Cryptography fundamentals

Organization

- What is cryptography ?
- How does cryptography relates to cybersecurity ?
- Symmetric cryptography
- Public key cryptography
- Hybrid cryptography

What is the subject ?

- Cryptography is a set of technologies to provide :
  - Data protection
  - Data transfer protection
  - Authentication of entities

What is data protection ?

- Data protection means ensuring some **properties** on data and data transfers
  - Confidentiality
    - « I am sure that nobody but authorized users can read the data »
  - Integrity
    - « I am sure that the data has not been tampered with »
  - Authenticity
    - « I am sure that the data has really been issued by the claimer issuer »
  - Non-repudiation
    - Non-repudiation is the assurance that someone cannot deny the validity of something

What is system protection ?

- Protection of data handled by the system
- System availability
  - « I am sure that the system is always available »
    - « Available » means that I can use the system **at any time** for **any service** it is designed to provide
- ...

Confidentiality

- Goal : to keep some piece of information **secret**

- Operations related to confidentiality
  - Ciphering / encryption
    - Process to turn a piece of information from a readable form to anyone into a non-readable form to anyone but **authorized users**
  - Deciphering / decryption
    - The reverse process

Integrity

- Goal : to **check** that some piece of information has not been **modified**
  - It does **not** prevent tampering (data modification), but it prevents such a modification to get **unnoticed**

- Operations related to integrity
  - **Generation** of a data footprint before transmitting the data
  - **Verification** of the data footprint after the data has been received

Authenticity

- Goal : to check that the originator of a piece of information really is who she/he/it claims to be
  - It does not prevent to change the data issuer identity, but it prevents this modification to get **unnoticed** → spoofing prevention

- Operations related to authenticity
  - **Generation** of a data signature (i.e. proof of identity) or authentication tag (i.e. proof of knowledge) before transmitting the data
  - **Verification** of the signature or tag after the data has been received

Threats

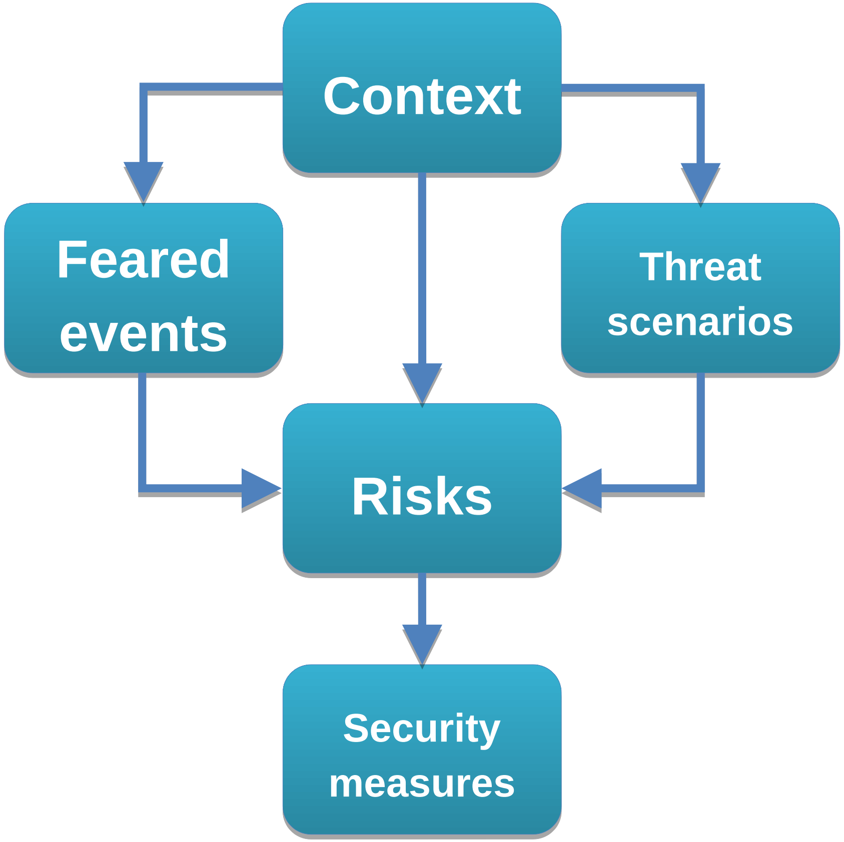
- What are the threats regarding the system and the data it handles ?
  - Unauthorized access to the system ⇒ authenticity
  - Sensitive data exposure ⇒ confidentiality
  - Impossibility to access the service ⇒ availability
  - Transfer of file whose origin is unsure, or whose content has been modified ⇒ authenticity, integrity
  - Modification of data to hide one’s real identity ⇒ non-repudiation
  - ...

Risk analysis

- The threats are always relative to a data / system property

- A method for **analysing risks** is of utter importance

Example : EBIOS method



Context	Why and how do we manage risks ? What is the study subject ?
Feared events	What are the feared events ? Which are the worst ?
Threat scenarios	What are all the possible scenarios ? Which are the most likely?
Risks	What are the risks ? How do we choose to treat them ?
Security measures	What measures should be taken ? Are the residual risks acceptable ?

Cybersecurity and cryptography

- Once threats are known, protective actions can be taken
  - The system and data can be secured

