Cryptography - Principles –Cryptographie et Sécurité des Communications–

Lionel Morel

Telecommunications - INSA Lyon

Fall-Winter 2021-22

Context

Previously

- Cesear Cipher
- One-Time Pads
- Enigma
- **Cryptology = Cryptography + Cryptanalysis**

Today's objectives

- Encryption / Decryption (Confidentiality)
- Verification (Integrity)
- Signature (Authenticity)

Kerchoffs Principle (in “La Cryptographie Militaire” 1883)

1° Le système doit être matériellement, sinon mathématiquement, indéchiffrable ;

2° Il faut qu’il n’exige pas le secret, et qu’il puisse sans inconvénient tomber entre les mains de l’ennemi ;

3° La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;

4° Il faut qu’il soit applicable à la correspondance télégraphique ;

5° Il faut qu’il soit portatif, et que son maniement ou son fonctionnement n’exige pas le concours de plusieurs personnes ;

6° Enfin, il est nécessaire, vu les circonstances qui en commandent l’application, que le système soit d’un usage facile, ne demandant ni tension d’esprit, ni la connaissance d’une longue série de règles à observer.

- The adversary knows the system [Shannon]
- $\neq$ Security by Obscurity
- Largely accepted in cryptography
- Can be more widely applied to InfoSec (Information System Security) in general.

Confusion and Diffusion (Shannon, 1949)

Confusion

- Each bit in the ciphertext should **depend on several parts of the key**
- Usually implemented using **Substitutions**, aka S-Boxes

Diffusion

- Encryption/decryptions should imply an **avalanche effet.** Formally (in the original Shannon description): changing a single bit in the plaintext changes half of the bits in the cipher-text (eg at the block granularity)
- Usually implemented using **Permutations** (P-Boxes)

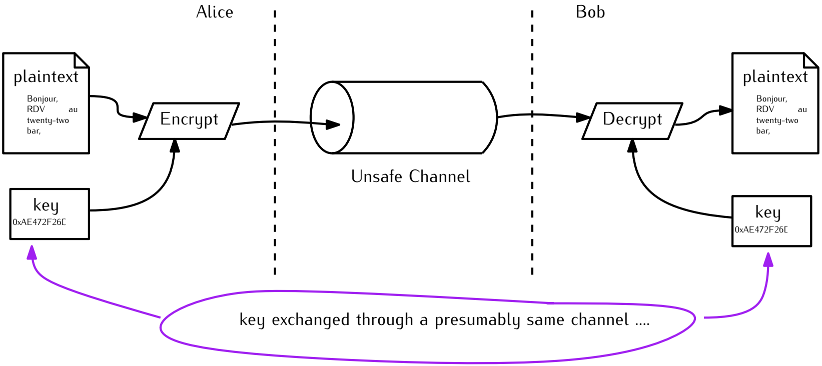
Precautions

- Use recognized libraries (eg OpenSSL), not your own implementation
- Prefer open-source implementations (easier to identify bugs and backdoors)[1]
- In this class, a lot of simplified versions (same on wikipedia)

[1] https://www.theguardian.com/world/2013/sep/05/nsa-how-to-remain-secure-surveillance

Symmetric Cryptography

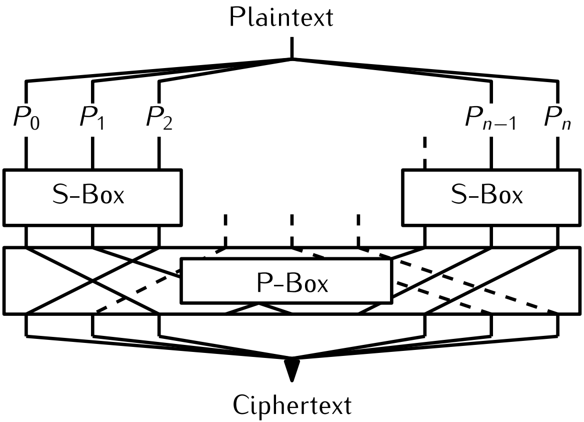
Symmetric Cryptography - Principles



- Encryption, Decryption, Signature and Verification use the same key
- Used implementations are quite efficient.
- A key for each pair of communicating entities

⇒ Rapid explosion in the number of keys

Symmetric ciphers - Basic Principles



Built as a network of substitution/permutation functions:

- Substitution: replace *n* bits by a pre-determined (but moving) table. Must be one-to-one (to allow reversibility of encryption function)
- Permutation: exchange bits

Symmetric ciphers - Baisc Principles

Block cipher

- Treat input as fixed-size blocks (between 64 and 128 bits)
- (+) More secure
- (−) Requires padding

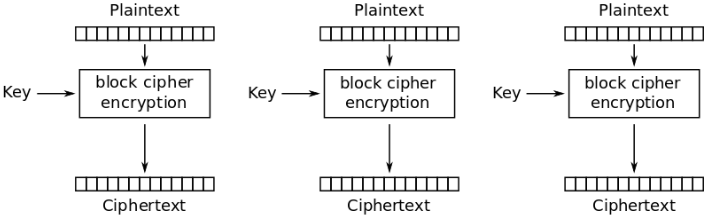
Stream cipher

- Treat input one byte at a time
- The encryption of one byte depends on the current state of the cipher (hence of its history of encryption),
- (+) fast HW implementation
- (−) Security less guaranteed

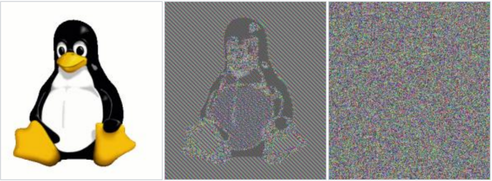
Symmetric ciphers - Operation Modes

Electronic Code Book:

- Message is divided into blocks and each block is encrypted/decrypted separately



- Lacks diffusion

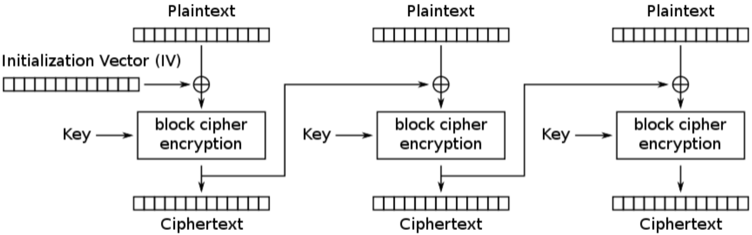


Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

Symmetric ciphers - Operation Modes

Cipher Block Chaining

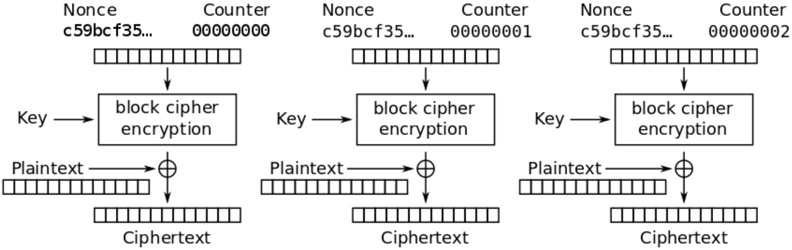
- Initialization Vector to make all cipher message unique



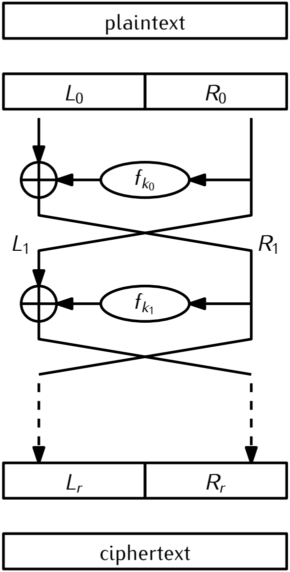
- encryption cannot be parallelized

Symmetric ciphers - Operation Modes

CounTeR



Feistel

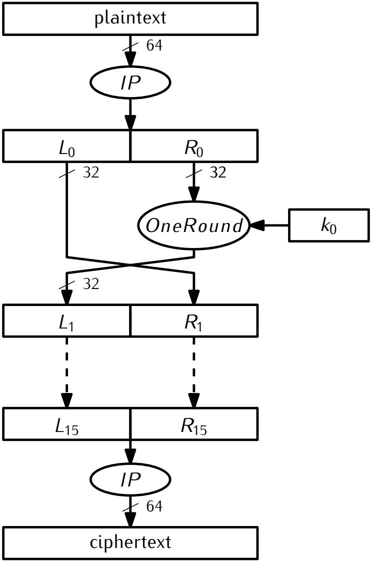


- block cipher
- r rounds
- key k is spilt into r subkeys: $(k_0, \ldots, k_{r-1})$
- plaintext = (L_0, R_0)
- $(L_{i+1}, R_{i+1}) = (R_i, L_i \oplus f_{k_i}(R_i))$
- General structure used in all other ciphers

