

The status bar at the bottom of the Notepad window shows 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.


```
C:\Users\cs2lab\Desktop>certutil -hashfile secret.txt SHA256
SHA256 hash of secret.txt:
f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065
CertUtil: -hashfile command completed successfully.
```

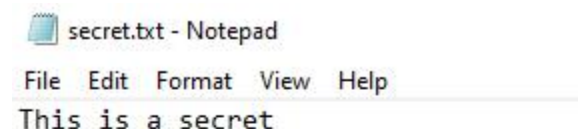
The hash value of the file `secret.txt` calculated using the SHA256 algorithm is:
f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065

The output size of the SHA256 hash algorithm is 256 bits, which translates to 64 hexadecimal characters or 32 bytes.

A larger output size of a hash algorithm is preferable because it provides a larger hash space, making it computationally infeasible for an attacker to find two different inputs that produce the same hash value (a phenomenon known as a collision). This property ensures the integrity and uniqueness of the hashed data. Additionally, a larger output size increases the resistance against brute-force attacks and other cryptographic attacks, enhancing the overall security of the system.

Question 6: Do the checksums match after the change? How different are the checksums? Also, is the protocol you're using (i.e. sending the checksum in clear text with the message) secure? Why/why not?

First, we created a text file named `secret.txt` with the content "This is a secret". Then, we calculated the SHA256 hash of the file using the command "`C:\certutil -hashfile secret.txt SHA256`", and obtained the hash value
f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065.



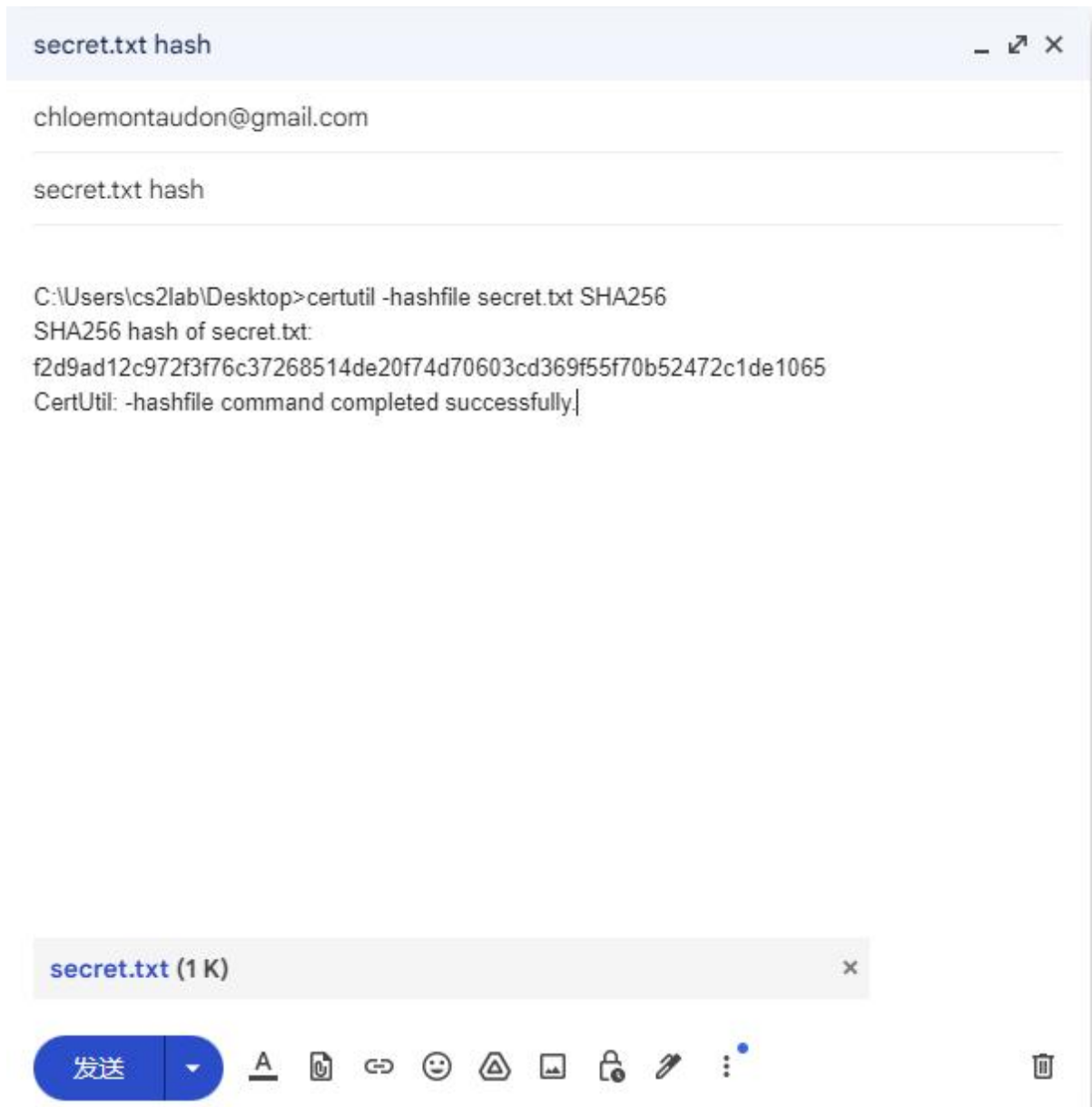
```
CMD
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\cs2lab>cd C:\Users\cs2lab\desktop

C:\Users\cs2lab\Desktop>certutil -hashfile secret.txt SHA256
SHA256 hash of secret.txt:
f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065
CertUtil: -hashfile command completed successfully.

C:\Users\cs2lab\Desktop>_
```

Next, we sent the secret.txt file as an attachment to a team member and wrote the SHA256 hash value f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065 in the email.



