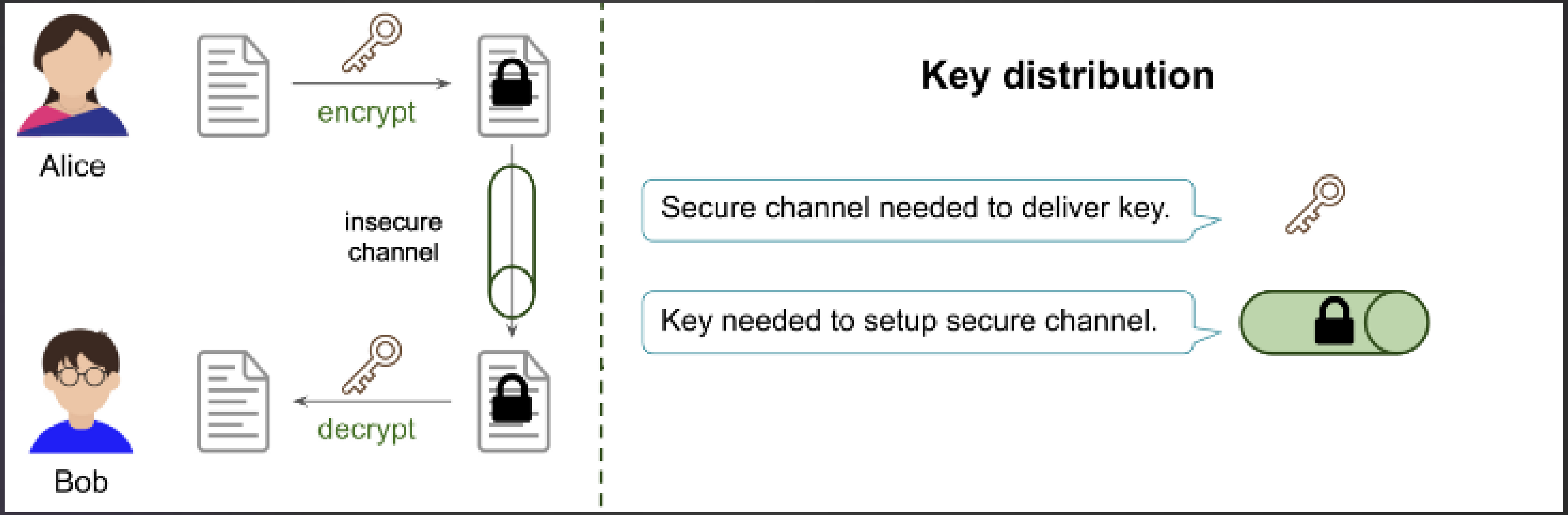
SYMMETRIC ENCRYPTION

Symmetric encryption

- **Single shared key** for encryption and decryption process.
- If **third party gains access** to this key you will need to throw that key away and you will **distribute a new key** both the sender and the recipient.
- Use a **shared key** algorithm (also called shared secret) only for people who need to either encrypt or decrypt this information.
- It’s **difficult to distribute** these keys to everyone who might need it.
- **Very fast to encrypt and decrypt data** (compared to asymmetric encryption).
- You can **combine symmetric and asymmetric encryption** (example by encrypting a symmetric key using asymmetric encryption).

Symmetric encryption

- Sender and receiver share the same key for both encryption and decryption.


Alice
encrypt
insecure channel
Bob
decrypt
Key distribution
Secure channel needed to deliver key.
Key needed to setup secure channel.

