THE UNIVERSITY OF QUEENSLAND
AUSTRALIA
CREATE CHANGE

Business Information Security

- Week 06: Cryptography Part 1 (Ch. 8)

Dr Alex Pudmenzky

Semester 2, 2024

Private key cryptography (symmetric cryptography) and hashing

Introduction – some definitions

- **Encryption**: converting original message into a form unreadable by unauthorised individuals – a confidentiality strategy. Does it physically protect the data from access?
- **Cryptography**: the science of making and using codes to secure transmission of information
- **Cryptanalysis**: process of obtaining original message from encrypted message without knowing algorithms
- **Cryptology**: an area of science (mathematics, language theory, engineering); combines cryptography and cryptanalysis - the science of encryption

More terminology

- **Algorithm** – a procedural description of an activity

- **Cipher:** an algorithm for performing encryption or decryption

- **Plain text** – the message/data/information we apply the encryption algorithm to
- **Cipher text** – the result after we encrypted the plain text

- **Key** – the mapping instrument used within the cipher
- **Key space** – the complexity of the key

- **Stream cipher** - each plain text bit transformed into cipher bit one bit at a time
- **Block cipher** - message divided into blocks (e.g., sets of 8- or 16-bit blocks) and each is transformed into encrypted block of cipher bits using algorithm and key

Certain encryption approaches can also be used for authentication

Some important trends in cryptography

- With emergence of technology, need for encryption in information technology environment greatly increased

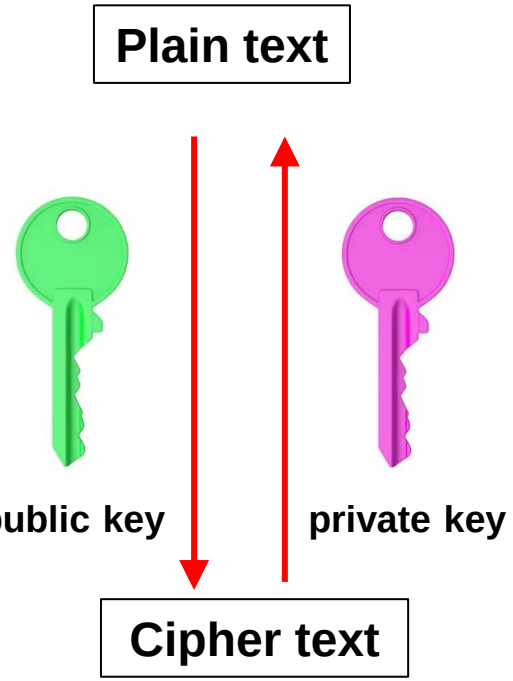
- The Internet (i.e., the original protocols) initially did not have any security focus for transmission of data – *we shall talk about this more in a few weeks*.

- All popular Web browsers and email user agents use built-in encryption features for security (i.e. certain security protocols are built into the user applications).
  - *We look at **S/MIME** (email) and **TLS** (web) in significant business detail next week*

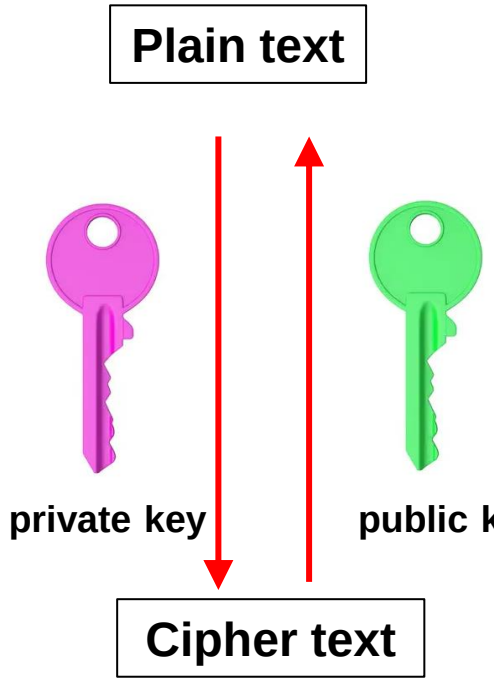
- Restrictions on the export of cryptosystems began after WWII. (The **Clipper chip** was a chipset that was developed and promoted by the United States National Security Agency (NSA) as an encryption device (combating “going dark”), with a built-in backdoor, intended to be adopted by telecommunications companies for voice transmission. It was announced in 1993 and by 1996 was entirely defunct).

Cryptography

Two main paradigms (both heavily used): **Private key (symmetric)** and **Public key (asymmetric)**.



Confidentiality because only the holder of the private key (the intended recipient) can decrypt the message



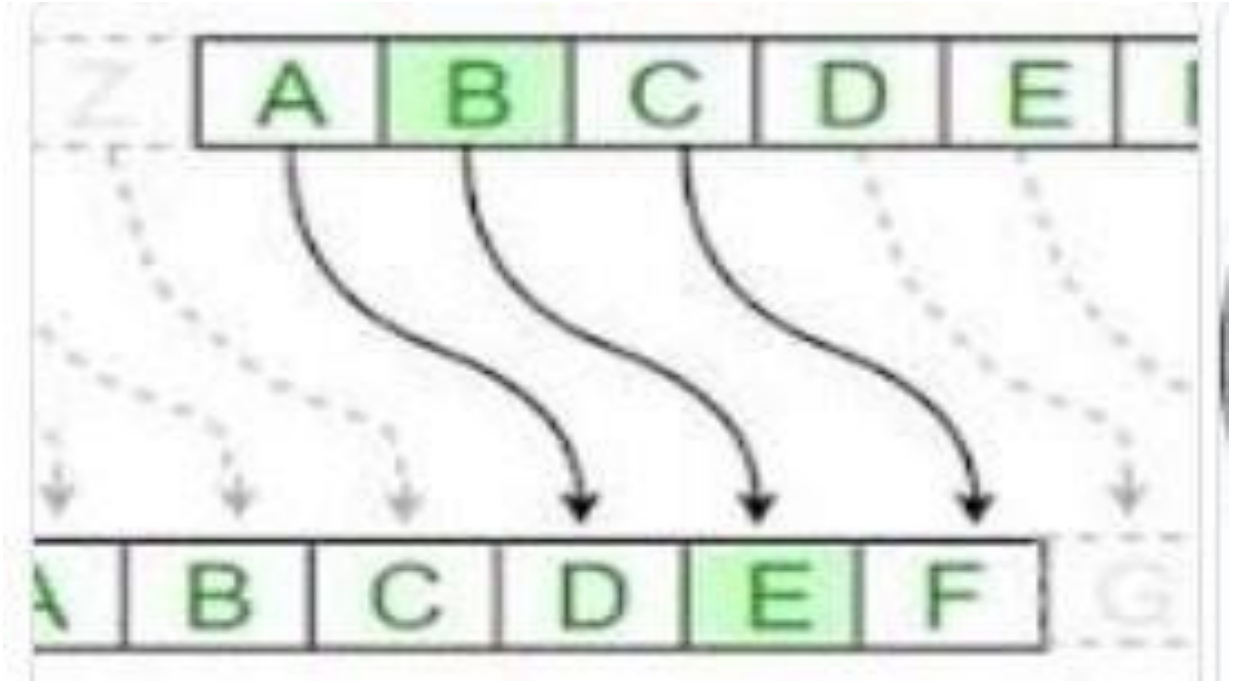
Authentication and **Integrity** because it confirms that the message was signed by the holder of the private key and that the message has not been altered.

Symmetric key
Has been around a long time (Caesar cipher).

Asymmetric key
Theory evolved in late 70s, the first major change in cryptography in a long time.

The Caesar cipher

The Caesar cipher (100 BC) is one of the earliest known and simplest ciphers. It is a type of **substitution cipher** in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on.



Stream cipher - How to create

The basic idea behind a stream cipher is to generate a **keystream** – a sequence of bits or bytes – that is then combined with the plaintext to produce the ciphertext. The keystream is generated based on an initial key and, often, an additional value called an **Initialization Vector (IV)**.

Setup

Key: 10101010 (8 bits) - must be secret

Initialization Vector (IV): 11001100 (8 bits) - does not have to be secret

XOR

Keystream: 01100110

Plaintext "**f**": 01100110 ENCODING

XOR

Ciphertext: 00000000 (**NULL**)

Keystream: 01100110 DECODING

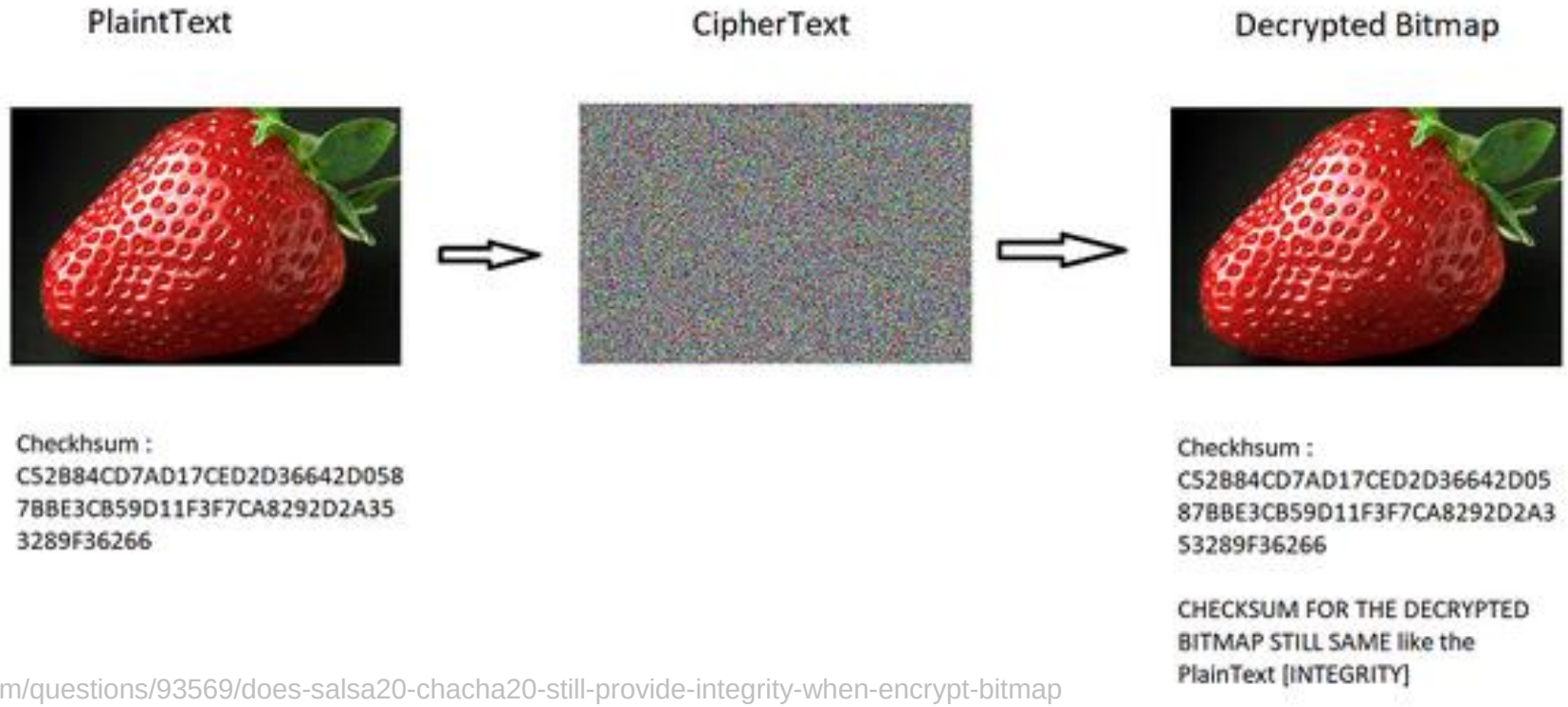
XOR

Plaintext "**f**": 01100110

Stream cipher - Application

RC4: One of the most well-known stream ciphers, RC4, was widely used in protocols like SSL/TLS and WEP. However, RC4 is now considered insecure due to several vulnerabilities and should be avoided.

