Matthew Regan

Cyber Security

Unit code: COMP232

**CONTENTS**

[Task 1 2](#_Toc3814969)

[Task 2 4](#_Toc3814970)

# Task 1

Comparison of methods for message authentication

Hash functions (SHA-256)

A hash function is the process of mapping data of an arbitrary size onto data of a fixed size. For example, we can take a series of names as input (Alice, Bob, Carol) and map them to a set of integers (1, 2, 3). In this example I used a single bit to encrypt the inputs where 1 returns Alice, 2 returns Bob and 3 returns Carol. Of course, more sophisticated hash mapping is used in real-world implementations. More commonly in modern encryption hash functions encrypt data with 256 bits and only identical messages will produce identical hash functions. A single difference of character should produce an entirely new hash function. For this reason, it is infeasible to try and recreate the input data given a hash function with the only reliable method being a brute-force search. A property referred to as irreversible.

The benefit of hashing functions is they are deterministic. Any computer that understands a hashing function such as SHA-256 can compute the hash of a given sentence and get the same answer as every single other computer using the same hash function.

RSA + SHA1

RSA is a cryptographic technique named after its creators (Rivest-Shamir-Adleman). It is one of the first public-key cryptography techniques. The goal of RSA encryption is establishing a secure connection between two or more parties across a network. This is accomplished using a pair of keys, one public and the other private. Data encrypted with the public key can be decrypted using the matching private key and vice versa (public decrypts public). Using keys to encrypt data becomes more computationally expensive as the size of the data we are trying to transmit increases. A solution to this is using a hash function such as SHA1 to translate our messages into inputs of fixed length. We can then encrypt the hash value with a private key to produce a digital signature. A digital signature works in much the same way a physical signature does. It provides proof that the message sent was truly from who we thought it was (message authentication) and the sender cannot later deny that they did not send the message (non-repudiation). By using the senders corresponding public key, we can verify a person’s digital signature.

DSA

DSA (or Digital Signature Algorithm) belongs to the same family of public-key cryptography as RSA. Where they differ is RSA can be used to encrypt/decrypt data as well as provide a digital signature, whereas DSA can only produce a digital signature. Without the assistance of additional encryption algorithms DSA will merely attach a digital signature to un-encrypted data.

HMAC-SHA256

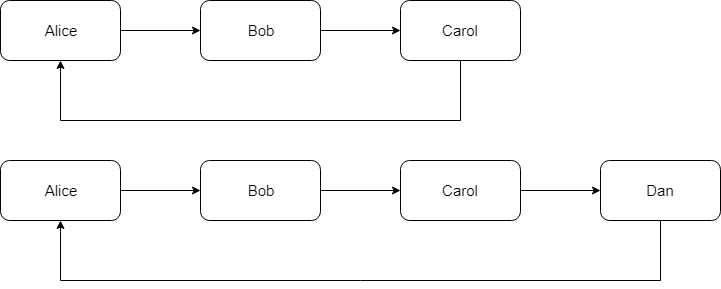
HMAC, short for Keyed-Hash Message Authentication code is a cryptographic method to verify data integrity and the authenticity of a message. The goal of HMAC is the same as any MAC except a hash function is used in conjunction to generate the MAC. The level of security provided by a HMAC is reliant upon the hash function utilized. In this example SHA256 will produce larger hash values than any SHA1 function improving its cryptographic strength.

# Task 2

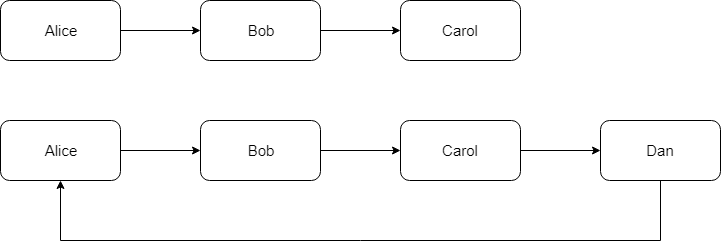
Key exchange for four parties

The Java program given in Lab 6-7 is a Diffie-Hellman’s Key exchange between 3 parties. In this program information travels along a line from neighbour to neighbour. The goal of Diffie-Hellman’s key exchange is to establish secure communication between two or more parties. To achieve this a secret value must be established between parties that can be used to decrypt incoming messages. Before each party can calculate the common secret key, they first need to share an agreed upon value, referred to as a public key. A sends their public key to B, B sends their public key to C and C sends their public key to A. Once each party has their neighbours public key a secret value can be independently calculated. This avoids sharing any secret information across a network.

