2: Introduction

- Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries.
- The f5 + 2 main goals of cryptography are:
    - confidentiality: keeping information secret
    - integrity: messages are not altered during transmission and/or storage
    - authentication: knowing who sent / received a message
    - non-repudiation: preventing the denial of past actions
    - non-equivocation: preventing the making of contradictory statements
    - + anonymity + transparency
- Caesar Cipher. shifting the letters by an agreed upon number
- Symmetric (DES, AES) / Asymmetric (public key) (RSA)

3: Transpositions & Substitutions

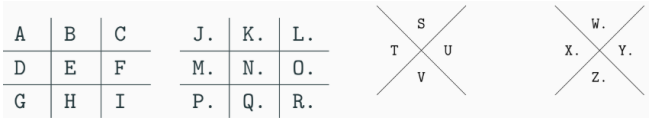
- Transposition:
    - permutes the ordering of the units of a plaintext according to a regular system (single characters, pairs, triplets, etc.)
    - does not change the frequency of the units of the plaintext.
- Rail Fence / Zig-Zag Cipher (transposition):
    - Remove whitespace and pad the plaintext so that its length is a multiple of four: MEETMETONIGHTZZZ
    - Rewrite the plaintext by alternating between two rows
    - Concatenate the two rows and divide into groups of four
    - Decrypt: sSplit the ciphertext into two halves
- Twisted Path / Route Cipher (transposition):
    - Remove whitespace / pad the plaintext: length is a multiple of 4: MEETMETHURSDAYNIGHTZ
    - Rewrite the plaintext into a 4x5 matrix
    - Trace a path and divide into groups of 4
    - The cipher can be extended in many ways: add a key for column order, repeat with a different path each time, include the grid dimension

M	E	E	T	M
E	T	H	U	R
S	D	A	Y	N
I	G	H	T	Z

- Substitutions:
    - replaces the units of a plaintext according to a regular system. The units may be single characters, pairs, triplets, etc..

    - Monoalphabetic: the mapping from plaintext units to ciphertext units is fixed over the entire message (a “T” mapped to “K” is always “K”). The advantage is that you don’t have to carry the entire mapping around with you; you can easily regenerate it.
    - Polyalphabetic: the mapping from plaintext units to ciphertext units can change at different locations
- Monoalphabetic substitution:
    - Simple / old technique: write the reversed alphabet under itself.
    - Pigpen Cipher (Mason’s cipher): Draw 2 tic-tac-toe patterns and two Ś patterns. Include the dots in the compartments.

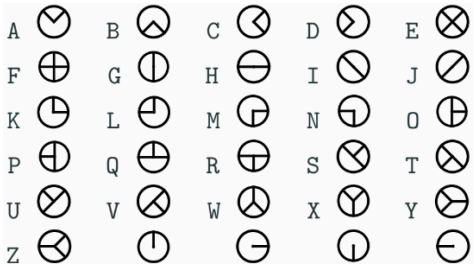
    - Write the alphabet in the compartments left-to-right and top-to-bottom



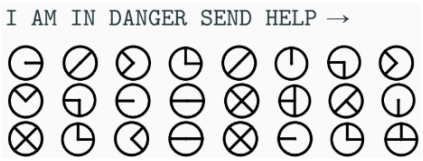
    - Shift Cipher / Caesar Cipher (monoalphabetic): write the alphabet, then underneath, write the alphabet moved to the left or to the right by k places
    - Keyword Cipher: write the alphabet, then write the keyword followed by all the other letters (no duplicates, and last letters are unchanged)
    - Polybius Checkerboard: substitute 2 digit numbers for each unit of plaintext.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V	W	X	Y/Z

- Polyalphabetic substitution:
    - Shadow’s Code: the 4 extras symbols at the end: they can be inserted at any point in the ciphertext. The first extra symbol tells you to give the page a quarter-turn clockwise before you decrypt the next four symbols. The next extra symbol tells you to turn the page to the normal position until you reach the next extra symbol



I AM IN DANGER SEND HELP →



- The Date Shift Cipher: take a date (21st October 1973, rewrite it as 102173, and write it repeatedly above the plaintext. shift M forward 1 letter, shift Y forward 0 letters, shift R forward 2 letters, etc.. decrypt the ciphertext, shift the letters backwards by the same amounts

```
102173 102 173 1021
MYRTLE HAS BIG FEET
```

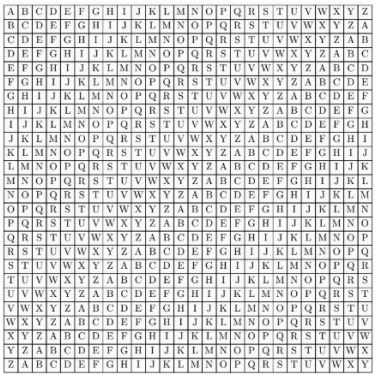
- Playfair Cipher: To encrypt a pair of letters use the following rules:
  - If the pair lie on the same row, use the letters immediately to the right of each letter (wrapping if necessary).
  - If the pair lie on the same column, use the letters immediately below each letter (wrapping if necessary).
  - Otherwise, each letter is replaced by the letter in the same row that is also in the same column as the other letter
  - To encrypt a message, divide the message into pairs. If both letters of the same pair are alike, insert an X between the letters. If only one letter remains at the end, insert an additional X.

4	H	M	V	L	3	Y	D
X	K	B	5	P	Z	E	O
N	7	W	U	F	T	6	J
G	R	2	Q	C	A	I	S

- PO → ZX (Rule 1)
- CL → LP (Rule 2)
- TH → 73 (Rule 3)

```
IW IL LA RX RI VE AT FO UR PM →
26 CY 3C GK 2S Y5 3A JP 7Q BL
```

- Vigenere Cipher: requires a 26x26 matrix and a keyword. Repeatedly write the keyword above the message. encrypt the first letter of the keyword with the first letter of the plaintext. To decrypt, repeatedly write the keyword above the ciphertext then lookup columns and rows



4: How to Break Transposition & Substitution Ciphers

- All languages have letters that are more frequently used (including at the end)
- “The” and “and” are the most common 3-letter words. “Q” is always followed by “U”
- Transpositions don’t change the frequency of the units of the plaintext, but change the frequency of bigrams / trigrams
- Anagramming: permute the units of ciphertext and identify candidates where the frequency of bigrams / trigrams match the language of the plaintext.
- Break Monoalphabetic Substitution Ciphers: Shannon: a 30+ letter cryptogram has only 1 solution. Less than 20: more than 1 solution.

```
TO ?E O? NOT TO ?E THAT I? THE ??ESTION
ZU HO UD CUZ ZU HO ZSGZ AE ZSO JKOEZAUC
```

- Break Polyalphabetic Substitution Ciphers: Vigenere Cipher seems ‘unbreakable’ (up to 26 different monoalphabetic substitutions,resistant to freq analysis)
- Kasiski method (1868): repetitions in the ciphertext suggested repetitions in the plaintext and that the distance between such repetitions were a multiple of the length of the keyword (Babbage)
- Pigeonhole principle: say the key is KING: If there are only four ways to encrypt the word the, and the original message contains more than four uses of the word “the”, then it is inevitable that some of the four possible versions will be repeated in the ciphertext