Salsa20/ChaCha20: These are modern stream ciphers that are designed to be secure, fast, and efficient. ChaCha20, for example, is used in the HTTPS protocol for securing web traffic.



https://crypto.stackexchange.com/questions/93569/does-salsa20-chacha20-still-provide-integrity-when-encrypt-bitmap

Block cipher (operates on fixed length groups of bits called blocks)

A **block cipher** is a type of symmetric encryption algorithm that encrypts data in fixed-size blocks, typically 64 or 128 bits at a time. It is one of the fundamental building blocks of modern cryptography, widely used to secure data in various applications, including SSL/TLS for internet security, disk encryption, and more.

Two widely used block cipher algorithms: **AES (Advanced Encryption Standard)** and **DES (Data Encryption Standard)**.

DES (Data Encryption Standard)
DES is one of the earlier block cipher algorithms developed in the 1970s. It operates on 64-bit blocks and uses a 56-bit key. DES uses a 16-round Feistel structure†, a common design pattern for block ciphers, where each round involves substitution, permutation, and mixing with the key.

AES (Advanced Encryption Standard)
AES is the current standard for block ciphers, designed to be secure and efficient. It supports key sizes of 128, 192, or 256 bits and operates on 128-bit blocks.

† A Feistel structure is a symmetric block cipher design that splits the data into two halves and iteratively applies a round function, combining and swapping the halves to achieve encryption and decryption using the same algorithm and key.

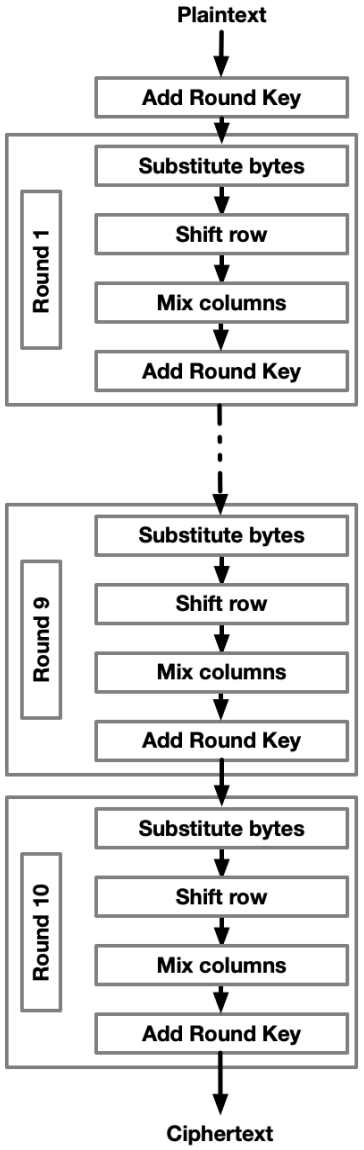
Block cipher algorithm

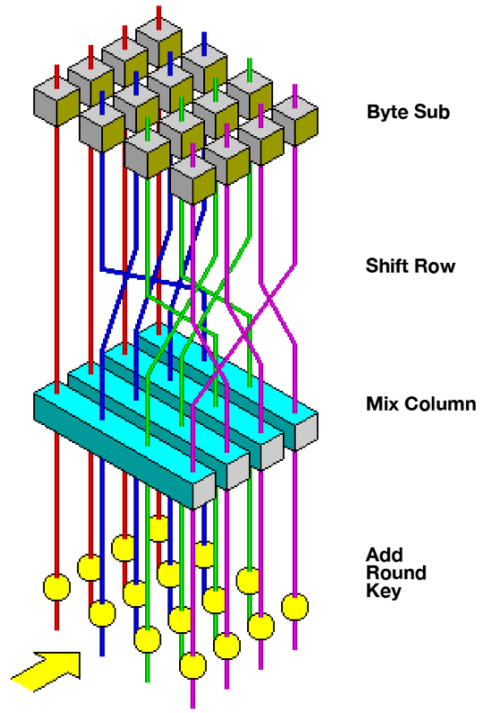
AES Encryption Process

1. Key Expansion: The original key is expanded into multiple round keys using a key schedule algorithm.
2. Initial Round:
   1. **AddRoundKey**: The plaintext block is XORed with the first round key.
3. Main Rounds (9, 11, or 13 rounds, depending on the key size):
   1. **SubBytes**: Each byte of the block is replaced with a corresponding byte from an S-box (substitution box) for non-linear substitution.
   2. **ShiftRows**: Rows of the block are shifted cyclically to the left to introduce diffusion.
   3. **MixColumns**: Columns of the block are mixed using linear transformation to further spread the plaintext information across the block.
   4. **AddRoundKey**: The block is XORed with a round key derived from the original key.
4. Final Round: The last round omits the MixColumns step and consists of SubBytes, ShiftRows, and AddRoundKey operations.

Decryption Process

AES decryption is performed using the inverse operations of encryption, with the round keys applied in reverse order.





Symmetric key algorithms

Symmetric key cryptography is the oldest form but still heavily used. Simple definition:

- "*A single key is used to encrypt and decrypt*"

These two ciphers are the basis for most/all symmetric key ciphers:

- ***Substitution* cipher**: substitutes or exchanges one value for another
- ***Transposition* cipher:** rearranging the values within a block based on an established pattern.

Note: We are not discussing the details of the variations on these two ciphers, the Vigenere cipher (1553) or Vernam cipher (1917). However, some interactive tools exist to encrypt and decrypt the Vigenere cipher and others on the Internet:

- *https://cryptii.com/pipes/vigenere-cipher*
- *https://www.dcode.fr/vigenere-cipher*
- *https://www.guballa.de/vigenere-solver*

Substitution Example

Decrypt the following line of cipher text (a simple substitution cipher with a three character key has been used). Use the underscore character (_) to represent the space character.

ehtwrmsrohsrmtnsy

The key used in the above encryption strategy is

space => r

e => s

o => t

c) The algorithm for the above encryption and decryption strategy

Starting with the first character, consider each letter of the cipher/plain text in turn. If the letter appears in the key, make the substitution – else leave the letter unchanged. Move to the next letter.

Answer: show_me_the_money

Cryptanalysis of substitution

The most frequently occurring letters in the English language, in order of decreasing frequency, are:

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

ehtwrmsrohsrmtnsy

So cryptoanalysis of the cypher text would quickly see that R is the most commonly occuring character.

In English language the most commonly occuring character in a block of text is the space.

So immediately, if we were using cryptanalysis to try and hack, this we would quickly try R being replaced by the space.

ehtw ms ohs mtnsy

And immediately we would break the cypher text into words.

And by simple trial and error approach, we would quickly deduce that {space => r, e => s, o => t} is the encryption/decryption key.

show me the money

So, the point is that this type of encryption can be vulnerable to what we call frequency distribution in the underlying language, that means that cryptanalysis of this cypher text is easily produced.

Transposition Example

f) Decrypt the following line of cipher text (a simple transposition cipher with a four character key has been used). The underscore character (_) represents the space character.

_isheamn_is_nohj

g) The key used in the above encryption strategy is *{1234 becomes 4231}*

That is, within each block of four characters, the first and fourth characters are swapped.

_ish eamn _is_ nohj

his_ name _is_ john

Is the cipher used above *block* or *stream* and is it *symmetric* or *asymmetric*?

Cryptographic algorithms

Symmetric encryption

Uses same “secret key” (also known as private key) for encoding and decoding

- Encryption methods can be extremely efficient, requiring minimal processing
- Both sender and receiver must possess this key
- If either copy of key is compromised, an intermediate can decrypt and read messages
- Major difficulty: ‘the key distribution’, how do you get this key secretly to the other party?
- This is often done ‘out of band’

Consider how your access to the UQ network was initially set up

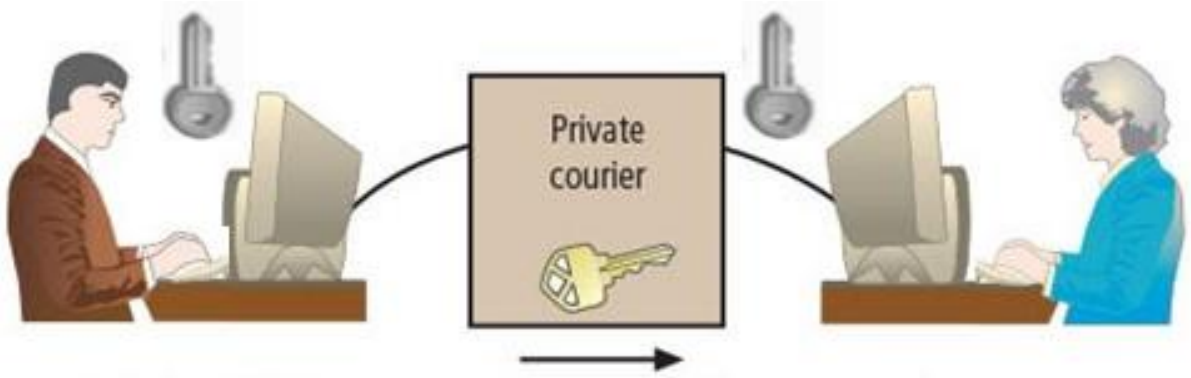
The '*key distribution*' problem

- Simply stated:
  - **How to get a key from Bob to Alice – maintaining the secrecy of the key and**
  - **ensuring the key is only used by Alice for the appropriate time period**