Upon receiving the email, the recipient computed the SHA256 hash of the secret.txt file and confirmed that the hash value matched the one provided, thus verifying the integrity of the file and ensuring it had not been tampered with.

CMD

```
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\cs2lab>cd C:\Users\cs2lab\desktop

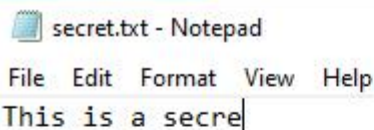
C:\Users\cs2lab\Desktop>certutil -hashfile secret.txt SHA256
SHA256 hash of secret.txt:
f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065
CertUtil: -hashfile command completed successfully.

C:\Users\cs2lab\Desktop>cd C:\Users\cs2lab\Desktop\1234

C:\Users\cs2lab\Desktop\1234>certutil -hashfile secret.txt SHA256
SHA256 hash of secret.txt:
f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065
CertUtil: -hashfile command completed successfully.

C:\Users\cs2lab\Desktop\1234>S_
```

We made a slight modification to the text, changing it to "This is a secre", and then recalculated the hash of the text. The original hash value was f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065.



secret.txt - Notepad

File Edit Format View Help

This is a secre|

```
C:\Users\cs2lab\Desktop\1234>certutil -hashfile secret.txt SHA256
SHA256 hash of secret.txt:
f2d9ad12c972f3f76c37268514de20f74d70603cd369f55f70b52472c1de1065
CertUtil: -hashfile command completed successfully.

C:\Users\cs2lab\Desktop\1234>certutil -hashfile secret.txt SHA256
SHA256 hash of secret.txt:
571bc35ddb4f1272d5e067a5bac1c7dbf9eb5dae65b74a102ea7cd1dfae1784b
CertUtil: -hashfile command completed successfully.

C:\Users\cs2lab\Desktop\1234>
```

The checksums are not match after the change. The hash value changed into 571bc35ddb4f1272d5e067a5bac1c7dbf9eb5dae65b74a102ea7cd1dfae1784b

As for the security of the protocol used (sending the checksum in clear text with the message), it is not secure. Clear text transmission of the checksum alongside the message does not provide any cryptographic protection against tampering. An attacker could easily intercept the message and modify both the message and the checksum to match the new message. This could be done without the recipient being aware of the alteration, as there are no mechanisms in place to detect such changes. Therefore, this protocol lacks integrity verification and is vulnerable to tampering.

Exercise 4. Public Key Cryptography

Question 7: What key-size and key-type did you chose? I.e., what type of public key cryptography algorithms. Why?

```
C:\Users\cs2lab>gpg --full-generate-key
gpg: invalid option "--full-generate-key"

C:\Users\cs2lab>gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (sign only)
 (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 1024
Requested keysize is 1024 bits
Please specify how long the key should be valid.
  0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Alice
Email address: zhangsiwei729084343@gmail.com
Comment:
You selected this USER-ID:
"Alice <zhangsiwei729084343@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: revocation certificate stored as 'C:\\Users\\cs2lab\\AppData\\Roaming\\gnupg\\openpgp-revocs.d\\14E936D3B3E583A9EC652F82D0D66578E208B41F.rev'
public and secret key created and signed.

pub   rsa1024 2024-02-19 [SC]
       14E936D3B3E583A9EC652F82D0D66578E208B41F
uid           Alice <zhangsiwei729084343@gmail.com>
sub   rsa1024 2024-02-19 [E]
```

The command `gpg --full-generate-key` is used to generate a key pair. GPG prompts the user to select the type and size of the key. We choose the RSA encryption and 1024 bits as the key size.

Here are concise reasons for choosing RSA with a 1024-bit key size for short text transmission. Firstly, smaller key sizes require less computational resources, enabling faster encryption and decryption, suitable for short messages. Moreover, smaller keys result in smaller public keys, reducing transmission overhead, ideal for short text transmission with limited bandwidth.

Question 8: Are there any security issues with the secret key file you generated? Why/why not?

The secret key file (typically named `secring.gpg`) contains Bob's private key. This private key is crucial for decrypting messages intended for Bob. Security issues arise if the secret key file is compromised. If an unauthorised person gains access to Bob's secret key file, they could decrypt any messages encrypted for Bob, potentially compromising the confidentiality of sensitive information.

Question 9: How did you make sure that Alice used Bob's public key? If you cannot check Bob's public key, how could someone who has complete control of the communication channel between Alice and Bob perform a man-in-the-middle attack?

