# UF1
# Seguretat i criptografia

# Continguts UF1

- Principis i pràctiques de programació segura
- Criptografia

# Principis i pràctiques de programació segura

## Principis bàsics

Aquests principis estan orientats a la seguretat del programari. El seu coneixement i com s'implementen són la base de la programació segura.

- Protecció contra la divulgació
- Protecció contra l'alteració
- Protecció contra la destrucció
- Qui por fer-lo servir?
- Quin drets i privilegis te l'usuari?
- Capacitat de construir evidencia històrica
- Gestió de configuració, de les sessions i dels  errors/excepcions

# Principis i pràctiques de programació segura

## Pràctiques bàsiques

A continuació s'enumeren algunes de les bones pràctiques recomanades per assegurar el desenvolupament de programari segur.

- Comprovar les entrades al costat del client i del servidor
- Encriptar peticions i respostes
- Utilitzar HTTPS com a entrades de domini
- Utilitzar algoritmes d'encriptació i hashing actualitzats
- No guardar dades sensibles a dintre de cookies
- Comprovar la aleatorietat de la sessió
- Establir una política de clau d'accés forta
- Comprovar la pujada i descarrega de fitxers
- Assegurar que les llibreries de tercers són segures
- Oculta la informació del servidor web

# Criptografia

## Criptografia

És l'estudi de formes de convertir informació des de la seva forma original cap a un codi incomprensible

- Algunes aplicacions de la criptografia inclouen caixers automàtics, contrasenyes, comunicació segura i comerç electrònic.
- De la informació original en diem el text pla . Llavors passa per un procés de xifrat que converteix la informació original en un codi il·legible per tothom que no tingui els mitjans per desxifrar-lo i la clau per fer-ho
- Intenta resoldre dos dels problemes de seguretat en el maneig de dades: la privadesa i l'autenticitat
- Tipus de criptografia:
  - **De clau simètrica o privada**
  - **De clau asimètrica o pública**

# Criptografia

## Criptografia de clau simètrica o privada

Es refereix a aquells mètodes de xifratge en què l'emissor i receptor comparteixen la mateixa clau

Actualment els algoritmes de xifratge més utilitzats son els de xifratge per blocs i el mes coneguts i utilitzats són: DES/TripleDES i AES