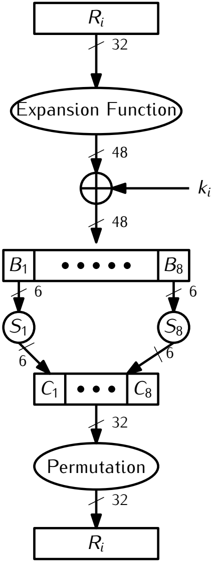
Symmetric Cryptography - DES

- Expands Feistel algorithm, by introducing:
  - More permutations
  - Substitution Boxes (S-Boxes)
- Designed (and initially published) in 1975.
- Block-cipher

DES - General Algorithm



DES - One Round



48-bits subkey obtained through a key-schedule algorithm using the original 64-bits key as input

DES - Weaknesses and Attacks

- Most practical attack to date: still brute force (ie trying out all possible key in turn).
- Key size in DES was reduced from 128 bits to 56 bits (after discussions with ... NSA) “to fit on a single chip”
- Practically cracked (brute-forced) in 1997
- Attacks faster than brute-force:
  - Differential cryptanalysis: requires 2^{47} chosen plaintexts
  - Linear cryptanalysis: requires 2^{43} chosen plaintexts

Example: Differential Cryptanalysis

Principle:

- Choose two plaintexts x and y such that: $y = x \oplus \Delta_x$
- Compute the corresponding cyphertexts and for each S-Box S:
  - $S(x)$
  - $S(y) = S(x \oplus \Delta_x)$
- Compute difference on S-Boxes:
  - $\Delta_y = S(x \oplus \Delta_x) \oplus S(x)$
- Repeat this for many plaintexts and several key hypothesis $k_i i \in \{0, n\}$
- key k_j that minimizes Δ is deemed "most probable".

Limits:

- In practice requires 2^{47} well-chosen plaintext (so that Δ_x is "not too big")
- Limits: choose the "right" plaintexts

3DES

- Standardized in 1998 to compensate for the weaknesses of DES
- DES has a 56-bits key
- 3DES **chains 3 DES together**:
  - Encrypt = Encrypt(k_1)$\rightarrow$Decrypt(k_2)$\rightarrow$Encrypt(k_1)
  - Decrypt = Decrypt(k_1)$\rightarrow$Encrypt(k_2)$\rightarrow$Decrypt(k_2)
  - Key: 112 bits ($k_1|k_2$)
- Developped in parallel of AES (waiting for AES to be defined)

AES - Advanced Encryption Standard

- Supersedes DES
- Standardized in 2001
- NIST-organized competition with 5 finalists:
  - IBM proposed MARS
  - RSA proposed RC6
  - Serpent by Anderson, Bihman, Knudsen
  - Twofish by Bruce Schneier et al
  - Rijndael, by Daemen and Rijmen
- Rijndael's was elected by community after a thourough international comparative effort (including NSA, companies, academics), based on security, performance (speed, memory usage).
- NB: no-patent allowed (imposed by the NIST)

Algorithm

```
void AES_Run_secure(void){
  int i;
  addRoundKey();
  for(i = 0; i < 9; i++){
      subBytes();
      shiftRows();
      mixColumns();
      computeKey(rcon[i]);
      addRoundKey();
  }
  subBytes();
  shiftRows();
  computeKey(rcon[i]);
  addRoundKey();
}
```

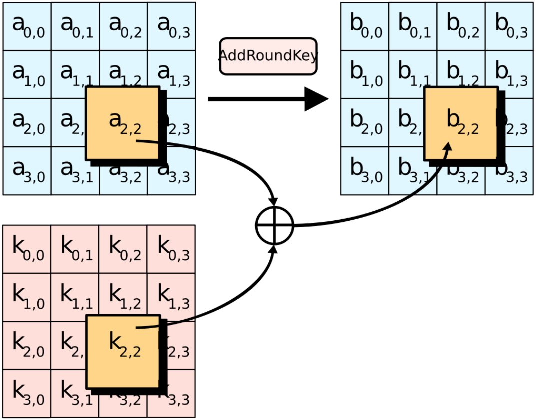
AES explained[2]

- **KeyExpansion** — round keys are derived from the cipher key using the AES key schedule. AES requires a separate 128-bit round key block for each round plus one more.