Without independently verifying Bob's public key, Alice cannot be certain that she is encrypting the message with the correct key. If an attacker can intercept Alice's attempt to obtain Bob's public key and substitute their public key instead, they can decrypt the message intended for Bob.

If an attacker gains control of the communication channel between Alice and Bob during the key exchange process, they can intercept Bob's public key and replace it with their own. As a result, Alice would encrypt the message with the attacker's public key, allowing the attacker to decrypt and read the message.

Question 10: You've only done encryption so far. Alice can be certain that only Bob can read the message (if precautions against man-in-the-middle attacks have been taken), but can Bob be certain that the message comes from Alice? Why/why not?

Bob cannot be certain that the message comes from Alice solely based on the fact that it was encrypted with his public key. Encryption ensures confidentiality, meaning only Bob can decrypt and read the message, but it does not guarantee the authenticity of the sender.

To establish the authenticity of the sender, additional measures are required, such as digital signatures or message authentication codes (MACs). These cryptographic techniques can be used to verify that the message was indeed created by Alice and has not been tampered with during transmission.

Exercise 5. Public Key Cryptography – Digital Signatures

Now clear all trusted keys from both Alice and Bob. I.e. they should not trust any other key. Have Charlie sign both Alice and Bob's keys. Then introduce Charlie's key to both Bob and Alice. Now sign and encrypt a message from Alice to Bob. Get the respective keys and check Charlie's signature, i.e. trust Charlie to sign for others with a high degree of certainty.

Now we have three entities:

1. Alice - yanxiangduamanda@outlook.com
2. Bob - amanda1990925@gmail.com
3. Charlie - yanxiangduamanda@foxmail.com

1. No trust

We use the `gpg --edit-key` command to edit the keys of both Alice and Bob. In edit mode, we will clear all trusted keys to ensure that neither of them trusts any other keys anymore.

```

cmd CMD - gpg --edit-key C3DDC07350A162A8
ssb rsa1024/A42ABDE99EE4DA56
   created: 2024-02-20  expires: never      usage: E
[ultimate] (1). Alice <yanxiangduamanda@outlook.com>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

gpg> gpg --edit-key C3DDC07350A162A8

Invalid command (try "help")

gpg> gpg --edit-key C3DDC07350A162A8

Invalid command (try "help")

gpg> --edit-key C3DDC07350A162A8

Invalid command (try "help")

gpg> ^C
C:\Users\cs2lab\Desktop>gpg --edit-key C3DDC07350A162A8
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec rsa1024/C3DDC07350A162A8
   created: 2024-02-20  expires: never      usage: SC
   trust: ultimate      validity: ultimate
ssb rsa1024/FA33D74CEF419918
   created: 2024-02-20  expires: never      usage: E
[ultimate] (1). Bob <amanda19990925@gmail.com>

gpg> trust
sec rsa1024/C3DDC07350A162A8
   created: 2024-02-20  expires: never      usage: SC
   trust: ultimate      validity: ultimate
ssb rsa1024/FA33D74CEF419918
   created: 2024-02-20  expires: never      usage: E
[ultimate] (1). Bob <amanda19990925@gmail.com>

Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I don't know or won't say
  2 = I do NOT trust
  3 = I trust marginally
  4 = I trust fully
  5 = I trust ultimately
  m = back to the main menu

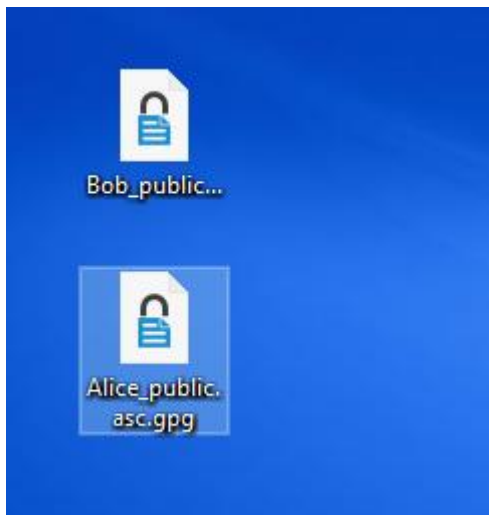
Your decision? 2

sec rsa1024/C3DDC07350A162A8
   created: 2024-02-20  expires: never      usage: SC
   trust: never          validity: ultimate
ssb rsa1024/FA33D74CEF419918
   created: 2024-02-20  expires: never      usage: E
[ultimate] (1). Bob <amanda19990925@gmail.com>
Please note that the shown key validity is not necessarily correct
unless you restart the program.

gpg> _

