Answer Sheet
Assignment – A06

Cryptography

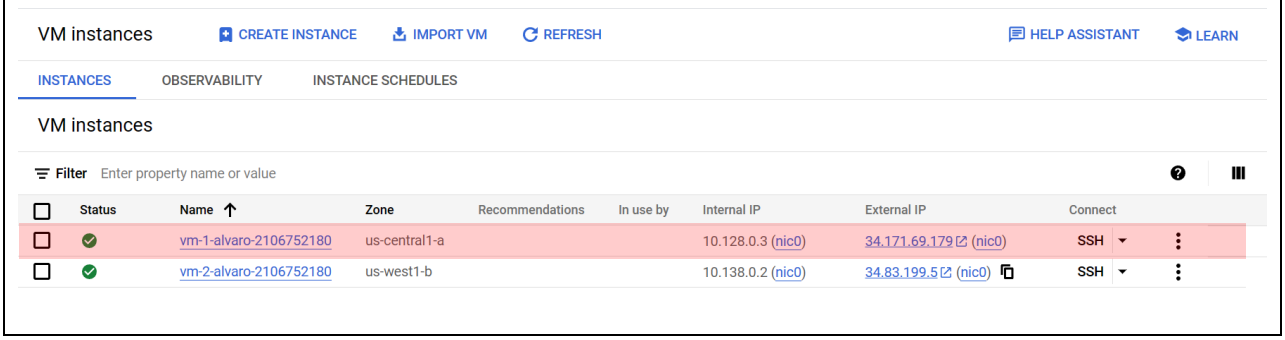
Name : Alvaro Austin
Student ID : 2106752180

Notes: All answers must be supported by relevant screenshot(s). You **must highlight part of the screenshot** or **describe which part of the screenshot (fields, value, etc.) specifically** to support your answer. Answers with screenshots that have no highlight/description **will not be scored**.

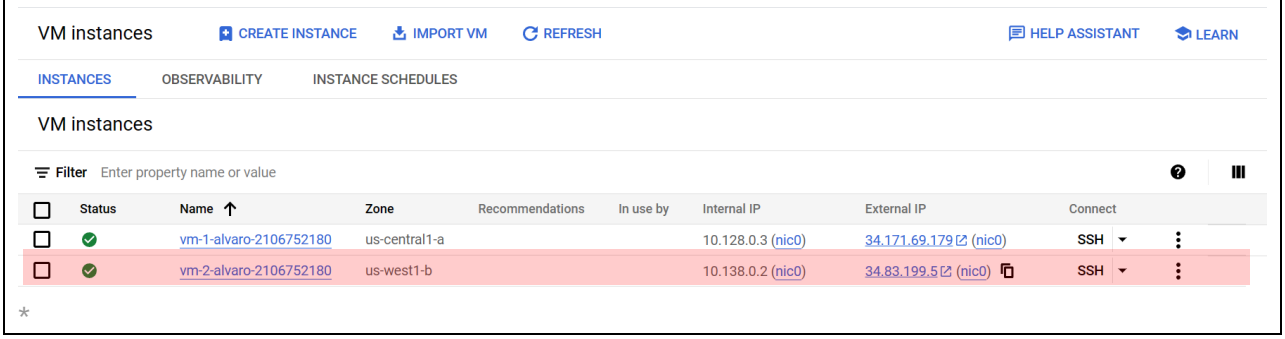
Initial Information “Server” and “Client

Virtual Machine Name	Acting As	Private IPv4 Address	Public IPv4 Address
vm-1-alvaro-2106752180	Server	10.128.0.3	34.171.69.179
vm-2-alvaro-2106752180	Client	10.138.0.2	34.83.199.5

Screenshot of Server Virtual Machine



Screenshot of Client Virtual Machine



Specification Answers

[20 points] Part A: Symmetric Encryption Coding

Screenshot of Client and Server interaction (screenshot the CLI)