SYMMETRIC ALGORITHMS

RC2, RC4, RC5 and RC6 Algorithms

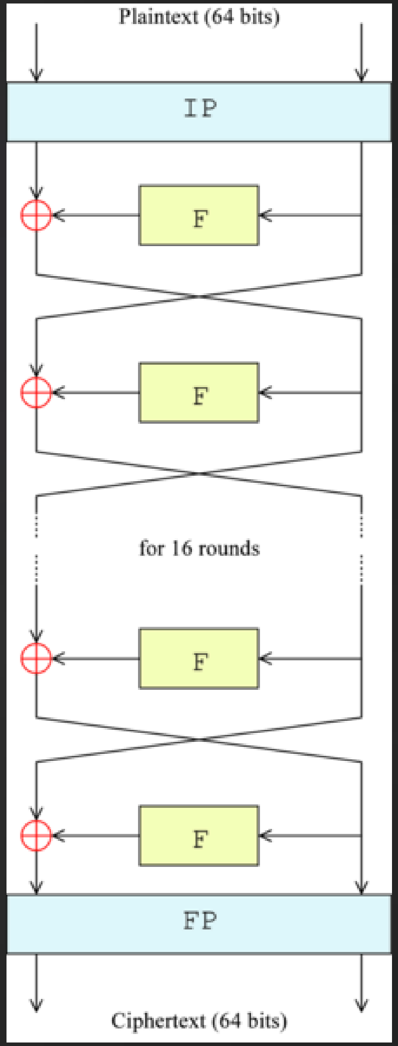
- RCx for Rivest Cipher.
- All of these are vulnerable (**RC2** is vulnerable to a related-key attack, **RC4** have led to very insecure protocols such as WEP, etc.).

Algorithms	**Category**	**Block sizes**	**Key sizes**	**Rounds**
RC2	Block cipher	64 bits	8–1024 bits	16 of type MIXING, 2 of type MASHING
RC4	Stream cipher	NA	40-2048 bits	1-255
RC5	Block cipher	32, 64 or 128 bits	0-2040 bits	1-255
RC6	Block cipher	128 bits	128, 192, or 256 bits	20

Data Encryprion Standard (DES)

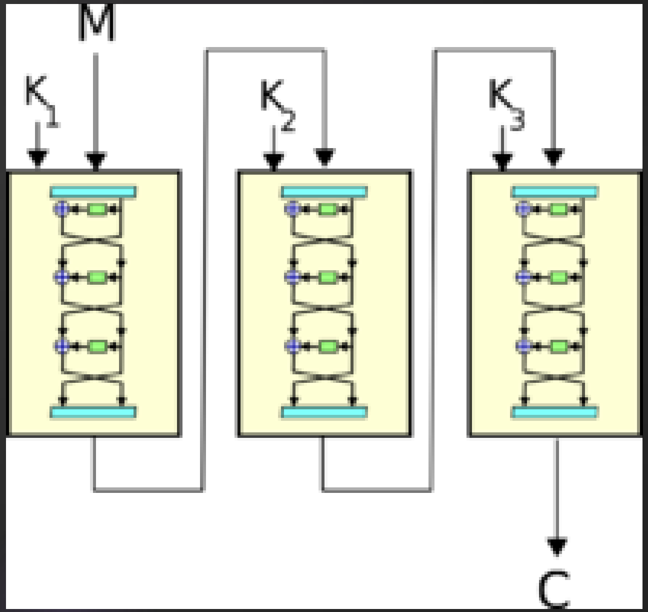
- **DES** is designed to encrypt and decrypt blocks of data consisting of 64 bits under control of a 56 bit key.
  - Block sizes : 64 bits
  - Key sizes : 56 bits (+8 parity bits)
  - Rounds : 16 (Initial Permutation to Final Permutation with function F)

- **DES** has been considered **insecure** right from the start because short block size of 64 bits makes DES vulnerable to block collision attacks if it is used to encrypt large amounts of data with the same key.



Triple DES (3DES)

- **3DES** applies the DES cipher algorithm three times to each data block.
  - Block sizes : 64 bits
  - Key sizes : 168, 112 or 56 bits
  - Rounds : 3x16
- **3DES** is also considered **insecure** for same reason as DES. Feasibility of brute-force / collision attacks.
- **Vulnerability**
  Sweet32: Birthday attacks on 64-bit block ciphers



Blowfish

- **Blowfish** it is about 5 times faster than 3DES.
  - Block sizes : 64 bits
  - Key sizes : 32 at 448 bits
  - Rounds : 16

- **Blowfish** is currently considered **insecure**.