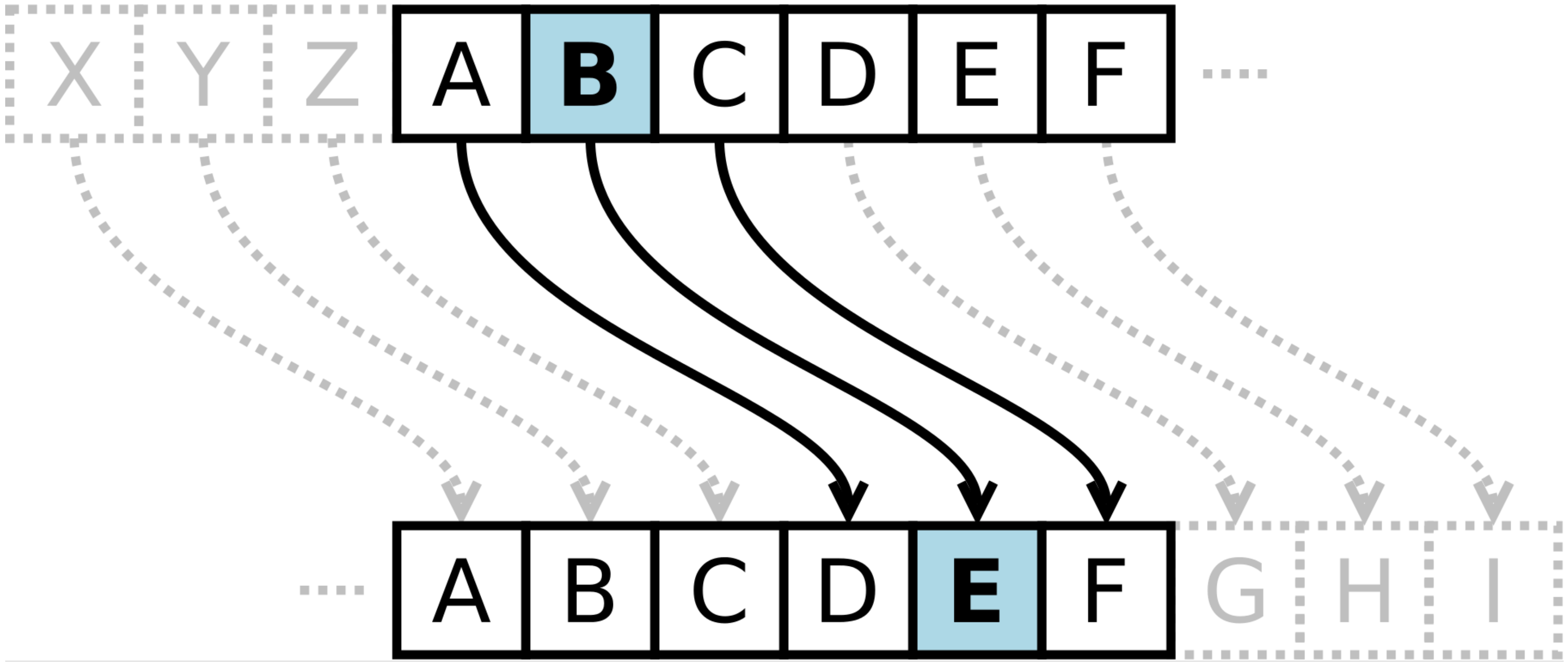
- Protective actions often rely on **cryptography**
  - Cryptography, when used to try to **break** cryptographic security systems, is called cryptanalysis

Confidentialy assurance

- Data must be ciphered by the issuer then deciphered by the receiver(s)
- ...and only the receiver(s) can perform the deciphering operation

- It exists many ways to ensure this property
  - An old example : the Caesar cipher

Caesar cipher



Caesar cipher

- Assumption : the ciphering process **is** the secret
  - Otherwise, the code can easily be broken by brute force attack or by frequential analysis

- How many attempts at most would yield the solution ?

The Kerckhoffs's principle

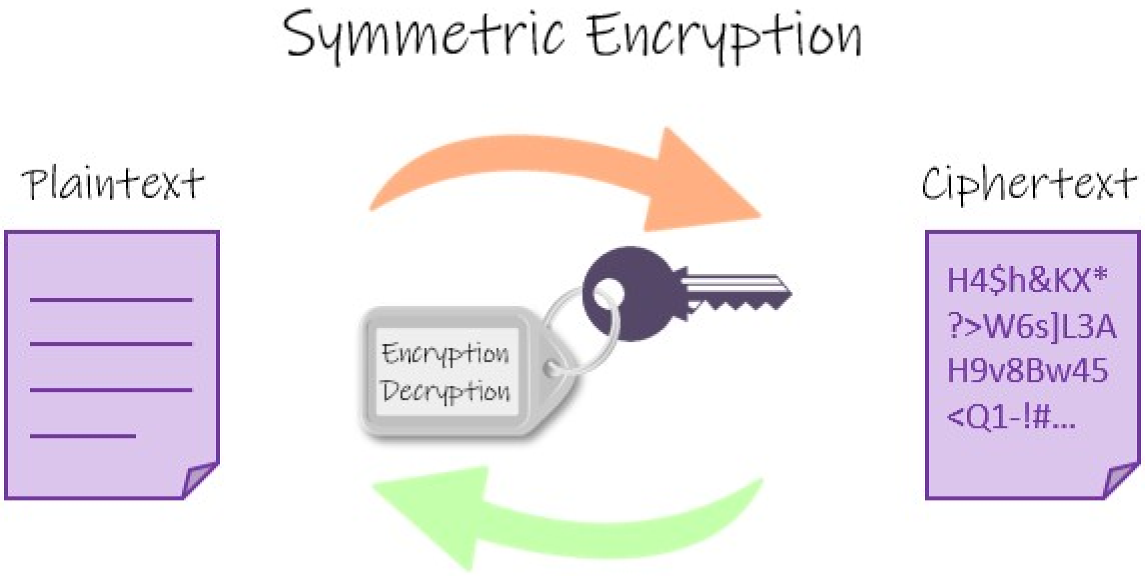
- Modern approach : the Kerckhoffs's principle (end of 19th century)
  - The ciphering process is **not** secret
  - The secret is an **input** to either the ciphering and/or deciphering process
  - Making public the ciphering/deciphering process actually leads to **more secure** algorithms
    - More secure : why is that ??

Symmetric cryptography

- Confidentiality is generally ensured using a symmetric ciphering algorithm
  - Symmetric : the **same** secret (called a key) is used for both ciphering and deciphering of sensitive data

- Example : AES (Advanced Encryption Standard)

Symmetric cryptography



@101computing.net

Symmetric cryptography

- AES (Advanced Encryption Standard)
  - Invitation to tender organized by the NIST (US), between 1997 and 2000
  - A Belgian team won the selection process
  - 128-bit block cipher algorithm
    - Data to be ciphered is split in 128-bit long block
  - Like **any** block cipher algorithm, it only describes **part** of the encryption process…
    - What is missing ??