5: Symmetric Cryptography: Classical to Modern

- Symmetric Encryption Decryption pattern: C = E(K,P) and P = D(K,C) (P,C,K are bit strings)
- One-Time Pad (secure but impractical):
    - The K has to be has long as the plaintext and can only be used once
    - Encryption / Decryption: XOR to P and K: C = (P XOR K) D = (C XOR K)

For example, suppose $P = 01101101$ and $K = 10110100$, then:

$$C = P \oplus K = 01101101 \oplus 10110100 = 11011001$$

and:

$$P = C \oplus K = 11011001 \oplus 10110100 = 01101101$$

- Security Notion (achieved if an attacker working in the attack model can't achieve the security goal):
    - Attack model: The things an attacker can and cannot do.
    - Security goal: The thing an attacker is trying to do (or learn).
        - Indistinguishability (IND): Ciphertexts should be indistinguishable from random strings.If an attacker chooses two plaintexts and then receives a ciphertext of one of the two (chosen at random), they should not be able to tell which plaintext was encrypted.
        - Non-malleability (NM): Given a ciphertext C1 = E(K, P1), it should be impossible to create another ciphertext, C2, whose corresponding plaintext, P2, is related to P1 in a meaningful way
        - IND-CPA (most important as it means ciphertexts can't leak any info, through randomised encryption) denotes indistinguishability against chosen-plaintext attacks. NM-CCA denotes non malleability against chosen-ciphertext attacks.
    - Security notion: Attack model + security goal.
- Kerckhoffs's Principle: a cryptosystem should be secure even if everything about the system, except the key, is public knowledge.
- Attack model:
    - Black-Box Model: The attacker does not have access to the cipher's physical implementation but can observe inputs (plaintext) and/or outputs (ciphertext). They may be able to interact with the cipher by providing chosen plaintext and chosen ciphertext.
    - Gray-Box Model: The attacker has access to the cipher's physical implementation.

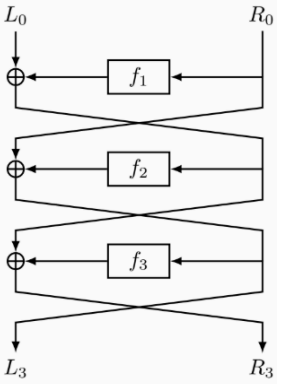
6: Symmetric Cryptography: Block Ciphers

- Block cipher: combines a core algorithm that works on blocks of data with a mode of operation, or a technique to process sequences of blocks of data.
    - Two values characterise a block cipher: the block size and the key size (DES's blocks have 64-bits and AES's blocks have 128-bits). Lengths are powers of two.
    - A block cipher is not a single complex algorithm, but a repetition of rounds.
    - There are two main techniques to construct a block cipher (Feistel scheme (used in DES), Substitution-permutation network (used in AES)).

    - A 3 round block cipher: C = F3(F2(F1(P))) where F1, F2 and F3 are the round functions and P is the plaintext.
    - Each round has an inverse function: P = F'3(F'2(F'1(C))).
    - Round functions are usually identical algorithms parameterised by a value called round key. They're derived from the main key, K, using a key schedule algorithm (K1, K2). Each round key is different.
- Feistel scheme
    - If plaintext: 0011 1111 and key: 1100 0011: ciphertext: 0000 1100.
        - Set L0 = 0011 and R0 = 1111.
        - Then L1 = 1111 and R1 = 1111 XOR 1100 XOR 0011 = 0000
        - Then L2 = 0000 and R2 = 0000 XOR 0011 XOR 1111 = 1100

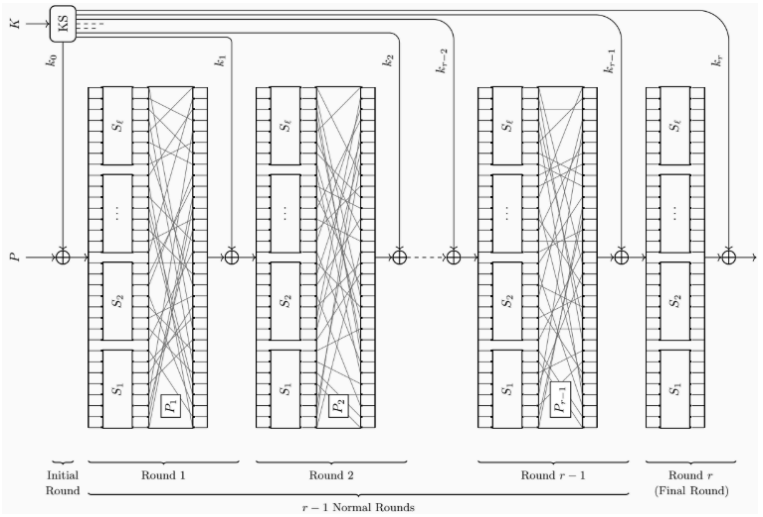
In the 1970s, Horst Feistel designed the *Feistel scheme*:

1. Split the block into two halves, L_0 and R_0.
2. Set $L_1 = R_0$ and $R_1 = L_0 \oplus f_1(R_0)$.
3. Repeat for L_2 and R_2, etc.
4. Concatentate L_n with R_n to produce the final output.



A 3-Round Feistel Scheme

- Substitution-permutation network (combination of both):
    - Substitution operations: performed by S-boxes (produces confusion). S-boxes substitute bits for other patterns. They're non-linear (a change in the input produces a non-proportional change in the output). S-boxes are implemented as fixed lookup tables
    - Permutation operations performed by P-boxes (produces diffusion). P-boxes permute or transpose bits. They can be as simple as changing the order of the bits (transposition) but can also use linear algebra and matrix multiplications to transform the bits.
    - Both can be key-dependent (their behaviour depends on the key) or key-independent (their behaviour does not depend on the key).



7: The Advanced Encryption Standard (AES)

- AES (og name Rijndael) is the most-used cipher in the world.
- Before: DES (56-bit keys: too small) / 3DES (168-bit keys to provide 112-bit security).
- AES processes 128 bits blocks. Secret key: 128 (most common because it is the fastest), 192, or 256 bits.
- AES manipulates bytes within blocks with 128 bits as a 16-byte 2D array (S means state). Bytes / columns / rows of this array are transformed to produce the ciphertext:

s_0	s_4	s_8	s_{12}
s_1	s_5	s_9	s_{13}
s_2	s_6	s_{10}	s_{14}
s_3	s_7	s_{11}	s_{15}

- AES uses a substitution-permutation network with rounds (10 for 128-bit keys, 12 for 192-bit keys, 14 for 256-bit keys)
- Each round (except the last) has four steps:
    - SubBytes (substitution): Replaces each byte using a specified S-box. (The row is determined by the most significant nibble and the column is determined by the least significant nibble. The value 0x9a is converted into 0xb8)

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

- ShiftRows (permutation): Shifts the rows using a specified pattern (not the 1st row).

s_0	s_4	s_8	s_{12}
s_1	s_5	s_9	s_{13}
s_2	s_6	s_{10}	s_{14}
s_3	s_7	s_{11}	s_{15}