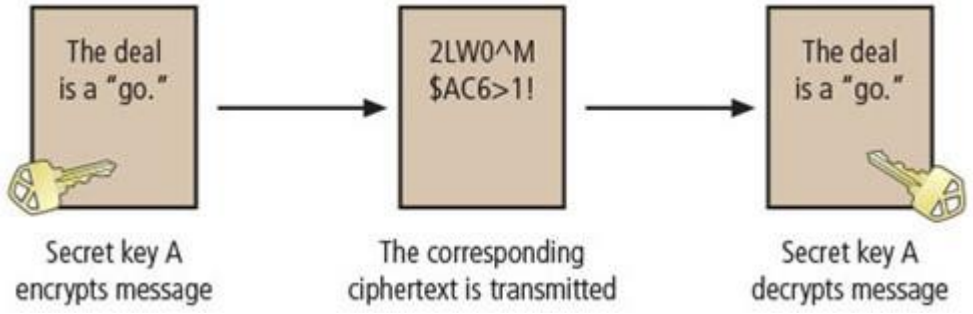
- Key distribution becomes very much harder as the cryptosystem scales up in size (i.e., more participants - one key for each?)
- Key distribution is a very significant issue in designing modern security protocols

Consider your access to the UQ computing infrastructure – what is the 'key', how is key granting and key distribution administered – the significance (difficulty/cost) of 'out of band' solutions

Symmetric key cryptography



Bob at ABC corp. generates a secret key. He must somehow get it to Alice at XYZ corporation, **out of band**. Once Alice has it, Bob can use it to encrypt messages, and Alice can use it to decrypt and read them.



Encrypting binary data

- All alpha-numeric data (whether stored, processed, transmitted) is represented in *binary* format (i.e., as a *number consisting of digits 0 and 1 only*)
- This binary data is converted back to the appropriate 'human type' when we need it
- We already talked about how to convert data like this using the ASCII, extended ASCII, and Unicode tables last week
- **All encryption/decryption is performed on binary data** (not alpha-numeric data). That is, **all data (text, numbers, graphics, everything) is represented at the memory/processor level as series of digits 0 and 1**

Example "**Alex1.**"

A	**l**	**e**	**x**	**1**	**.**
01000001	**01101100**	**01100101**	**01111000**	**00110001**	**00101110**
41	**6C**	**65**	**78**	**31**	**2E**

To encrypt this we select an encryption key that will also be stored in binary format – **so we end up with all binary data!**

Full ASCII Table 1/2

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

Binary	Hex	Decimal	esc	Char	Description
0000 0000	0	0	\0	NUL	Null character
0000 0001	1	1		SOH	Start of Header
0000 0010	2	2		STX	Start of Text
0000 0011	3	3		ETX	End of Text
0000 0100	4	4		EOT	End of Transmission
0000 0101	5	5		ENQ	Enquiry
0000 0110	6	6		ACK	Acknowledgment
0000 0111	7	7	\a	BEL	Bell
0000 1000	8	8	\b	BS	Backspace
0000 1001	9	9	\t	HT	Horizontal Tab
0000 1010	A	10	\n	LF	Line Feed
0000 1011	B	11	\v	VT	Vertical Tab
0000 1100	C	12	\r	FF	Form Feed
0000 1101	D	13		CR	Carriage Return
0000 1110	E	14		SO	Shift Out
0000 1111	F	15		SI	Shift In
0001 0000	10	16		DLE	Data Link Escape
0001 0001	11	17		C1 (XON)	Device Control 1
0001 0010	12	18		DC2	Device Control 2
0001 0011	13	19		DC3(XOFF)	Device Control 3
0001 0100	14	20		DC4	Device Control 4
0001 0101	15	21		NAK	Negativ Acknowledgemnt
0001 0110	16	22		SYN	Synchronous Idle
0001 0111	17	23		ETB	End of Transmission Block
0001 1000	18	24		CAN	Cancel
0001 1001	19	25		EM	End of Medium
0001 1010	1A	26		SUB	Substitute
0001 1011	1B	27	\e	ESC	Escape
0001 1100	1C	28		FS	File Separator
0001 1101	1D	29		GS	Group Separator
0001 1110	1E	30		RS	Reqst to Send
0001 1111	1F	31		US	Unit Separator

Binary	Hex	Decimal	Char
0010 0000	20	32	(space)
0010 0001	21	33	!
0010 0010	22	34	"
0010 0011	23	35	#
0010 0100	24	36	$
0010 0101	25	37	%
0010 0110	26	38	&
0010 0111	27	39	'
0010 1000	28	40	(
0010 1001	29	41	)
0010 1010	2A	42	*
0010 1011	2B	43	+
0010 1100	2C	44	,
0010 1101	2D	45	-
0010 1110	2E	46	.
0010 1111	2F	47	/
0011 0000	30	48	0
0011 0001	31	49	1
0011 0010	32	50	2
0011 0011	33	51	3
0011 0100	34	52	4
0011 0101	35	53	5
0011 0110	36	54	6
0011 0111	37	55	7
0011 1000	38	56	8
0011 1001	39	57	9
0011 1010	3A	58	:
0011 1011	3B	59	;
0011 1100	3C	60	<
0011 1101	3D	61	=
0011 1110	3E	62	>
0011 1111	3F	63	?

Binary	Hex	Decimal	Char
0100 0000	40	64	@
0100 0001	41	65	A
0100 0010	42	66	B
0100 0011	43	67	C
0100 0100	44	68	D
0100 0101	45	69	E
0100 0110	46	70	F
0100 0111	47	71	G
0100 1000	48	72	H
0100 1001	49	73	I
0100 1010	4A	74	J
0100 1011	4B	75	K
0100 1100	4C	76	L
0100 1101	4D	77	M
0100 1110	4E	78	N
0100 1111	4F	79	O
0101 0000	50	80	P
0101 0001	51	81	Q
0101 0010	52	82	R
0101 0011	53	83	S
0101 0100	54	84	T
0101 0101	55	85	U
0101 0110	56	86	V
0101 0111	57	87	W
0101 1000	58	88	X
0101 1001	59	89	Y
0101 1010	5A	90	Z
0101 1011	5B	91	[
0101 1100	5C	92	\
0101 1101	5D	93	]
0101 1110	5E	94	^
0101 1111	5F	95	_

Binary	Hex	Decimal	Char
0110 0000	60	96	`
0110 0001	61	97	a
0110 0010	62	98	b
0110 0011	63	99	c
0110 0100	64	100	d
0110 0101	65	101	e
0110 0110	66	102	f
0110 0111	67	103	g
0110 1000	68	104	h
0110 1001	69	105	i
0110 1010	6A	106	j
0110 1011	6B	107	k
0110 1100	6C	108	l
0110 1101	6D	109	m
0110 1110	6E	110	n
0110 1111	6F	111	o
0111 0000	70	112	p
0111 0001	71	113	q
0111 0010	72	114	r
0111 0011	73	115	s
0111 0100	74	116	t
0111 0101	75	117	u
0111 0110	76	118	v
0111 0111	77	119	w
0111 1000	78	120	x
0111 1001	79	121	y
0111 1010	7A	122	z
0111 1011	7B	123	{
0111 1100	7C	124	|
0111 1101	7D	125	}
0111 1110	7E	126	~
0111 1111	7F	127	

https://forum.arduino.cc/t/ascii-character-table/179647 (I corrected some mistakes I found in this web version)

Full ASCII Table 2/2

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

ASCII does not define characters code above 127. The characters in the table below are from the true type font "MS Linedraw" that came with early MS Windows operating systems (*replicates the MS DOS extended character set*).

Binary	Hex	Decimal		Char
1000 0000	80	128		
1000 0001	81	129		ü
1000 0010	82	130		é
1000 0011	83	131		â
1000 0100	84	132		ä
1000 0101	85	133		à
1000 0110	86	134		å
1000 0111	87	135		ç
1000 1000	88	136		ê
1000 1001	89	137		ë
1000 1010	8A	138		è
1000 1011	8B	139		ï
1000 1100	8C	140		î
1000 1101	8D	141		ì
1000 1110	8E	142		Ä
1000 1111	8F	143		Å
1001 0000	90	144		É
1001 0001	91	145		æ
1001 0010	92	146		Æ
1001 0011	93	147		ô
1001 0100	94	148		ö
1001 0101	95	149		ò
1001 0110	96	150		û
1001 0111	97	151		ù
1001 1000	98	152		ÿ
1001 1001	99	153		Ö
1001 1010	9A	154		Ü
1001 1011	9B	155		¢
1001 1100	9C	156	£	British pound sterling
1001 1101	9D	157	¥	Chinese yuan / Japanese yen
1001 1110	9E	158	₧	Spanish peseta
1001 1111	9F	159	ƒ	florin sign (Dutch gulden)

Binary	Hex	Decimal	Char
1010 0000	A0	160	á
1010 0001	A1	161	í
1010 0010	A2	162	ó
1010 0011	A3	163	ú
1010 0100	A4	164	ñ
1010 0101	A5	165	Ñ
1010 0110	A6	166	ª
1010 0111	A7	167	º
1010 1000	A8	168	¿
1010 1001	A9	169	⌐
1010 1010	AA	170	¬
1010 1011	AB	171	½
1010 1100	AC	172	¼
1010 1101	AD	173	¡
1010 1110	AE	174	«
1010 1111	AF	175	»
1011 0000	B0	176	░
1011 0001	B1	177	▒
1011 0010	B2	178	▓
1011 0011	B3	179	│
1011 0100	B4	180	┤
1011 0101	B5	181	╡
1011 0110	B6	182	╢
1011 0111	B7	183	╖
1011 1000	B8	184	╕
1011 1001	B9	185	╣
1011 1010	BA	186	║
1011 1011	BB	187	╗
1011 1100	BC	188	╝
1011 1101	BD	189	╜
1011 1110	BE	190	╛
1011 1111	BF	191	┐

Binary	Hex	Decimal	Char
1100 0000	C0	192	└
1100 0001	C1	193	┴
1100 0010	C2	194	┬
1100 0011	C3	195	├
1100 0100	C4	196	─
1100 0101	C5	197	┼
1100 0110	C6	198	╞
1100 0111	C7	199	╟
1100 1000	C8	200	╚
1100 1001	C9	201	╔
1100 1010	CA	202	╩
1100 1011	CB	203	╦
1100 1100	CC	204	╠
1100 1101	CD	205	═
1100 1110	CE	206	╬
1100 1111	CF	207	╧
1101 0000	D0	208	╨
1101 0001	D1	209	╤
1101 0010	D2	210	╥
1101 0011	D3	211	╙
1101 0100	D4	212	╘
1101 0101	D5	213	╒
1101 0110	D6	214	╓
1101 0111	D7	215	╫
1101 1000	D8	216	╪
1101 1001	D9	217	┘
1101 1010	DA	218	┌
1101 1011	DB	219	▉
1101 1100	DC	220	▄
1101 1101	DD	221	▌
1101 1110	DE	222	▐
1101 1111	DF	223	▀