My variant of the three-person Diffie-Hellman key exchange to include a fourth person operates in the exact same manner as the three-person protocol. The core idea of passing information to your neighbour remains true when adding a fourth person. Instead of:

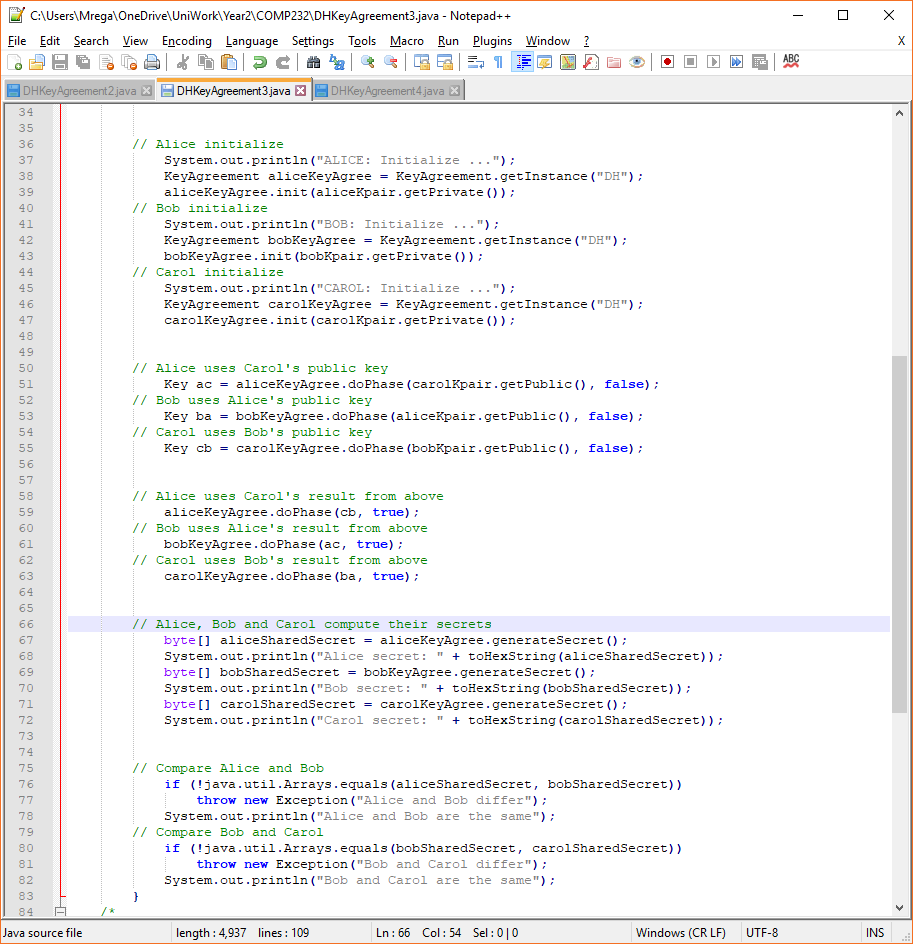


I now include a fourth person (“Dan”) and the passing of information looks more like:



In theory the process of inserting neighbours can be extended to include any number of people.

In the code for the 3-man Diffie-Hellman’s key exchange every participant generates a key and passes it to their neighbour. The recipient then attach’s their own key to the received key. This final key is passed to the “final” neighbour in the loop who now posses a single key encompassing both the other participants keys.



In my 4-man Diffie-Hellman key exchange I take the same principle and perform an additional passing of keys between neighbours because the final key needs to encompass three neighbours instead of two. The result is Alice will have a single key containing Bob, Carol and Dan’s individual keys. This is true for all participants in the key exchange.