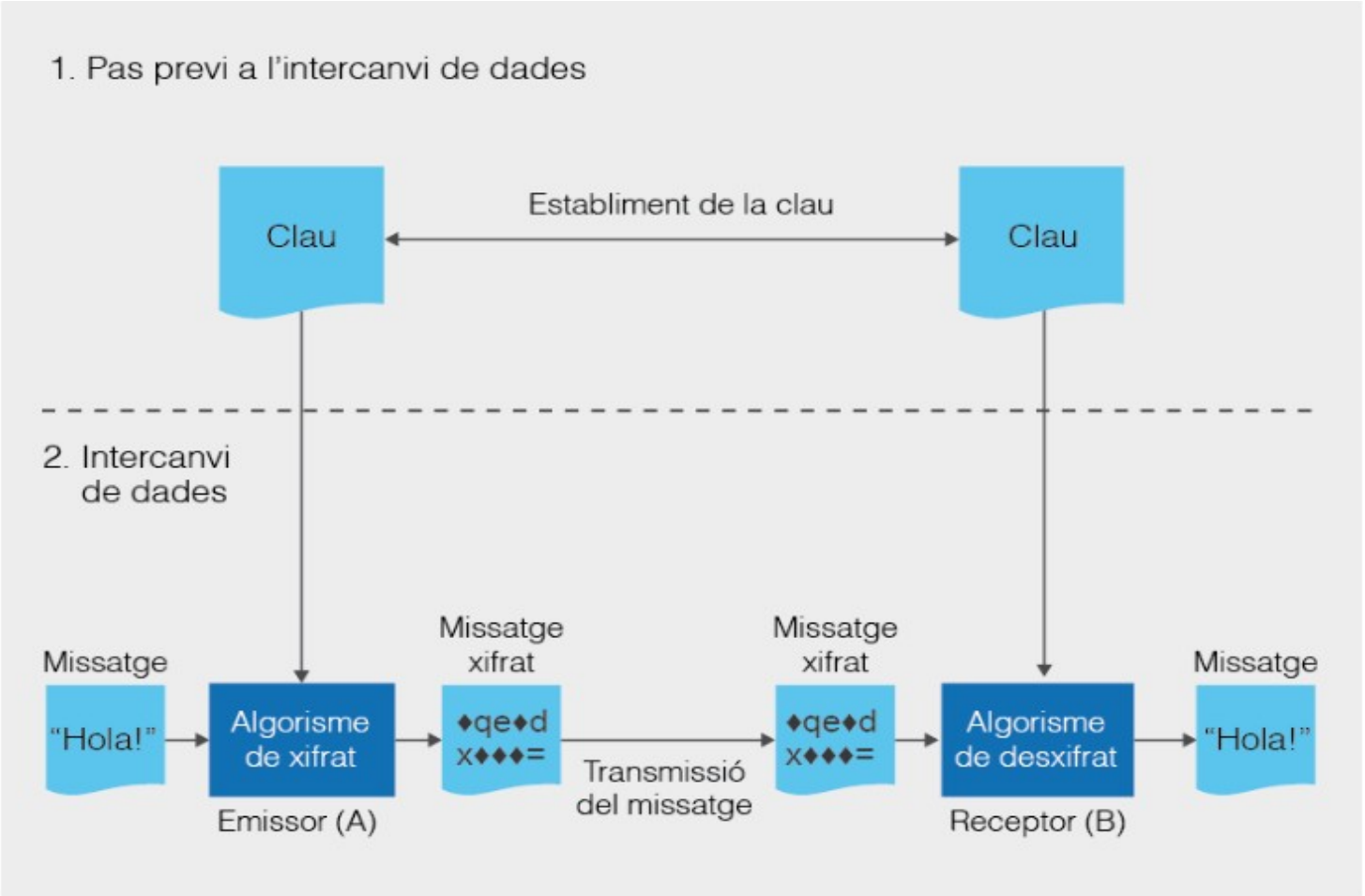
Les principals aplicacions del xifrat de clau simètrica són: encriptació ATM, privadesa dels correus electrònics, datàfons, missatgeria instantània (ex. Wassap)  i accés remot segur

Per poder realitzar un intercanvi de missatges xifrats primer s’ha de decidir quina serà la clau i com es compartirà/proporcionarà la clau a l’altra part implicada

# Criptografia

## Criptografia de clau simètrica o privada

Esquema del procés de xifrat basat en clau simètrica



# Criptografia

## Criptografia de clau simètrica o privada

### Generació de claus simètriques

Una clau no és més que una seqüencia de bytes però per treballar en Java farem servir instancies de la classe *javax.crypto.SecretKey* i per generar-les farem servir la classe *javax.crypto.KeyGenerator*

```
public SecretKey keygenKeyGeneration(int keySize) {
    SecretKey sKey = null;
    if ((keySize == 128)||(keySize == 192)||(keySize == 256)) {
      try {
        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        kgen.init(keySize);
        sKey = kgen.generateKey();

      } catch (NoSuchAlgorithmException ex) {
        System.err.println("Generador no disponible.");
      }
    }
    return sKey;
  }
```

# Criptografia

## Criptografia de clau simètrica o privada

### Claus simètriques basades en contrasenya

Com que una clau criptogràfica és una seqüencia de bytes prou llarga i incomprensible per poder intercanviar-la es poden generar a partir d'una contrasenya. Per poder fer-ho necessitem un algorisme que ens converteixi aquesta contrasenya en la clau

Un algorisme de hash és una transformació criptogràfica que compleix:

- Donada una mateixa entrada, sempre resulti en exactament la mateixa sortida i que dues sortides diferents vinguin sempre d'entrades diferents (bijectiu)
- A partir només d'un resultat, ha de ser pràcticament impossible endevinar quina ha estat l'entrada original que l'ha generat (irreversible)

Els algorismes hash que es fan servir son; SHA-1 que genera una sortida de 160 bits, SHA-256 que genera 256 bits i el més actual el SHA-3