```

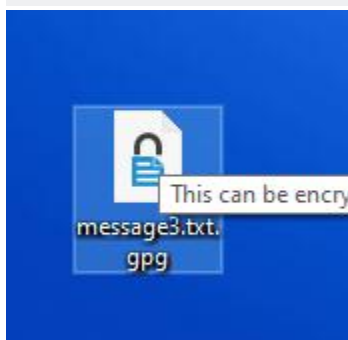
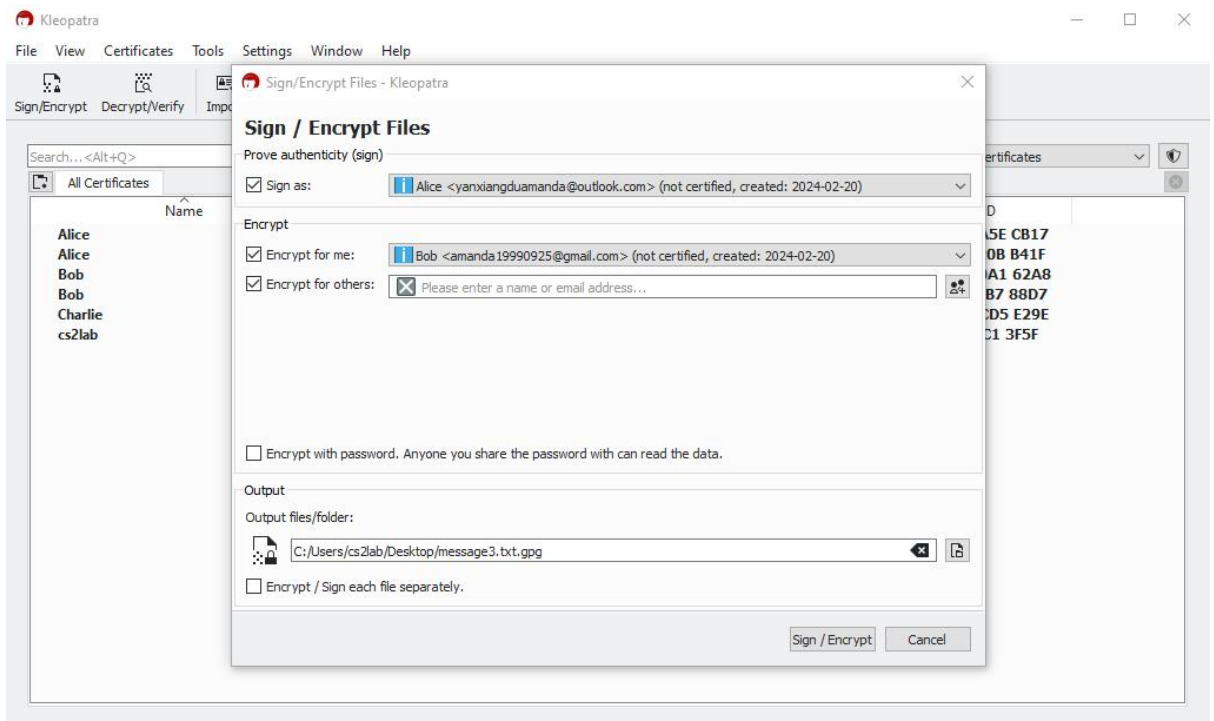
1. Charlie will sign both Alice's and Bob's keys.



2. Import Charlie's public key

Charlie exports his public key as the 'charlie_public_key.gpg' file and attaches it to an email, which he then sends to both Bob and Alice. Upon receiving the email, they download the attachment and import Charlie's public key into their keyrings using the command 'gpg --import charlie_public_key.gpg'.

3. Sign and encrypt a message from Alice to Bob and get message3.txt.gpg.



4. Alice signs and encrypts the message using her private key, then sends the encrypted message to Bob. Upon receiving the message, Bob decrypts it using his private key and verifies the signature. Additionally, Bob checks Charlie's signature on the message to confirm its authenticity.

```
C:\Users\cs2lab\Desktop>gpg --import Charlie_public.asc
gpg: key 1A6D4A746CD5E29E: "Charlie <yanxiangamandadu@foxmail.com>" not changed
gpg: Total number processed: 1
gpg: unchanged: 1

C:\Users\cs2lab\Desktop>gpg --verify gpg message.txt(alice to bob).gpg
gpg: can't open 'gpg': No such file or directory
gpg: verify signatures failed: No such file or directory

C:\Users\cs2lab\Desktop>gpg --verify message.txt(alice to bob).gpg
gpg: can't open 'message.txt(alice to bob)': No such file or directory
gpg: verify signatures failed: No such file or directory

C:\Users\cs2lab\Desktop>gpg --verify gpg message3.txt.gpg
gpg: can't open 'gpg': No such file or directory
gpg: verify signatures failed: No such file or directory

C:\Users\cs2lab\Desktop>gpg --verify gpg message3.txt.gpg
gpg: can't open 'gpg': No such file or directory
gpg: verify signatures failed: No such file or directory

C:\Users\cs2lab\Desktop>gpg --decrypt message3.txt.gpg
gpg: encrypted with rsa1024 key, ID FA33D74CEF419918, created 2024-02-20
"Bob <amanda19990925@gmail.com>"
hahahagpg: Signature made 02/20/24 16:07:36 W. Europe Standard Time
gpg: using RSA key 0DF417BD4F569FDDBCD60D961A6D4A746CD5E29E
gpg: Good signature from "Charlie <yanxiangamandadu@foxmail.com>" [ultimate]
```

Bob checks Charlie's signature on the message to confirm its authenticity. Therefore, Bob can determine that Charlie has signed the message by checking Charlie's signature on the message.