Binary	Hex	Decimal	Char
1110 0000	E0	224	α
1110 0001	E1	225	ß
1110 0010	E2	226	Γ
1110 0011	E3	227	π
1110 0100	E4	228	Σ
1110 0101	E5	229	σ
1110 0110	E6	230	µ
1110 0111	E7	231	τ
1110 1000	E8	232	Φ
1110 1001	E9	233	Θ
1110 1010	EA	234	Ω
1110 1011	EB	235	δ
1110 1100	EC	236	∞
1110 1101	ED	237	ø
1110 1110	EE	238	ε
1110 1111	EF	239	∩
1111 0000	F0	240	≡
1111 0001	F1	241	±
1111 0010	F2	242	≥
1111 0011	F3	243	≤
1111 0100	F4	244	⌠
1111 0101	F5	245	⌡
1111 0110	F6	246	÷
1111 0111	F7	247	≈
1111 1000	F8	248	°
1111 1001	F9	249	•
1111 1010	FA	250	·
1111 1011	FB	251	√
1111 1100	FC	252	ⁿ
1111 1101	FD	253	²
1111 1110	FE	254	■
1111 1111	FF	255	□

Encryption algorithm

- Start with plain text
- Generate an ‘encryption key’
- Encrypt using XOR, bit shifting, substitution, transposition, modular arithmetic
- Very simple, very fast and very efficiently – implemented in hardware, not software
- This introduces the concept of algorithm efficiency and effectiveness.

KRYPTOR FPGA: OPEN-SOURCE HARDWARE END-TO-END ENCRYPTION

https://www.electronics-lab.com/kryptor-fpga-open-source-hardware-end-end-encryption/

Encrypting "CAT" with key "VVV"

Text value	Binary value		
CAT as bits	01000011	01000001	01010100
	C	A	T
XOR	V	V	V
VVV as key	01010110	01010110	01010110
Cipher	00010101	00010111	00000010

Binary	Hex	Decimal	Char
0100 0000	40	64	@
0100 0001	41	65	A
0100 0010	42	66	B
0100 0011	43	67	C
0100 0100	44	68	D
0100 0101	45	69	E
0100 0110	46	70	F
0100 0111	47	71	G
0100 1000	48	72	H
0100 1001	49	73	I
0100 1010	4A	74	J
0100 1011	4B	75	K
0100 1100	4C	76	L
0100 1101	4D	77	M
0100 1110	4E	78	N
0100 1111	4F	79	O
0101 0000	50	80	P
0101 0001	51	81	Q
0101 0010	52	82	R
0101 0011	53	83	S
0101 0100	54	84	T
0101 0101	55	85	U
0101 0110	56	86	V
0101 0111	57	87	W
0101 1000	58	88	X
0101 1001	59	89	Y
0101 1010	5A	90	Z
0101 1011	5B	91	[
0101 1100	5C	92	\
0101 1101	5D	93	]
0101 1110	5E	94	^
0101 1111	5F	95	_

Cipher strength

- The inherent weaknesses of ‘simple’ substitution and transposition ciphers is that they are open to analysis via frequency and statistical studies
- The strength of a cipher (regardless of whether it is symmetric or asymmetric) is dependent upon:
  - **Key length** (e.g. modern symmetric key is 128 bit)
  - **Algorithm correctness** (are there any ‘bugs’ in the algorithm)
- Perhaps paradoxically, we get ‘more’ algorithm correctness by publicizing the details of the cipher. The best algorithms are public – why? How can this ensure/enhance security?

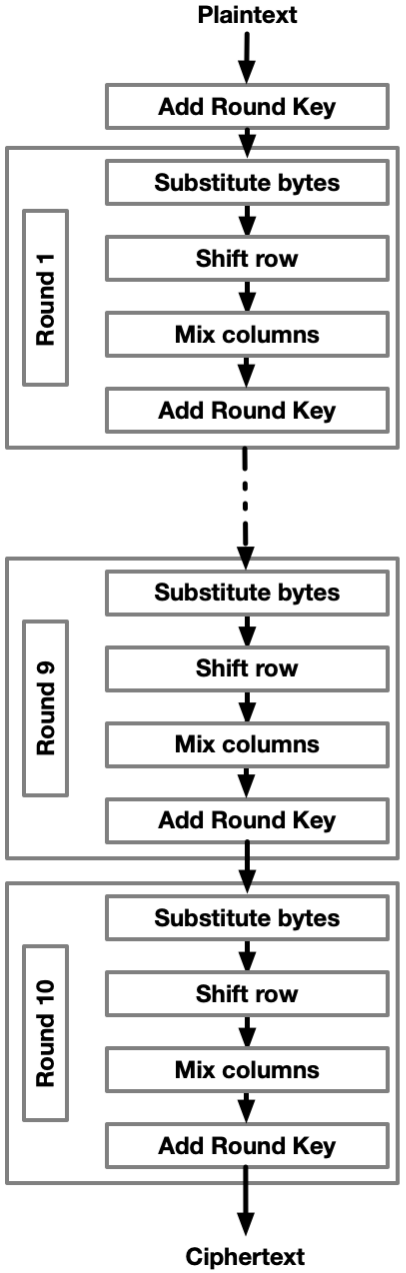
Graphic model of Advanced Encryption Standard (AES).

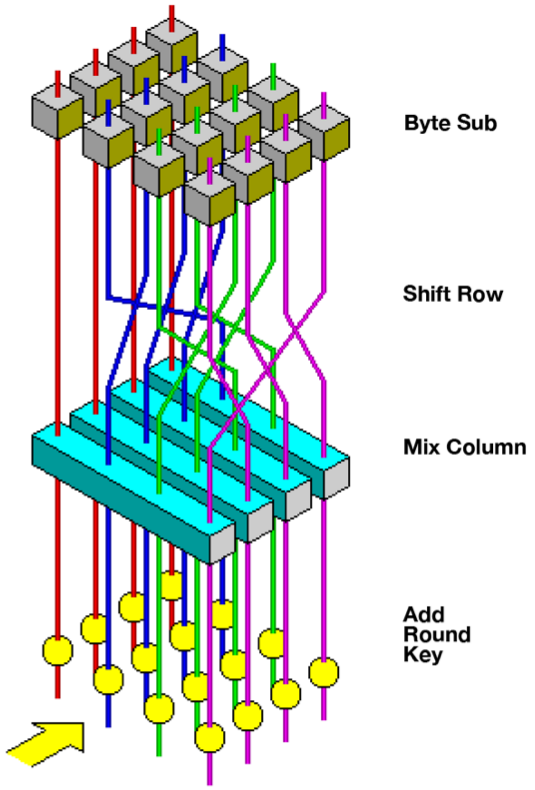
AES, a sample of symmetric encryption, is very strong. The use of 128- and 256-bit keys makes it impervious to cracking.

Contrary to movies and TV, the only way to get unauthorised access to AES encrypted information is to steal or guess the key.

Guessing is impractical due to the number of possible keys. A 128-bit key has 2^128 = 340,282,366,920,938,463,463,374,607,431,768,211,456 possible keys.

An n-bit key has 2^n possible keys because for each of the n bits, we have 2 options (0 or 1).





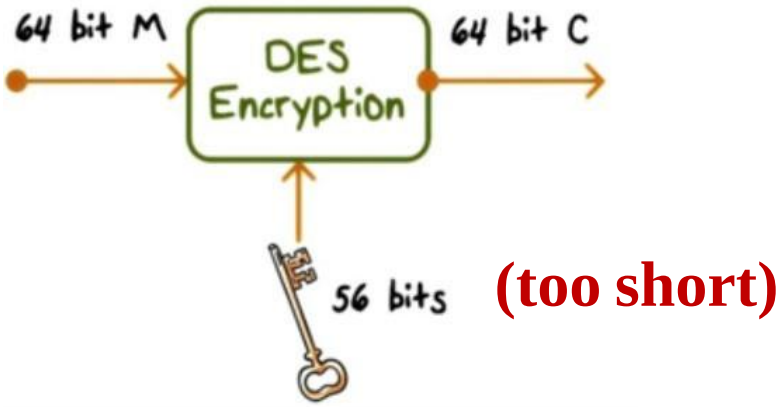
Cipher/key-length strength

- **When using ciphers, the size of the key is very important**.
  - Symmetric key and public key systems should be equally secure – they are not equal in computational efficiency (i.e. time to encrypt, memory usage, etc.)
- **Strength of many encryption applications and cryptosystems measured by key size**
  - E.g., 128 bit length for symmetric, 1024 bit length for asymmetric
  - Cracking a 128-bit key with modern hardware is going to take around 500 billion years. However, quantum computers would be able to do it faster (~185 years atm).
- For cryptosystems, security of encrypted data is not dependent on keeping the encryption algorithm secret
  - Cryptosystem security depends on keeping some or all of the elements of crypto-variable(s) or key(s) secret

Key Length (Bits)	Maximum Number of Operations (Guesses)	Maximum Time to Crack	Estimated Average Time to Crack
16	65,536	0.0000003 seconds	0.00000016 seconds
24	16,777,216	0.00008 seconds	0.00004 seconds
32	4,294,967,296	0.02 seconds	0.01 seconds