[2] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

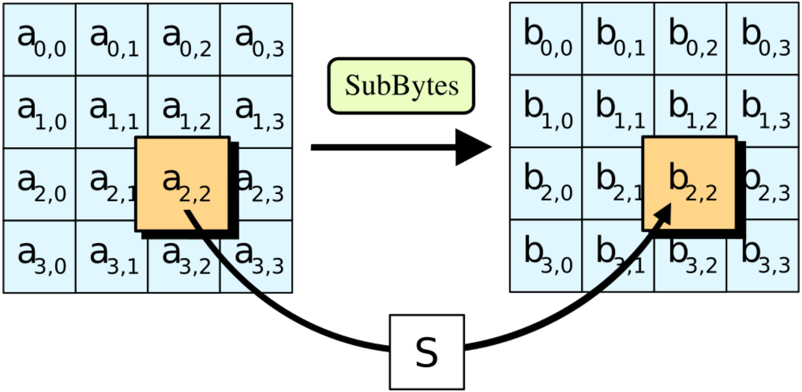
AES (cont'd)

- **Initial round key addition**:
  - **AddRoundKey** – each byte of the state is combined with a byte of the round key using bitwise xor.



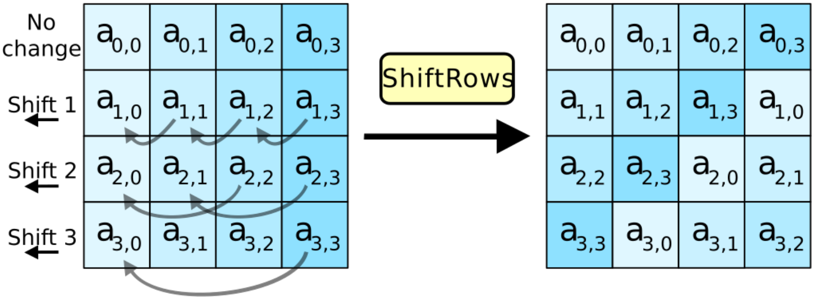
AES - SubBytes

SubBytes = a non-linear substitution step where each byte is replaced with another according to a lookup table.



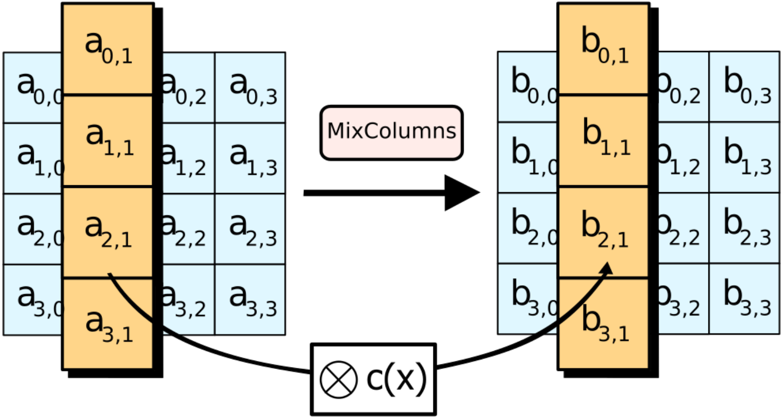
ShiftRows

ShiftRows = a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.



MixColumns

MixColumns = a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
AddRoundKey



AES - Weaknesses and Attacks

- Side-channel attacks are practical:
  - 6-7 blocks plaintexts needed
  - $\Rightarrow$ requires HW protections
- Related-key attacks exists:
  - $2^{99.5}$ time and space complexity
  - btw: age of universe ˜ 2^{70}
  - Anyway totally impractical (because keys are well-chosen to be independant in crypto-systems)

Symmetric Cryptogaphy - Conclusions

- Overall very effecient (linear in the size of data to encrypt)
- Arithmetic/Logical operations are simple: xor.
- Requires a shared key!
- Solutions to this:
  - Avoid the need for a common key
  - Find a way to securely share a common key

Key Sharing Problem

- Symmetric cryptography uses same key to encrypt and decrypt
- Problem: how to share this key
- Hypothesis: there is no secure channel to exchange the key