Question 11: Does Charlie have to be present when Alice and Bob communicate with each other? Do they have to communicate with Charlie beyond having trusted his key to sign for other keys? Why/why not? Were there any problems with your understanding of what was going on, and how the web of trust works? What/why? Do you have any suggestions for improvements in that regard?

1. Charlie doesn't need to be present.
2. Once Alice and Bob have trusted Charlie's key to sign for other keys, they can communicate securely without directly involving Charlie in their communication.
3. Upon reflection, the phrase "Then introduce Charlie's key to both Bob and Alice" might have initially been misunderstood. At first, it might have been interpreted simply as the action of sharing Charlie's public key with Bob and Alice. However, a deeper understanding reveals that this step involves more than just sharing the key; it entails the process of Bob and Alice consciously importing Charlie's key into their cryptographic systems, thereby establishing trust in Charlie's identity within their respective networks.

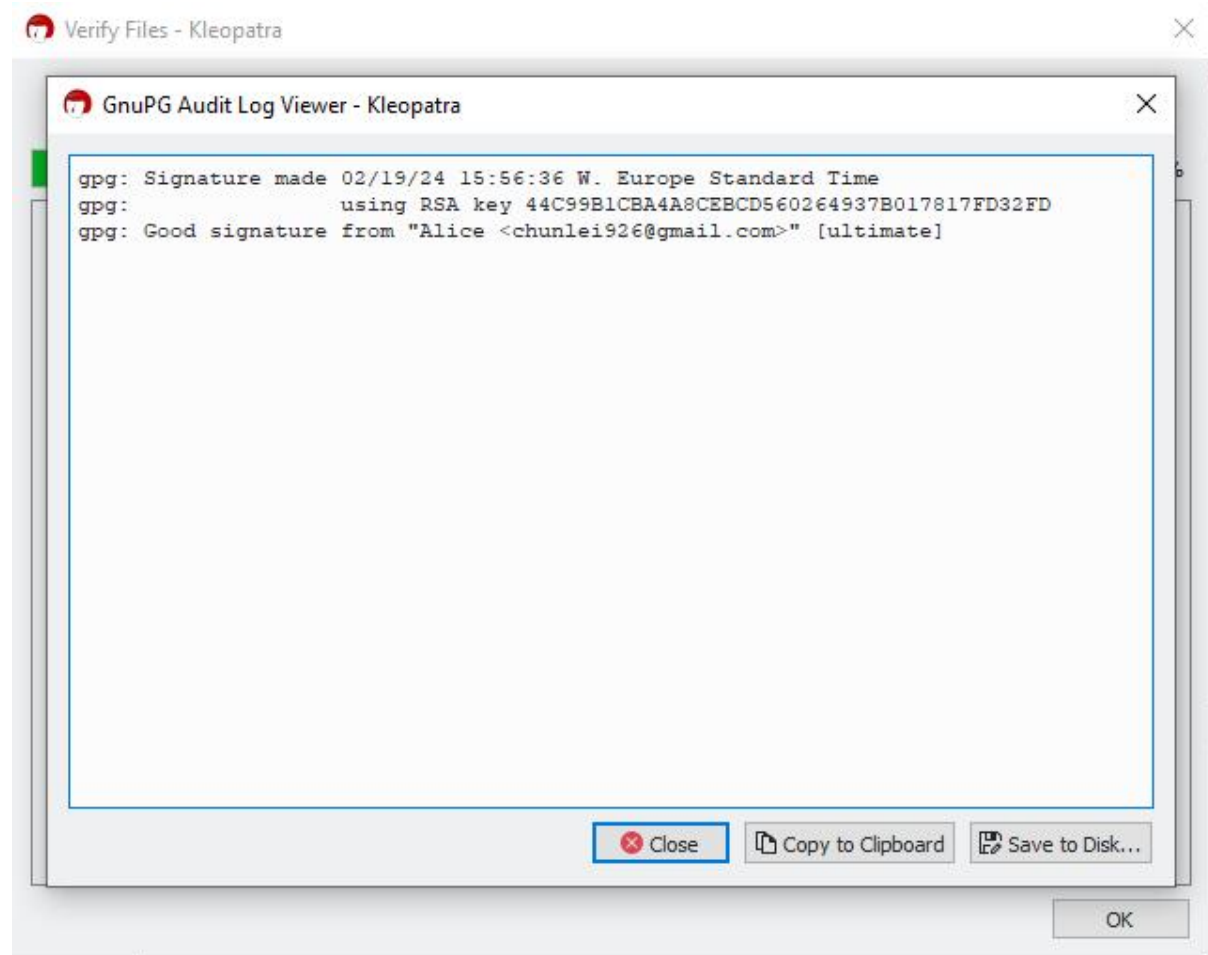
4. The web of trust is a decentralized trust model used in public key cryptography, particularly in systems like Pretty Good Privacy (PGP) and GNU Privacy Guard (GPG).
 - 1)Key Generation: Each user generates a pair of cryptographic keys—a public key and a private key. The public key is shared openly, while the private key is kept secret.
 - 2)Key Signing: Users can sign each other's public keys to attest to their authenticity. When a user signs another user's public key, they are essentially vouching for the fact that they have verified the owner's identity and that the key belongs to the claimed identity.
 - 3)Trust Level: Users can assign trust levels to the keys they have signed. For example, they may fully trust the keys of close friends or colleagues, while trusting keys signed by strangers to a lesser extent.
 - 4)Transitive Trust: Trust in the web of trust is transitive. If Alice trusts Bob's key and Bob trusts Charlie's key, then Alice can also implicitly trust Charlie's key, even if she has never directly interacted with Charlie.
 - 5)Verification: Users can verify the authenticity of a public key by checking its signatures and the trust levels assigned to it. If a key has been signed by multiple trusted users, it is considered more trustworthy.
5. To improve the web of trust, especially concerning key management, consider the following suggestions:
 - 1)Automated Key Renewal: Implement mechanisms for automating key renewal and expiration processes. This ensures that users regularly update their keys to maintain security and validity without manual intervention.
 - 2)Key Backup and Recovery: Provide robust key backup and recovery mechanisms to prevent data loss in case of key compromise or device failure.
 - 3)Enhanced Key Discovery: Improve key discovery mechanisms to make it easier for users to find and verify keys belonging to trusted contacts.