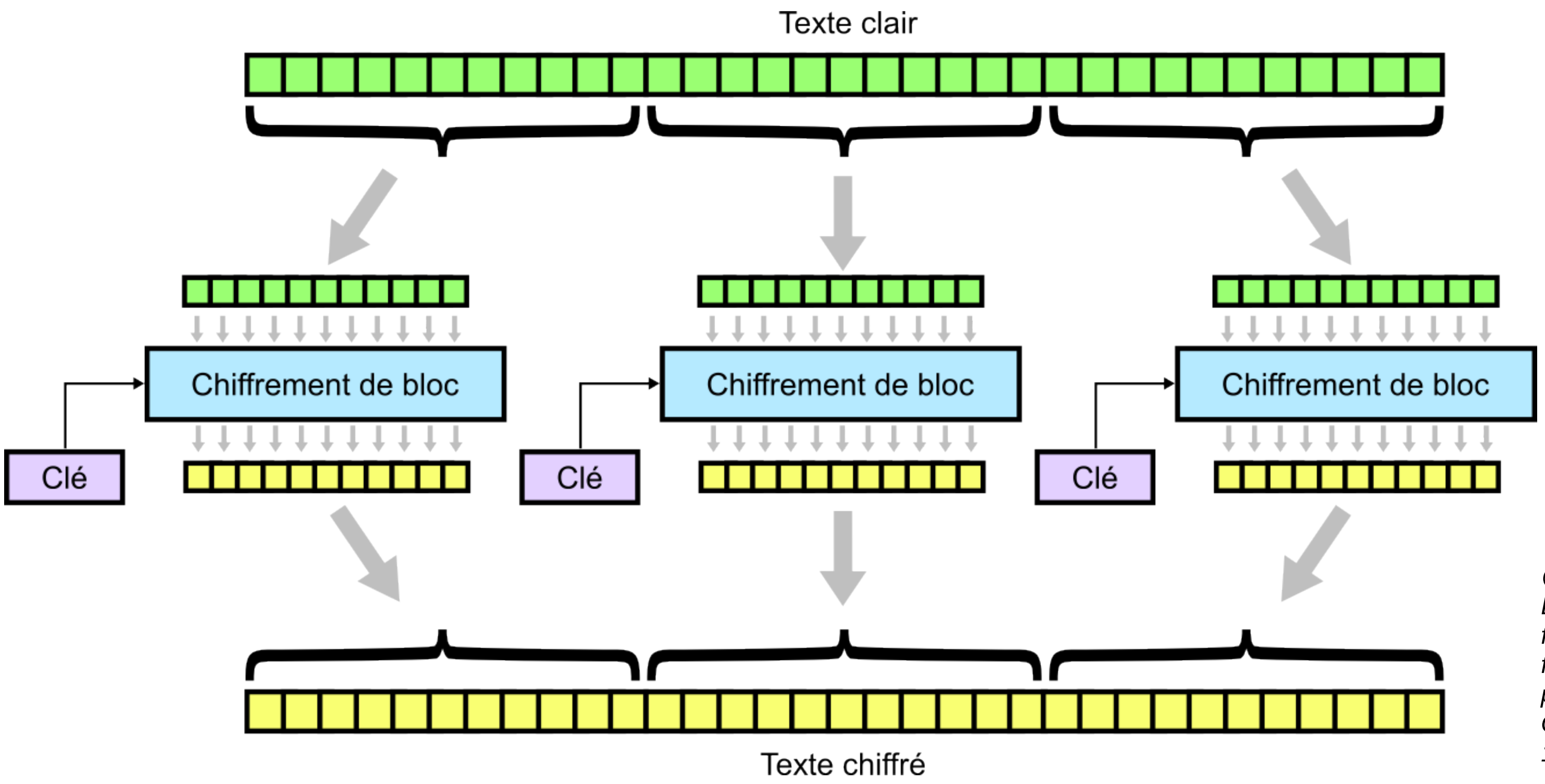
Block cipher mode of operation

- **How** is applied the 128-bit block algorithm on the complete data (which is generally much larger than 128 bits) ?

Block cipher mode of operation

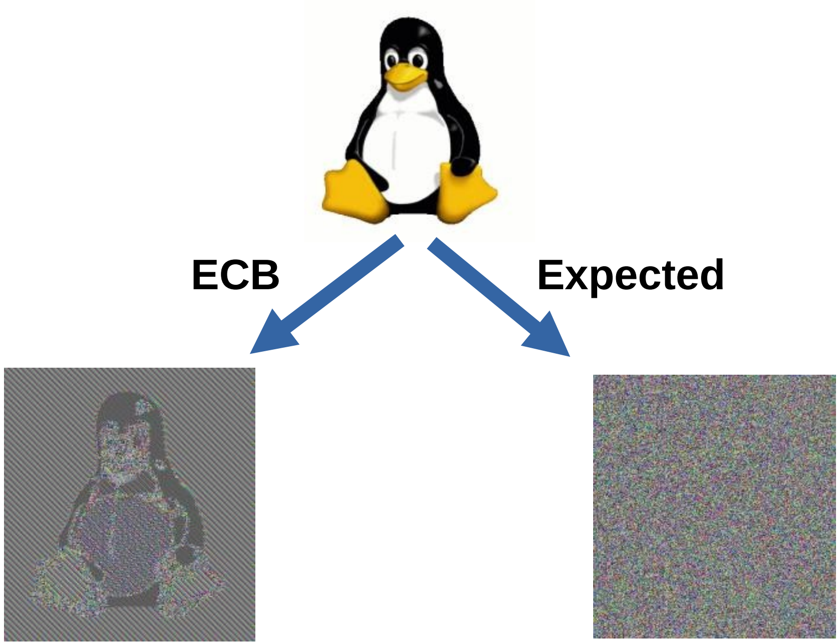
- **Naive** solution : apply the algorithm on each 128-bit block of the message ⇒ ECB (Electronic Code Block)

Block cipher mode of operation



Block cipher mode of operation

- This solution fails to provide a high security level because each **same** 128-bit block of clear text produces **always** the same output



Other operating modes

- Each 128-bit block of data to be ciphered is XOR-ed with a 128-bit block which is either :
  - The result of the ciphering of the previous block (Cipher Feedback Mode or CFB)
    - Use of an Initialisation Vector (IV) to provide an initial value
  - The result of the ciphering of pseudo-random block or 128 bits (Cipher Block Chaining or CBC)
    - Use of an Initialisation Vector (IV) to provide an initial value to be ciphered
  - The result of the ciphering of a 128-bit block based on a pseudo-random number (nonce) to which is added a **counter** starting usually at 0 and incremented by 1 each time (Counter Mode or CTR)

Other operating modes

- As a matter of fact, the ciphering algorithm is **NOT** applied on the data itself

Block cipher mode of operation / Key takeaways

- Specifying the block ciphering algorithm used to cipher a plaintext (AES, Blowfish, 3DES, …) is **not** enough
  - The mode of operation is **as important** as the block algorithm

- The overall **performance** of ciphering plaintext depends heavily on the operating mode
  - Currently, the best ratio performance/processing power is obtained using variants of the Counter mode
    - As it allows for **parallel processing** of the plaintext blocks
  - The **key length** (128 bits, 192 bits, 256 bits, …) is of primary importance