Diffie-Hellman Key Exchange

Alice

let's agree on :
p = 23 (prime)
and g = 5
(primitive root modulo 23)

Bob

Alice chooses a secret key P_A

Bob chooses a secret key P_B

$g^{P_A} \bmod p$

$g^{P_B} \bmod p$

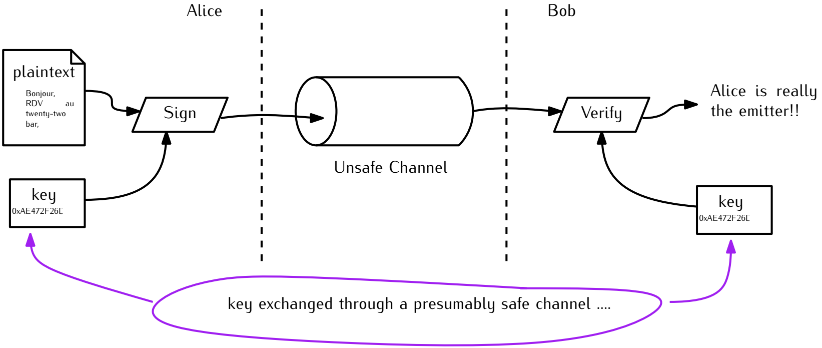
Alice computes
$(g^{P_B} \bmod p)^{P_A} \bmod p$
$= g^{P_B P_A} \bmod p$

Bob computes
$(g^{P_A} \bmod p)^{P_B} \bmod p$
$= g^{P_A P_B} \bmod p$
$= g^{P_B P_A} \bmod p$

Alice and Bob now share a common key $g^{P_A P_B} \bmod p$

Hash

Cryptographic Hash



- eg Hash-based Message Authentication Code
- Only sender and recipient can sign/verify the message

Cryptographic Hash - Principle

- Compute a “footprint”
- The message can be of any size, the footprint is of fixed size
- Pseudo-unique identification of message
- Used for:
  - Integrity checks
  - Cryptographic signature
  - PRNG
  - Hashed password storage

Cryptographic Hash - Good Properties

- **Pre-image resistance**: no one can reverse the hash function (to find input from output)
- **Second pre-image resistance**: unicity of hash. Given an input and the corresponding hash, one cannot find another input with the same hash.
- **Collision-resistance**: no-one can produce two different inputs with the same hash
- **Randomness**

Cryptographic Hash - today’ state of affairs

Existing (and used) implementations

- MD5: please don’t use anymore: “cryptographically broken and unsuitable for further use”
- SHA-1: not recommanded anymore (since 2017)
- SHA-2: still not planned for removal
- SHA-3: standardized in 2015

Current situation:

- Hash functions are critical in crypto!
- SHA-2 is still safe but is conceptually close to SHA-1 and might share some weaknesses with it
- SHA-3 considered “as safe” but built completely differently

Asymmetric Cryptography

(general) Asymmetric Cryptography

- Each participant u has a pair of keys Pub_u and $Priv_u$.
- u sends Pub_u to v
- v sends Pub_v to u
- u can encrypt its messages to v using a combination of Pub_v and $Priv_u$
- v can decrypt messages from u using a combination of Pub_u and $Priv_v$

Note:

- Relies on “hard mathematical problems”:
  - Discrete logarithm
  - Factorization of large numbers
- Usually slow (exponentiation)

RSA

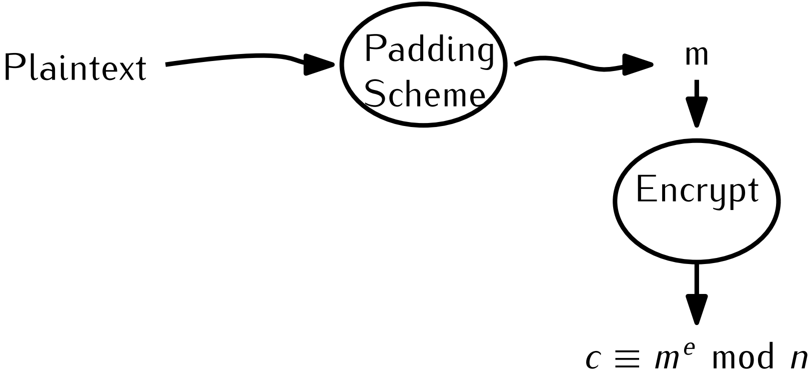
- Invented in 1977 by Rivest, Shamir and Adleman
- MIT Patent in 1983, expired in 2000
- Security based on the difficulty of factorizing large integers