Farem servir un hash que ens generi prou bits per poder utilitzar-los com a clau agafant els que necessitem

# Criptografia

## Criptografia de clau simètrica o privada

### Claus simètriques basades en contrasenya

La classe *MessageDigest* permet executar algorismes de hash sobre unes dades

*SecretKeySpec* permet obtenir la clau AES passant-li com a paràmetre els bytes resultants del hash que es necessiten per crear la clau

```
public SecretKey passwordKeyGeneration(String text, int keySize) {
    SecretKey sKey = null;
    if ((keySize == 128)||(keySize == 192)||(keySize == 256)) {
      try {
        byte[] data = text.getBytes("UTF-8");
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] hash = md.digest(data);
        byte[] key = Arrays.copyOf(hash, keySize/8);
        sKey = new SecretKeySpec(key, "AES");
      } catch (Exception ex) {
        System.err.println("Error generant la clau:" + ex);
      }
  }
    return sKey;
  }
```

# Criptografia

## Criptografia de clau simètrica o privada

### Algorismes de xifrat en bloc

Els algorismes de xifrat en bloc, treballen amb una mida de bloc fitxa

**Funcionament**:

- Les dades a xifrar es divideixen en blocs de la mida adient
- Si el darrer bloc no arriba a la mida s'afegeix un **padding** que són dades generades per l'algorisme
- Es processa cada bloc individualment usant la clau completa
- El resultat final s'obté concatenant tots els resultats parcials en el mateix ordre que s'han anat processant

Poden funcionar principalment en dos modes d'operació:

- **ECB** (Electronic Code Book, Llibre de Codis Electrònic)
- **CBC** (Cyclic Block Chaining, Encadenat de Blocs Cíclic)

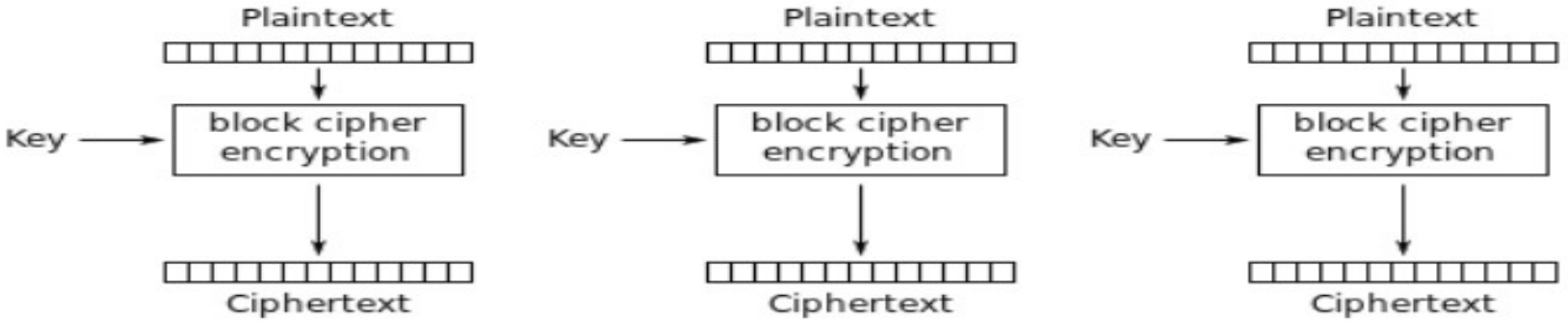
Dels dos algorismes de clau simètrica de xifrat en bloc DES/tripleDES, AES treballarem amb l'AES