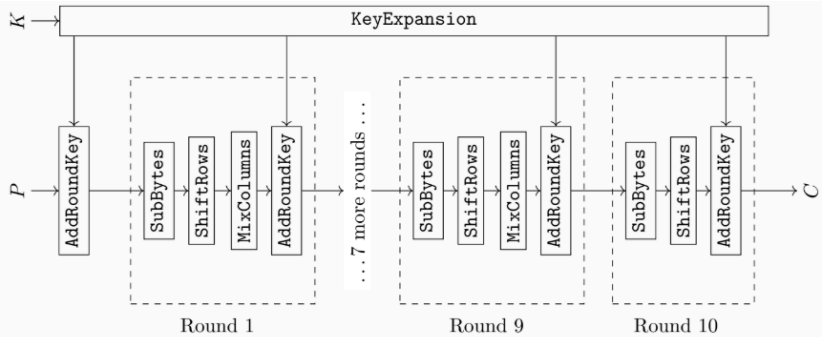
$\rightarrow$

s_0	s_4	s_8	s_{12}
s_5	s_9	s_{13}	s_1
s_{10}	s_{14}	s_2	s_6
s_{15}	s_3	s_7	s_{11}

- MixColumns (permutation): Linear transformation to each of the 4 columns (Each column is left-multiplied by a matrix to produce a new column)

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- AddRoundKey: XORs a round key to the internal state.

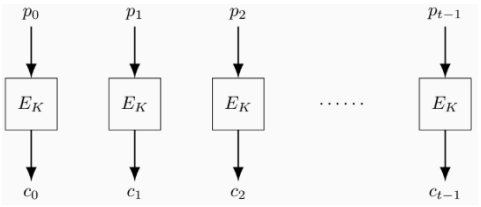


AES uses 10 rounds for 128-bit keys.

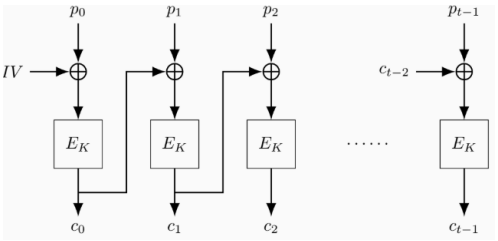
- To decrypt a ciphertext, AES unwinds each operation by using its inverse function: Only the inverse of AddRoundKey's XOR is unchanged because the inverse of an XOR is another XOR
- Key schedule in AES: KeyExpansion:
- Creates 11 round keys (K0,K1,K2, ...,K10) of 16-bytes each for the 128-bit key.
- Uses the same S-box as SubBytes and a combination of XORs.
- Given any round key,an attacker can determine all other round keys as well as the main key by reversing the algorithm (this is an imperfect defense against attacks)
- AES Implementations: no production-level AES code with SubBytes() ShiftRows() MixColumns() functions (too inefficient). Fast AES software uses table-based implementations: they replace the functions with a hardcoded combination of XORs and lookups in tables. These are vulnerable to cache timing attacks.
- AES native instructions (AES-NI) solve this problem: dedicated assembly instructions, implemented in hardware: no AES round as a sequence of low-level assembly instructions, but the hardware does it.

8: Modes of Operation

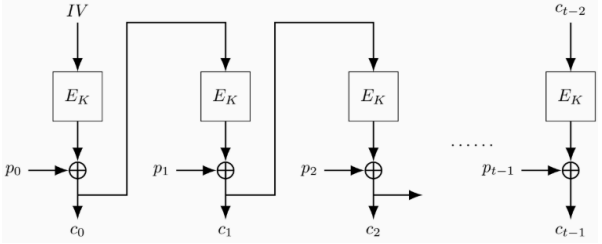
- The biggest danger in using AES, and many other block ciphers, is not the core algorithm but the mode of operation
- Mode of operation: how to repeatedly apply a block cipher's single-block operation to securely transform amounts of data larger than a block
- Electronic Codebook Mode (ECB): simple but insecure. Processes each plaintext block independently. Not semantically secure: identical ciphertext blocks reveal identical plaintext blocks



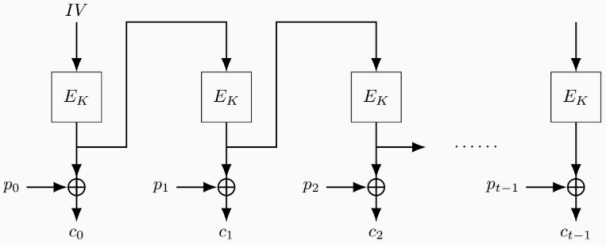
- Cipher Block Chaining Mode (CBC): Instead of encrypting the ith block using Ci = Ek (pi), CBC uses ci = Ek(pi XOR ci´- 1) to chain the blocks together
    - each ciphertext block dependent on all the previous blocks.
    - It ensures that identical plaintext blocks won't produce identical ciphertext blocks.
    - the random initialisation vector (IV) guarantees that 2 identical plaintexts will encrypt to distinct ciphertexts when calling the cipher twice with 2 distinct initial values.
    - The decryption process needs to know the IV that was used to encrypt, so the IV is sent along with the ciphertext, in the clear



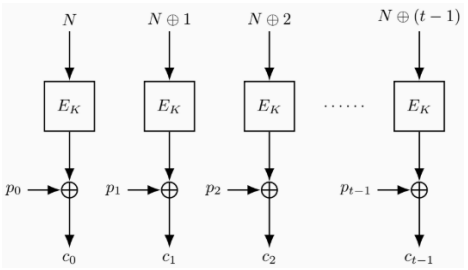
- Cipher Feedback Mode (CFB): similar to CBC (CBC is better)
    - Encrypt the IV and use it as a one-time pad for the first block
    - Encrypt the first ciphertext block and use it as a one-time pad for the second block, etc



- Output Feedback Mode (OFB): similar to CFB, but CTR is better and also faster.
    - Encrypt the IV and use it as a one-time pad for the first block.
    - Encrypt the first pad block and use it as a one-time pad for the second block, etc
    - interesting property that the pad blocks can be generated in advance



- Counter Mode (CTR): hardly a block cipher mode: it turns a block cipher into a stream cipher that just takes bits in and spits bits out
    - The nonce N is similar to an IV but it doesn't need to be cryptographically random. It shouldn't be used more than once
    - It is parallelisable and you can start encrypting even before knowing the message by picking a nonce and computing the stream that you'll later XOR with the plaintext

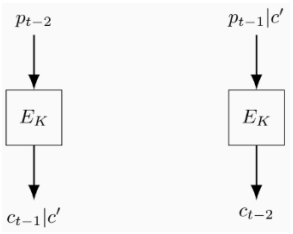


- Summary:

ECB	Insecure
CBC	Popular choice
CFB	Academic interest; use CBC instead
OFB	Academic interest; use CTR instead
CTR	Popular choice

