Encryption and Digital Signatures for A-level Computing

(originally written for the AQA CPT5 module in 2006)

There are various methods of encrypting messages, most of which use the concept of a "key". An encryption key is a bit pattern which is used to process the "plaintext" to produce the encrypted text, or "ciphertext". In order to decrypt this ciphertext, producing the original plaintext, a decryption key must be used. In some algorithms, the encryption and decryption keys could be the same (a "symmetrical" algorithm). More commonly, a pair of keys are used - either one could encrypt the plaintext, and the other could then decrypt the ciphertext.

For CPT5 you don't need to know the details of exactly how the keys are generated, or the mathematical way in which they are used. If you are interested, then look up the RSA algorithm, or PKI (public key infrastructure). (A Google search will help, or the wikipedia is a good starting point). You do need to know how the keys are used in sending secure messages, and the methods expected by the examiners are those of PKI technology. A pair of keys are used: one is known only to one person, and is called that person's "private" key. The other key of the pair can be released to anyone, and is therefore called that person's "public" key. The private key should be kept secret - if it becomes known to anyone else, then security is compromised. The degree of security that this method provides is based on the facts that

- messages encrypted with one key of the pair can be decrypted only with the other key;
- it is virtually impossible given one key to work out the other key of the pair. (This depends on the key length, i.e. number of bits in the key, but as the key length increases, the time taken for the fastest supercomputer to find the corresponding key also increases, and can easily reach years or millions of years).

Simple level

Suppose B wants to send a message to A, so that only A can read the message. If only A can read it, then it follows that it must be decrypted by A with A's private key. No-one else knows A's private key, hence only A could decrypt it. In order that it should be decryptable by A's private key, it must have been encrypted by the corresponding key of the pair, i.e. A's public key, which, being a public key, can be obtained by B.
In other words:

To send a message to a particular recipient so that only that recipient can read it, you **obtain that recipient's public key, encrypt the message with it, and send the encrypted message**. No matter who intercepts the message, it can be **decrypted only by the intended recipient, with the recipient's private key**.

This is perfectly secure for simply sending messages to specified recipients such that no-one else can read them. However, there is a problem from the recipient's - A's - point of view. How does A know that the message has actually come from B? Anyone could obtain A's public key and send an encrypted message to A, but claim to be B. If the nature of the message needs some guarantee of the identity of the sender, then a greater degree of security is needed, such as using a Digital Signature.

Digital Signatures

When you send a letter to someone, you sign your name at the end. The purpose of this is to authenticate you as the sender, and written signatures have long been used to authenticate the identity of a person. A digital signature serves the same purpose - if A receives a message from B containing B's digital signature, then A can be certain that that message did actually originate from B and it is not a forgery. (As always, this is subject to private keys being kept private, and keys being long enough to prevent computer cracking of the keys. You could argue that this is in fact much more secure than traditional written signatures etc. ever were). Here's one method:

1. B writes the (plaintext) message for A, including a date & time stamp.

2. B then uses a hash formula to create a "message digest" of the time-stamped message. (This is something like a checksum, whose value depends on the whole message including its timestamp. Any subsequent tampering with the message would invalidate the message digest).

3. The message digest is immediately encrypted with B's private key (meaning that only B could have produced this message digest. This is effectively a digital signature).

4. The encrypted message digest is appended to the plaintext message, and the resulting text is encrypted with A's public key (so that it will be readable only by A).

5. The encrypted text is sent.

6. A receives the ciphertext, and decrypts it with A's private key, getting the plaintext message plus a still-encrypted message digest.