Alice and Bob don't need to communicate with Charlie beyond having a trusted key to sign for other keys. Because Alice and Bob can communicate with each other safely just with the signed key from Charlie.

Problem: We don't know what is the exact meaning of introducing Charlie's key to both Bob and Alice. **Why:**

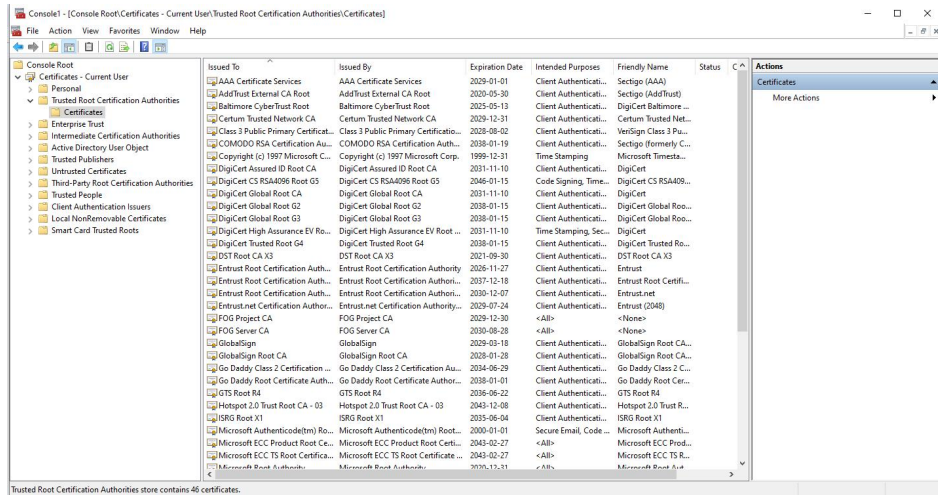
In our understanding trust web needs the user to trust somebody's key with the user's signature. In this way, we can ensure the safety of the communication.

Suggestion:

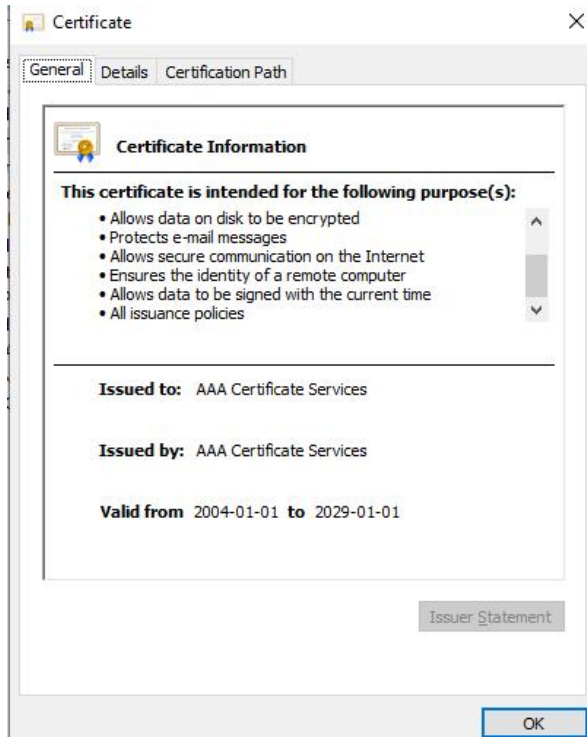


Exercise 6. Public Key Cryptography - Digital Certificates

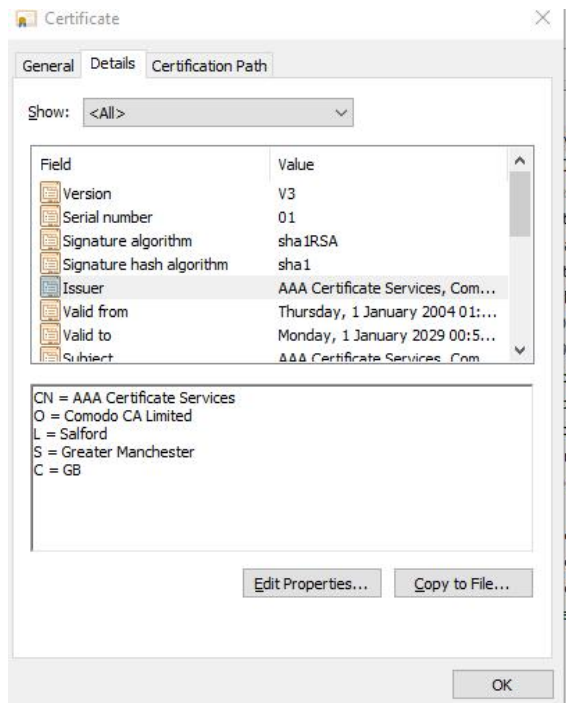
Question 12: What is the publisher of each certificate? (Supply a written description of the publisher and the information provided within the certificate). What information does the Certificate Window tell you? What is the purpose for which the certificate is intended?



First Certificate



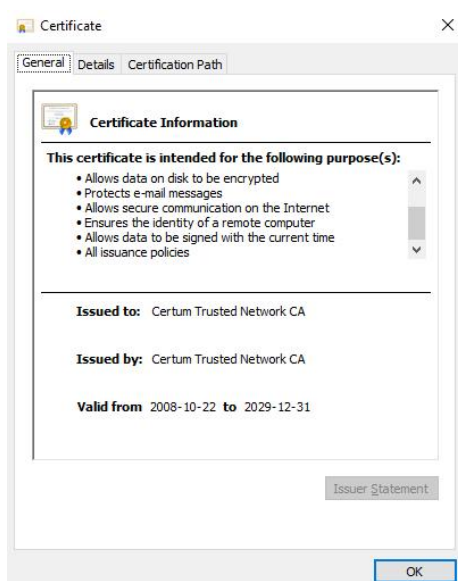
The publisher is AAA Certificate Services, which is the common name of the publisher. The organization is Comodo CA Limited, which is located in Salford, Greater Manchester, U.K.



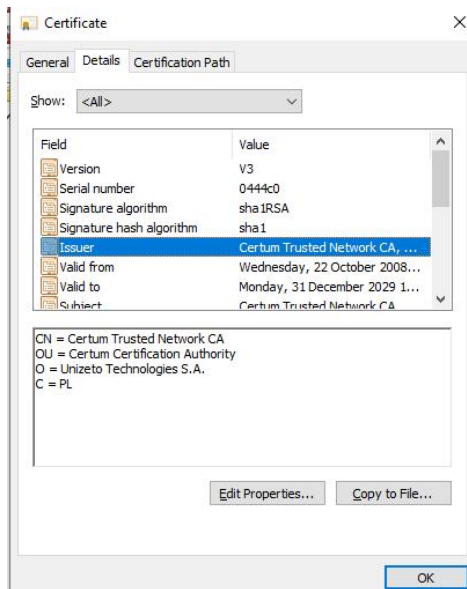
There are versions, serial number validation time, signature algorithm, signature hash algorithm, public key, and public key parameter provided by the certificate window.

The purpose of this certificate is to prove your identity to a remote computer, ensure software came from a software publisher, protect software from alteration after publication, allow data on disk to be encrypted, protect email messages, allow secure communication on the Internet, ensure the identity of a remote computer, allows data to be signed with the current time and all issuance policies.

Second Certificate



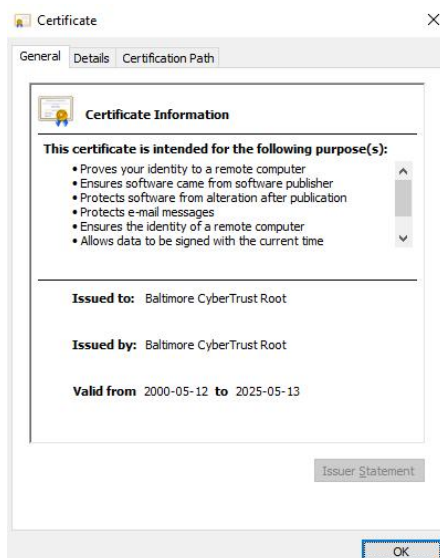
The publisher is Certum Trusted Network CA, which is the common name of the publisher. The organization is Certum Certification Authority, which is located at Unizeto Technologies S.A, Poland.



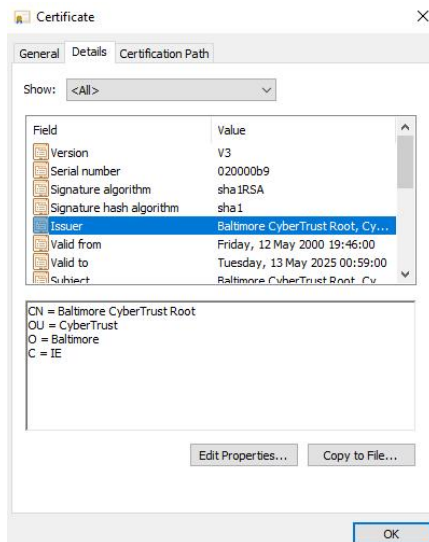
There are versions, serial number validation time, signature algorithm, signature hash algorithm, public key, and public key parameter provided by the certificate window.