Information:
vm-1-alvaro-2106752180: server (
as mentioned above)
vm-2-alvaro-2106752180: client (

as mentioned above)

```
alvaro_austin@vm-1-alvaro-2106752180:~$ go run compnetcsui/a06/server
Hello Server!
Here's the Symmetric Key ERcJdOdLYOXGIRMNWGafNHNbLRAcSNEY
Testing Message 2106752180
alvaro_austin@vm-1-alvaro-2106752180:~$
```

```
alvaro_austin@vm-2-alvaro-2106752180:~$ go run compnetcsui/a06/client
Hey there Client
Symmetric Key received
Received Message 2106752180
alvaro_austin@vm-2-alvaro-2106752180:~$
```

Related code (Just for safety :D)

Server.go

Client.go

[20 points] Part A: Symmetric Encryption Analysis

How does symmetric encryption work in securing the message?

Symmetric encryption work in securing the message by encrypting and decrypting the message with a single key that were generated. This key is then use to create a cipher block that will be used to encrypt and decrypt a message. In this assignment, we can identify few steps to securing the message:

1. Key generation

Key generation is the step to generate key that will be used across client and server. By having this key, once we have establish a secure connection with the server (using TCP), we could then send the key. Afterwards, the server will use this key to create a new AES cipher block for the sake of encryption and decryption.

2. Encryption
   After the server create his new cipher block from the key that were received from the client, then this cipher block is used to convert a plain text to a cipher text. The encryption algorithm operates on fixed size block of data. This cipher text is then transmitted to the recipient (client)
3. Transmission
   The cipher text is then transmitted over a secure protocol, named TCP.
4. Decryption
   After the recipient received the encrypted text. They who already have cipher block just decrypt the cipher text to the original text message.

Mention one problem from a security standpoint of symmetric encryption and point out which part of the code shows that problem.

One problem from a security standpoint of using symmetric encryption is the key distribution. As we already know, symmetric encryption uses the same key for both encryption and decryption from both sides. Hence, the key must be securely shared between the sender and the recipient. If an attacker try to intercept or obtains our shared key, then they could decrypt and intercepted ciphertext and gain access to original plain text.

In the client.go, when we sent the cipher key to the server:

```
// This part will send the symmetric key to the server
_, err = conn.Write(cipherKey)

if err != nil {
    fmt.Println("Error message:", err.Error())
}
```

In this part, a malicious attacker could potentially listen for instance using MiM attack, could get our cipher key. This key will create a cipher block that's similar across application. Hence, the attacker could see our plain text.

Cryptohelper.go

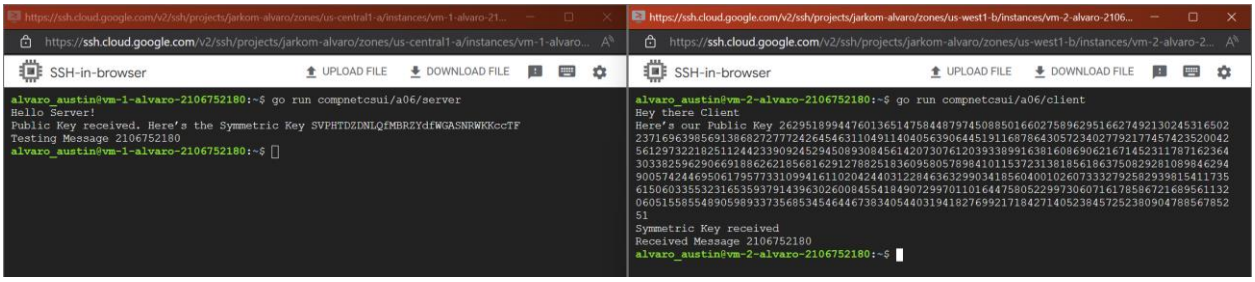
```go
// This function creates an AES cipher block using the cipher key specified in the cipherKey
// argument. Returns the associated AES cipher block and an error (if any).
func ImportSymmetricKey(cipherKey []byte) (cipherBlock cipher.Block, err error) {
    cipherBlock, err = aes.NewCipher(cipherKey)
    if err != nil {
        return nil, err
    }

    return cipherBlock, nil
}
```

In cryptohelper.go, you can see that, once somebody gets our cipher key, then they could easily decrypt our ciphertext because cipher block was created with the same cipher key.