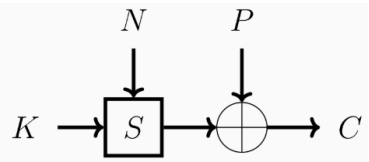
- Arbitrary Length Plaintexts:
    - ECB, CBC and CFB, require the length of the plaintext to be a multiple of the block size
    - Two options to handle plaintexts whose lengths are not a multiple of the block size:
    - Padding a Message (expands a message to fill a complete block by adding extra bytes to the end of the message)
        - PKCS#7 (or RFC 5652) is very popular

        - Works with any block length up to 255 bytes
        - Assuming 16-byte blocks:
        - If there is one byte leftover (example: the message is 1 byte, 17 bytes, or 33 bytes long) pad the message with 15 bytes of 0x0f
        - If there are two bytes leftover, pad with 14 bytes of 0x0e
        - If there are three bytes leftover, pad with 13 bytes of 0x0d
        - Etc..
        - If the length of the message is already a multiple of 16, the block length, add 16 bytes of 0x10
        - Why: allow for the unambiguous removal of the padding after decryption. prioritizes the integrity of the decryption process by eliminating any uncertainty about how to remove the padding. This prevents potential data loss or misinterpretation
    - Ciphertext Stealing: more complex and less popular than padding
        - The exact details depend on the mode of operation
        - Pros:
            - Ciphertexts are exactly the same length as plaintexts.
            - Plaintexts can be of any bit length, not just bytes (example: we can encrypt a message of 131 bits)
            - It is not vulnerable to Padding Oracle Attacks
            -
        - It alters the processing of the last two blocks of the message.
        - A portion of the second-last block's ciphertext is stolen (c' in the figure below) to pad the last plaintext block

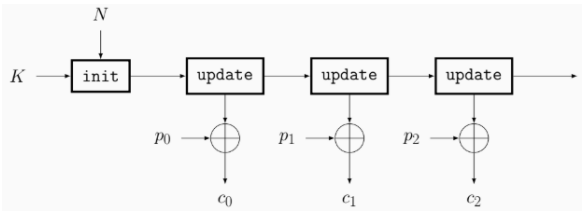


9: Stream Ciphers

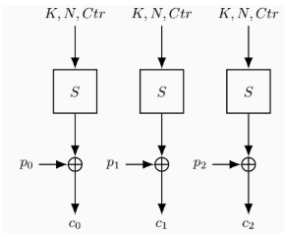
- Symmetric ciphers can be either block ciphers or stream ciphers
- They generate a deterministic stream of random bits from the key and encrypt the plaintext by XORing it with the random bits
- They used to be not secure, but have become much better now
- Stream ciphers produce a deterministic stream of random bits called the keystream.
- A stream cipher computes KS = S(K, N), encrypts using C = P XOR KS, and decrypts using P = C XOR KS. N is a nonce (a number only used once) and unlike IV it need not be cryptographically random
- The encryption and decryption processes are the same: XOR bits with the keystream



- A message can be encrypted with K1 and N1, and then another message with K1 and N2
- you should never reuse the same key-nonce pair
- There are 2 types of stream ciphers: stateful and counter-based
- Stateful stream ciphers:
    - have a secret internal state that evolves throughout keystream generation.
    - The cipher initialises the state from the key and the nonce
    - It calls an update function to update the state value and produce one or more keystream bits from the state



- Counter-based stream ciphers:
    - They produce chunks of keystream from a key, a nonce, and a counter value
    - Unlike stateful stream ciphers, no secret state is memorised during keystream generation
    - It's similar to counter mode (CTR) for block ciphers. CTR can turn a block cipher into a counter-based stream cipher



- 2 further types of stream ciphers: Hardware-Oriented vs. Software-Oriented
- Hardware-Oriented:
    - Many hardware-oriented stream ciphers are implemented using linear feedback shift registers (LFSRs). They can be implemented very efficiently in hardware using shift registers and XOR gates
    - They can work at the bit-level
    - Grain-128a (Stateful): secure and fast
    - A5/1 (Stateful): broken and not secure
- Software-oriented:
    - software-oriented stream ciphers tend to rely on primitives such as table lookups, word-oriented operations, or more complex functions

- RC4 (Stateful): used for early WiFi encryption WEP. It's weak and not secure
  - To generate a keystream, RC4 uses a 2 step secret internal state
    - A permutation of all 256 possible bytes (denoted by S)
    - Two 8-bit index-pointers (denoted by i and j)
  - The order of the permutation is initialised with a variable length key, typically between 40 and 2048 bits using a key-scheduling algorithm
  - The index-pointers are initialised to zero
  - For as many iterations as are needed, RC4 modifies the state and outputs a byte of the keystream

For each iteration, we:

1. Increment i mod 256.
2. Look up the ith element of S, $S[i]$, and add that to j mod 256.
3. Swap the values of $S[i]$ and $S[j]$.
4. Use the sum $S[i] + S[j]$ mod 256 as an index to fetch a third element of S. This is k, one byte of the keystream.

- Salsa20 (Counter-based): fast and secure. Chacha20 is an improvement
- AES in counter mode (CTR): secure

10: Diffie-Hellman

- Whitfield Diffie & Martin Hellman published "New Directions in Cryptography" in 1970
- they introduced the ideas behind asymmetric cryptography: public-key encryption and digital signatures
- Diffie-Hellman (DH) protocol: a public-key distribution scheme or key agreement protocol

Alice and Bob agree on $g = 5$ and $p = 23$.

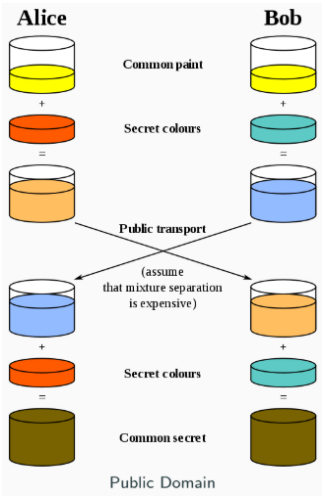
Alice chooses $a = 6$ and sends Bob $A = 5^6 \mod 23 = 8$.

Bob chooses $b = 15$ and sends Alice $B = 5^{15} \mod 23 = 19$.

Alice computes $K = B^a \mod 23 = 19^6 \mod 23 = 2$.

Bob computes $K = A^b \mod 23 = 8^{15} \mod 23 = 2$.