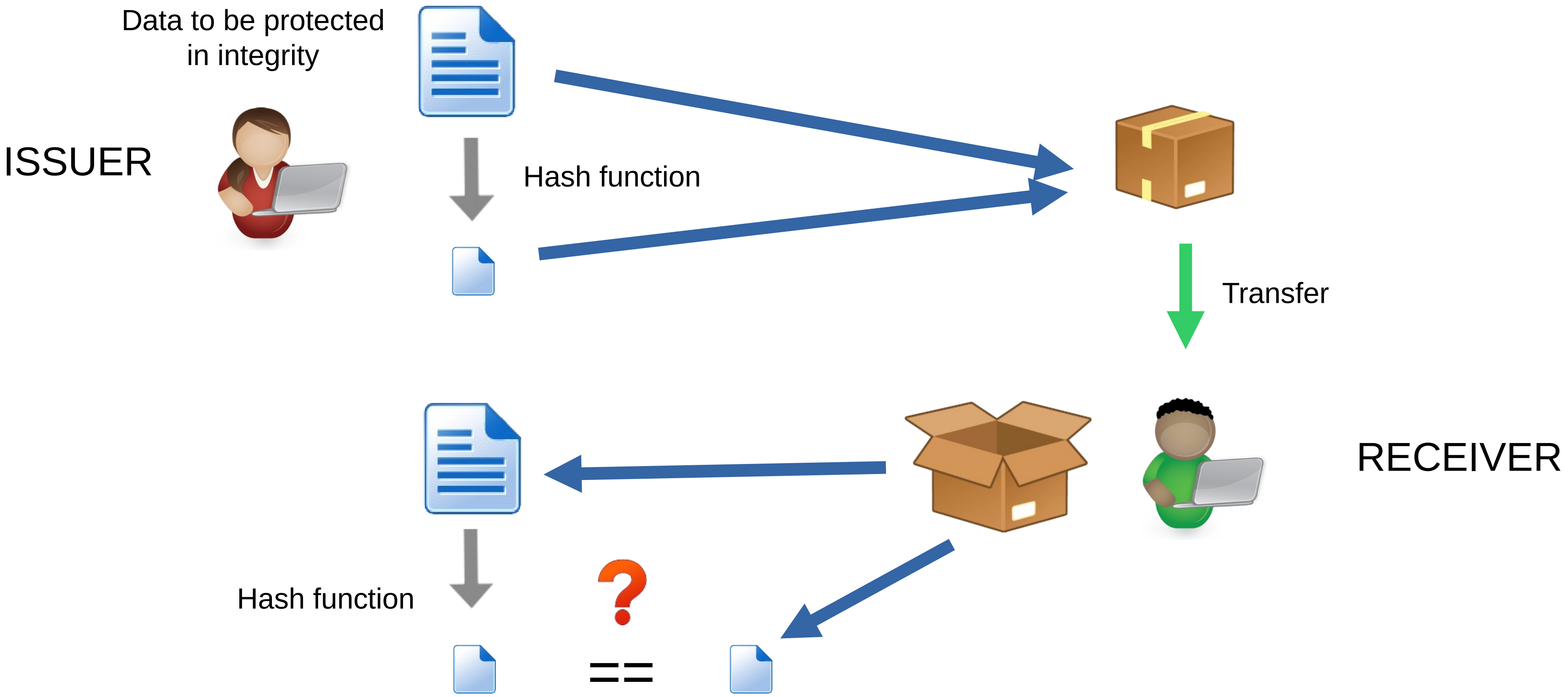
Integrity

- How can I be sure that the data I received has not been tampered with ?
- The solution is based on the computation of a small value that **represents** the data
  - This small value is called a hash of the data
  - The function used to compute the hash value is called a hash function
  - A hash function is a surjection from the set of input data to the set of hashed values
    - There may exist collisions when several input produce the same hash
    - A « good » hash function produces **few** collisions

Integrity / Procedure

- The data issuer computes the hash of the transferred data and **attaches it** to the data itself
- The data receiver computes the hash of the received data and **compares** this value with the hash received
  - If both hashes match, the integrity of the transferred data is deemed verified

Integrity / Procedure



Usual hash functions

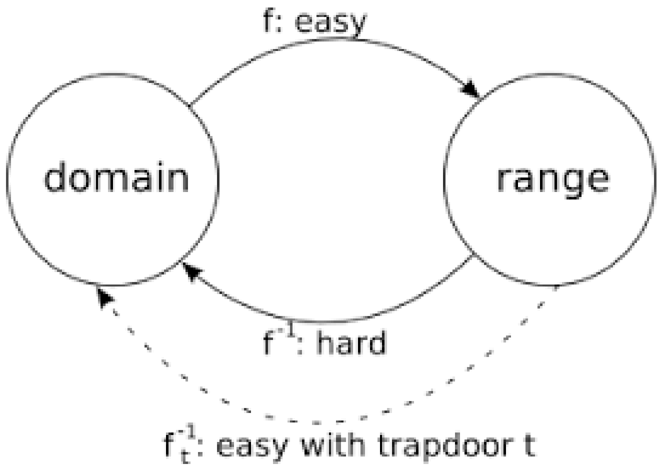
- MD4 / MD5
- CRC (Cyclic Redundancy Check)
- SHA-0 / SHA-1 (SecureHash Algorithm)
  - Not trusted anymore
- SHA-2
  - Several variants (SHA-256, SHA-512, SHA-512/256, SHA-512-224, …)
- SHA-3, aka. Keccak
  - Designed by (among others) one of the AES designers

Authenticity

- How can I be sure that the data has really been issued by the claimed issuer ?
- This problem can be solved using public-key cryptography (aka. asymmetric cryptography) as well as symmetric cryptography (within limits)
- Public-key cryptography relies on the use of very special mathematical functions : trapdoor functions (one-way functions with a trapdoor)
  - A hash function is an example of pure one-way function, as there is no trapdoor to determine the original based on the hash value

Public-key cryptography

- A trapdoor function is a function that is **easy** to compute in **one direction**, but **very difficult** to compute in **reverse direction** (finding its inverse function) without a special information called a **trapdoor**



Public-key cryptography

- In practice, asymmetric cryptography relies on an algorithm used to generate two keys : K_1 and K_2
- K_1 is a trapdoor for messages ciphered with K_2
  - A message ciphered with K_2 can « only » be deciphered using K_1
- K_2 is a trapdoor for messages ciphered with K_1
  - A message ciphered with K_1 can « only » be deciphered using K_2