Symmetric key algorithms

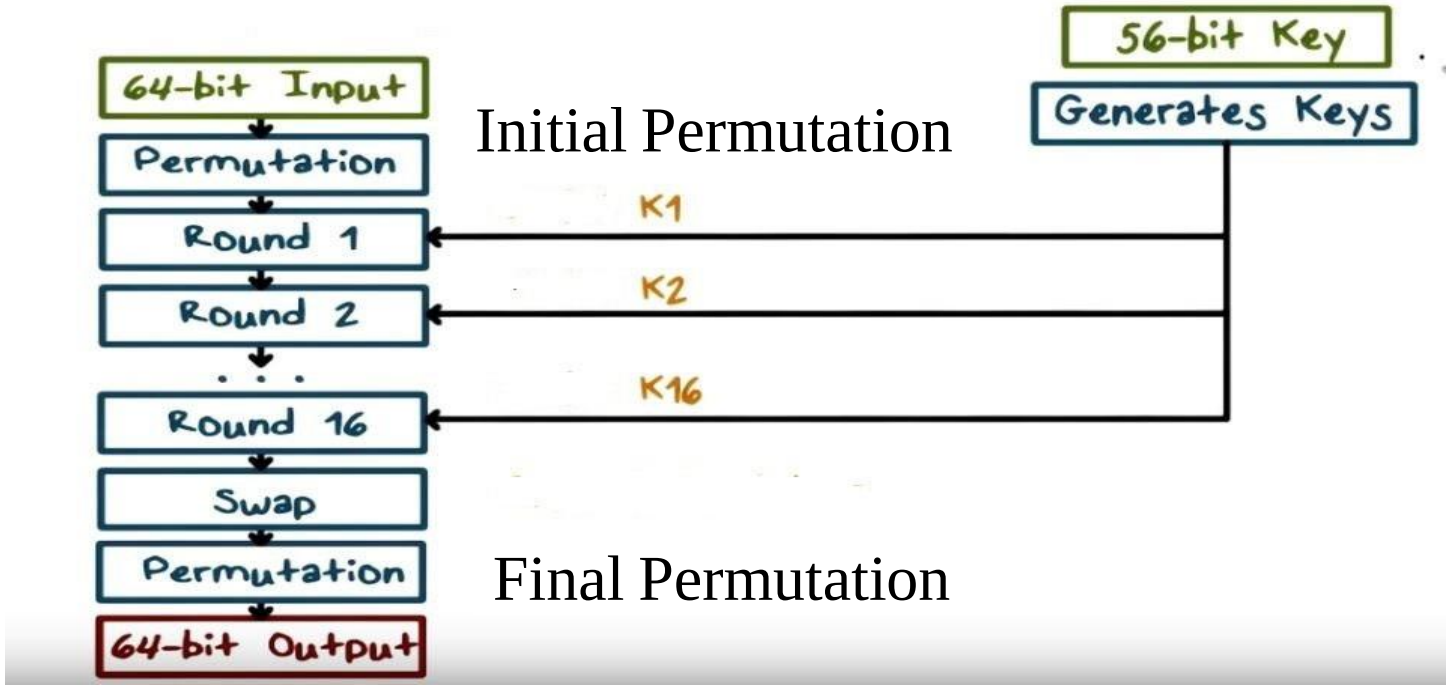
- Data Encryption Standard (DES): one of most popular symmetric encryption cryptosystems – no longer in use
  - 64-bit block size; multiple-rounds of encryption, 56-bit key
  - Published in 1977 – standardized in 1979 as federal standard for encrypting non-classified information
- Remember – the current safe symmetric key length is 128 bits



(too short)

Data Encryption Standard (DES)

64 bit block size, **56 bit key size** (fixed), **multiple rounds**, **key is changed with each round**, output from one round forms the input to the next – uses SUBSTITUTION and transposition



Permutation, another term for transposition

Symmetric key algorithms

- Triple DES (3DES): created to provide security far beyond DES
  - applies DES three times to each data block (FIPS approved in 1999, withdrawn in 2005) – key length 168 bits (three 56 bit DES keys) – deprecated by NIST in 2017 (unsafe)

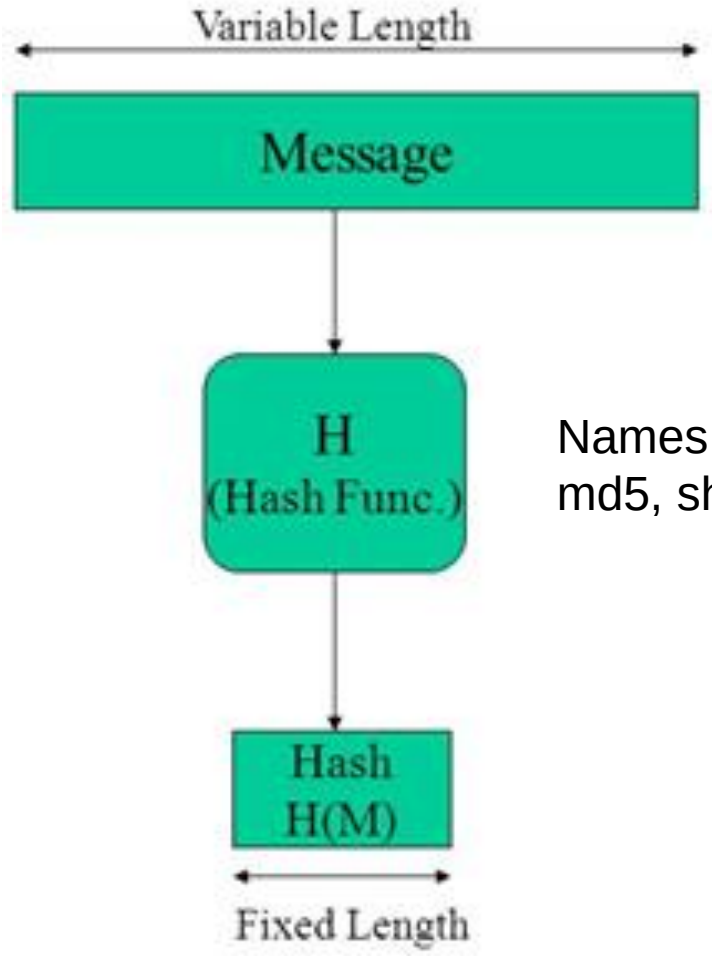
- **Advanced Encryption Standard** (AES): developed to replace both DES and 3DES
  - also known by its original (Dutch) name *Rijndael* Block Cipher.
  - Rijndael is a grouping of ciphers with different key and block sizes.
  - For AES, NIST selected three Rijndael members, each with a block size of 128 bits but three different key lengths 128, 192, and 256 bits.

Summary: Symmetric Key Cryptography

- Cryptography is used to secure most aspects of Internet and Web uses that require it, drawing on extensive set of protocols and tools designed for that purpose
- Symmetric key crypto uses one secret key to both encrypt and decrypt a message
- Data Encryption Standard (DES): was one of most popular symmetric encryption cryptosystems.
  - 64-bit block size; multiple-rounds of encryption, 56-bit key
  - Adopted by NIST in 1976 as federal standard for encrypting non-classified information
- Triple DES (3DES): created to provide security beyond DES
- Advanced Encryption Standard (AES): developed to replace both DES and 3DES – the main cipher is Rijndael Block Cipher
- Current safe symmetric key length is 128 bits

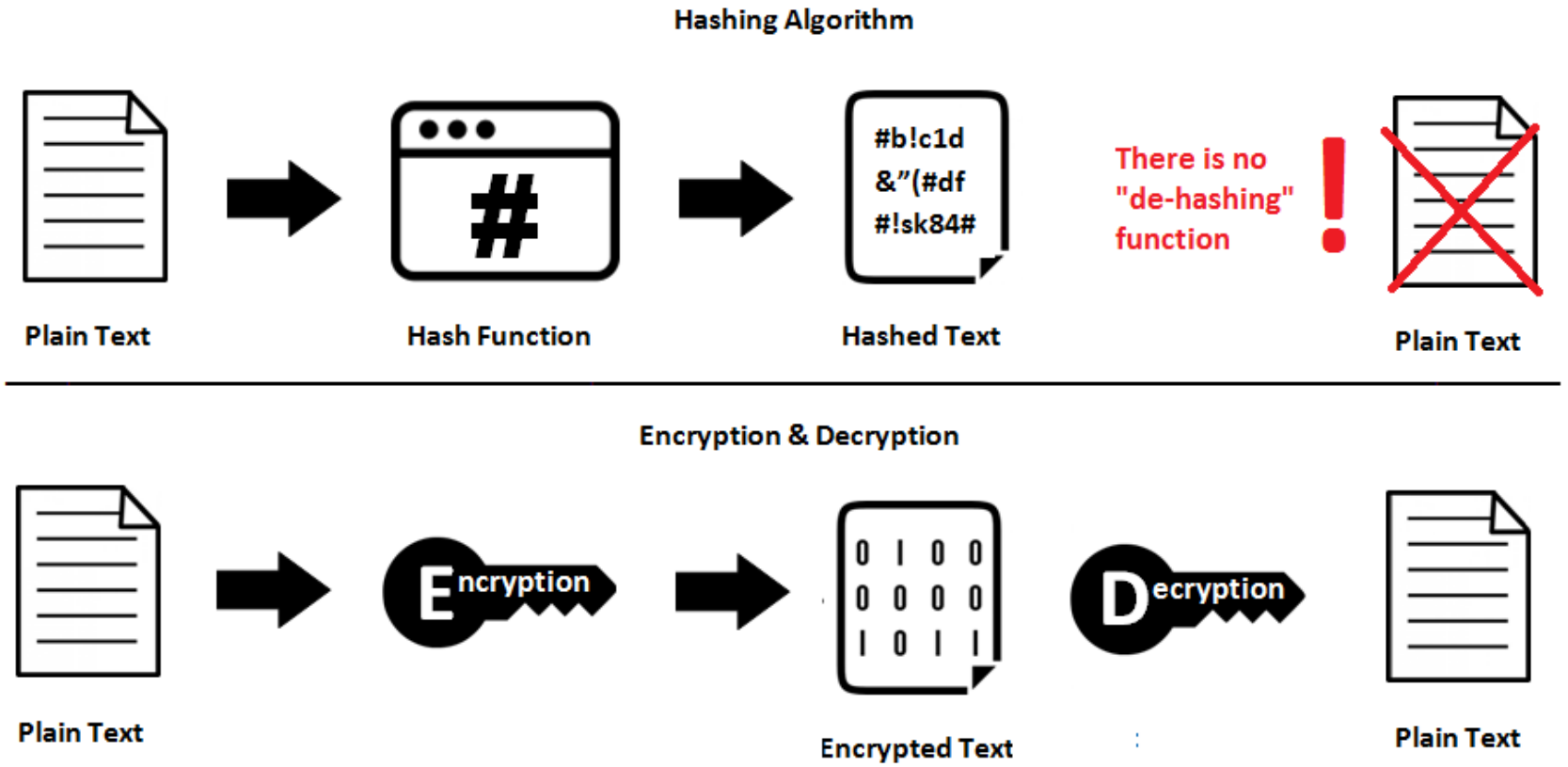
Hashing (cryptographic hash functions)

Generates fixed length fingerprints of arbitrarily large messages.



Names of some hash functions:
md5, sha1, sha256, sha512 and more...

Difference Between Hashing and Encryption



- **Hashing** is a one-way function that changes a plain text to a unique digest† that is irreversible.
- **Encryption** is a two-way function that includes encryption and decryption.