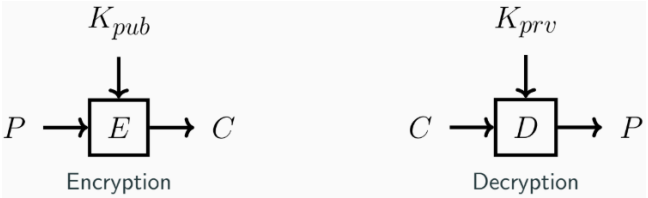
- Diffie-Hellman allows Person A and Person B to generate a shared secret in such a way that the secret cannot be known by Person C. Person A and Person B aren't sharing information during the key exchange, they're creating a secret together



- Very useful for symmetric encryption
- Even if the communication is recorded and analyzed later there is no way to figure out what the secret was
- This is an example of forward secrecy: the secret was never stored, transmitted, or made available to anyone other than the two parties
- Used for HTTPS

11: Public-Key Cryptography

- In asymmetric encryption, or public-key encryption, there are two keys: one to encrypt and another to decrypt
- The encryption key is called a public-key and is publicly available to anyone who wants to send the recipient encrypted messages
- The decryption key, however, must remain secret and is called a private-key



- We can write C = E(Kpub, P) and P = D(Kprv, C)
- A public-private keypair (Kpub and Kprv) is determined solely by the recipient
    - If you want to send me a message, you must first request me to generate a keypair
    - I generate a keypair, one part of which I send to you (Kpub) and the other part I keep to myself (Kprv)
    - You encrypt the message using Kpub and send it to me
    - I decrypt the message using Kprv
    - If I want to reply to your message, I must request you to generate a keypair
- If someone knows Kprv , they may be able to easily compute Kpub

- However, if someone knows Kpub, they cannot easily compute Kprv
- Rivest-Shamir-Adleman (RSA): involves Chinese remainder algorithm, Euler's totient function, etc..
- RSA works because:
    - Same as Kid-RSA
    - The hard problem is the integer factorisation problem (IFP)
    - No algorithm is known that can factor an arbitrary integer in polynomial time
    - Neither the existence nor non-existence of such algorithms has been proved, but it is generally suspected that they do not exist

Alice chooses: $a = 9$, $b = 11$, $a' = 5$, $b' = 8$ and computes:

- $M = ab - 1 = (9 * 11) - 1 = 98$
- $e = a'M + a = (5 * 98) + 9 = 499$
- $d = b'M + b = (8 * 98) + 11 = 795$
- $n = (ed - 1)/M = ((499 * 795) - 1)/98 = 4048$

Alice's K_{pub} is $(499, 4048)$ and her K_{prv} is $(795, 4048)$.

Let's encrypt and decrypt the message $P = 538$.

$C = eP \mod n = 499 * 538 \mod 4048 = 268462 \mod 4048 = 1294$.

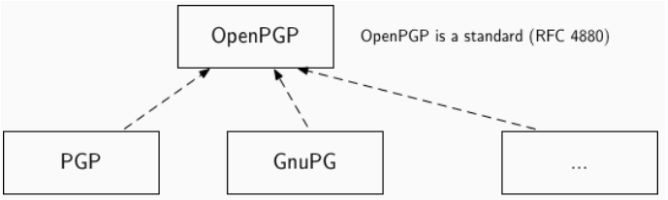
$P = dC \mod n = 795 * 1294 \mod 4048 = 1028730 \mod 4048 = 538$.

- Kid-RSA is a simplified version of RSA:
    - Alice chooses four numbers: a, b, a' and b' and computes:
        - M = ab - ´1
        - e = a' M + a
        - d = b' M + b
        - n = (ed - 1) / M
    - Alice's Kpub is (e, n) and her Kprv is (d, n)
    - To encrypt a plaintext P, one uses C = eP mod n
    - To decrypt the ciphertext C, Alice uses P = dC mod n (the plaintext must be a number that is both relatively prime to, and less than, n)
- Kid-RSA works because:
    - It's easy to perform multiplication in modulo arithmetic
    - It's hard but not too hard to perform division in modulo arithmetic
- Other Public-Key Encryption Algorithms:
    - ElGamal: it's based on the Diffie-Hellman Protocol and It relies on the hardness of the discrete logarithm problem rather than the integer factorisation problem. It produces larger ciphertexts
    - Elliptic Curve Cryptography (ECC): It is also based on the hardness of the discrete logarithm Problem and has many performance advantages
- Attack Models & Security Goals:
    - similar to those for symmetric encryption
    - However since the public-key is public an attacker can perform encryption queries as required
    - The default model for asymmetric encryption is therefore the chosen-plaintext attacker (CPA). Ciphertext-only attacks (COA) and known-plaintext attacks (KPA) are unrealistic since the attacker has access to the public-key

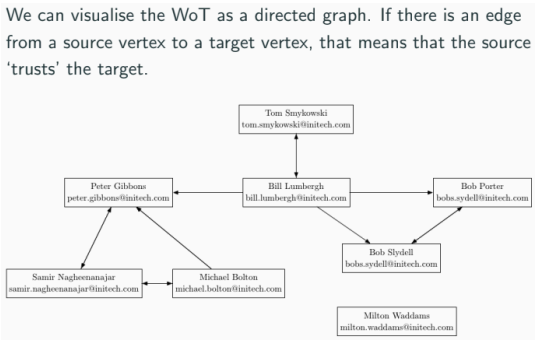
12: Digital Signatures

- A digital signature is a scheme for presenting the authenticity of digital messages or documents
- A valid digital signature gives a recipient reason to believe that:
    - the message was created by a known signatory (authentication)
    - the message was not altered after being signed (integrity)
    - the signatory cannot deny having signed the message (non-repudiation)
- Digital signatures are often used to implement electronic signatures (any electronic data that carries the intent of a signature. They have legal significance).
- Where M is the message, S is the signing process, N is the signed message, V is the verification process, Kpub is the public-key and Kprv is the private-key: we can write N = S(M,Kprv) and M = V(N,Kpub)

- A public-private keypair (Kpub and Kprv) is determined solely by the signatory
    - If you want to verify a message from me, you must request me to generate a keypair
    - I generate a keypair, one part of which I send to you (Kpub) and the other part I keep to myself (Kprv)
    - I sign the message using Kprv and send it to you
    - You verify the signed message using Kpub
    - If I want to verify a message from you, I must request you to generate a keypair
- Digital Signature Algorithms
    - RSA
    - ElGamal
    - DSA (Digital Signature Algorithm): a variant of ElGamal, standardised by NIST
    - ECDSA (Elliptical Curve Digital Signature Algorithm): a variant of DSA
    - Schnorr: efficient and generates short signatures
- OpenPGP (Pretty Good Privacy): it defined standards for programs to communicate freely but securely by using an enhanced version of the PGP protocol and a variety of different hashing and encryption algorithms



- OpenPGP provides five features:
    - Encrypt a message with a public-key (confidentiality)
    - Decrypt a message with a private-key (confidentiality)
    - Sign a message with a private-key (authentication, integrity, non-repudiation)
    - Verify a signature with a public-key (authentication, integrity, non-repudiation)
    - Establish identity with the Web of Trust (trust)
- To setup OpenPGP, create and store:
    - A keypair by specifying: name, email, comment (UID), Key algorithm (ex: RSA), Key length (ex: 2048 bits), Key expiration date
    - A revocation certificate (optional but recommended): this won't revoke the keypair immediately, but it will allow to revoke the keypair in the future even if access to the private-key is lost
    - A public-key can be distributed using a website, email signature, keyservers
- Public-Key Infrastructure (PKI) Web of Trust:
- OpenPGP relies the Web of Trust: a friend-of-a-friend honour system that relies on physical meetings where people verify identities
    - strong set: the largest collection of strongly connected PGP keys
    - any two keys in the strong set have a path between them
    - what can go wrong: poor usage (emailing your private-key and passphrase to yourself), fake keys, poor key-signing practices, software / hardware / people compromise



