Compression, Encryption and Hashing

Learning Aims

- Know why sound and images are often compressed
- Understand how other files can be compressed
- Understand the difference between **lossless and lossy compression**
- Explain the advantages and disadvantages of different compression techniques
- Explain **run length encoding** and **dictionary-based compression**
- Define **symmetric and asymmetric encryption**
- Understand how and why **hashing** may be used to encrypt data

Key words

Compression

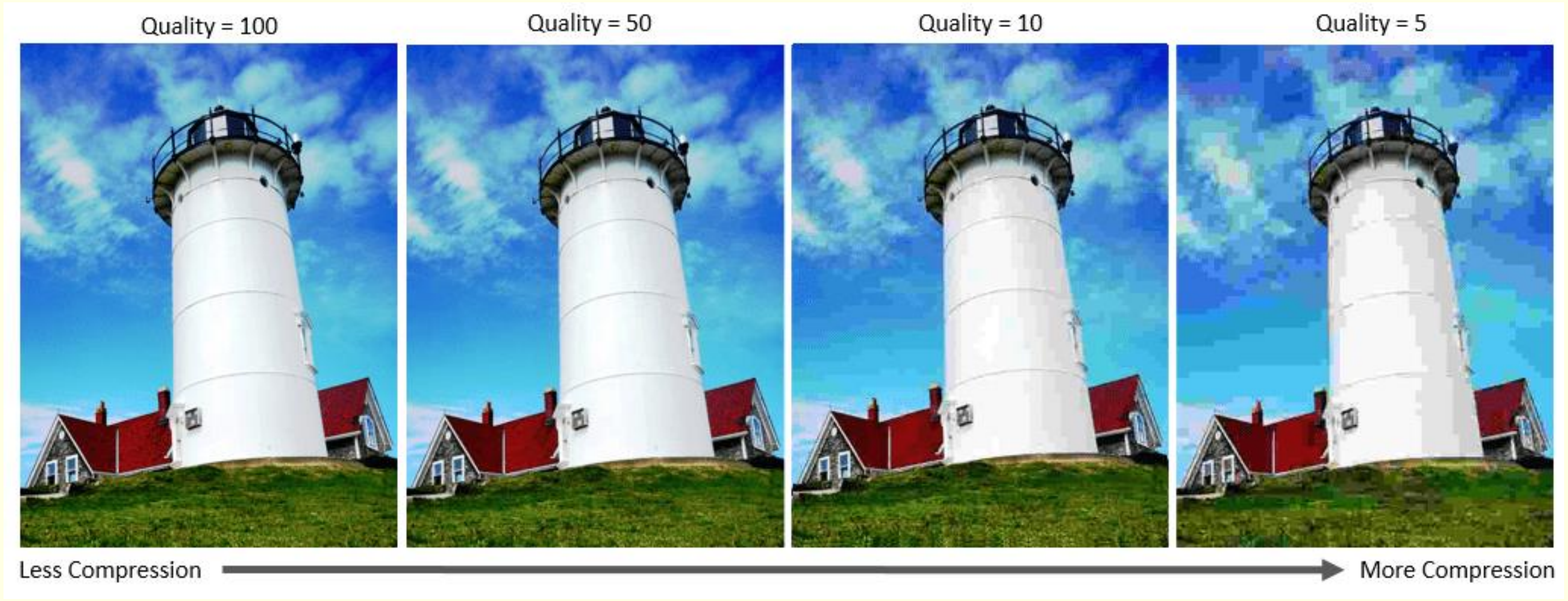
- Lossless compression
- Lossy compression
- Sharing and transmission of data
- Buffering
- Run length encoding
- Dictionary-based compression

Encryption

- Symmetric encryption
- asymmetric encryption
- plaintext
- ciphertext
- Key
- Hashing

Why use compression?

- File compression techniques were developed to **reduce** the **storage space** of files on disk.
- With disk storage becoming larger and cheaper, this is less important these days, but the reduction of file size has become even more important in the **sharing and transmission of data.**



Sharing and transmission of data.

- Internet Service Providers (ISPs) and mobile phone networks impose limits and charges on bandwidth.

- Images on websites need to be in a compressed format to enable a web page to **load quickly** – even on a fast connection, music and video streaming must take advantage of compression in order to **reduce buffering.**

What is buffering?

- In streaming audio or video from the Internet, buffering refers to downloading a certain amount of data to a temporary storage area or buffer, before starting to play a section of the music or movie.

Lossy vs Lossless Compression

Lossy compression

Removes some data to reduce file size, so the original file cannot be recreated when uncompressed.

Used when you need to save space or when a file can afford to lose some data

Lossless compression

- Lossless will not permanently remove data
- Lossless can be fully reconstructed/restored
- Quality (of text/graphics/sound) is not lost
- Any loss of text would be noticeable/would make it unreadable/unusable
- Lossless rewrites data in a more efficient format

Lossless compression

- Lossless compression works by recording patterns in data rather than the actual data. Using these patterns and a set of instructions on how to use them, the computer can **reverse the procedure** and **reassemble an image**, sound or text file with exact accuracy and no data is lost.
- This is most important with the compression of program files, for example, where a single lost character would result in an error in the program code.
- A pixel with a slightly different colour would not be of huge consequence in most
- cases.
- Lossless compression usually results in a **much larger file** than a **lossy file**, but one that is still significantly smaller than the original.

Compression of sound and video

The compression of sound and video works in a similar way.

MP3 files use lossy compression to remove frequencies too high for most of us to hear and to remove quieter sounds that are played at the same time as louder sounds.

The resulting file is about 10% of original size, meaning that 1 minute of MP3 audio equates to roughly 1MB in size.

Voice is transmitted over the Internet or mobile telephone networks using lossy compression and although we have no problem in understanding what the other person is saying, we can recognise the difference in quality of a voice over a phone rather than in person.

The apparent difference is **lost data.**

Methods used for Lossless compression

The purpose of lossless compression is to reduce the amount of storage space needed to save some information, without losing any information hence **without losing quality**.