- **Blowfish's** use of a 64-bit block size (as opposed to e.g. AES's 128-bit block size) makes it **vulnerable to birthday attacks**. In 2016, the SWEET32 attack demonstrated how to leverage birthday attacks to perform plaintext recovery / decrypting ciphertext against ciphers with a 64-bit block size.

Twofish

- **Twofish** is the successor of Blowfish
  - Block sizes : 128 bits
  - Key sizes : 128, 192 or 256 bits
  - Rounds : 16
- **Twofish** is currently considered **secure** (finalist of the AES competition).
- **Twofish** is slightly slower than AES.
- Attractive alternative to the current AES if it becomes vulnerable.

Advanced Encryption Standard (AES)

- **AES** is a standardization process initiated by NIST to ask cryptologists to design a new block cipher algorithm for the US government.
- **Winner of the competition is Rijndael algorithm.**

- **AES** supersedes DES / 3DES. It use a method called Square.
  - Block sizes : 126 bits
  - Key sizes : 128, 192 or 256 bits
  - Rounds : 10, 12 or 14 (depending on key size)

- **Vulnerability** for all key sizes
  Biclique attack (based on MITM).

ASYMMETRIC ENCRYPTION

Asymmetric encryption

- Asymmetric encryption **involves both** a public key and a private key.
- **Public key** is publicly available and **Private key** must be kept secret.
- It'**s not the same key** that is used for encryption and decryption.
- They are intimately linked by a complex mathematical function (**one-way function**) is a function that is easy to compute on every input, but hard to invert. But a person with a piece of information (**private key**) can easily decode the message.
- Asymmetric algorithms have two modes of operation (**encryption mode** and **signature mode**).

Asymmetric encryption

- Example :

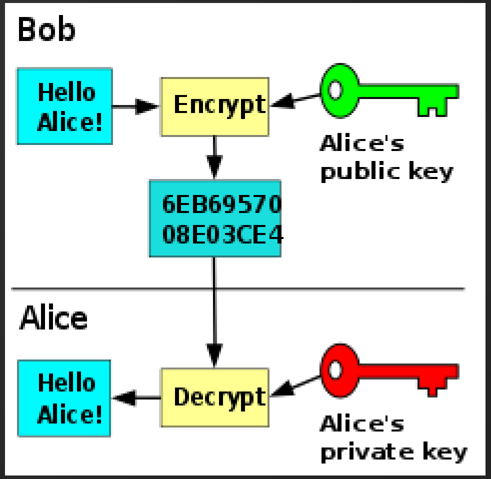
Suppose Alice wants to receive a secret message from Bob on a channel that could be listened to by a passive attacker Eve.

1. Alice transmits to Bob a one-way function for which she alone knows the private key.
2. Bob uses the function sent by Alice to encrypt his secret message.
3. Alice receives the encrypted message then decodes it using the private key.
4. If Eve also receives the message while it is circulating on the public channel, she cannot decode it, because she has no knowledge of the private key.

Asymmetric encryption scheme

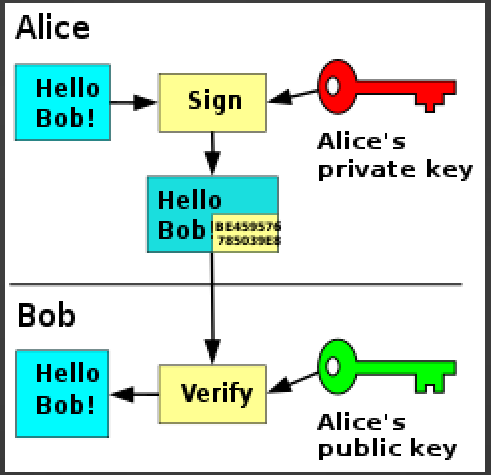
- The sender encrypts with the recipient's public key, the recipient decrypts with her private key.

- In an asymmetric key encryption scheme, anyone can encrypt messages using the public key, but only the holder of the paired private key can decrypt.
- Security depends on the secrecy of the private key.