[20 points] Part B: Asymmetric Encryption Coding

Screenshot of Client and Server interaction (screenshot the CLI)

Code is in Scele submission
vm-1-alvaro-2106752180: server (as mentioned above)
vm-2-alvaro-2106752180: client (as mentioned above)



[20 points] Part B: Asymmetric Encryption Analysis

How does asymmetric encryption work in securing the message?

Asymmetric encryption work in securing the message by creating a pair of public and private key. The public key is freely shared and used for encryption, while the private key is kept secret and used for decryption. There are some similar steps with symmetric encryption, but the steps are:

1. Key generation
   Create a pair of public and private key (even though in this assignment we just generate a private key, then the private key has their own public key). Especially in this assignment, we will use this public key that can be distributed freely for encryption. Particularly in this assignment, we will use this public key to encrypt client's symmetric key to server to make a more secure symmetric key distribution.
2. Public key Distribution
   This step is a little bit different than symmetric encryption. In asymmetric encryption, we provide public key to others that can freely used to encrypt their message so that we can use it.
3. Encryption

Normally, encryption uses the public key receiver by the sender. But in this assignment, we still use symmetric key for encryption. Although, except for symmetric key encryption, we use the public key server so that server can decrypt using their private key.

4. Transmission
   The cipher text is then transmitted over a secure protocol, named TCP.
5. Decrypt
   Normally, decryption uses the private key receiver to get the plain text. In this assignment, as I said, we still use symmetric key for normal text. Although, we used private key of the server to receive symmetric key from the client.

Does asymmetric encryption solve the problem that symmetric encryption has? How?

Yeah, I will talk about the security standpoint as it's the most relevant point for this assignment. As I already mentioned above, that asymmetric encryption uses public key of the receiver to encrypt the message then the receiver could use their private key to decrypt back. This will make the security of the communication more secure. Because as we already know, private key of the receiver must not be shared publicly. Hence, by having a private key exclusive to only the receiver, then the message could only be read by the receiver thus fixing the security standpoint that somebody can listen to the communication.

In this assignment, we uses both symmetric and asymmetric encryption. This is happened because the problem above shows that once the attacker know the symmetric key by listening to the communication could automatically get the symmetric key. Therefore, by encrypting the symmetric key message using public key of the receiver, we can securely sent the symmetric key to the receiver without worrying about attacker listening to our communication.

The code that shows where asymmetric encryption solve symmetric encryption problem is:

```
encryptedMessage, err := cryptohelper.AsymmetricEncrypt
(pubKey, "Public Key received. Here's the Symmetric Key")

if err != nil {
    fmt.Println("error message:", err.Error())
    return
}

_, err = conn.Write(encryptedMessage)
time.Sleep(time.Second)

if err != nil {
    fmt.Println("error message:", err.Error())
    return
}

encryptedCipherKey, err := cryptohelper.AsymmetricEncrypt
(pubKey, string(cipherKey))

if err != nil {
    fmt.Println("error message:", err.Error())
    return
}

_, err = conn.Write(encryptedCipherKey)
time.Sleep(time.Second)
```

[20 points] Part C: Comparison Analysis

Notice that the asymmetric code still uses symmetric key. Why does asymmetric encryption still use symmetric key? Why not just use asymmetric encryption/keys only?

The reason we still uses symmetric keys is that asymmetric encryption is slower and more resource intensive
compared to symmetric encryption. Symmetric encryption, on the other hand, is faster and more efficient
for bulk data encryption. By combining both asymmetric and symmetric encryption, we can achieve a secure
and efficient encryption system that leverages the strengths of each type of encryption.

If we only use asymmetric encryption/keys then as I said, then the computational overhead will increase and reduces the performance for our program. Hence, reducing user experience for others.