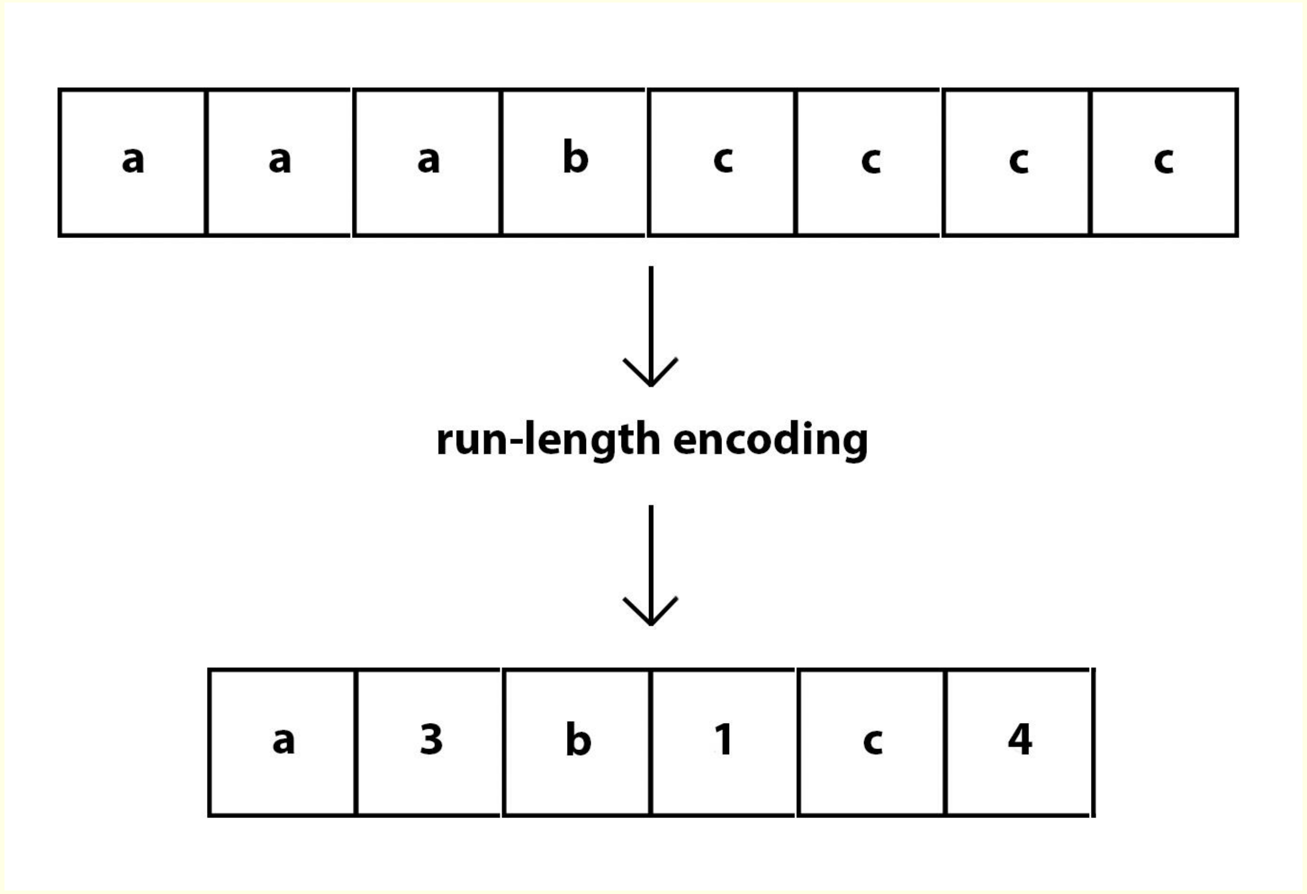
Lossless compression can be used to store text-based information in a more effective way.

The methods used for lossless compression are:

Method 1: Run-length Encoding
Method 2: Dictionary Coding

Go to 101 Computing for more information: https://www.101computing.net/lossless-compression/

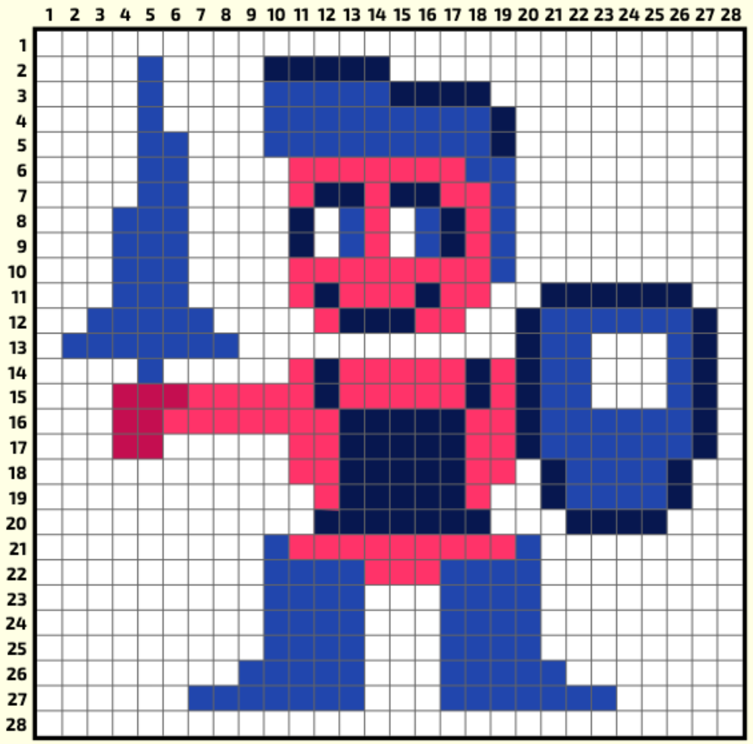
Run length encoding (RLE)



Can you explain how run-time encoding works using the image above?

Run length encoding (RLE)

- RLE is a **lossless** compression technique. It works by finding runs of repeated binary patterns and replacing them with a single instance of the pattern and a number that specifies how many times the pattern is repeated.
- Consider the simple bitmap image.



⬜ **White (W)**

🟦 **Blue (B)**

⬛ **Navy (N)**

🟥 **Pink (P)**

🟪 **Magenta (M)**

The image size is 28 pixels x 28 pixels. Each pixel is represented in binary by a colour code that specifies the colour of a pixel.

To define five colour codes, 3 bits are required (2, cubed, equals, 8,23=8); a colour depth of 3 bits.

The image will be represented as a sequence of colour codes. On the row numbered 6, starting from the left, there are four white pixels, two blue pixels, four white pixels, seven pink pixels, two blue pixels, and nine white pixels. These 28 pixels will require 84 bits of storage. In binary, the row would be stored as:

000 000 000 000 010 010 000 000 000 000 001 001 001 001 001001 0 01 010 010 000 000 000 000 000 000 000 000 000

Compressing the file using RLE

- Instead of storing the colour code for every pixel, the data can be compressed by storing **a single colour code**, followed by the **number of times that the colour is repeated in a run** (i.e. before another colour is found).

- As well as the code, the number of repetitions must be stored. Assuming that five bits are used to store the number of repetitions, the compressed row would look like this:

000 00100 010 00010 000 00100 001 00111 010 00010 000 01001

In RLE all data is expressed as a pair of values (**colour and count**).

In the compressed row shown, the code for white (000,000) is followed by the number four in binary (00100,00100), because there are four white pixels at the start of the run.

This is followed by the code for blue (010,010) and the number two in binary (00010,00010), and so on.

The total number of bits needed to represent the row in compressed form is 48, compared to the 84 bits needed for the uncompressed row.

This technique is known as **lossless**. The exact data in the original file can be re-created from the information in the compressed file by reversing the process.

Dictionary-based compression techniques

- Suppose that instead of sending a complete message, a copy of the Oxford English dictionary was sent alongside a coded message using the page number and the position of the word on that page.
- The word ‘pelican’ falls on page 249 as the 7th word on that page.
- This could be send as 249,7 – using only 2 bytes; considerably fewer than the 7 bytes it would take to send the complete word.

(Ignore, for now, the additional space that it would take to send the dictionary with it!)

- Dictionary based compression works in a similar way.
- The compression algorithm searches through the text to find suitable entries in its own dictionary (or it may use a known dictionary) and translates the message accordingly.