Digital Signatures (not encrypted)

- The sender signs with her private key, the recipient verifies the signature with the sender's public key.



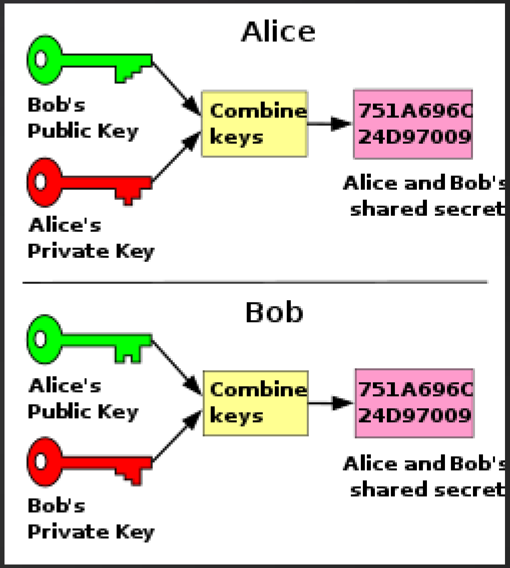
ASYMMETRIC ALGORITHMS

Diffie–Hellman (DH)

- **DH is a key exchange method** for securely exchanging cryptographic keys over a public channel.

- Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical means, such as paper key lists transported by a trusted courier.

- This method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.



Alice

Bob's Public Key

Alice's Private Key

Combine keys

751A696C 24D97009

Alice and Bob's shared secret

Bob

Alice's Public Key

Bob's Private Key

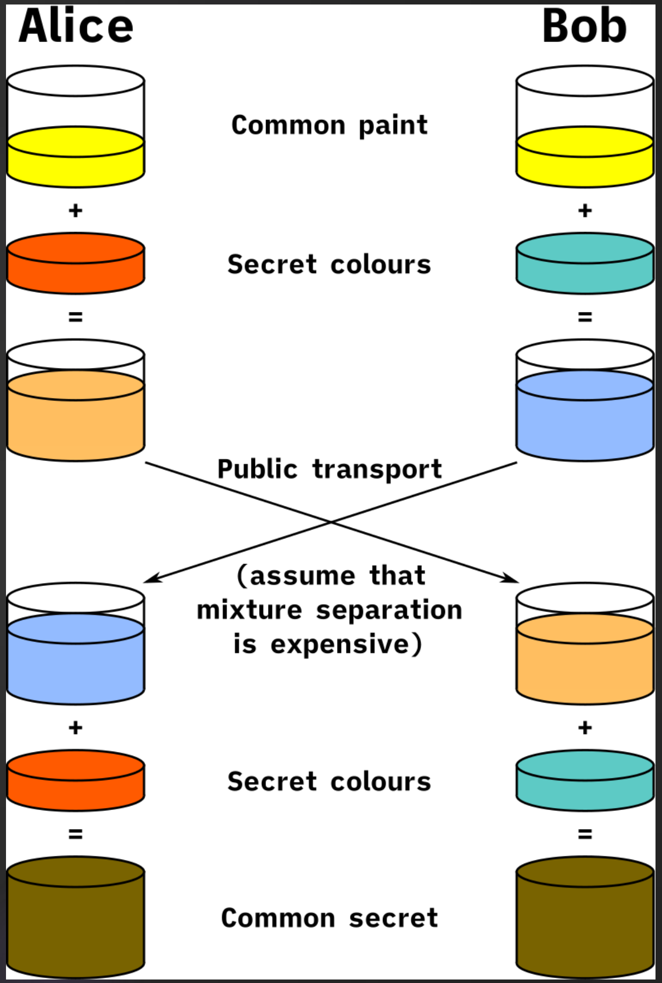
Combine keys

751A696C 24D97009

Alice and Bob's shared secret

Diffie–Hellman (DH)

- Example with colors rather than numbers:
  - Alice and Bob, publicly agree on an arbitrary starting color that doesn't need to be kept secret.
  - Each person selects a secret color that they keep to themselves.
  - Each mix their own secret color with their mutually shared color.
  - Exchange publicly the two mixed colors.
  - Each of them mixes the color they received from the partner with their own private color.
  - If a third party listened to the exchange, it would only know the common color and the first mixed colors, but it would be difficult for this party to determine the final secret.
- To determine the color by calculation or with brute force is very time consuming.



