GRIFFITH COLLEGE DUBLIN

Tutorial Cover Sheet

Student name:	BANDARU KIRAN SUBRAHAMANYA SAI		
Student number:	3178784		
Faculty:	Computing Science		
Course:	Computing Science	**Stage/year:**	1/1
Subject:	Information Security		
Study Mode:	Full time **yes**	*Part-time*	
Lecturer Name:	Sheresh Zahoor		
Assignment Title:	Tutorial 03		
No. of pages:	06		
Disk included?	Yes	*No*	
Additional Information:	(ie. number of pieces submitted, size of assignment, A2, A3 etc)		
Date due:	01/05/2025		
Date submitted:	01/05/2025		

Plagiarism disclaimer:

I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.

I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.

Signed: _ B.kiransai___ Date: 01/05/2025_

Please note: Students **MUST** retain a hard / soft copy of **ALL** assignments

Q1 : **List the five services offered by PGP**

Sol:

1. Authentication

PGP verifies that a message really comes from the person who claims to have sent it.
It does this by using **digital signatures** — basically, a way to "sign" a message electronically to prove its origin.

2. Confidentiality

PGP keeps your messages **private and secure**.
It scrambles the message so that only the person it's meant for can read it — even if someone else tries to peek along the way.

3. Compression

Before encrypting a message, PGP **compresses** it.
This reduces the message size, saving space and speeding up transmission. It also adds an extra layer of security by making patterns in the data harder to spot.

4. Email Compatibility

PGP converts messages into a special **ASCII** format (called Radix-64 encoding) to make sure they can easily travel across different email systems, even ones that don't handle binary data well.

5. Segmentation

If a message is too large for email systems to handle in one piece, PGP **breaks it into smaller chunks**.
Each piece is sent separately and can be reassembled on the receiver's side without losing any part of the original message.

Q2 : **Explain how hashing is used in digital signatures, supported by a labeled diagram to illustrate the process.**

Sol : 1. Hashing the Message

First, the sender (you) takes the original message and runs it through a hash function (like SHA-256).
This creates a fixed-size summary of the message, called a hash or digest.

The idea is simple:
No matter how big your message is, the hash will always be a short, unique fingerprint of the content**.**

2. Signing the Hash

Instead of encrypting the entire huge message (which would be slow), you encrypt the hash using your private key.
This encrypted hash becomes your digital signature.

Your private key is like your personal seal only you have it!

3. Sending the Message + Signature

You send both:

- The original message
- The digital signature (encrypted hash)

to the receiver.

4. Verifying at the Receiver's End

When your friend (the receiver) gets your message, they do two things:

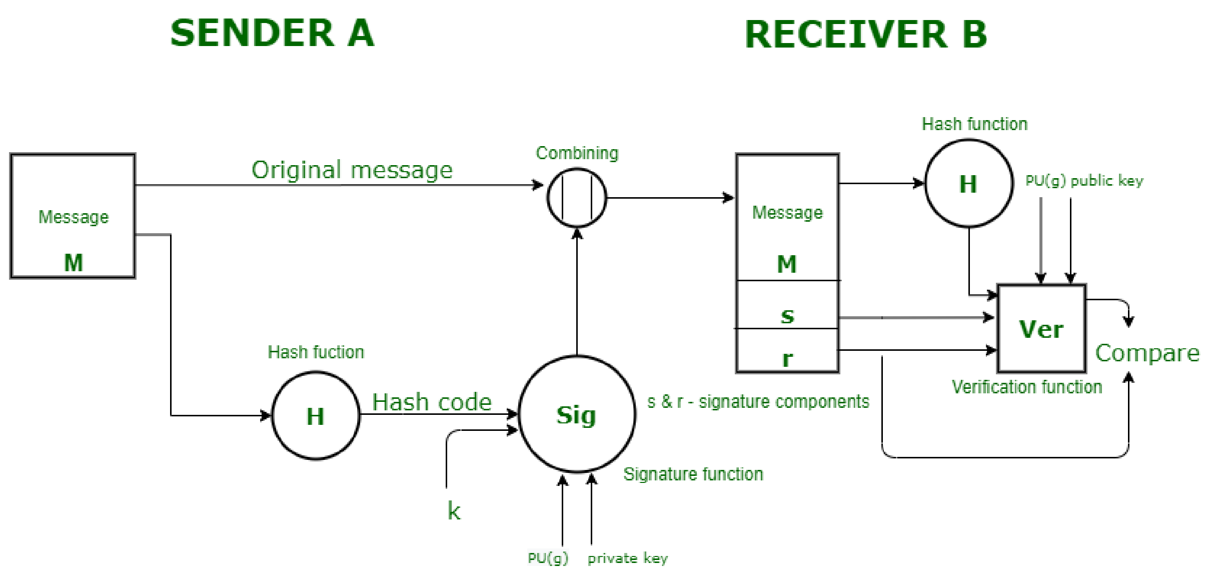
1. The receiver first **creates a hash** of the message they received, using the **same hash function** you used.
2. Then, they **decrypt your digital signature** using **your public key** to recover the **original hash** you had created and signed.