Dictionary-based compression example

- Using the dictionary table above, the saying

"Do unto others as you would have others do unto you"

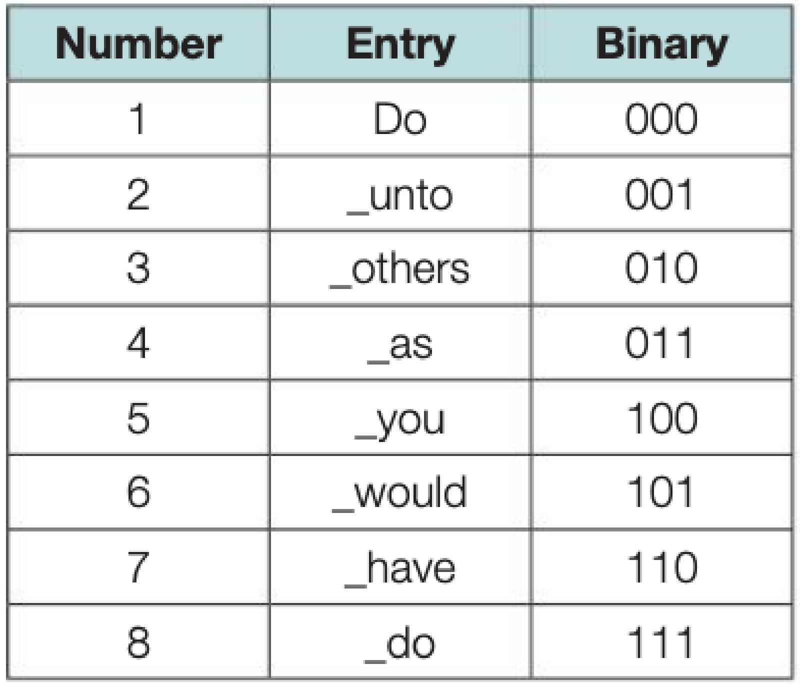
- would be compressed as:

1 2 3 4 5 6 7 3 8 2 5

or in binary using only 33 bits.

This compares to 51 characters or 51 bytes – a reduction of 92%.

This still ignores the fact that the dictionary must also be stored with the text, but with a longer body of text to be compressed, a dictionary becomes quite insignificant in size compared with the original, and the original message can still be reassembled perfectly.



Number	Entry	Binary
1	Do	000
2	_unto	001
3	_others	010
4	_as	011
5	_you	100
6	_would	101
7	_have	110
8	_do	111

Homework: Long Answer Question

A charitable organisation is trying to make the works of William Shakespeare available to more people.

The organisation looks at using either run length encoding or dictionary encoding to compress the file described in part (a).

Discuss the two compression methods and justify which you would recommend. You may refer to the extract of text below to illustrate your argument.

What's in a name? that which we call a rose
By any other name would smell as sweet;
So Romeo would, were he not Romeo call'd, [12]

Encryption Overview

Encryption is a way to protect a message or file by scrambling its contents so that only someone with the right "key" can unlock it. If someone unauthorised gets the file, they won't be able to read it unless they also have the key to decrypt it.

There are two main types of encryption: **symmetric and asymmetric.**

Symmetric encryption uses the same key for both encrypting and decrypting the data. However, this can be risky because the key has to be sent along with the scrambled data, and if someone intercepts it, they can easily unlock the message.

Asymmetric encryption is more secure. It uses two keys:

- A public key that anyone can use to encrypt the data.
- A private key that only you know and use to decrypt the data.

The public key can't be used to unlock the message, so even if someone has it, they can't read the data. The private key is kept secret, and it's nearly impossible to figure out the private key from the public key. This makes asymmetric encryption a very safe way to protect information.

Encryption

Encryption is the transformation of data from one form to another to prevent an unauthorised third party from being able to understand it.

The original data or message is known as **plaintext.**

The encrypted data is known as **ciphertext.**

The encryption method or algorithm is known as the **cipher**, and the secret information to lock or unlock the message is known as a **key**.

Ciphers

The Caesar cipher and the Vernam cipher offer polar opposite examples of security.

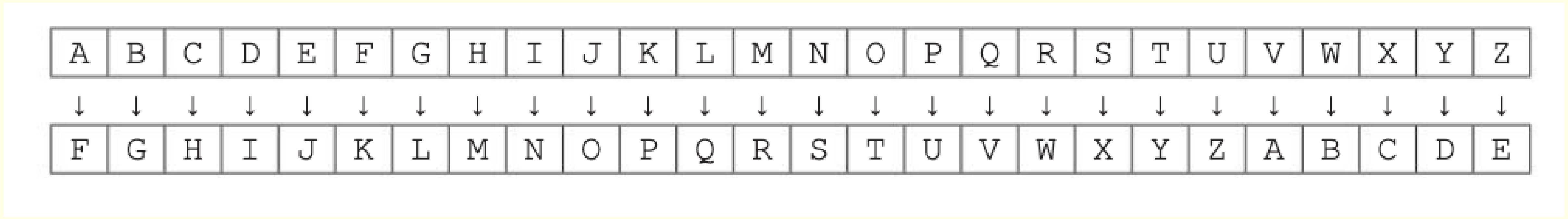
Where the Vernam offers perfect security, the Caesar cipher is very easy to break with little or no computational power.

There are many other methods of encryption – some of which may take many computers many years to break, but almost all of these are still breakable and the principles behind them are similar.

The Caesar cipher

Julius Caesar is said to have used this method to keep messages secure. The Caesar cipher (also known as a shift cipher) is a type of substitution cipher and works by shifting the letters of the alphabet along by a given number of characters; this parameter being the key.

Below is an example of a shift cipher using a key of 5.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E

Brute force & frequency analysis

You will no doubt be able to see the ease with which you can decrypt a message using this system.

DGYDQFH WR ERUGHU DQG DWWDFN DW GDZQ

Even if you had to attempt a **brute force attack** on the message above, there are only 26 different possibilities.

Otherwise you might begin by guessing the likelihood of certain characters first and go from there, known as **frequency analysis.**

Using cryptanalysis on longer messages, you would quickly find the most common ciphertext letter and could start by assuming this was an E, for example; or perhaps an A. (Hint.)

The Vernam cipher - XOR Encryption algorithm

The Vernam cipher, invented in 1917 by the American scientist Gilbert Vernam, is the only cipher **still proven to be unbreakable**.

All others are based on computational security and are theoretically discoverable given enough time, ciphertext and computational power.

One-time pad

The encryption key or one-time pad must be equal to or longer in characters than the plaintext, be truly random and be used only once.