# Criptografia

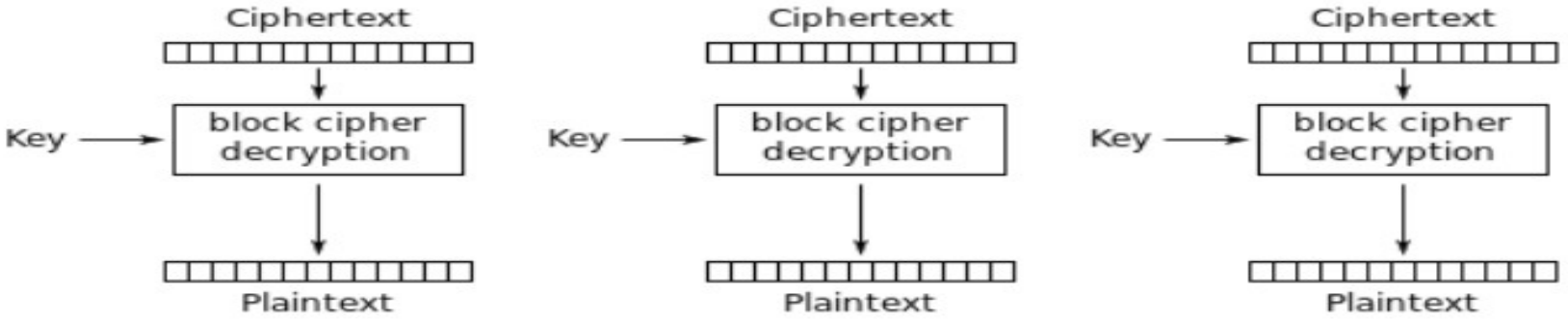
## Criptografia de clau simètrica o privada

### Algorismes de xifrat en bloc mode ECB

Al mode d'operació ECB cada bloc es xifra és independent de l'altre




Electronic Codebook (ECB) mode encryption




Electronic Codebook (ECB) mode decryption

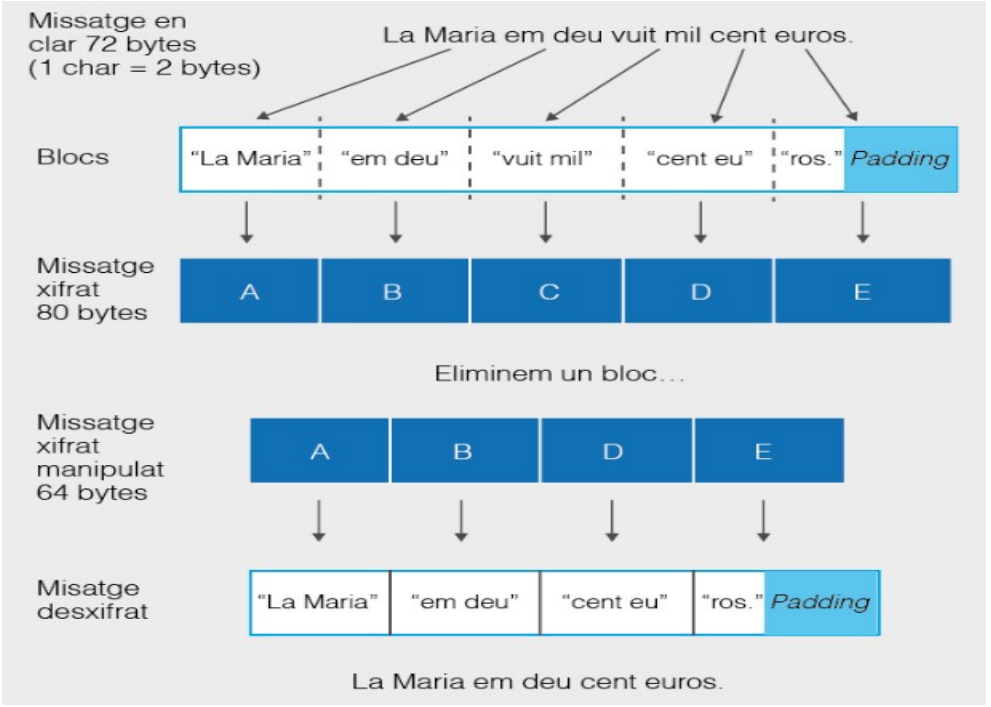
# Criptografia

## Criptografia de clau simètrica o privada

### Algorismes de xifrat en bloc mode ECB

El mode d'operació ECB te dos problemes degut a que cada bloc és independent de l'altre:

- Es pot extreure un bloc xifrat del missatge xifrat sense que ens adonessin
- Si al missatge hi ha blocs idèntics el xifratge serà idèntic i això ajudaria en un atac per desxifrar-lo