† A compilation or summary of material or information.

https://www.ssl2buy.com/wiki/difference-between-hashing-and-encryption

Hash functions

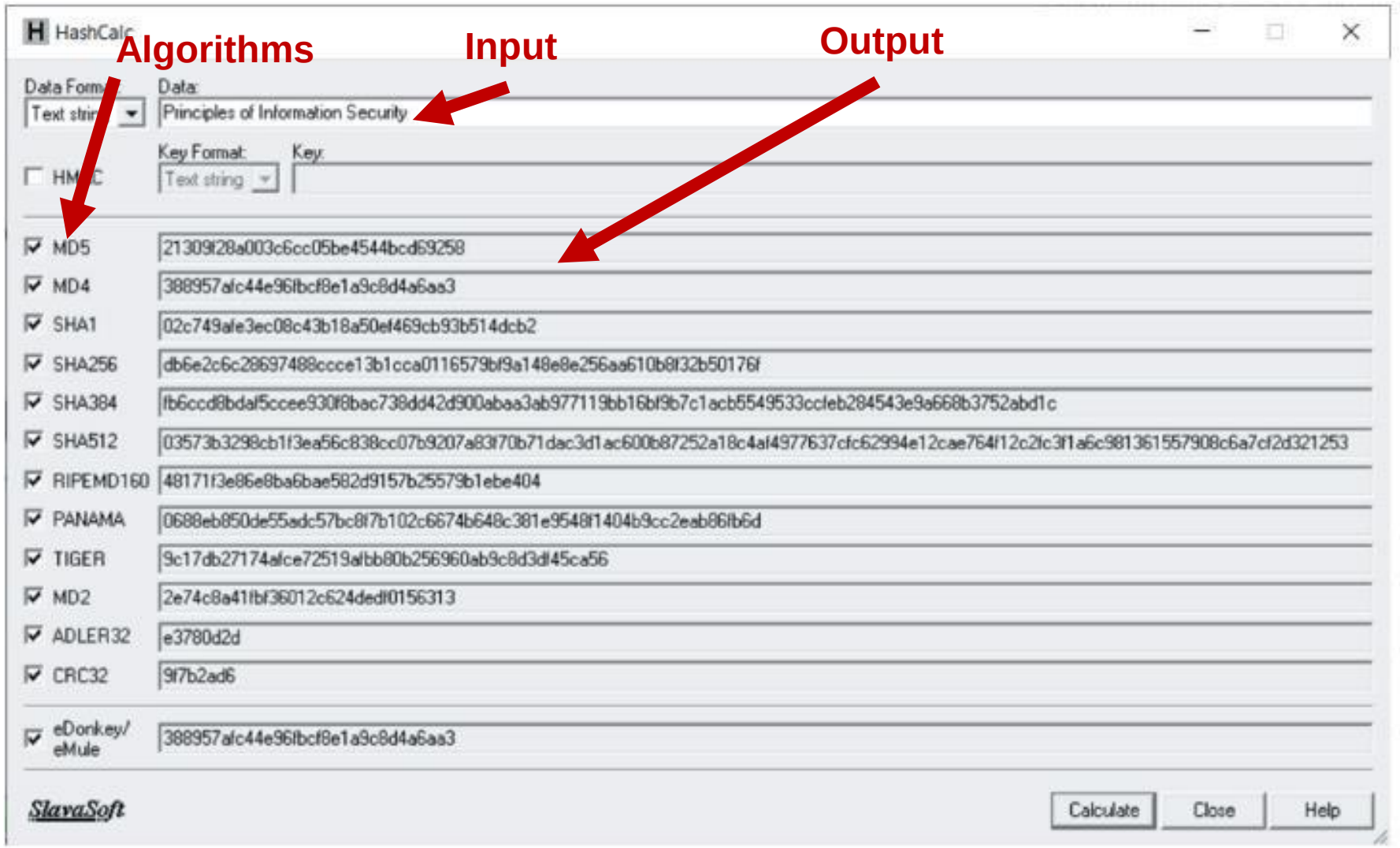
Example- Hash Algorithm: MD5

Original Data	Digest	digest size
a	0cc175b9c0f1b6a831c399e269772661	128 bit
ab	187ef4436122d1cc2f40dc2b92f0eba0	128 bit
abc	900150983cd24fb0d6963f7d28e17f72	128 bit
abc123	e99a18c428cb38d5f260853678922e03	128 bit
abc12345	d6b0ab7f1c8ab8f514db9a6d85de160a	128 bit
abcdefg12345	6e97687128e8b80da01a5400813ae4ad	128 bit
abcdefghijklmn1234567890	3d4f26c33ab8c83f1edbdec1bf92b904	128 bit
abcdefghijklmnopqrstuvwxyz1234567890	7f7d52a001b9d2c71b6bae1f189f41f3	128 bit

- Computationally inexpensive mathematical algorithms that generate fixed length message summary/digest to confirm message identity and confirm no content has changed provides integrity control
- What is a hash collision? Two different inputs producing same hash – very bad!
- Irreversible unique output
- Hash algorithms: publicly known functions that create hash value:
  - SHA group (SHA-1, SHA-256, SHA-384, SHA-512),
  - MD family (MD2, MD4, MD5, MD6),
  - RIPEND family, Tiger, Whirlpool, …

  MD4, MD5, SHA-1 are no longer considered secure due to possible hash collisions
- Also heavily used with public key cryptography to deliver digital signatures, also used in Blockchain applications

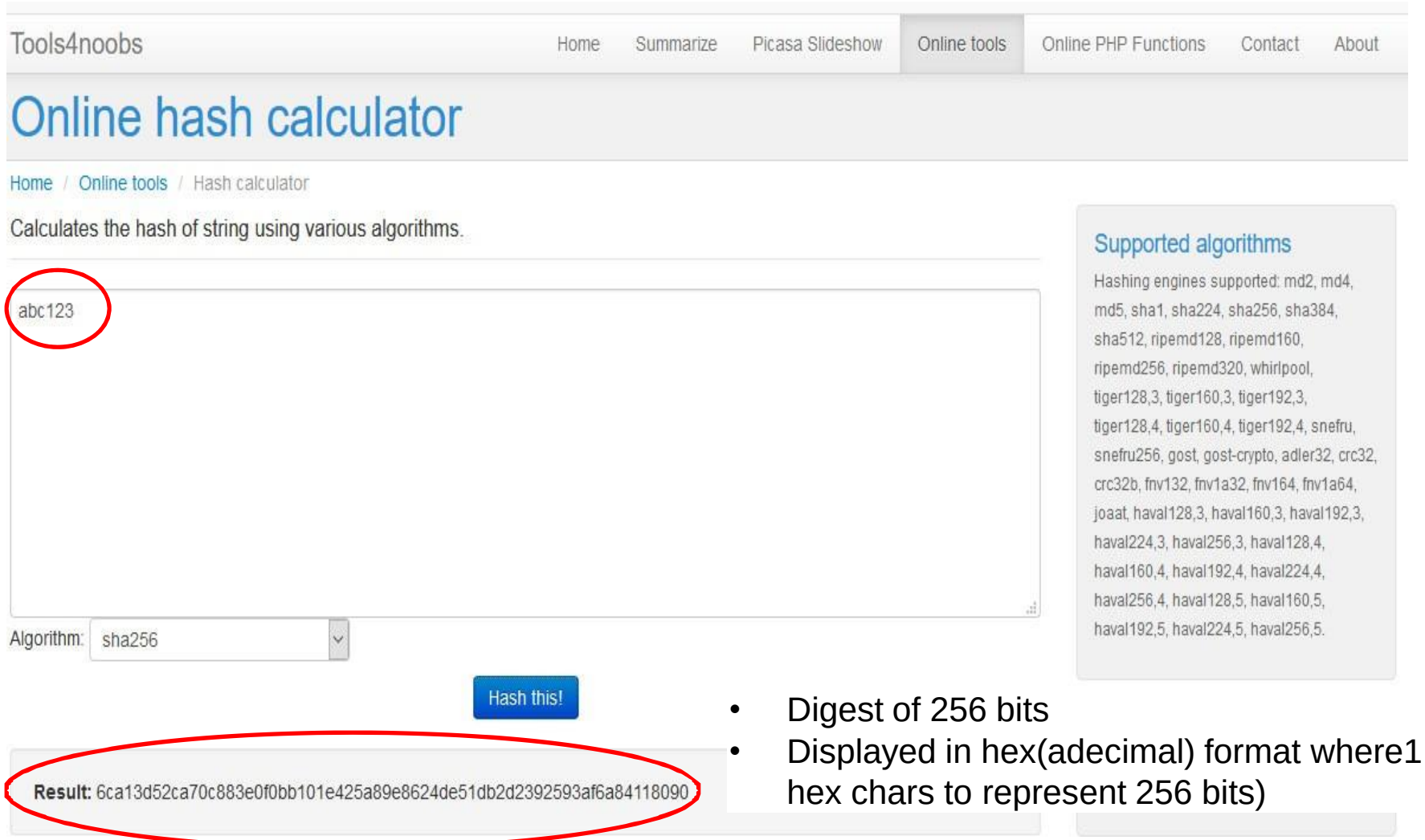
Hash function examples



Source: SlavaSoft HashCalc.

https://www.tools4noobs.com/online_tools/hash/

Online hash calculator



- Digest of 256 bits
- Displayed in hex(adecimal) format where1 hex character = 4 bits (64 hex chars to represent 256 bits)

Summary: Hash functions

- Mathematical algorithms that generate message summary/digest to confirm message identity and confirm no content has changed, i.e. provide an integrity control.
- One way – unique output (fingerprint) – computationally inexpensive
- Hash algorithms: publicly known functions that create hash value – e.g. the SHA and the MD families
- Hashing is NOT encryption
- Hashing is also heavily used with public key cryptography to deliver digital signatures.