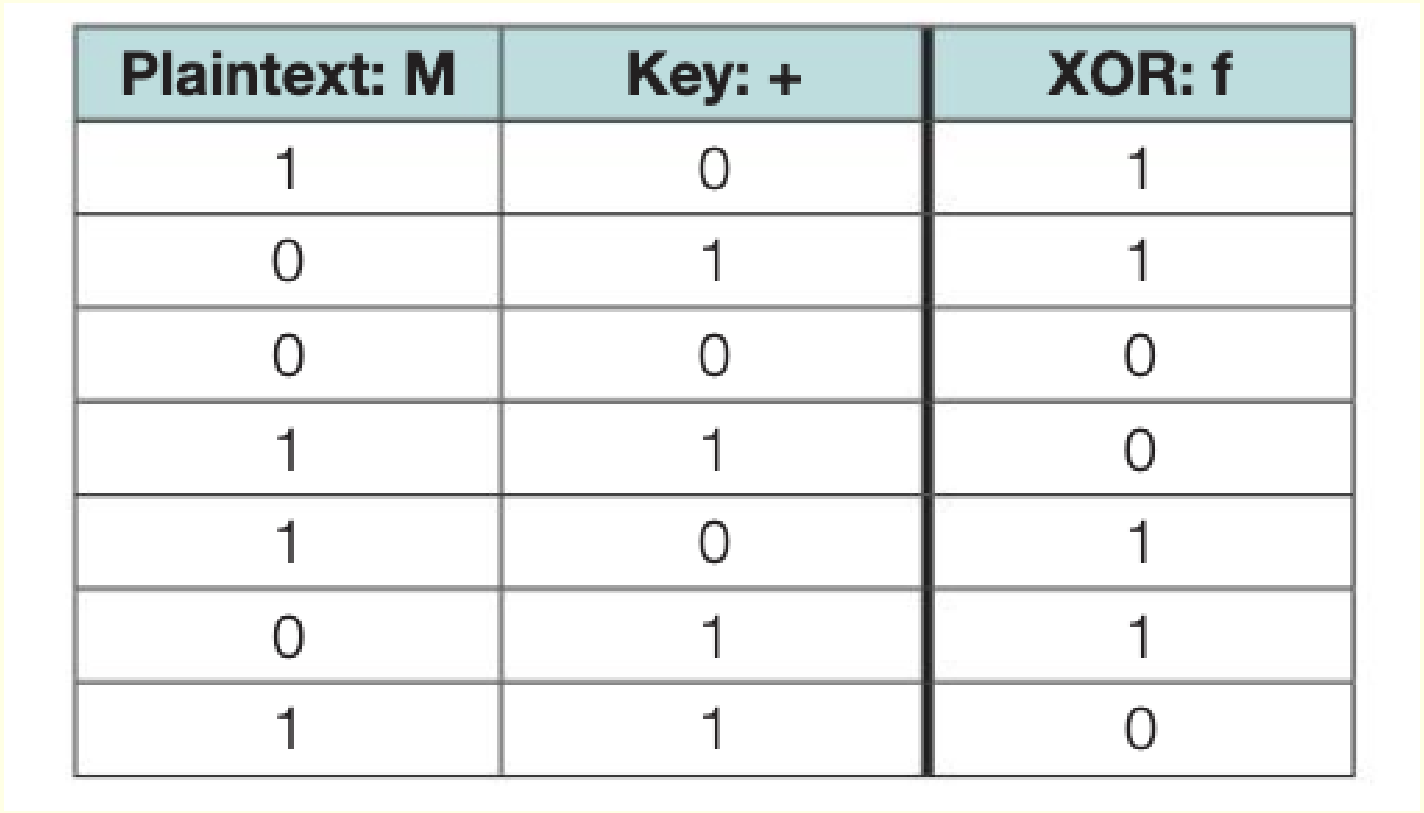
One-time pads are used in pairs where the sender and recipient are both party to the key.

Both must meet in person to securely share the key and destroy it after encryption or decryption.

Since the key is random, so will be the distribution of the characters meaning that no amount
of cryptanalysis will produce any meaningful results.

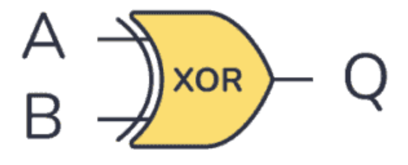
The bitwise exclusive or XOR

An XOR operation is carried out between the binary character value of the first character of the plaintext and the first character of the one-time pad.



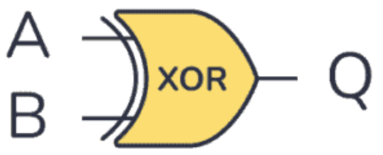
Plaintext: M	Key: +	XOR: f
1	0	1
0	1	1
0	0	0
1	1	0
1	0	1
0	1	1
1	1	0

XOR, or exclusive or, is a logical operator that determines if the number of true inputs is odd

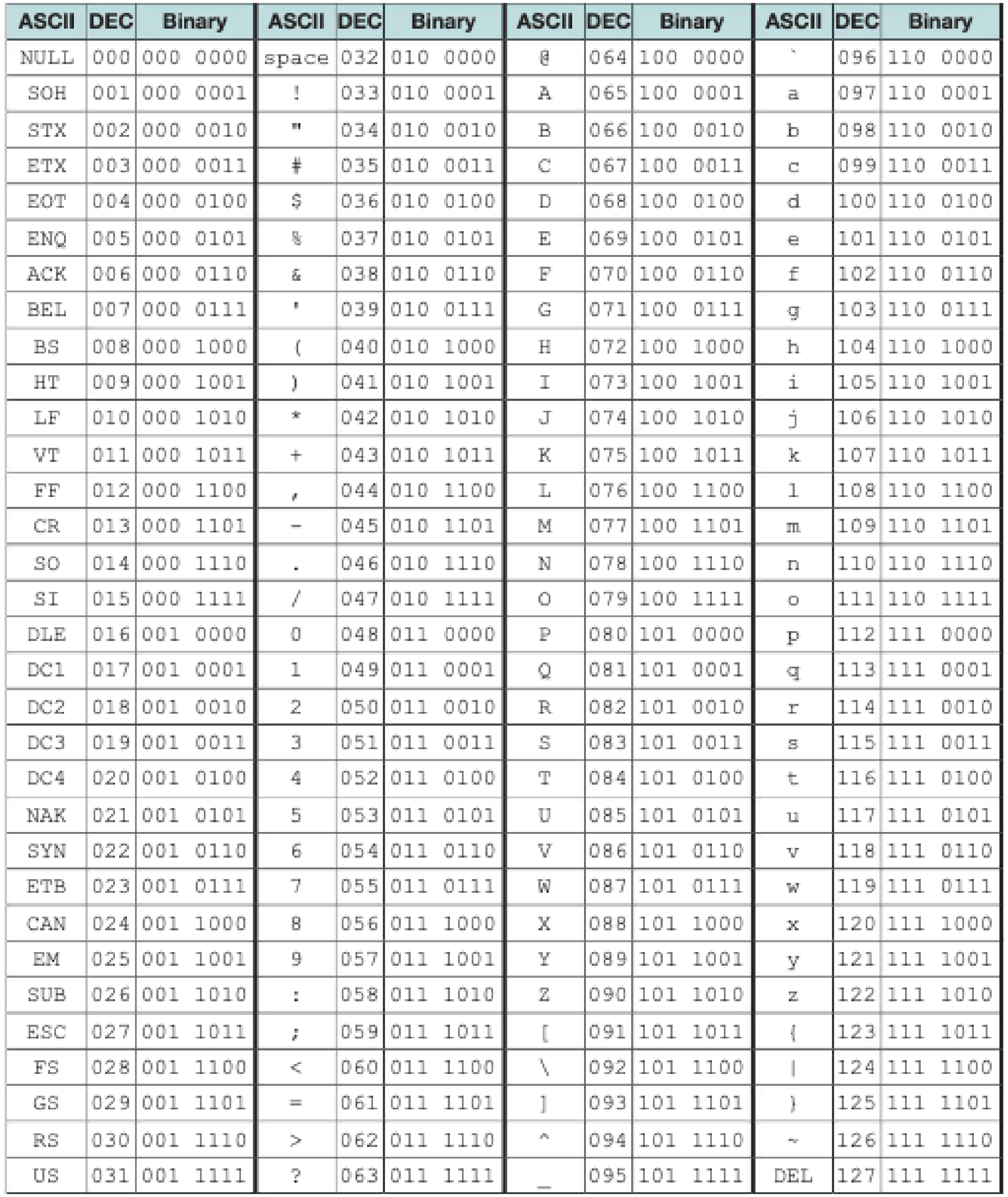


A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Using the ASCII chart and the XOR operator, what ciphertext character will be produced from the letter E with the key w?