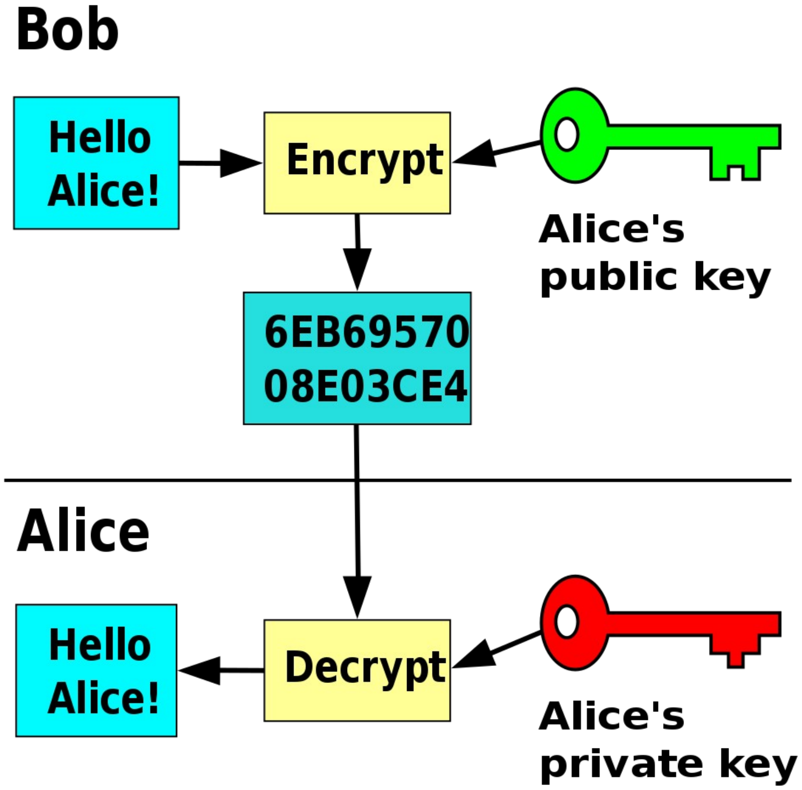
Public-key cryptography

- One of the keys, for instance K_1, is considered a private key, while the other (K_2) is deemed a public key
- Each entity (user, component, system…) is assigned a **pair** of such keys : a private one and a public one
  - The private one must be kept absolutely **secret** by its owner
    - It must **never** be transferred to another entity
  - The public one **must** be published so that secure communication can take place

Public-key cryptography

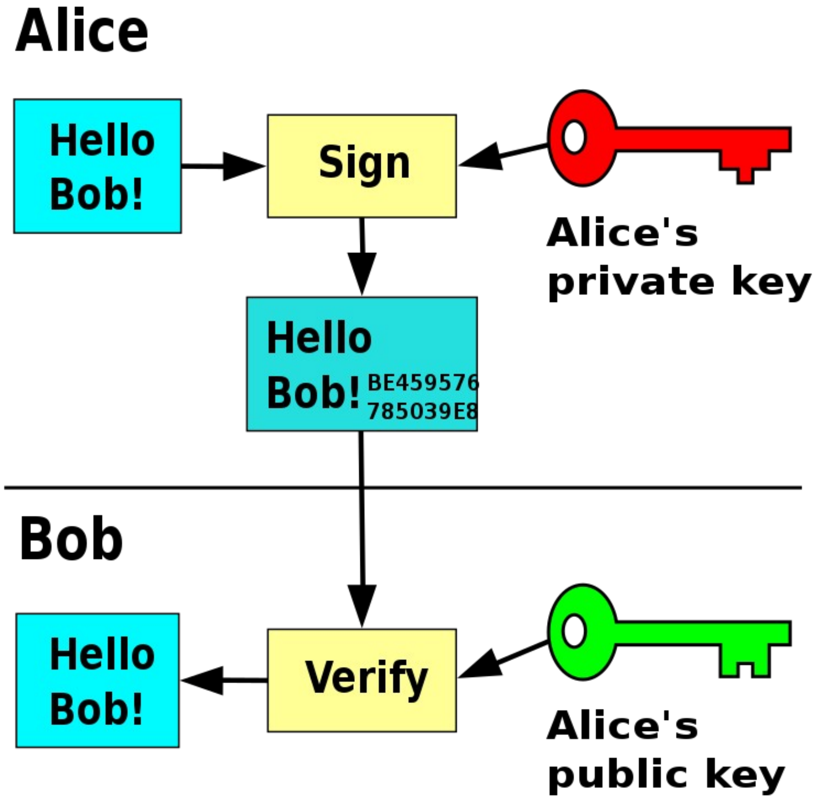
- Any entity (user, component, system…) knowing some entity's public key can send it a message **ciphered** with this **public** key
  - Only the **private key owner** can decipher the message ⇒ confidentiality is ensured
-
- Conversely, the **private key owner** entity can send a message **ciphered** with its **private** key
  - Any entity knowing the issuer's **public** key can decipher the message ⇒ this ensures the issuer authenticity

Public-key cryptography / Confidentiality



@David Gothberg - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1028460

Public-key cryptography / Authenticity



NB. In this example, the message is **NOT** ciphered ⇒ **please don't mix** confidentiality and authenticity !

@David Gothberg - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1028460

Public-key cryptography / Usages

Processing performed by A	Meaningful operation ?	Property

Public-key cryptography / Usages

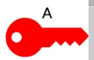
Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key		

Public-key cryptography / Usages

A **B**

Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key	YES	

Public-key cryptography / Usages

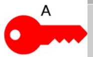


Processing performed by A	Meaningful operation ?	Property
Cipher with A’s private key	YES	Authentication

Public-key cryptography / Usages

Processing performed by A	Meaningful operation ?	Property
Cipher with A’s private key	YES	Authentication
Cipher with A’s public key		

Public-key cryptography / Usages

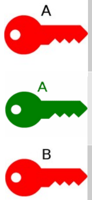



Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key	YES	Authentication
Cipher with A's public key	Possibly (e.g. backup)	

Public-key cryptography / Usages

Processing performed by A	Meaningful operation ?	Property
Cipher with A’s private key	YES	Authentication
Cipher with A’s public key	Possibly (e.g. backup)	Confidentiality