Public key cryptography (asymmetric cryptography) and digital signatures

Overview

- **Public key crypto** - completing the theory discussion of last part – focus **public key cryptography**
- **Digital Signatures** (this concept builds on the **hash** strategy)

- Next week:
- **Digital Certificates** (for distribution of public keys with trust – a certificate contains a **digital signature**)
- **Public Key Infrastructure** (PKI) – for distribution of public keys with trust
- We look at the two important protocols in business: SSL/**TLS**, **S/MIME**

Asymmetric cryptography – the theory

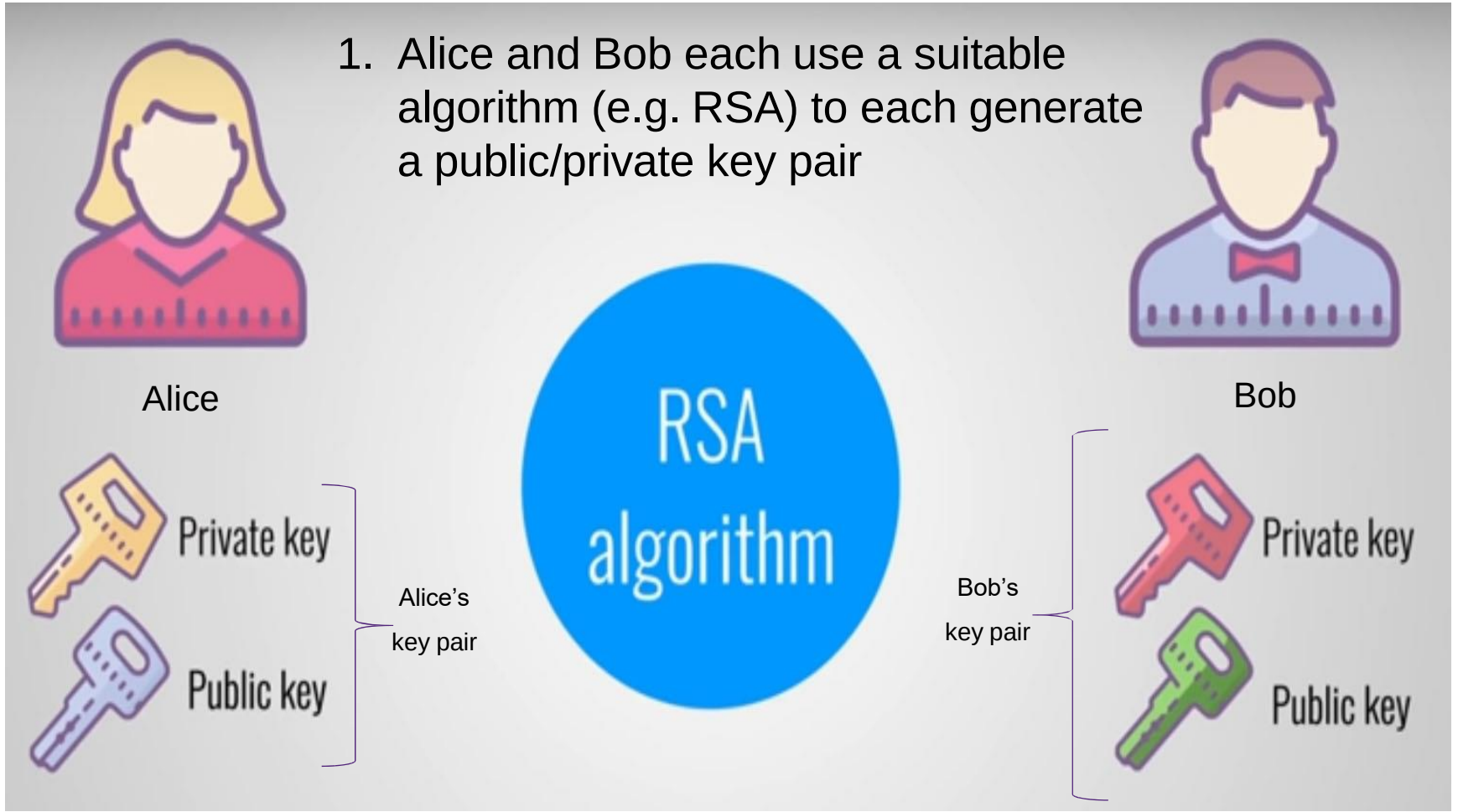
- A new approach (from the 70's – exact date/place unknown)
- **Two keys** – one strictly private, one distributed to the world (public)!
  - Both keys are generated simultaneously and are *mathematically* related
  - Either key can encrypt – whichever one is used, the other (matched) key must decrypt

Example: I have a public key A and a matched private key B. If I generate a message, say M, what happens as per the following:

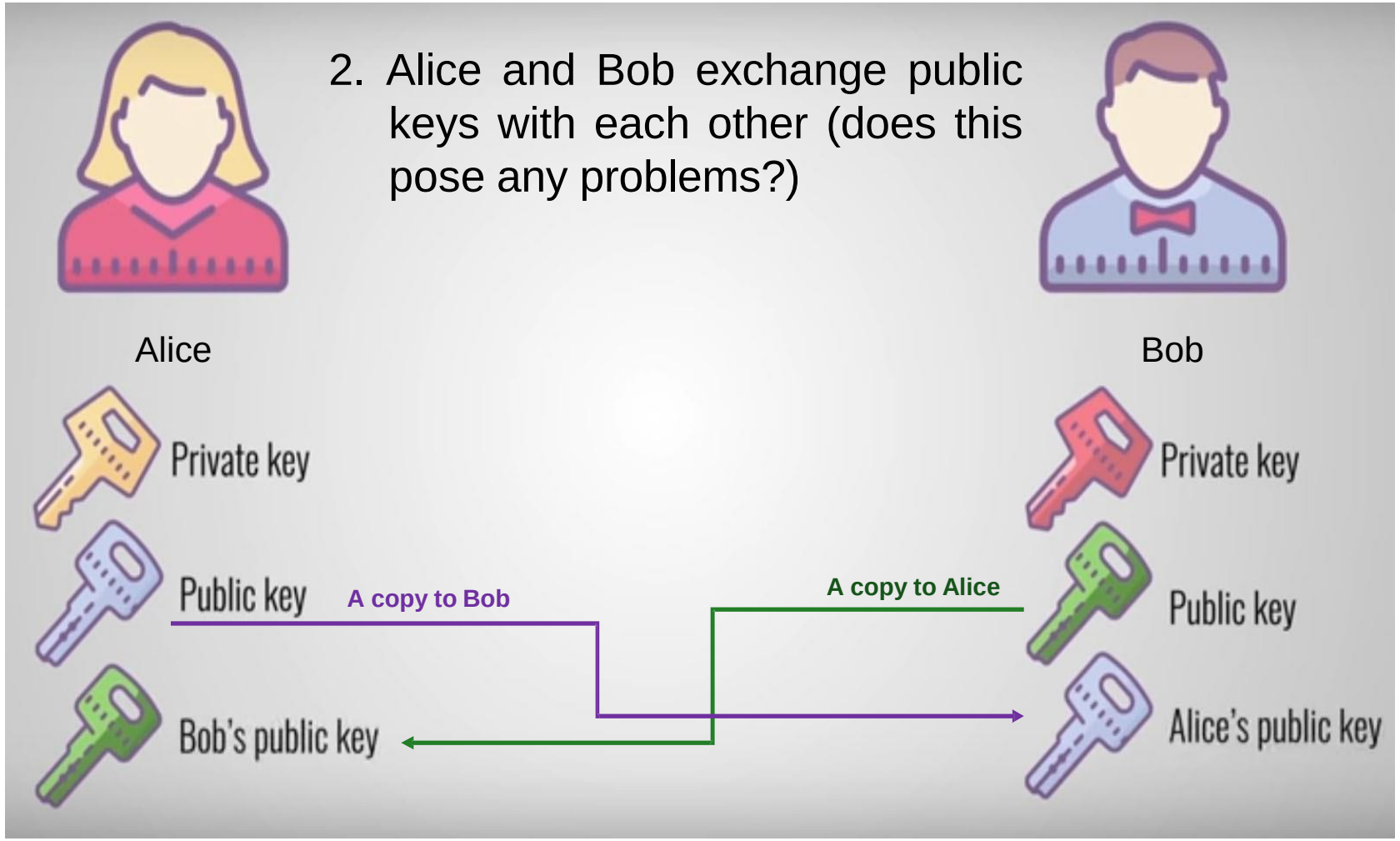
- **Encrypting with the public key delivers confidentiality** (why?)
- **Encrypting with the private key delivers authentication** (why?)

Two types of security: **confidentiality and authentication**!

Asymmetric cryptography – the practice (1)

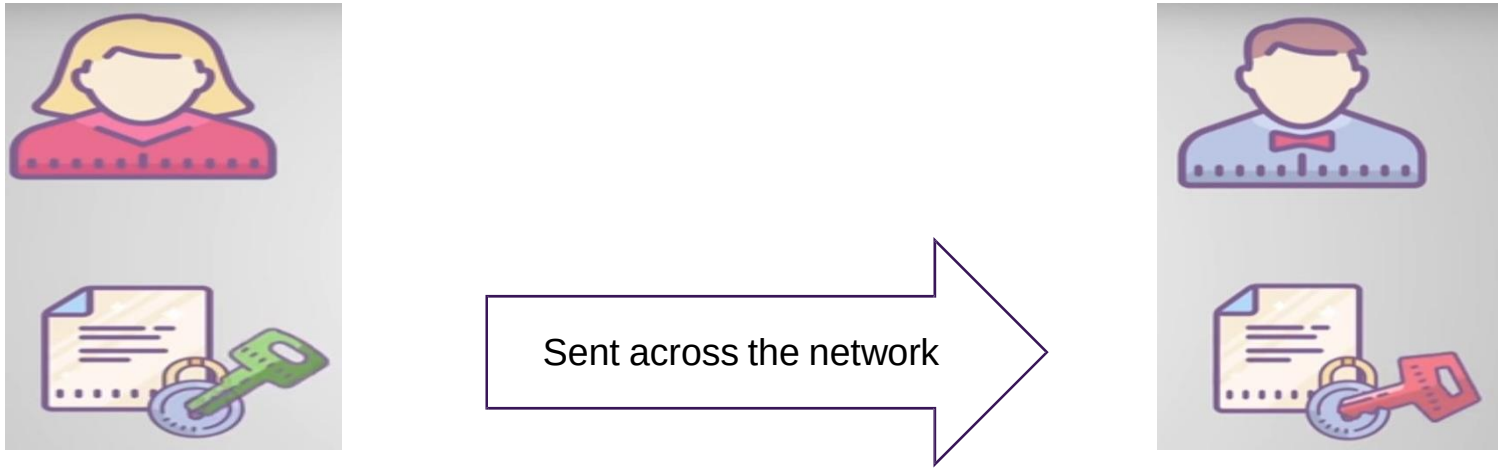


Asymmetric cryptography – the practice (2)



Asymmetric cryptography – the practice (3)

2. Next, Alice creates a sensitive document, encrypts it with Bob's public key, and sends it to Bob. Once she has encrypted this document, Alice cannot decrypt it (why?).
3. Upon receipt of the encrypted document, Bob uses his private key to decrypt it.



Some obvious questions

Q1. Can Bob be sure the message has been sent by Alice (why/why not)?

Q2. Can the message be intercepted by a third party (i.e. copied from the network) (why/why not)?