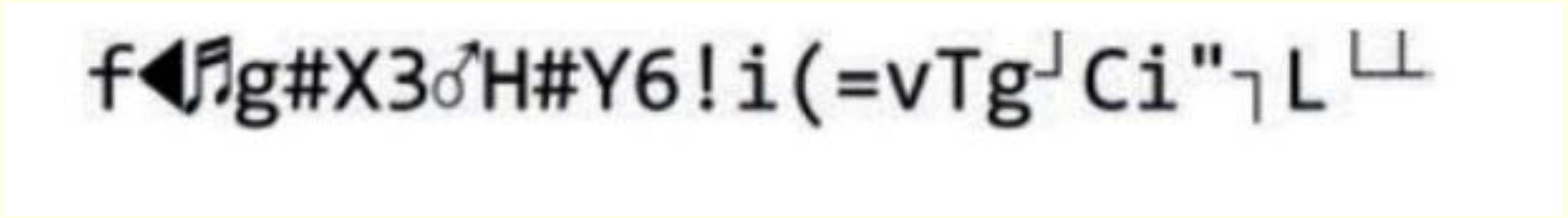
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



ASCII	DEC	Binary	ASCII	DEC	Binary	ASCII	DEC	Binary	ASCII	DEC	Binary
NULL	000	000 0000	space	032	010 0000	@	064	100 0000	`	096	110 0000
SOH	001	000 0001	!	033	010 0001	A	065	100 0001	a	097	110 0001
STX	002	000 0010	"	034	010 0010	B	066	100 0010	b	098	110 0010
ETX	003	000 0011	#	035	010 0011	C	067	100 0011	c	099	110 0011
EOT	004	000 0100	$	036	010 0100	D	068	100 0100	d	100	110 0100
ENQ	005	000 0101	%	037	010 0101	E	069	100 0101	e	101	110 0101
ACK	006	000 0110	&	038	010 0110	F	070	100 0110	f	102	110 0110
BEL	007	000 0111	'	039	010 0111	G	071	100 0111	g	103	110 0111
BS	008	000 1000	(	040	010 1000	H	072	100 1000	h	104	110 1000
HT	009	000 1001	)	041	010 1001	I	073	100 1001	i	105	110 1001
LF	010	000 1010	*	042	010 1010	J	074	100 1010	j	106	110 1010
VT	011	000 1011	+	043	010 1011	K	075	100 1011	k	107	110 1011
FF	012	000 1100	,	044	010 1100	L	076	100 1100	l	108	110 1100
CR	013	000 1101	-	045	010 1101	M	077	100 1101	m	109	110 1101
SO	014	000 1110	.	046	010 1110	N	078	100 1110	n	110	110 1110
SI	015	000 1111	/	047	010 1111	O	079	100 1111	o	111	110 1111
DLE	016	001 0000	0	048	011 0000	P	080	101 0000	p	112	111 0000
DC1	017	001 0001	1	049	011 0001	Q	081	101 0001	q	113	111 0001
DC2	018	001 0010	2	050	011 0010	R	082	101 0010	r	114	111 0010
DC3	019	001 0011	3	051	011 0011	S	083	101 0011	s	115	111 0011
DC4	020	001 0100	4	052	011 0100	T	084	101 0100	t	116	111 0100
NAK	021	001 0101	5	053	011 0101	U	085	101 0101	u	117	111 0101
SYN	022	001 0110	6	054	011 0110	V	086	101 0110	v	118	111 0110
ETB	023	001 0111	7	055	011 0111	W	087	101 0111	w	119	111 0111
CAN	024	001 1000	8	056	011 1000	X	088	101 1000	x	120	111 1000
EM	025	001 1001	9	057	011 1001	Y	089	101 1001	y	121	111 1001
SUB	026	001 1010	:	058	011 1010	Z	090	101 1010	z	122	111 1010
ESC	027	001 1011	;	059	011 1011	[	091	101 1011	{	123	111 1011
FS	028	001 1100	<	060	011 1100	\	092	101 1100	|	124	111 1100
GS	029	001 1101	=	061	011 1101	]	093	101 1101	}	125	111 1101
RS	030	001 1110	>	062	011 1110	^	094	101 1110	~	126	111 1110
US	031	001 1111	?	063	011 1111	_	095	101 1111	DEL	127	111 1111

The Vernam cipher

- Using this method, the message “Meet on the bridge at 0300 hours” encrypted using a one-time pad
- of +tkiGeMxGvnhoQ0xQDIIIVdT4sIJm9qf will produce the ciphertext:



f◄♫g#X3♂H#Y6!i(=vTg┘Ci"┐L╨

The encryption process will often produce strange symbols or unprintable ASCII characters as in the above example, but in practice it is not necessary to translate the encrypted code back into character form, as it is transmitted in binary.

To decrypt the message, the XOR operation is carried out on the ciphertext using the same one-time pad, which restores it to plaintext.

Cryptanalysis and perfect security

Other ciphers that use non-random keys are open to a cryptanalytic attack and can be solved given enough time and resources.