- Hash-Then-Sign (supported by OpenPGP):
    - asymmetric cryptography (encrypting, decrypting, signing, verifying) is slow for large messages
    - it is faster to hash the message and then sign the hash
    - 'Sign-then-encrypt' is better than 'encrypt-then-sign': 'Encrypt-then-sign' doesn't prove that the signatory was aware of the context of the plaintext
- Hybrid Cryptosystems: combines the convenience of a public-key cryptosystem with the efficiency of a symmetric-key cryptosystem

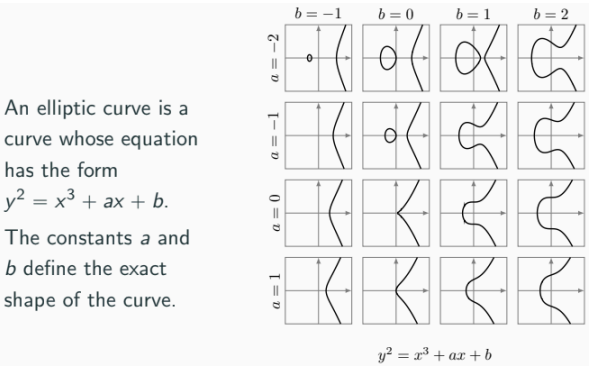
    - Public-key cryptosystems are convenient but they often rely on complicated mathematical computations
    - A hybrid cryptosystem is constructed using any 2 separate cryptosystems:
        - a key encapsulation scheme, which is a public-key cryptosystem
        - a data encapsulation scheme, which is a symmetric-key cryptosystem
    - The key encapsulation scheme encrypts the session key. The data encapsulation scheme uses the session key to encrypt the plaintext.
    - Used by TLS and the OpenPGP

13: Elliptic Curves

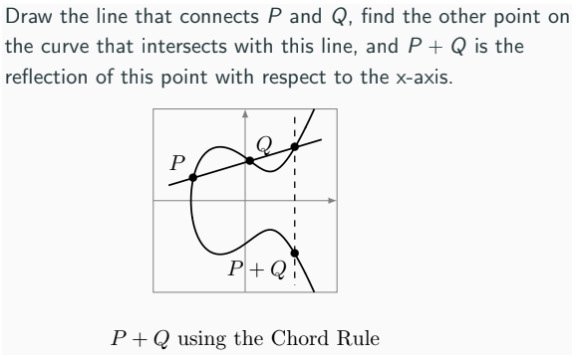
- Elliptic Curve Cryptography (ECC): an alternative approach to asymmetric cryptography.
- It requires smaller keys compared to non-ECC to provide equivalent security.
- It is more powerful and efficient than alternatives like RSA and classical Diffie-Hellman
- NIST Recommended Key Sizes

Security (bits)	Symmetric Algo.	RSA Key Size	ECC Key Size
112	3DES	2048	224
128	AES-128	3072	256
192	AES-192	7680	384
256	AES-256	15 360	512

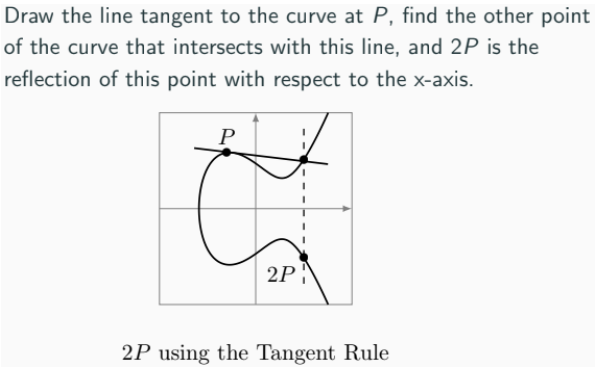
- RSA is based on the integer factorisation problem and Diffie-Hellman is based on the discrete logarithm problem: no efficient algorithms are known for classical (non-quantum) computers
- Trapdoor Function:
    - a function that is easy to compute in one direction, but difficult to compute in the opposite direction without special information
    - The bigger the spread between the difficulty of going in one direction when compared with going in the other, the more secure the system will be
    - It is needed to build a public-key cryptosystem
    - For RSA, the easy computation is the multiplication of two prime numbers and the difficult computation is the factorisation of the product of two prime numbers (the integer factorisation problem)
- Elliptic curve:
    - a curve on a plane, and a curve's equation defines all the points that belong to that curve



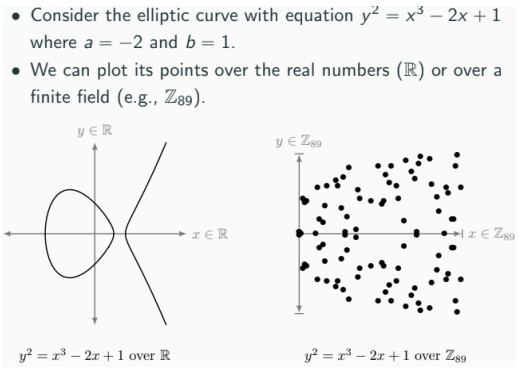
- We can add two points on an elliptic curve using the Chord Rule



- We can double a point, P, on an elliptic curve using the Tangent Rule:



- Multiplication of a Point by a Constant:
  - We can compute kP by applying the addition operation k - 1times: P + P + . . . + P. However, this is slow for large k.
  - We can efficiently compute kP by combining the addition and doubling operations. For example: we can efficiently compute 9P using three doublings and one addition: 9P = P * 2 * 2 * 2 + P = 9P
- Elliptic Curve over a Finite Field



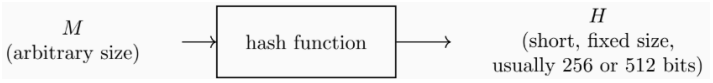
    - The elliptic curve on the finite plane is topologically equivalent to a torus (‘doughnut’).
    - The finite field allows us to limit our infinitely large plane to a finite one
    - If we perform additions or doublings on the finite field, we stay within the finite field
    - Given a start point P on an elliptic curve over a finite field, we can efficiently compute an end point kP
    - But given a start point and an end point, we cannot efficiently deduce k: we cannot simply ‘divide’ kP by P to get k
    - This is the trapdoor function. It is a better trapdoor function than factoring
    - Just as the security of RSA depends on the size of the numbers used, the security of an elliptic curve-based cryptosystem depends on the number of points on the curve.
    - To know the number of points on an elliptic curve (its cardinality): Zp contains approximately p points (ex: Z89 contains approximately 89 points)
- Elliptic Curve Cryptography:
    - It’s the next generation of public key cryptography. It#s based on currently understood mathematics, and it provides a more secure foundation than first generation public key cryptography systems like RSA
- Elliptic curves for key agreement, digital signatures and public-key encryption:
    - Elliptic Curve Diffie-Hellman Key Agreement (ECDH)
    - Elliptic Curve Digital Signature Algorithm (ECDSA)
    - Elliptic Curve Integrated Encryption Scheme (ECIES): this is a hybrid cryptosystem
- An elliptic curve is a curve whose equation has the form y2 = x3 + ax + b. To choose a and b we use some de facto standard curve:
    - NIST Curves (criticised because only the NSA knows the origin of the b coefficient in their equations)
    - Curve255192 (used in Chrome, Apple, etc.. Not an NIST standard)
    - Secp256k13 (popular in the blockchain)