Public-key cryptography / Usages



Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key	YES	Authentication
Cipher with A's public key	Possibly (e.g. backup)	Confidentiality
Cipher with B's private key		

Public-key cryptography / Usages

Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key	YES	Authentication
Cipher with A's public key	Possibly (e.g. backup)	Confidentiality
Cipher with B's private key		

Public-key cryptography / Usages

Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key	YES	Authentication
Cipher with A's public key	Possibly (e.g. backup)	Confidentiality
Cipher with B's private key		[redacted]

Public-key cryptography / Usages

A **B**

Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key	YES	Authentication
Cipher with A's public key	Possibly (e.g. backup)	Confidentiality
Cipher with B's private key		[redacted]
Cipher with B's public key		

Public-key cryptography / Usages

Processing performed by A	Meaningful operation ?	Property
Cipher with A's private key	YES	Authentication
Cipher with A's public key	Possibly (e.g. backup)	Confidentiality
Cipher with B's private key		
Cipher with B's public key	YES	

Public-key cryptography / Usages

Processing performed by A	Meaningful operation ?	Property
Cipher with A’s private key	YES	Authentication
Cipher with A’s public key	Possibly (e.g. backup)	Confidentiality
Cipher with B’s private key		[redacted]
Cipher with B’s public key	YES	Confidentiality

Main algorithms

- The main public-key algorithms are :
  - **RSA** (Rivest Shamir Adleman)
    - Widely used in the web
  - **El Gamal**
  - **DSA** (Digital Signature Algorithm)
    - A variant of El Gamal
  - **Elliptic curve cryptography**
    - A family of public-key algorithms

Limitations

- Ciphering / deciphering operations are **very slow**, compared to symmetric cryptography
  - Many more CPU cycles to perform encryption / decryption
  - Keys are much longer (usually 2048 / 4096 bits, instead of 128 / 192 / 256 bits)
- Only very small messages can be ciphered (a few hundreds of bytes only – around the **key size**)
  - Why not iterate over successive blocks of data (mode of operation) ?

    ⇒ **much too costly** as far as processing power is concerned

Authenticating data / Procedure

- So how is authenticity ensured in practice, using PKC ?
- A **hash** of the data to be authenticated is produced
- The hash is **ciphered** by the data issuer ciphered using its **private** key
  - One says that the data has been signed by the issuer
- Thus **any** user knowing the issuer **public key** can decipher the hash value
- The data and its hash are transferred to destination

Authenticating data / Procedure

- The receiver **computes** the hash of the received data
- The receiver **compares** the received hash with the computed one
  - If the hash values are **the same**, it is a **proof** that the received message has been produced by the private key owner and that the message has not been tampered with
  - Except for collisions of course

Public key authenticity

- Using a public key to check data authenticity is good and well, but can a public key be implicitely trusted ?

Public key authenticity

- No ! It is a well-known attack ⇒ man in the middle (MITM)
- The solution is to authenticate the public key itself
  - Certificate (like X509 certificates used for web browsing using HTTPS)
  - Public Key Infrastructure (PKI)
  - Certification Authority (CA)
  - Certificate Revocation List (CRL) for certificate expiration date

  ⇒ authenticating public keys and managing their **lifecycle**

Confidentiality revisited

- Due to technical limitations, confidentiality of medium to large data cannot be ensured using public-key infrastructure
  - Too slow and requires powerful processing power
  - Maximum size of data to cipher severely limited
- So let’s go for symmetric cryptography
- But, as far as security is concerned, can symmetric cryptography be **easily** used ?

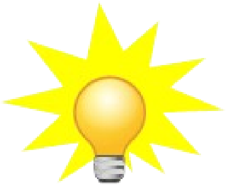
Secret sharing

- Symmetric cryptography relies on the use of a **shared secret**

- The question is : how can I share a secret key if the communication channel between the communicating peers is **not** safe ?

Secret sharing

- A solution to this problem has officially produced by **Diffie and Hellman** in 1976
  - That was the starting point of asymmetric cryptography



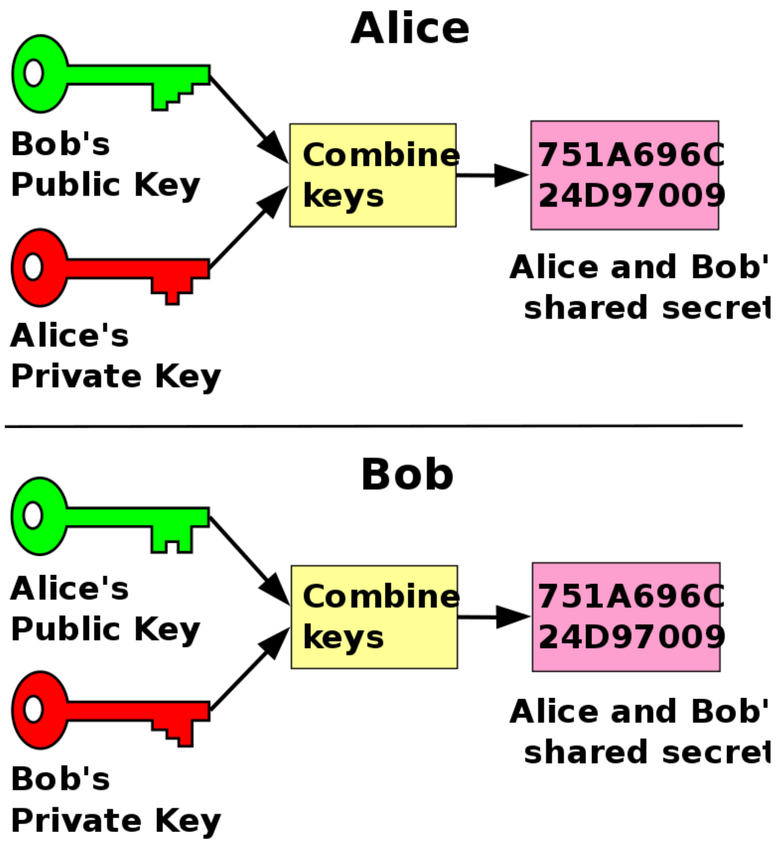
Key exchange/ Procedure

- The solution to build a **confidential communication channel** between two peers is public-key cryptography :
  - Each peer transmits its public key to the other side
  - Each peer **combines** one's own secret (private key) with the peer's public key to build a **shared secret**
  - This shared secret is used as a **symmetric ciphering key**
  - The nice point is that each side has defined the same secret **without exchanging sensitive information**