Even ciphers that use a computer generated random key can be broken

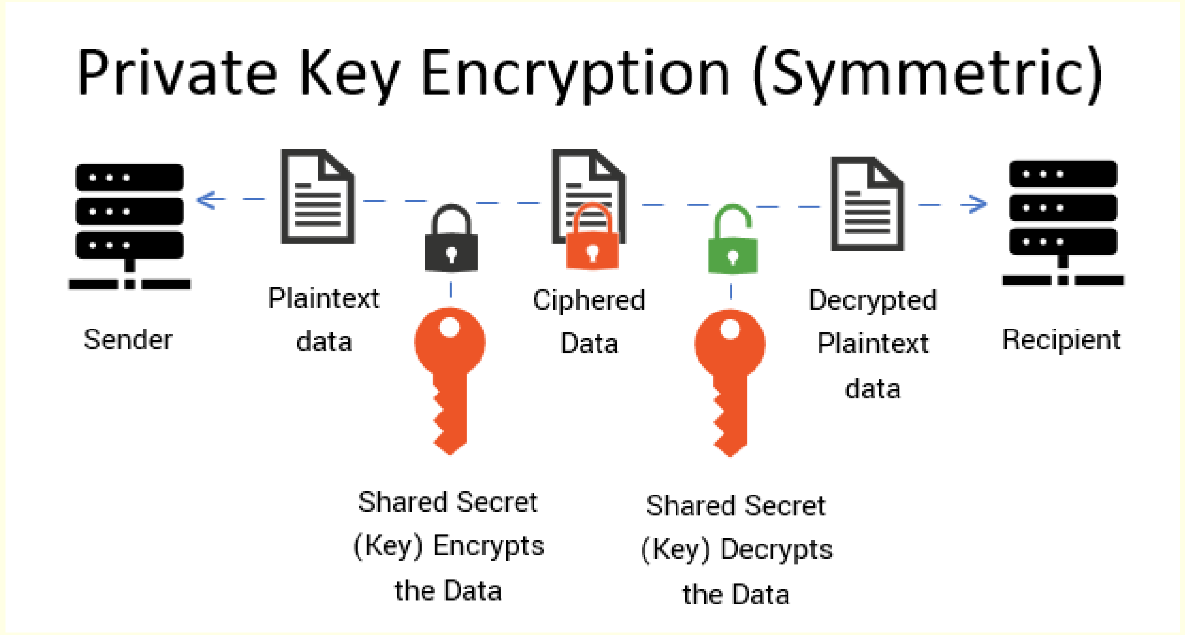
since mathematically generated random numbers are not actually random; they just appear to be random.

A truly random sequence must be collected from a physical and unpredictable phenomenon such as white noise, the timing of a hard disk read/write head or radioactive decay.

Only a truly random key can be used with a Vernam cipher to ensure it is mathematically impossible to break.

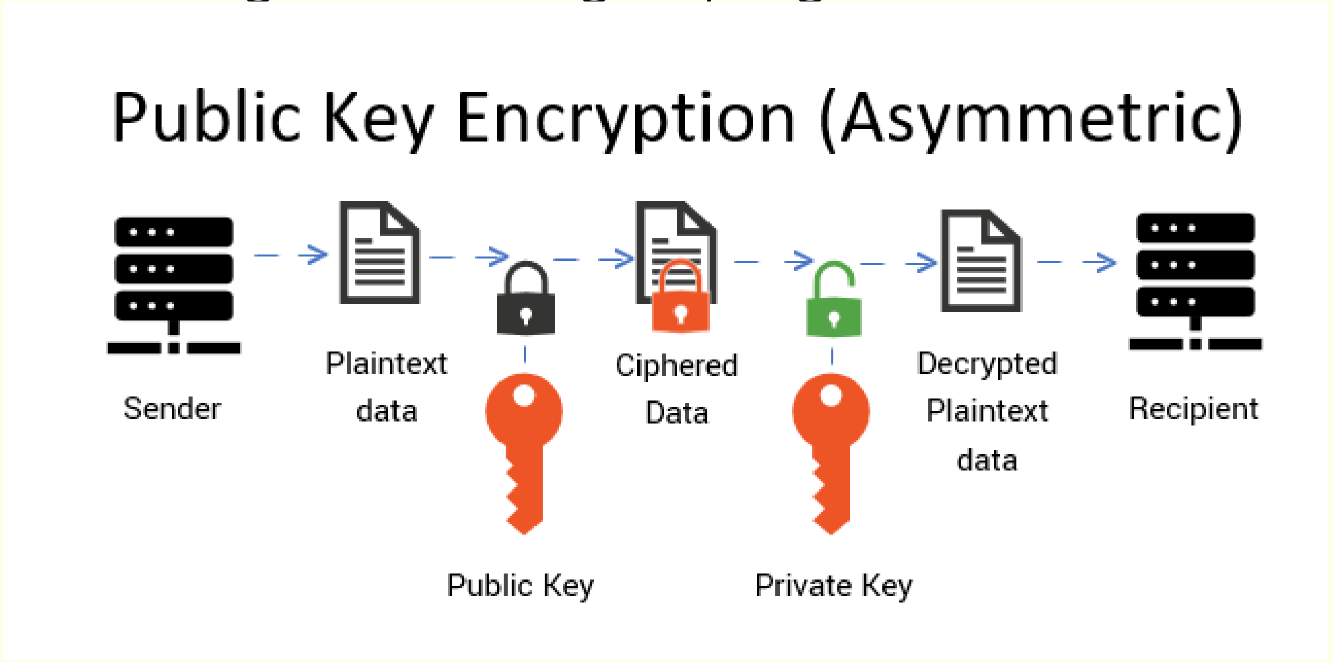
Symmetric (private key) encryption

- Symmetric encryption, also known as private key encryption, **uses the same key to encrypt and decrypt data.**
- This means that the key must also be transferred (known as key exchange) to the same
- destination as the ciphertext, which causes obvious security problems.
- The key can be intercepted as easily as the ciphertext message to decrypt the data.
- For this reason asymmetric encryption can be used instead.

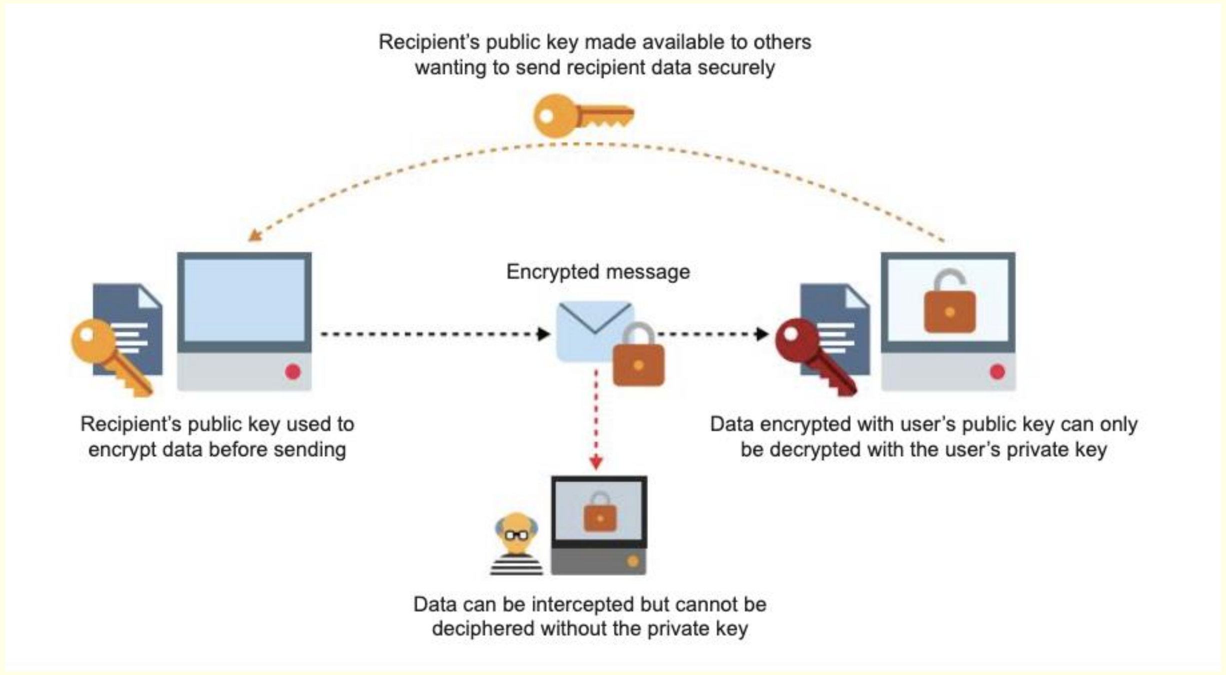

Private Key Encryption (Symmetric)
Sender
Plaintext data
Shared Secret (Key) Encrypts the Data
Ciphered Data
Shared Secret (Key) Decrypts the Data
Decrypted Plaintext data
Recipient

Asymmetric (public key) encryption

- Asymmetric encryption uses two separate, but related keys.
- **Public key**, is made public so that others wishing to send you data can use this to encrypt the data. This public key cannot decrypt data.
- **Private key** is known only by you and only this can be used to decrypt the data.
- It is virtually impossible to deduce the private key from the public key.
- It is possible that a message could be encrypted using your own public key and sent to you by a malicious third party impersonating a trusted individual.
- To prevent this, a message can be digitally 'signed' to authenticate the sender.