14: Hashing

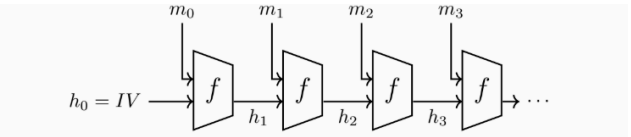
- Hash Functions such as MD5, SHA-1, SHA-256, SHA-3, and BLAKE2 are used for digital signatures, integrity verification, message authentication, passphrase storage,

key agreement protocols, cryptographic commitments and many other cryptographic protocols

- A hash function is any function that can be used to map data of arbitrary size to data of a fixed size. It returns hash values, hash codes, digests, or simply hashes



- Non-cryptographic hash functions detect accidental errors and provide no security
- Cryptographic hash functions are preimage resistant and collision resistant
- A preimage of a given hash value, H, is any message, M, such that Hash(M) = H
- Preimage resistance: the security guarantee that given a arbitrary hash value, an attacker will never find a preimage of that hash value
    - First-preimage resistance: given a hash value H, it is practically impossible to find a message M that hashes to H
    - Second-preimage resistance: given a message M1, it is practically impossible to find another message M2 that hashes to the same value that M1 does
- Collision resistance: the security guarantee that it is hard to find two messages that hash to the same value
    - Pigeonhole principle: if you have m holes and n pigeons to put into those holes, and if n is greater than m, at least one hole must contain more than one pigeon
    - It is faster to find collisions than preimages thanks to the birthday attack
    - Birthday paradox: a group of only 23 persons will include 2 persons having the same birth date with probability 1/2
- Avalanche effect: even if 2 messages are only slightly different, their has value will be totally different
- The cryptographic strength of hash functions stems from the unpredictability of their outputs
- Building Hash Functions:
- Iterative hashing: split a message into chunks and process each chunk consecutively using a similar algorithm
    - Compression-Based Hash Functions: usually based on block ciphers



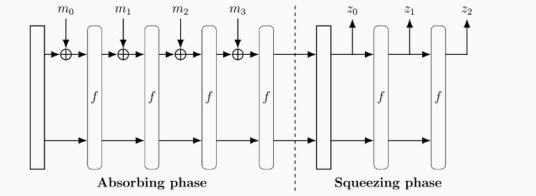
m_i is the ith chunk of the message and h_i is the hash of the ith chunk of the message.

h_0 is an initialisation vector.

Each m_i acts as a "key" and each h_i acts as a "plaintext".

    - Permutation-Based Hash Functions
        - Sponge functions are a simpler (and newer) alternative to compression functions

    - Instead of using a block cipher to mix message bits with the internal state, sponge functions perform XOR operations



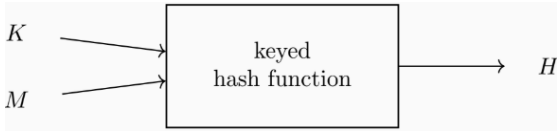
m_i is the ith chunk of the message and z_i is the ith bit of the final hash.

- Popular Hash Functions

Name	Internals	Status
MD5	Compression function	Broken in 2005.
SHA-0	Compression function	Broken in 1995–1998.
SHA-1	Compression function	Broken in 2005–2017.
SHA-2[1]	Compression function	Similar to SHA-1 but secure.
SHA-3[2]	Permutation function	Secure

15: Keyed Hashing

- Keyed Hash Functions:
    - non-keyed hash: produces an output that depends only on a message
    - keyed hash: produces an output that depends on both a key and a message. Only allows those that possess the key to compute and verify a hash



    - Other than using brute-force, it shouldn't be possible to find: an input that hashes to a particular output (preimage resistance), and two inputs that hash to the same output (collision resistance)
- Keyed hashing forms the basis of two types of important cryptographic algorithms:
    - message authentication codes (MACs)
    - pseudorandom functions (PRFs)
- Message Authentication Codes (MACs):
    - protects the integrity and authenticity of a message M using a key K by computing a value T = MAC(K, M), called the authentication tag of the message
    - The authentication tag ensures that the message wasn't corrupted in transit (integrity) and that it was sent by the right person (authenticity)

    - Secure communication systems often combine a cipher and a MAC to protect a message's confidentiality, integrity, and authenticity(WireGuard, SSH, TLS)
- Pseudorandom Functions (PRF):
    - A PRF transforms a message M using a key K into a pseudorandom output by computing a value PRF(K, M). Its output is indistinguishable from a random output
    - A key derivation function (KDF) uses a PRF to generate keys. A password-based KDF (PBKDF) can transform a password into a cryptographic key. A KDF can also transform a master key into any number of subkeys
    - Identification schemes use PRFs to generate a response from a random challenge. A server sends a random challenge message, M, and the client responds with PRF(K, M) to prove that it knows K
- Popular Keyed Hash Functions

Name	Comments
HMAC-*	Hash-based MAC is a family of functions based on hash functions, e.g., HMAC-SHA-256.
*-CMAC	Cipher-based MAC is a family of functions based on block ciphers, e.g., AES-CMAC-128.
Poly1305	Designed by Daniel J. Bernstein in 2005. Fast and secure. Difficult to implement.
SipHash	Co-designed by Daniel J. Bernstein in 2012. Easy to implement. Suitable for small messages.

16: Applications of Hashing

- Verifying the Integrity of Messages
    - compare the hash of a message (or message digest) calculated before / after transmission to determine if any changes were made to the message
    - The hash establishes a chain of trust as long as the hash is securely distributed
    - Error detecting codes such as cyclic redundancy checks (CRCs) only prevent against non-malicious modifications of a message
    - Digital forensics tools like FTK Imager, compute hashes to form chains of custody
- Signature Creation & Verification
    - Most digital signature schemes require a hash to be calculated over the message
    - This allows the signature calculation to be performed on a small, fixed size hash rather than a message of arbitrary size
    - Be careful when choosing a key size and a hash function for a digital signature
    - The security in bits for a hash function is half the size of the hash due to the birthday attack

    - The security of a system depends on the weakest link: If your digital signature scheme uses an RSA key with 15 360-bits but the SHA-224 hash function, then the security of your system is 112-bits; not the 256-bits provided by the RSA key

Security (bits)	...	RSA Key Size	...	Hash Function
112	...	2048	...	SHA-224
128	...	3072	...	SHA-256
192	...	7680	...	SHA-384
256	...	15 360	...	SHA-512