7. A obtains B's public key, and uses it to decrypt the message digest.

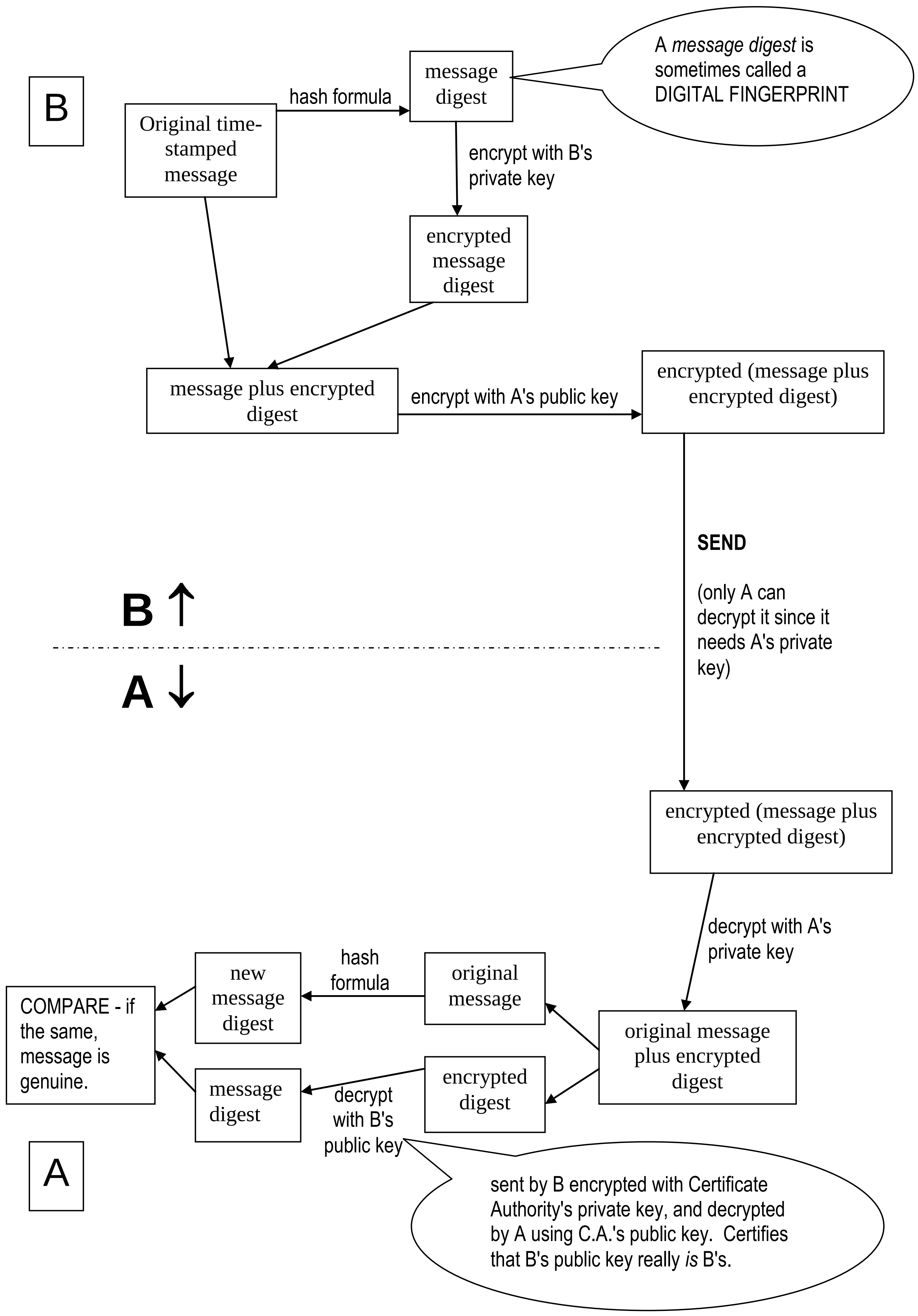
8. A uses the same hash algorithm (there's no secrecy needed with this - it merely computes a value which depends on the entire text, similarly to a checksum) on the plaintext message, generating a message digest of the received message.

9. A then compares the decrypted message digest with that computed from the received message. If they're the same, then this confirms that the received message is indeed what was sent, i.e. it hasn't been tampered with.

The weak link in this is that there is a possibility that the message could have originated from someone purporting to be B, e.g. C. For B above, read C throughout, where C claims to be B. In step 7, C manages to replace B's public key by C's, a not-too-difficult process. This problem can be ameliorated by the use of a Digital Certificate. Basically, this requires the use of a "trusted third party" - a Certificating Authority. Such an authority investigates the identity of an individual, then supplies that individual with an encrypted public key, encrypted by the Certificate Authority's private key. If both sender and recipient accept the Certificate Authority as being trustworthy, then B's public key obtained in this way must be genuinely B's, and not C's. Step 7 above would read:

7. A receives from B also a copy of B's public key, which has been encrypted by the Certificate Authority's private key, and which A now decrypts using the Certificate Authority's public key.

Wikipedia is a good source of further information and links on this topic, e.g.
http://en.wikipedia.org/wiki/Certificate_authority


B
Original time-stamped message
hash formula
message digest
A *message digest* is sometimes called a DIGITAL FINGERPRINT
encrypt with B's private key
encrypted message digest
message plus encrypted digest
encrypt with A's public key
encrypted (message plus encrypted digest)
SEND
(only A can decrypt it since it needs A's private key)
B ↑
A ↓
encrypted (message plus encrypted digest)
decrypt with A's private key
original message plus encrypted digest
original message
encrypted digest
hash formula
new message digest
decrypt with B's public key
message digest
COMPARE - if the same, message is genuine.
sent by B encrypted with Certificate Authority's private key, and decrypted by A using C.A.'s public key. Certifies that B's public key really *is* B's.
A