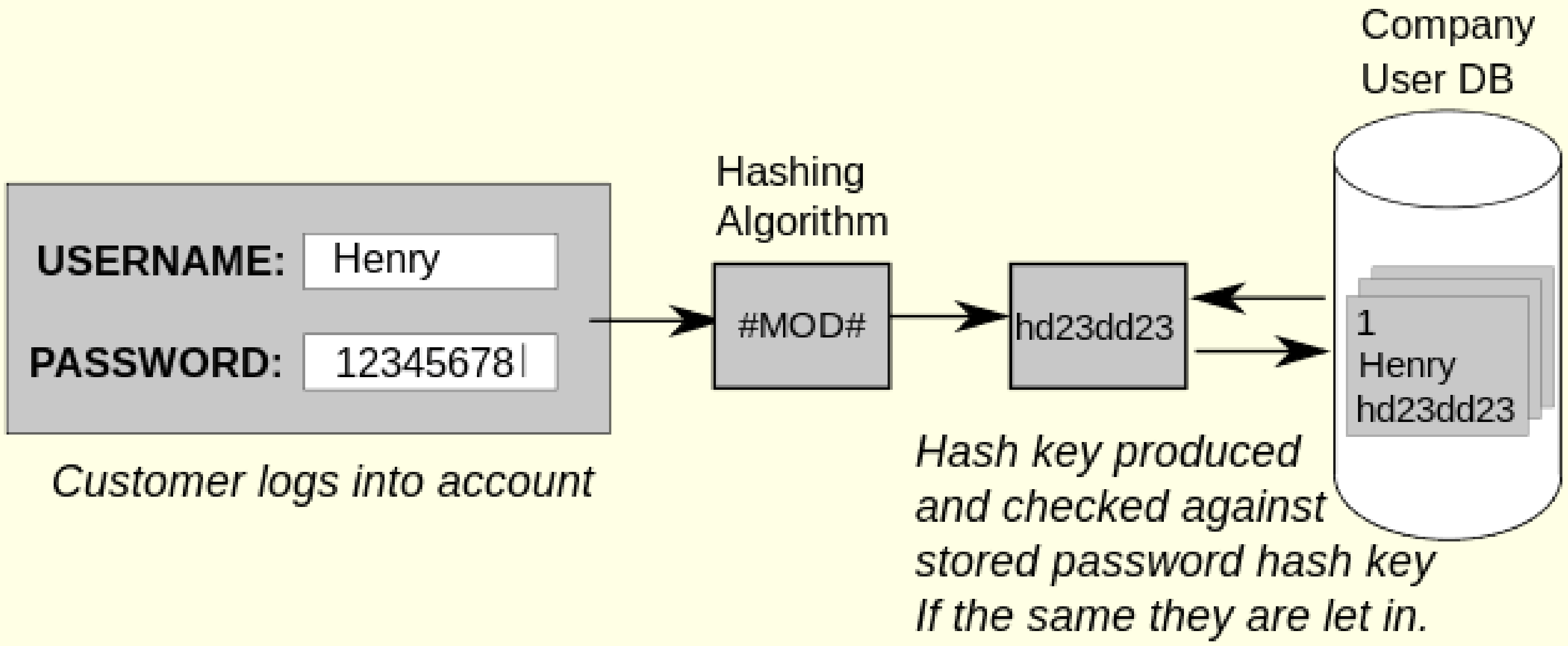


Asymmetric (public key) encryption


Recipient's public key made available to others wanting to send recipient data securely
Encrypted message
Recipient's public key used to encrypt data before sending
Data encrypted with user's public key can only be decrypted with the user's private key
Data can be intercepted but cannot be deciphered without the private key

Hashing

- Unlike the encryption techniques described, **it is one-way**; you cannot get back to the original.
- This is useful for **storing encrypted PINs and passwords** so that they cannot be read by a hacker.
- To verify a user's password, the software applies the hash function to the user input and compares it with the one stored.


USERNAME: Henry
PASSWORD: 12345678
Customer logs into account
Hashing Algorithm
#MOD#
hd23dd23
Company User DB
1
Henry
hd23dd23
Hash key produced and checked against stored password hash key If the same they are let in.

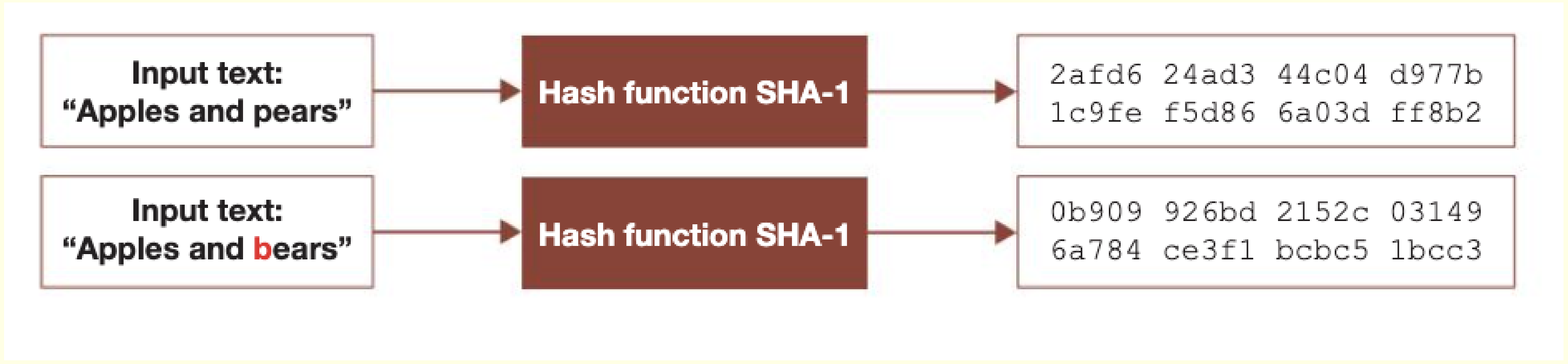
Cryptographic hash functions

A hash total is a mathematical value calculated from unencrypted message data.

This value is also referred to as a checksum or digest.

The process is irreversible and impossible to crack other than by trying all of the possible inputs until a match is found.