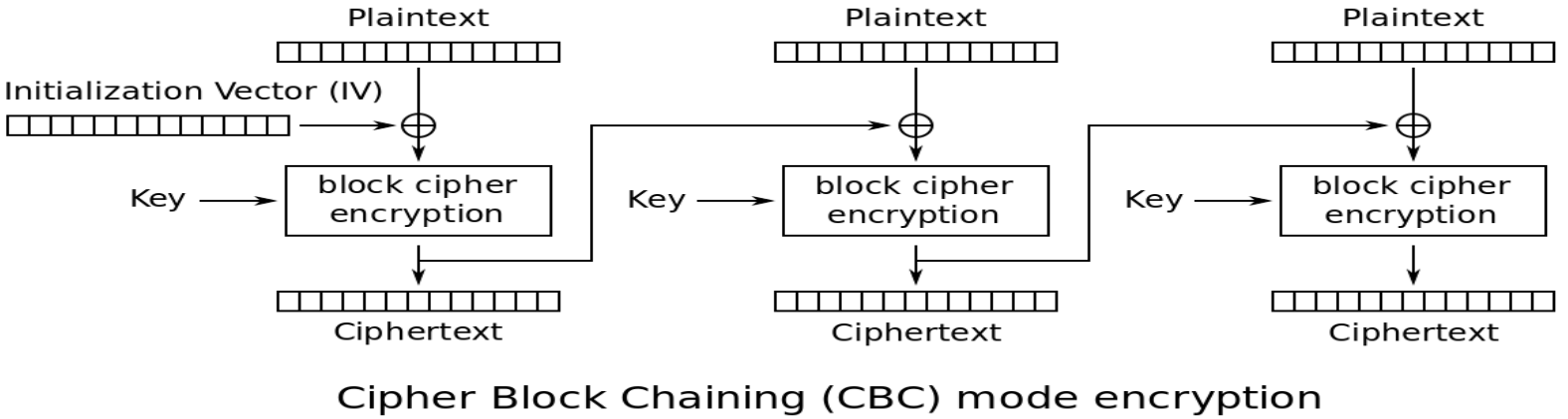


# Criptografia

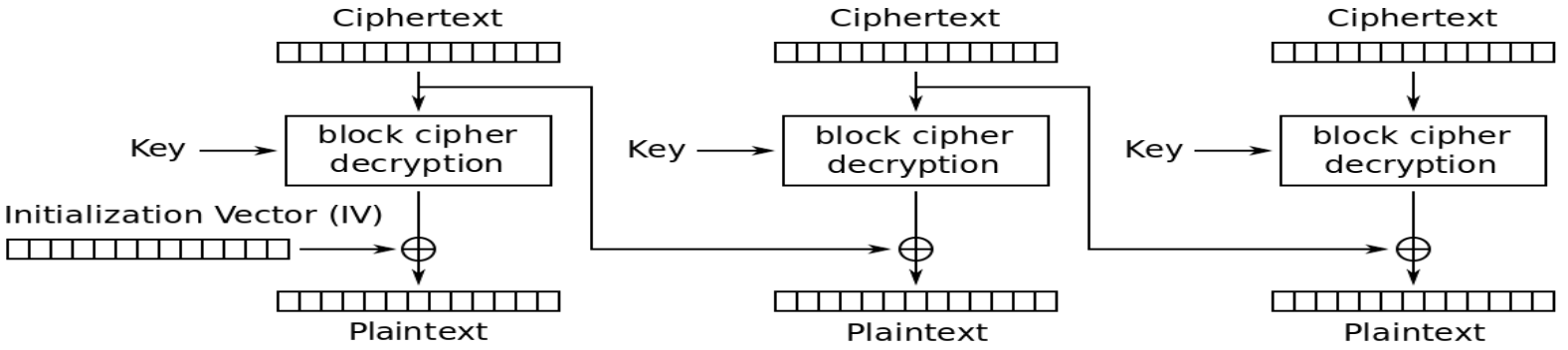
## Criptografia de clau simètrica o privada

### Algorismes de xifrat en bloc mode CBC

Al mode d'operació CBC fem que el xifrat d'un bloc depengui del resultat de xifrar el bloc anterior




Cipher Block Chaining (CBC) mode encryption




Cipher Block Chaining (CBC) mode decryption

# Criptografia

## Criptografia de clau simètrica o privada

### Xifrat AES en mode ECB

Per xifrar/desxifrar en Java farem servir la classe *Cipher. P*er obtenir una instància cridarem al mètode **getInstance** amb l'identificador de l'algorisme.