Diffie–Hellman (DH)

- Function used:
  - DH turned their research towards non-reversible functions like functions of the form
    y = f (x) where when we know x, it is very easy to calculate y, but when we know y, it is impossible to recalculate x.
    - For example, the function y = x * x is a reversible function. Knowing x, it is easy to calculate y, and knowing y, it is easy to calculate x. The reversible function of y = x * x is x = $\sqrt{y}$.
  - DH used modulo function, because they are not reversible.
    - For example y = x (modulo 7) is not a reversible function. Indeed, for x = 11, we get y = 4 but for y = 4, we get x = 11 or x = 18 or even x = 25 and an infinity of other solutions.
  - The function used is of the type **y = Y ^ x (modulo P)** which allows this secure exchange.

Diffie–Hellman (DH)

- Function used with numbers. Alice wants to communicate with Bob and Eve can listen to the exchanges.
  - Alice says she will use the function **y = 11 ^ x modulo 13**
    - Bob receives the message and Eve heard / read this information.
  - Alice chooses a secret number **A = 5** and Bob **B = 8**.
    - Eve knows neither A nor B, Alice doesn't know B and Bob doesn't know A.
  - Alice calculates the number **A‘ = 11 ^ A modulo 13**. **A' = 7** and sends this number **A'= 7** to Bob and at the same time Bob does the same with B‘. **B‘=11 ^ B modulo 13 = 9** and send this number to Alice.
    - Eve heard / read these informations **A’=7** and **B’=9**.
  - Alice takes Bob's number and calculates the result is **B' ^ A modulo 13=3** Idem for Bob and Alice's number A'=7 the result is **A' ^ B modulo 13=3**
    - **Eve cannot calculate these numbers without A or B.**

Rivest Shamir Adleman (RSA)

- The algorithm they have developed is also based on a one-way function or rather a function that is very difficult to reverse.
- This difficult to reverse function is the **factorization of a number as a product of prime factors**.
- It is very easy to choose two numbers (48 and 52 for example) and to calculate their product (2496).
- It is long and painful to extract the prime factors of the number 2397 which are 47 and 51.
- The principle of the RSA algorithm is therefore based on the difficulty of factoring a number, especially when this number is very large and when it is the product of two very large prime numbers as well.

Rivest Shamir Adleman (RSA)

- Function used with numbers (lcm mean lowest common multiple):
  - Alice chose two distinct prime numbers, such as p=61 and q=53.
    - She compute **n = pq** = 61*53 = 3233
    - **$\lambda(n) = \mathrm{lcm}(p - 1, q - 1)$** = lcm(60,52) = 780
    - She chose any number 1 < e < 780 that is coprime to 780 (E must not divide 780). Take **e = 17**
    - Alice calculates the modular multiplicative inverse **$d = e \pmod{\lambda(n)}$**.
      So **$d * e = 1 \pmod{\lambda(n)}$ ➔ $d * 17 = 1 \pmod{780}$ ➔ $413 * 17 = 1 \pmod{780}$**
      Because **$413*17 = 7021 \equiv 1 \pmod{780}$**
      You can verify this by doing the following calculation : **7021 mod 780 = 1**
    - Alice calculates the number d given by **e * d = 1 modulo ((p-1) * (q-1))** (77 in our example). This number represents Alice's private key.
    - Public key is (n = 3233, e = 17). For a plaintext message m, the encryption function is : **$c(m) = m^{17} \bmod 3233$**
      Private key is (n = 3233, d = 413). For an encrypted ciphertext c, the decryption function is : **$m(c) = c^{17} \bmod 3233$**
  - Exercice: Convert the word “Hello” to ASCII and replace “m” with the value to have the encrypted message “c” then decrypt it.