Since the hash total is generated from the entire message, even the slightest change in the message will produce a different total.


Input text: "Apples and pears"
Hash function SHA-1
2afd6 24ad3 44c04 d977b 1c9fe f5d86 6a03d ff8b2
Input text: "Apples and bears"
Hash function SHA-1
0b909 926bd 2152c 03149 6a784 ce3f1 bcbc5 1bcc3

Digital signatures

A **digital signature** is like a more secure version of a handwritten signature or stamp.

Here's how it works:

1. The sender creates a **hash value** (a unique summary of the message).
2. They use their **private key** to encrypt the hash value. This encrypted hash becomes the **digital signature**, proving the message came from them.
3. The digital signature is attached to the message.
4. Before sending, the sender encrypts the entire message (including the signature) with the recipient's **public key**.

Digital signatures

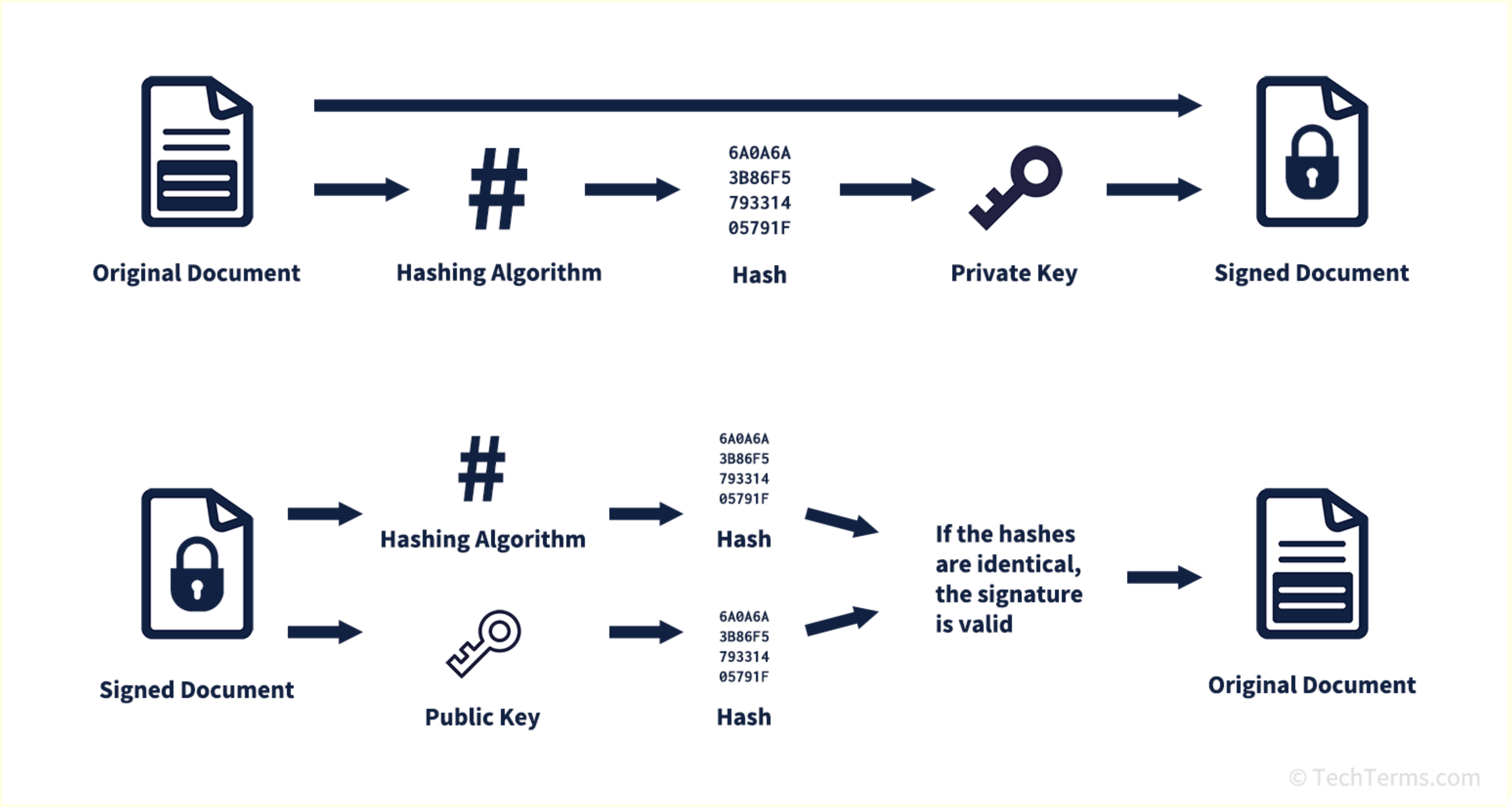
When the recipient gets the message:

1. They decrypt it using their **private key**.
2. They decrypt the digital signature using the sender's **public key**.
3. The recipient recalculates the hash value from the message. If this matches the hash in the digital signature, the message is confirmed to be authentic and unchanged.

To prevent someone from reusing the message later, a **timestamp** can be added.

Changing the timestamp or message would create a different hash, making tampering obvious.

Digital signatures



Digital signatures

- **Digital signatures** can be used with or without encryption.
- They can be used with most email clients or browsers making it easy to sign outgoing communications and validate signed incoming messages.
- If set up to use digital signatures, your browser should warn you if you download something that does not have a digital signature.
- This would also mean that anything sent by you, including online commercial and banking transactions, can be verified as your own.

Hoax digital signatures

Hoax digital signatures could be created using a bogus private key claiming to be that of a trusted individual.

In order to mitigate against this, a digital certificate verifies that a sender's public key is **formally** registered to that particular sender.

Digital certificates

- While digital signatures verify the trustworthiness of message content, a digital certificate is issued by official Certificate Authorities (CAs) such as Symantec or Verisign and verifies the trustworthiness of a message sender or website.
- This certificate allows the holder to use the Public Key Infrastructure or PKI.
- The certificate contains the certificate's serial number, the expiry date, the name of the holder, a copy of their public key, and the digital signature of the CA so that the recipient can authenticate the certificate as real.
- Digital certificates operate within the Transport layer of the TCP/IP protocol stack.

(TCP/IP will be covered in the network topic)

IoT and Security Issues

Encryption

Encryption is two-way, do data can be restored to it's original form.

Useful for data transmission as data intercepted cannot be decrypted with out the key.

So data been sent to the cloud will need to use encryption so the data can be read and analysed. There are two types of ency[tion

Symmetric and Asymmetric (will will study this in another lesson)

WIFI Security Password

Hashing is useful for data storage of a password.

To verify the device has access to the network, the hash of the input will be compared with the stored hash, to confirm correctness.

The password never sent across the wifi only the hash.

The password is hashed and stored in router.

Hashing is useful for information that needs to be verified but does not need to be known at any point, once hashed, it is impossible to return to it.

Hashing is one way mathematical process that produces a value for the input value.

Homework: Long Answer Question

Modern encryption is much stronger than the method described in the first part of this question.

Discuss the impact of modern encryption on society. You should refer to:

- The importance of asymmetric encryption and how it differs from symmetric encryption.
- Different circumstances in which symmetric and asymmetric encryption may be used. [9]