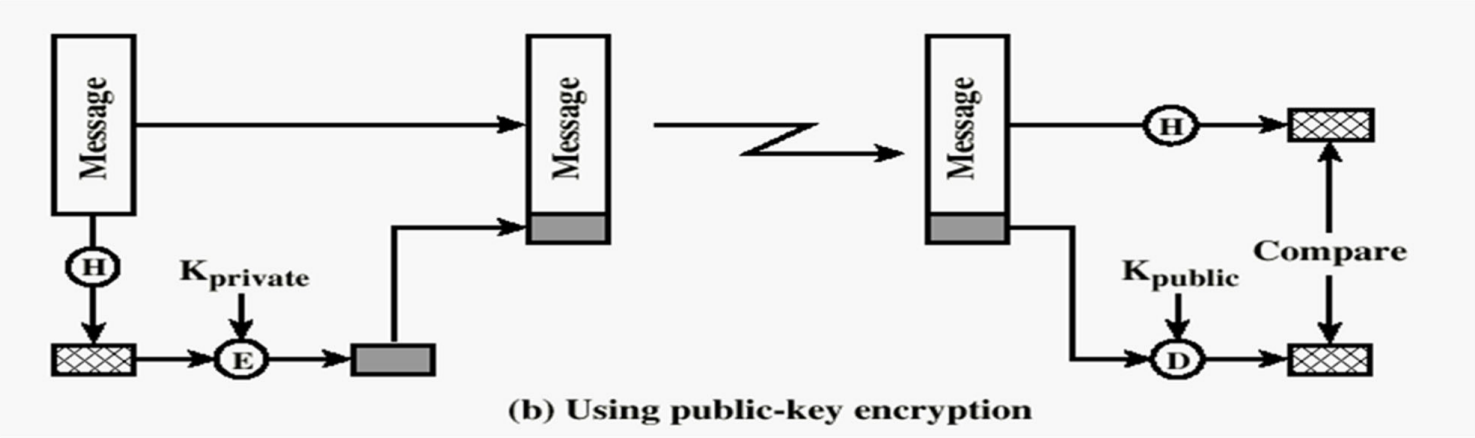
Q3. What 'type' of security is being controlled in this example?

Digital signatures are using asymmetric cryptography

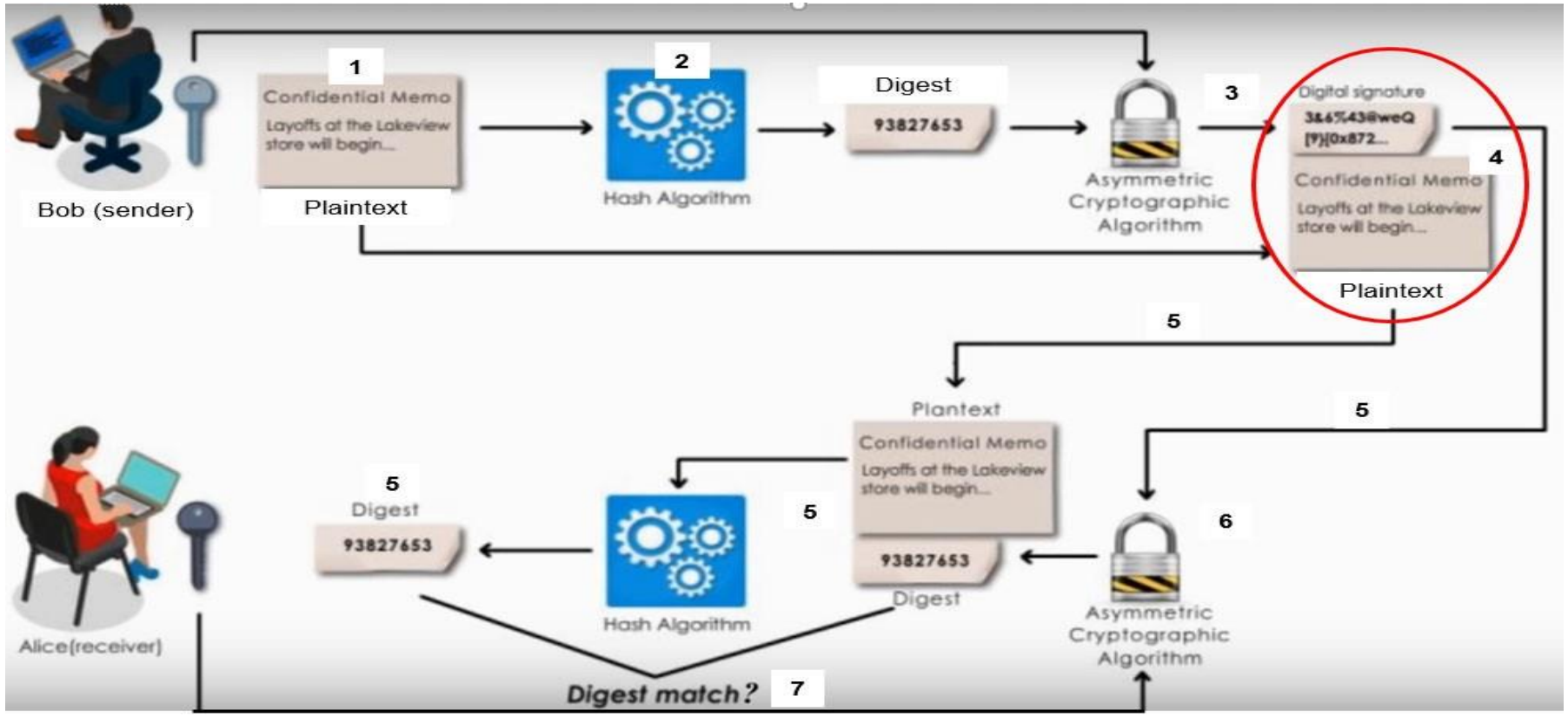
- Created in response to rising need to verify the integrity AND authenticity of information transferred via electronic systems. It is an electronic/digital verification of the sender/owner of the signature.
- **Hashing PLUS asymmetric encryption processes are used to create digital signatures**
- Integrity, authentication and **non-repudiation**: the process that verifies the message was sent by the sender, has not been altered since the signature was added, and thus cannot be refuted/repudiated
- Commonly used for software distribution (e.g. updates), financial transactions, important digital communications (e.g. email), and other cases where it is vital to detect forgery/unauthorized alterations
- Digital Signature Standard (DSS) is the NIST standard for digital signature algorithm usage by federal information systems

How to create a digital signature

- A digital signature is produced via a two-stage process: (1) **hashing**, and (2) **private key signing** (i.e. public key encryption).
- **RSA** is a main encryption algorithm in producing digital signatures. The process is shown below. This process is supported in law by many countries.
- The *Electronic Transactions Act* was enacted in 1999 and established electronic signatures in Australia. Under Australian law, contracts are held enforceable whether it's verbal, or the parties have signed the document with a **wet-ink** (physical) or **electronic signature**.



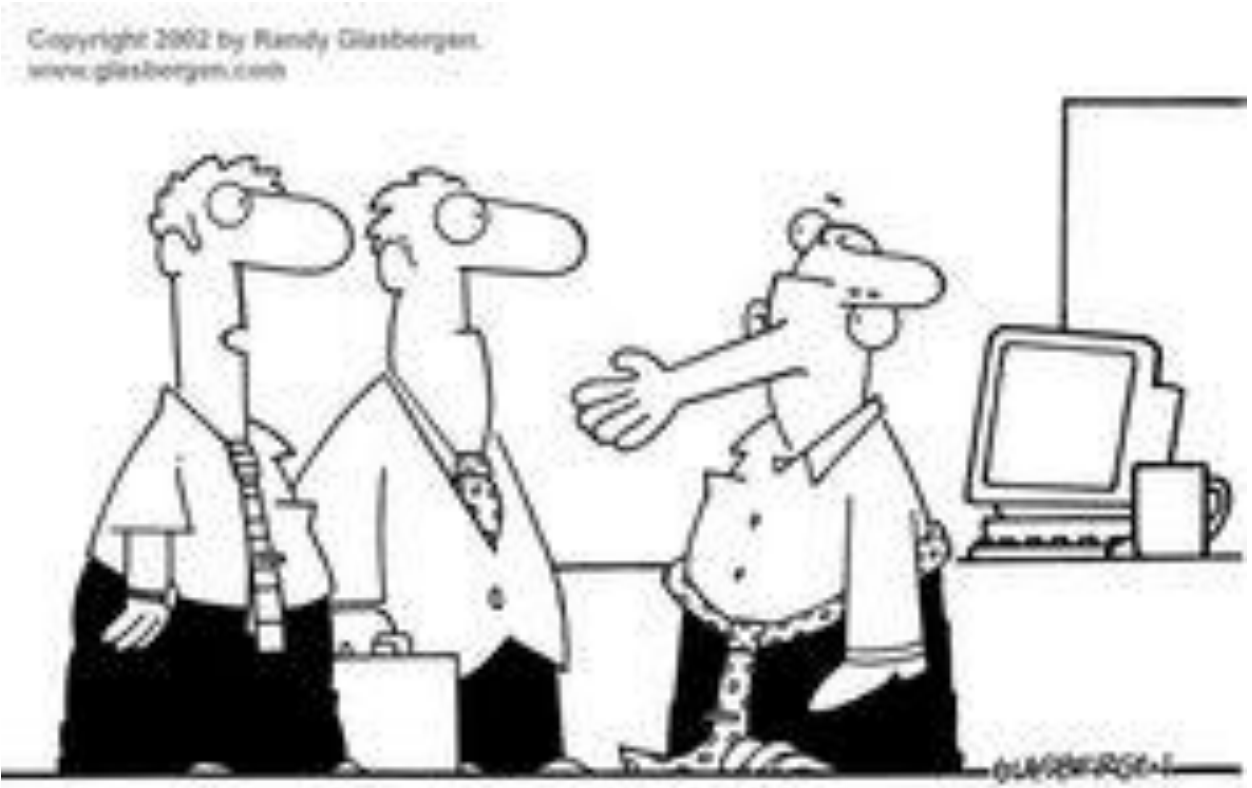
Digital signature creation in detail



Question: how does 'integrity', 'authentication' and 'non-repudiation' apply in the above example? Do we have confidentiality?

To be continued next week...

- **Digital Certificates**
- **Public Key Infrastructure (PKI)**
- **Protocols: SSL/TLS, S/MIME**



"That's our CIO. He's encrypted for security purposes."

```
        end;

    write(Endtext);
end.
CREATE TABLE product(
    class MultinomialNB(object):
    def __init__(self):
2})
        self.X = None
        self.y = None
    def __loading(self):
        self.list_labels = cl.Counter(
int acc(std::function<int(int, int)>
    auto it = operands.begin();
    int result = func(*it, *(++it));
    if (operands.size() > 2) {
        for (++it; it!=operands.end();
            result = func(result, *it);
        }
    }
    return result;
```

Thank you