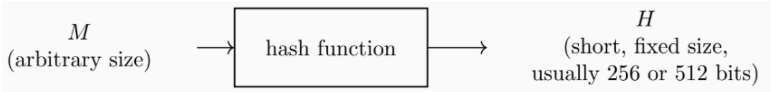
- Passphrase Storage & Verification
    - Don’t store the passphrases in the clear
    - Don’t encrypt the passphrases using a secret key and store the ciphertexts: both the database and the key could be compromised
    - Don’t hash the passphrases and store the hashes (though it’s better): The set of passphrases that people actually use is very limited and hash functions are the same everywhere. An attacker can hash many popular passphrases, creating huge tables mapping hash values to their corresponding passphrases (rainbow table)
    - Don’t hash the passphrases with salt and store the salt and hashes: an attacker can still generate salt-specific rainbow tables
    - BEST OPTION:
        - Generate a unique salt per user
        - Combine the salt with an application-wide pepper: a cryptographically random value and there is only one pepper per application (this acts like a secret key)
        - Configure an iterated hash function using appropriate parameters
        - Hash the passphrase
        - Store the salt, the parameters, and the salted, peppered, iterated hash in a database
- Popular Iterated Hash Functions
    - Many algorithms have a time–memory trade-off
    - Scrypt deliberately makes this trade-off costly in either direction to prevent GPU, FPGA, ASIC attacks
    - The minimum memory required to compute the function is 128 * N * r bytes

Name	Comments
PBKDF2-*	Family of functions based on keyed-hash functions, e.g., PBKDF2-HMAC-SHA-256. They are *compute-hard*. Avoid!
bcrypt	Based on the Blowfish block cipher. It is *memory-hard*. Old but good.
Scrypt (/'ɛs.krɪpt/)	Based on PBKDF2-HMAC-SHA256 and Salsa20. Also memory-hard.
Argon2	Winner of the Password Hashing Competition (PHC), organised by JP Aumasson, in 2012. Also memory-hard.

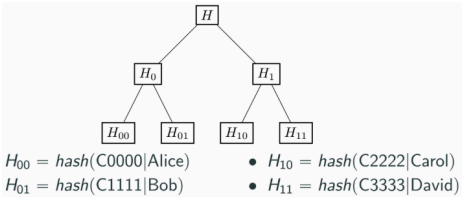
- File & Data Identification: Git and Magnet use SHA-1 hashes
- Proof-of-Work: a piece of data which is difficult (costly, time-consuming) to produce but easy for others to verify and which satisfies certain requirements. This imposes a cost on the sender

17: Merkle Trees

- Hash function: can be used to map data of arbitrary size to data of a fixed size. The values returned by a hash function are called hash values, hash codes, digests, or simply hashes



- Merkle tree (type of digital signature based on symmetric cryptography): a tree where every leaf node contains the hash of a piece of data and every non-leaf node contains the hash of its children. They're usually binary trees and the same hash function is used in all cases (ex SHA-256)



- The non-leaf nodes contain the hash of the hashes in their children:

$$H_0 = hash(H_{00}|H_{01})$$
$$H_1 = hash(H_{10}|H_{11})$$
$$H = hash(H_0|H_1) \text{ (the } root \text{ node)}$$

- Dummy leaf nodes containing random values are added to make the tree perfect: all non-leaf nodes have two children and all leaf nodes have the same depth / level

- Merkle proof:
    - Suppose I have a database containing the records of all ~8000 students at SETU Carlow. I can construct a binary Merkle tree with 14 levels and 8192 leaf nodes (214´1 “ 8192) to store the records of all of the Students. The first ~8000 leaf nodes will contain the hashes of the records of each student. The remaining leaf nodes will contain random values. I can compute the hashes for all of the non-leaf nodes, including H, the hash for the root node. I can advertise H to everyone by putting it on the SETU Carlow website or publishing it in a newspaper. This is a hash with a fixed size, usually 256 or 512 bits. It does not reveal anything about the registered students (count, records, etc.).
    - Can I prove to you without sending you the records of any other student that your record is stored in the database? Yes: I can send you the entire Merkle tree; you can verify that a leaf node contains the hash of your record and that the root node contains H
    - Can I prove this to you without sending you the records of any other student and without sending you the entire Merkle tree? Yes: I can do it by sending you just 13 hashes – one less than the number of levels in the Merkle tree!
- Why Merkle trees: integrity, validity, easy to construct, small (Bitcoin, Cassandra, etc)

18: Quantum Cryptography

- cryptographically relevant quantum computers (CRQCs) may exist in the future
- they have the potential to break many algorithms in asymmetric cryptography
- Quantum mechanics is a branch of physics that studies the behavior of subatomic particles, e.g., protons, neutrons, electrons, quarks, etc
- Subatomic particles behave very ‘strangely’:
    - Quantum Superposition: The double-slit experiment shows that subatomic particles behave both as a wave and a particle
    - Quantum Entanglement: Two subatomic particles can become entangled such that measuring the state of one instantly determines the state of the other, even if separated by vast distances
    - Quantum Tunneling: Subatomic particles can pass through barriers in a way that defies classical mechanics
    - Heisenberg Uncertainty Principle: You cannot simultaneously know a subatomic particle’s exact position and momentum
- Unlike classical computers, which operate on bits that are either 0 or 1, quantum computers operate on qubits, which can be |0> and |1> “simultaneously” (superposition).
- Quantum algorithms have three steps:
    - 1. Initial State Preparation: Qubits are initialised to a starting state
    - 2. Quantum Circuit/Algorithm Execution: Quantum gates manipulate and entangle the qubits
    - 3. Measurement: The qubits are measured and the quantum state collapses into classical bits

- Quantum algorithms rely on interference where subatomic particles can combine or cancel out each other's effects due to their wave-like nature
- Quantum algorithms exploit interference so that the "waves" of valid solutions reinforce each other but invalid solutions cancel each other out
- Quantum Speedup: occurs when a quantum algorithm can solve a problem "faster" than the best classical one
- Shor's Algorithm is a quantum algorithm that produces an exponential speedup when solving the integer factorisation problem (ILP), the discrete logarithm (DLP), and the elliptic curve discrete logarithm problem (ECDLP). has implications for asymmetric cryptography: it breaks RSA, DH, and ECC
- Grover's algorithm (quadratic speedup): search for an element within an unordered list. has implications for symmetric cryptography: it reduces the numbers of operations needed to explore many search spaces
- Post-Quantum Cryptography: proposes cryptographic algorithms (usually, asymmetric) that a quantum computer can't break. They rely on the hardness of problems that can't be efficiently solved by Shor's algorithm
    - There are four main types of post-quantum algorithms:
        - 1. Code-Based Cryptography: Relies on the hardness of problems found in error-correcting codes
        - 2. Lattice-Based Cryptography: Relies on the hardness of problems found in Lattice Theory. Provides a good combination of security and performance
        - 3. Multivariate Cryptography: Relies on the hardness of problems encountered when solving systems of multivariate equations
        - 4. Hash-Based Cryptography: Relies on the hardness of finding collisions or preimages for cryptographic hash functions
- Candidate PQC Algorithms:
    - (CRYSTALS-)Kyber: A lattice-based key encapsulation mechanism (KEM) that can encrypt / decrypt data and perform key-agreement
    - (CRYSTALS-)Dilithium: A lattice-based digital signature scheme
    - Falcon: A lattice-based digital signature scheme.
    - Smaller signatures than Dilithium.
    - SPHINCS+: A hash-based digital signature scheme
- How Long Do We Have: we know the theory, not the practice. Issues:
    - Qubits require the control and manipulation of subatomic particles
    - Qubits require extremely low temperatures to remain stable (coherence time)
    - Qubits can be adversely affected by their environment, like magnetic fields