The purpose of this certificate is to allow data on disk to be encrypted, protect email messages, allow secure communication on the Internet, ensure the identity of a remote computer, and allow data to be signed with the current time and all issuance policies.

Third Certificate



The publisher is Baltimore CyberTrust Root, which is the common name of the publisher. The organization is CyberTrust, which is located in Ireland.



There are versions, serial number validation time, signature algorithm, signature hash algorithm, public key, and public key parameter provided by the certificate window.

The purpose of this certificate is to prove your identity to a remote computer, ensure software came from a software publisher, protect software from alteration after publication, protect email messages, ensure the identity of a remote computer, and allow data to be signed with the current time.

Reference

- [1]
'What is IMAP ? IMAP and POP3', Cloudflare. Accessed: Feb. 23, 2024. [Online].
Available: <https://www.cloudflare.com/zh-cn/learning/email-security/what-is-imap/>
- [2]
'Digital signature', *Wikipedia*. Feb. 16, 2024. Accessed: Feb. 23, 2024. [Online].
Available:
https://en.wikipedia.org/w/index.php?title=Digital_signature&oldid=1208164452
- [3]
'File Transfer Protocol', *Wikipedia*. Feb. 05, 2024. Accessed: Feb. 23, 2024. [Online].
Available:
https://en.wikipedia.org/w/index.php?title=File_Transfer_Protocol&oldid=1203588080
- [4]
'HTTPS', *Wikipedia*. Feb. 22, 2024. Accessed: Feb. 23, 2024. [Online]. Available:
<https://en.wikipedia.org/w/index.php?title=HTTPS&oldid=1209562543>
- [5]
'Internet Message Access Protocol', *Wikipedia*. Jan. 31, 2024. Accessed: Feb. 23, 2024. [Online]. Available:
https://en.wikipedia.org/w/index.php?title=Internet_Message_Access_Protocol&oldid=1201573509
- [6]
'Man-in-the-middle attack', *Wikipedia*. Feb. 22, 2024. Accessed: Feb. 23, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Man-in-the-middle_attack&oldid=1209452131
- [7]
'Message authentication code', *Wikipedia*. Dec. 22, 2023. Accessed: Feb. 23, 2024. [Online]. Available:
https://en.wikipedia.org/w/index.php?title=Message_authentication_code&oldid=1191262880
- [8]
'Page Not Found - Newsoftwares.net Blog'. Accessed: Feb. 23, 2024. [Online].
Available: <https://www.newsoftwares.net/blog/dark-side-of-symmetric-key-encryption/https://www.kiteworks.com/risk-compliance-glossary/aes-256-encryption/>
- [9]
'Post Office Protocol', *Wikipedia*. Feb. 19, 2024. Accessed: Feb. 23, 2024. [Online].
Available:
https://en.wikipedia.org/w/index.php?title=Post_Office_Protocol&oldid=1208964171
- [10]

'Post Office Protocol', *Wikipedia*. Feb. 19, 2024. Accessed: Feb. 23, 2024. [Online]. Available:

https://en.wikipedia.org/w/index.php?title=Post_Office_Protocol&oldid=1208964171

[11]

'ROT13', *Wikipedia*. Feb. 14, 2024. Accessed: Feb. 16, 2024. [Online]. Available:

<https://en.wikipedia.org/w/index.php?title=ROT13&oldid=1207325632>

[12]

'ROT13', *Wikipedia*. Feb. 14, 2024. Accessed: Feb. 23, 2024. [Online]. Available:

<https://en.wikipedia.org/w/index.php?title=ROT13&oldid=1207325632>

[13]

'Secure Shell', *Wikipedia*. Feb. 19, 2024. Accessed: Feb. 23, 2024. [Online]. Available:

https://en.wikipedia.org/w/index.php?title=Secure_Shell&oldid=1208869682

[14]

'Simple Mail Transfer Protocol', *Wikipedia*. Feb. 12, 2024. Accessed: Feb. 23, 2024. [Online]. Available:

https://en.wikipedia.org/w/index.php?title=Simple_Mail_Transfer_Protocol&oldid=1206480192

[15]

'Telnet', *Wikipedia*. Feb. 22, 2024. Accessed: Feb. 23, 2024. [Online]. Available:

<https://en.wikipedia.org/w/index.php?title=Telnet&oldid=1209521814>

[16]

'What is the RSA algorithm? Definition from SearchSecurity', Security. Accessed: Feb. 23, 2024. [Online]. Available:

<https://www.techtarget.com/searchsecurity/definition/RSA>

[17]

'Why is HTTP not secure? The difference between HTTP and HTTPS', Gcore.

Accessed: Feb. 23, 2024. [Online]. Available: <https://gcore.com/learning/http-vs-https-security-comparison/>