RSA - Key generation

- Choose p and q, two prime numbers: random, kept secret
- Compute $n = pq$
- Compute $\lambda(n)$,
  - $\lambda(n) = lcm(\lambda(p), \lambda(q))$
  - $= lcm(p-1, q-1)$
  - $= \frac{pq}{gcd(p,q)}$ ... (gcd obtained with Euclid. algorithm)
- Choose e s.t.:
  - $1 < e < \lambda(n)$
  - $gdc(e, \lambda(n)) = 1$
- Compute $d = e^{-1} \bmod \lambda(n)$
  - d is the “private key exponent”

$Pub = (e, n)$ $\qquad$ $Priv = (d, n)$

RSA - Encryption



RSA - Decryption

c $\rightsquigarrow$ Decrypt $\rightsquigarrow$ $m \equiv c^d \bmod n$

RSA - Example

1. $p = 61$ and $q = 53$
2. n = pq = 3233
3. $\lambda(n) = lcm(p - 1, q - 1)$
4. $= \lambda(3233) = lcm(60, 52) = 780$
5. Choose $1 < e < 780$ (coprime to 780), eg $e = 17$
6. $d = e^{-1} \bmod \lambda(n)$
7. $= 413$ (as $1 = 17 * 413 \bmod 780$)
8. $\boxed{\text{Public key = } (e = 17, n = 3233)}$
9. $\boxed{\text{Private key = } (d = 413, m = 3233)}$
10. $c(m) = m^{17} \bmod 3233$
11. $m(c) = c^{413} mod 3233$
12. $m = 65 \rightarrow c = 65^{17} \bmod 3233 = 2790$
13. $2790 \rightarrow m = 2790^{413} \bmod 3233 = 65$

RSA - Properties & Limitations

- Finding d requires factorizing n: proven difficult (for p and q large)
- Implementation is tricky : good PRNG, acceptable e
- Relies on exponentiation which is **expensive** :

$$x^y = \underbrace{x * x * \dots * x}_{y \text{ times}}$$

  - Requires a (fast) multiplier
  - Remember y is big (if you want security)
  - still way more expensive than xor !

Key management

The key distribution problem

- To encrypt a message or check a signature, Alice needs Bob's public key
- Otherwise, it may encrypt a message thinking only Bob will read it, but maybe Charlie can read it instead
- How can she get this public key in a secure manner?
- Hard problem, no perfect solution

Note: Using the right key guarantees Bob **is** Bob, but not that Bob is honnest ...

Existing solutions

- Hierarchical certification authorities
- Web of trust (eg PGP)
- Direct exchange of keys

Hybrid Cryptography

Comparing Symmetric / Asymmetric cryptogaphy

Symmetric cryptography

- 1 key per pair of participants (n^2 keys)
- Fast: simple operations, easy to implement in HW

Asymmetric cryptography

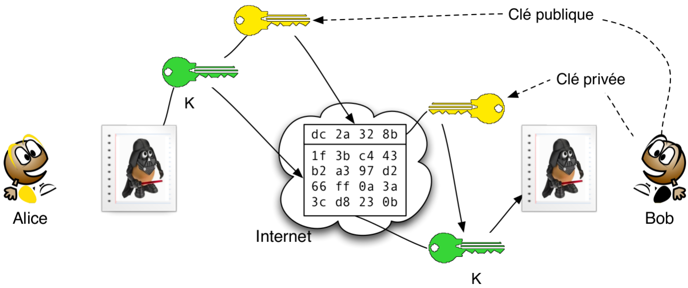
- 1 pair of key per participant ($2n$ keys)
- Slow: complex operations, eg exponentiations

Hybridcryptography

- Alice encrypts a symmetric key with the public key of Bob
- Alice encrypts the message with the symmetric key

$\Rightarrow$ Best of both worlds

The "best of both worlds"



- Alice encrypts message with Symm key k
- Alice encrypts k with Bob's public key
- Bob decrypts k with his private key
- Bob decrypts message with k

Next time

- Cryptographic protocols
- Public Key Authorities
- PGP