L'identificador es composa de tres valors separats per una barra '/':

- El nom de l'algorisme en majúscula.
- El mode d'operació, com ara "ECB" o "CBC".
- L'identificador del tipus de padding. N'hi ha de diferents, però se sol usar normalment "PKCS5PADDING"

```
public byte[] encryptData(SecretKey sKey, byte[] data) {
    byte[] encryptedData = null;
    try {
      Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
      cipher.init(Cipher.ENCRYPT_MODE, sKey);
      encryptedData =  cipher.doFinal(data);
    } catch (Exception  ex) {
      System.err.println("Error xifrant les dades: " + ex);
    }
    return encryptedData;
  }
```

Per desxifrar les dades cridarem inicialitzarem Cipher amb Cipher.DECRYPT_MODE

# Criptografia

## Criptografia de clau simètrica o privada

### Xifrat AES en mode CBC

A l'hora de programar en Java com que el primer bloc no té un bloc anterior el que se sol fer és generar un bloc especial anomenat Vector d'Inicialització (Initialization Vector, o IV), que és el que cal aplicar pel primer bloc juntament amb la clau

```
//Definició d'un IV estàtic. Per l'AES ha der ser de 16 bytes (un bloc)
  public static final byte[] IV_PARAM = {0x00, 0x01, 0x02, 0x03,
                                         0x04, 0x05, 0x06, 0x07,
                                         0x08, 0x09, 0x0A, 0x0B,
                                         0x0C, 0x0D, 0x0E, 0x0F};

  public byte[] encryptDataCBC(SecretKey sKey, byte[] data) {
    byte[] encryptedData = null;
    try {
      Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
      IvParameterSpec iv = new IvParameterSpec(IV_PARAM);
      cipher.init(Cipher.ENCRYPT_MODE, sKey, iv);
      encryptedData =  cipher.doFinal(data);
    } catch (Exception  ex) {
      System.err.println("Error xifrant les dades: " + ex);
    }
    return encryptedData;
  }
```

Al desxifrar el vector d'inicialització ha d'esser el mateix que es va usar per xifrar o no obtindreu el text en clar original encara que la clau sigui correcta

# Criptografia

## Criptografia de clau asimètrica o pública

Es refereix a aquells mètodes de xifratge en què la clau utilitzada per xifrar un missatge difereix de la clau utilitzada per desxifrar-lo

Els algorismes de clau pública generen un parell de claus, una de privada i una de pública

Cada usuari disposa d'un parell de claus:

- La clau pública. s'utilitza per xifrar i es distribueix sense preocupar-se de si algú la intercepta
- La clau privada. s'utilitza per desxifrar i esguarda de manera segura perquè ningú altre la sàpiga