Key exchange

- Communication based on symmetric ciphering can then take place
  - **No** real size limitation
  - **Very fast** communication channel

Key exchange

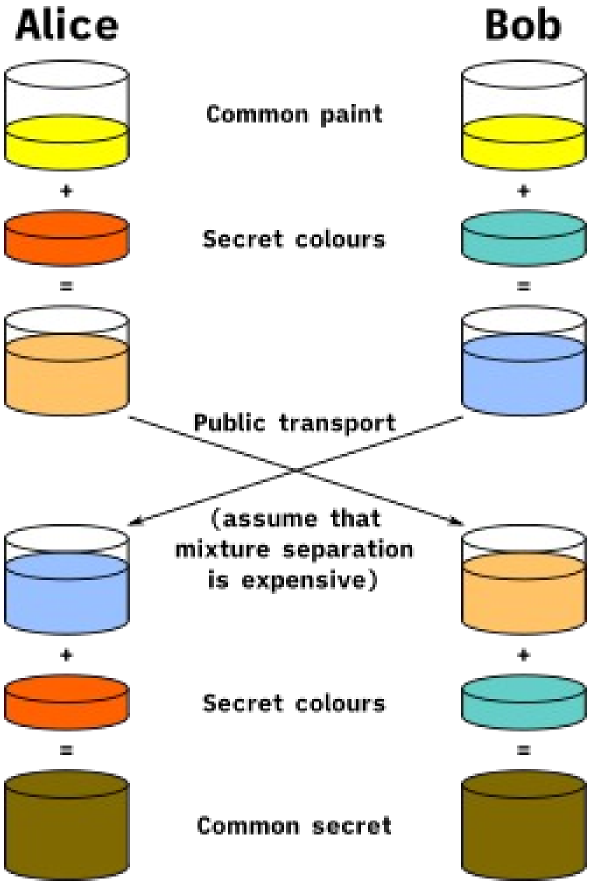


Key exchange

- The Diffie Hellman solution
  - Based on the **exponentiation** (modulus a prime number) of another prime number to a **secret value** on both communicating sides
  - The trick is associativity in this field :

    $(g^a)^b$ [mod p] = $(g^b)^a$ [mod p]
  - Determining a or b knowing g^a, g^b, g and p is **very difficult**

    ⇒ discrete logarithm problem
  - a and b are **trapdoors** to g^{ab}



@Original schema: A.J. Han Vinck, University of Duisburg-Essen

Key derivation

- The shared secret key is usually **not** used directly as a ciphering key, as each negotiation would yield the **same** key (provided that the public/private key pair would remain, which is the generally the case)
  - That would lead to easy replay attacks

- A new key is derived from the shared secret key
  - Using variable information

Key cryptoperiod

- The fact that a key (symmetric cryptography) or key pair (public-key cryptography) is used for **some time** leads to the concept of cryptoperiod
  - The time during which a key remains **valid**

- Cryptoperiods may vary widely
  - From a few minutes to a few years
  - It depends on the context in which the key is created and used

Ciphering suite

- Let us take the example of a file transfer protocol : SFTP

- Prior to file exchange, a negotiation phase takes place between the SFTP client and server
  - This negotiation phase aims at **answering** to some fundamental questions that **control the security level** of the file transfer
  - The result of this negotiation is the **choice** of algorithms and cryptographic parameters

    ⇒ this set of information is called a **ciphering suite**

Ciphering suite

- The questions that need to be answered are :
  - How to **create** a good secret key ? ⇒ Key generation process
  - How to **share** a secret ? ⇒ Key exchange algorithm
  - How to ensure **confidentiality** ? ⇒ Encryption algorithm
  - How to ensure **integrity** ? ⇒ Hash algorithm
  - How to ensure **authenticity** of **peers** ? ⇒ PKC algorithm
  - How to ensure **authenticity** of **file data** ? ⇒ MAC algorithm

Examples of ciphering suites (per function)

- Key exchange algorithms
  - diffie-hellman-group-exchange-sha256 Diffie Hellman, SHA 256
  - ecdh-sha2-nistp256 Ellip. curve DH with NIST P-235 and SHA-256
- Encryption ciphers
  - aes256-ctr AES in CTR mode with 256-bit key
  - twofish256-cbc Twofish in CBC mode with 256-bit key
- MAC algorithms
  - hmac-sha2-256 HMAC with SHA-256
  - Hmac-sha1 HMAC wit SHA-1

Examples of ciphering suites (complete)

- DHE-RSA-AES128-GCM-SHA256
  - Diffie Hellman with ephemeral asymmetric keys for **key exchange**, RSA for **authenticity**, AES with 128-bit keys and GCM as operating mode for **confidentiality** with SHA-256 for **hash** generation (GCM includes MAC)

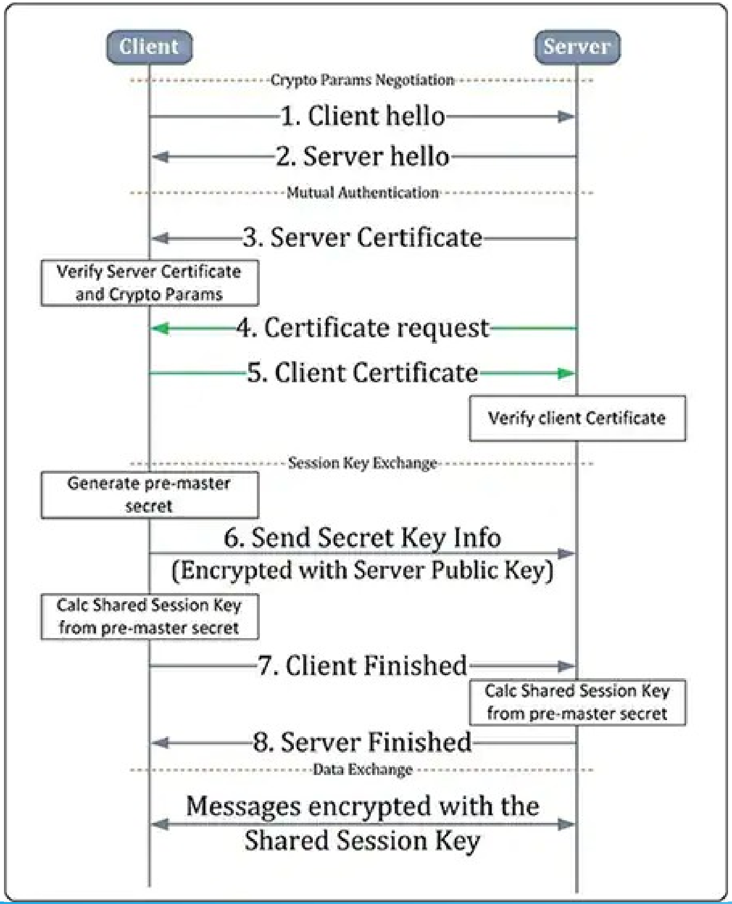
- ECDHE-ECDSA-AES256-GCM-SHA384
  - Diffie Hellman with ephemeral asymmetric keys on elliptic curve for **key exchange**, DSA on elliptic curve for **authenticity**, AES with 256-nit keys and GCM as operating mode for **confidentiality** with SHA-384 for hash generation

