1.3.1 Compression, Encryption and Hashing

Revision sheet

1 Compression

Compression is needed to reduce the storage space required to store data on disk as well as to reduce the amount of data that has to be transmitted across the internet. There are two types of compression

1.1 Lossy Compression

This works by removing non-essential information. For example, removing detail in images or removing frequencies humans can't here or quiet sounds that can't be heard when louder sounds are playing in audio (MP3 specifically).

1.2 Lossless Compression

This works by recording patterns in data rather than the actual data. With the right instructions, a computer can decompress the data back to its original state. This can be used in text or in images. In images, slight variation in pixel colour might be lost. Lossless compression usually results in a much larger file than that produced by lossy compression, they both are significantly smaller than the original data however.

1.2.1 Run Length Encoding

Run length encoding (RLE) records the value and the number of times that it repeats. For example, if you had an image which had the following pixel sequence *green, green, green, green, green, blue, blue*, then it would be possible to compress it down to *5 green, 2 blue*. This works best for data where there are lots of repeats. If there are not many repeating elements then you can result in negative compression because not only do you have to store the data itself, you have to store how many times it repeats - which could just be one time.

1.2.2 Dictionary Encoding

This works by the compression algorithm working through the text to be compressed and building a dictionary of the of words within the text. For example, the sentence *The quick brown fox jumped over the lazy brown mouse*, could be encoded into the following table.

Number	Entry
1	The
2	quick
3	brown
4	fox
5	jumped
6	over
7	lazy
8	mouse

Table 1: Example of dictionary encoding

Using the table above, the sentence can be compressed as *1 2 3 4 5 6 1 7 2 8*.

2 Encryption

Encryption is the transformation of data from one form to another to prevent an unauthorised third part from being able to understand it. A cipher is applied to the plaintext to achieve the ciphertext, a key can then be applied to the ciphertext to return it to the plaintext form.

2.1 Symmetric Encryption

Also known as private key encryption, uses the same key to encrypt and decrypt the data. The key must also be transferred - in a process called the key exchange which causes security problems as both the ciphertext and key could be intercepted.

2.2 Asymmetric Encryption

This uses two keys, one private and one public. Both the keys are related. The public key is made public so that others wishing to send the recipient data can use this to encrypt the data. The private key is held only by the recipient, which they can then use to decrypt the ciphertext on its arrival. It is possible that a message could be encrypted

using the public key by a malicious third party, a message can be signed which authenticates the sender and prevents the malicious attacks.

3 Hashing

Hashing functions provide a one-way method of encrypting data. The data is processed by a mathematical function which will usually return a fixed length string. This is very useful for storing passwords or PINs as they cannot be read. To authenticate that a password entered is correct, the user entered password would be passed through the hashing algorithm, then the hash in the database would be compared with it.

3.1 Hash Totals

Also referred to as a checksum or digest, this is the mathematical value calculated from unencrypted message data. A single letter change in the input can result in a completely different hash total.

3.2 Digital Signatures

This is like a handwritten signature but more secure. The sender of the message encrypts the hash total of the message with their private key, this forms the signature - it could only be them who encrypted it as only they have their private key. The signature is attached to the message which then gets encrypted with the recipients public key. On receiving the message, the recipient decrypts the message using their private key and then the digital signature using the senders public key. The recipient then uses the same hashing algorithm to generate a hash total for the body of the message. They then compare this hash total to the one decrypted from the sender, if the message hasn't been tampered with then they should be the same. A date or timestamp could be included in this signature to authenticate the message hasn't been resent at a later date.

3.3 Digital Certificates

Digital Signatures can be forged. Digital certificates verify that a senders public key is formally registered to that particular sender. They are issued by an official Certificate Authority and they verify the trustworthiness of a message sender or website. The certificate contains the certificate's serial number, expiry date, name of the holder, copy of the public key and the digital signature of the Certificate Authority who issued the signature so that the recipient can authenticate the certificate as real. Digital Certificates operate within the Transport Layer of the TCP/IP stack.