By comparing the two hashes, they can confirm whether the message is authentic and unchanged.

5. Comparing Hashes

Finally, they compare the two hashes:

- If the hashes match, it means the message is authentic and untouched.
- If they don't match, it means the message was changed or the signature is fake.



- Hashing makes messages short and unique.
- Signing the hash proves you sent it.
- Verifying the hash proves the message wasn't changed.

Q3 : **Using a labelled diagram, explain how a Message Authentication Code (MAC) is used to ensure message authentication in communication systems**

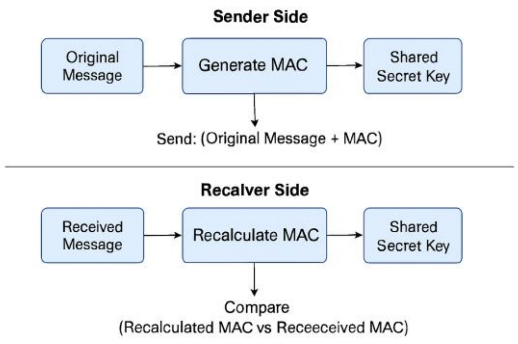
Sol :

Message Authentication Code (MAC):

Message Authentication Codes are like invisible security stamps added to online communications.
In secure connections like SSL/TLS, these MAC "tags" help you trust that:

- The message really came from the person it claims to be from, and
- The message wasn't tampered with while it was being sent.

In simple terms, a MAC acts like a seal of authenticity and integrity for your online conversations.



The diagram is divided into two main parts:

-> Sender Side and Receiver Side.

1. **Sender Side**

- Original Message:
  The sender starts with the message that they want to send.
- Shared Secret Key:
  Both sender and receiver share a secret key beforehand. This key is known only to them and is essential for the MAC process.
- Generate MAC:
  The message and the shared secret key are combined using a MAC algorithm (such as HMAC, CMAC, etc.).
  This generates a MAC code — a small, fixed-size value that uniquely represents the message + key.
- Send Message + MAC:
  The sender transmits both the original message and the generated MAC to the receiver.

2. **Receiver Side**

- Received Message:
  The receiver gets the message and the attached MAC from the sender.
- Shared Secret Key:
  The receiver uses the same shared secret key that was agreed upon earlier.
- Recalculate MAC:
  The receiver recalculates a new MAC by feeding the received message and the shared key into the same MAC algorithm.
- Compare:
  The receiver compares the recalculated MAC with the MAC received from the sender.

If they match, it means:

- The message really came from who it says it did — so you can trust the sender (authentication).
- The message wasn't changed or tampered with while it was being sent (integrity).

If they don't match, it means:

- The message may have been modified or forged (integrity/authentication failed)

Summary :

- A MAC uses both the message and a secret key to generate a secure code.
- Only someone with the same secret key can successfully create or verify the correct MAC.
- It ensures that the message is authentic and has not been changed during transmission.

Q4 : **List and briefly describe ten desirable properties of a secure cryptographic hash function.**

Sol:
Properties of good cryptographic hash functions are as follows:

Collision resistant

It is a property of a hash function *h* for which it is computationally very hard to find two distinct inputs, *A* and *B*, for which *h(A) = h(B)*.

Pre-image resistance

It's nearly impossible to work backwards from a hash and figure out the original input you can't reverse a hash to find out what was put in.

Second pre-image resistance

It's practically impossible for anyone to find another input that produces the same hash as a given input. e.g., Given *h(Cat) = AB38DA*, it is computationally very hard to find another input that maps to the same output.

Large output space

Outputs of cryptographic hash functions usually are very large integers (represented in binary as bits). The output of the SHA-256 hash function is 256 bits long, and the total number of possible outcomes is 2256=1.1579209×1077.2256=1.1579209×1077. Thus, using a brute force approach on this number of outputs will take years to compute, thus making it ineffective and our hash function more secure.

Deterministic

A given input *x* to a hash function *H* will always generate the same output.

Avalanche effect

A small change in the input will result in a completely different output hash.

Puzzle friendliness

A hash function is called puzzle-friendly when it's extremely hard to find a specific input that produces a desired output.

In fact, there’s no shortcut the only way to find the right answer is by brute force, meaning you have to keep guessing randomly until you get lucky.

Because of this, solving the puzzle typically takes about 2^n tries, where n is the number of bits in the hash output. The bigger the output size, the harder (and slower) it is to guess the correct input.

Fixed-length mapping

The output size of a hash function is fixed and independent of the input size. The output size of the SHA-256 hash function is 256 bits for any arbitrary-sized input.

Resistance to Length Extension Attacks

For some constructions, the hash should not allow an attacker to extend the input and predict the hash of the longer message.

Indistinguishability from Randomness

The output hash should appear random; no patterns should be easily detectable.