Hello protocol

- The choice of a ciphering suite is negotiated at communication startup between a client and a server : that's the hello protocol

- The client as well as the server may place **constraints** on which ciphering suites they would accept
  - This is a major parameter to secure a system

Hello protocol



Source : Texas Instruments

Authenticity using symmetric cryptography

- When transferring files using SFTP, the file emitter is authenticated using symmetric authentication
- An **authentication tag**, called a **MAC** (Message Authentication Code) is generated for each exchange
- This kind of authentication is based on a **shared secret** (the secret key that has been negotiated using public-key cryptography)
  - This is authenticity based on **proof of knowledge** (the receiver checks that the issuer **knows** this secret
  - Different from authenticity based on **proof of identity**

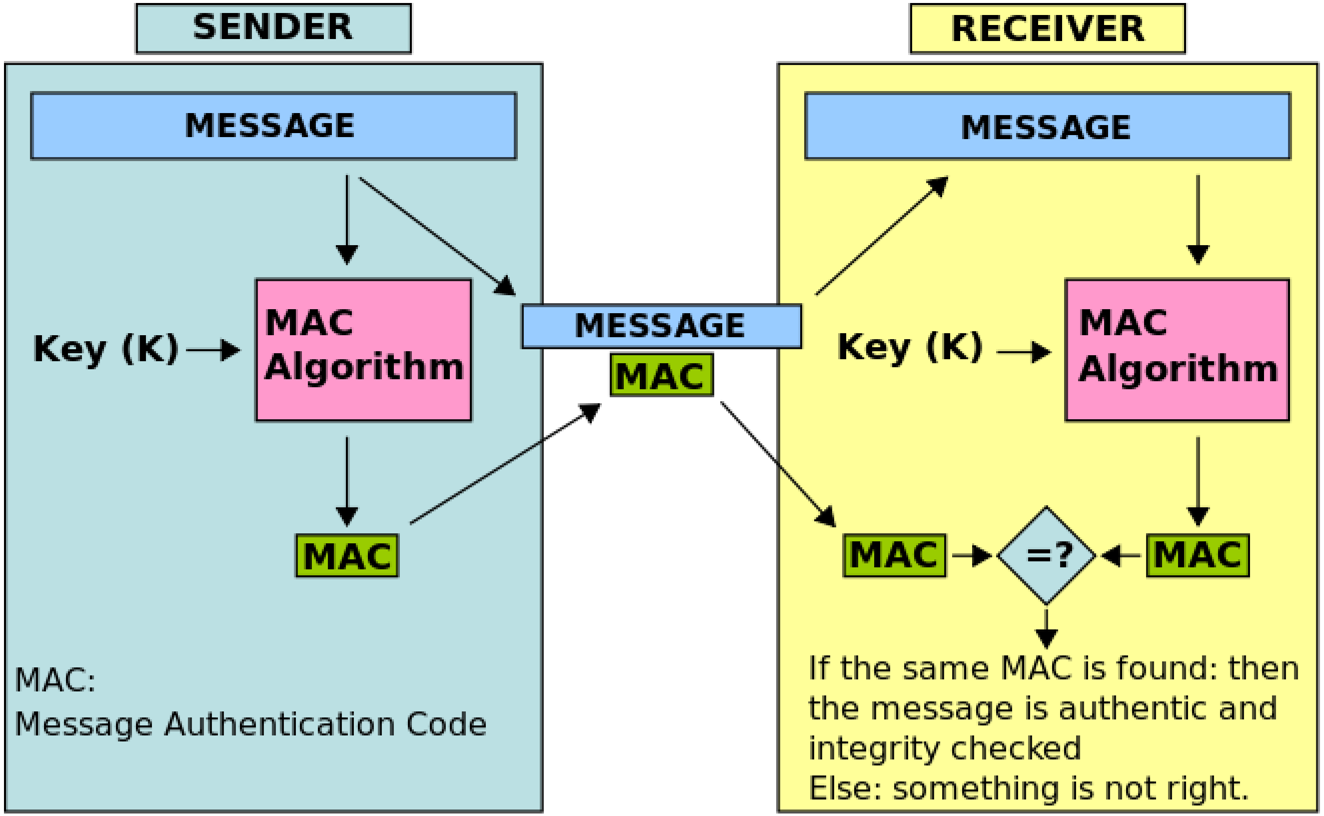
Authenticity using symmetric cryptography

- Consequently, does an authentication tag provides the non-repudiation property ?

MAC computation

- Several ways to compute a MAC have been defined
  - CMAC (Cipher Block Chain Message Authentication Code)
  - HMAC (Hash-based Message Authentication Code)
  - Universal hashing-based MAC
  - …
- Computing a MAC involves :
  - The secret that peers share : a **symmetric** key
  - A **hash** function or **block cipher**, applied on the whole **data** ⇒ integrity comes with it

MAC computation and verification



@Twisp, based on diagram by w:User:Smilerpt - self-made,This W3C-unspecified vector image was created with Inkscape., Public Domain, https://commons.wikimedia.org/w/index.php?curid=3410890

File producer authenticity

- When transferring files using SFTP, the file **sender** (SFTP server or client) is authenticated
  - But the file **producer** may **not be** the file sender
  - So… what could be done to ensure the **file producer authenticity** ?

Hybrid cryptography

- A solution is to use asymmetric cryptography to sign the file
  - The private key of the file **producer** is used to sign the file
    - A hash of the file is produced the signed with the private key of the file producer
  - This digital signature is **attached** to the file
  - Both information are transferred using SFTP
- This solution associates
  - Asymmetric cryptography (for file signature)
  - Symmetric cryptography (for channel ciphering and channel authenticity)

⇒ it is a kind of hybrid cryptography

Hybrid cryptography