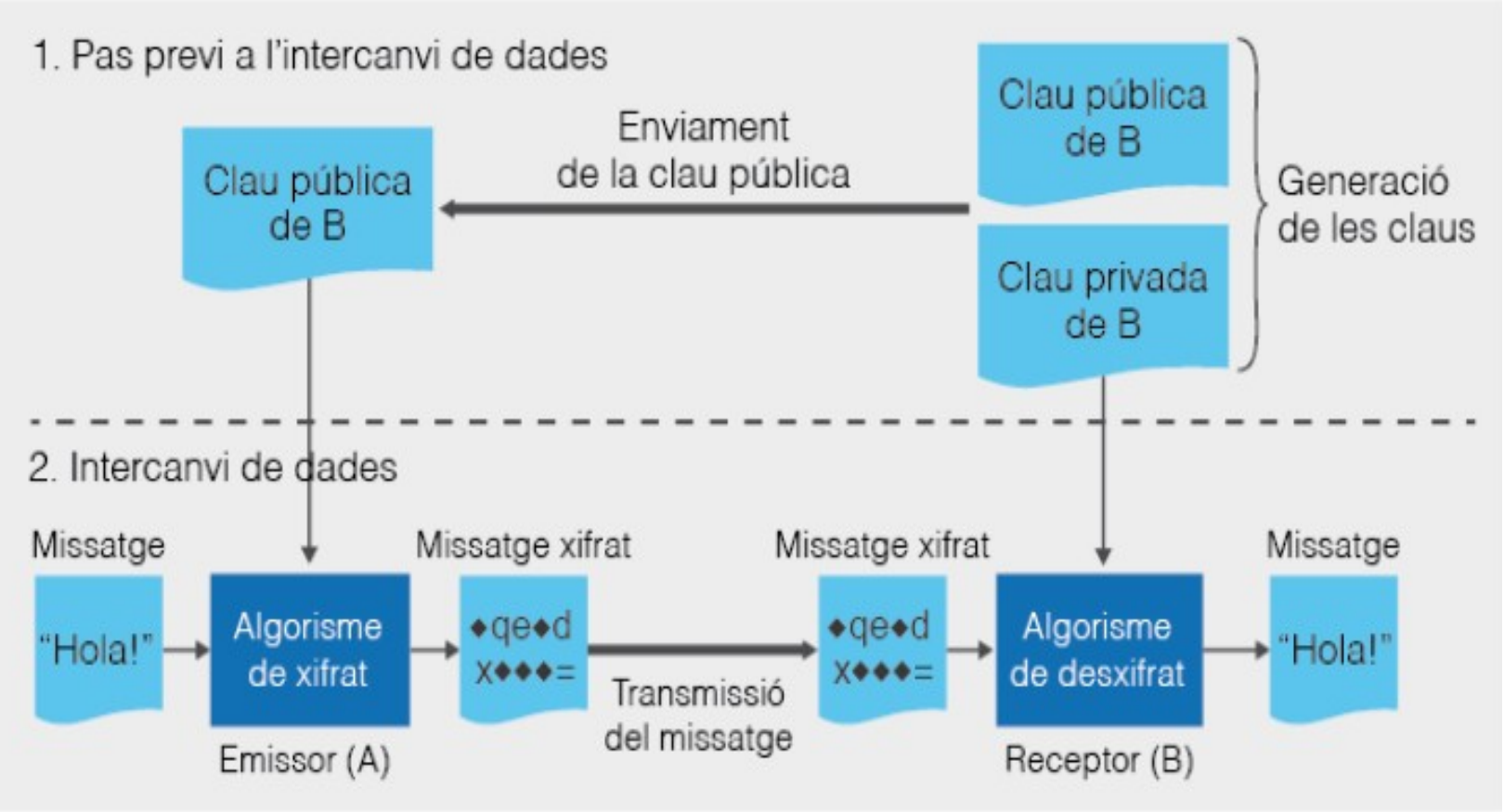
L'algorisme mes conegut és el RSA publicat al 1977 per Rivest, Shamir i Adleman

Les principals aplicacions del xifrat de clau publica són: Protocols de comunicació segurs (HTTPS), la signatura digital, missatgeria instantània

# Criptografia

## Criptografia de clau asimètrica o pública

Esquema del procés de xifrat basat en clau asimètrica o pública



# Criptografia

## Criptografia de clau asimètrica o pública

### Generació de claus privades i públiques

Els parells de claus asimètriques no són un conjunt arbitrari de bytes que es poden generar de qualsevol manera

Pel cas del RSA, la clau privada es genera a partir del producte de dos valors sencers primers molt grans i avui dia s'utilitza una clau de 2048 bits

A Java per generar parells de claus pública-privada es fa servir la classe *KeyPairGenerator*

```
public KeyPair randomGenerate(int len) {
    KeyPair keys = null;
    try {
      KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
      keyGen.initialize(len);
      keys = keyGen.genKeyPair();
    } catch (Exception ex) {
      System.err.println("Generador no disponible.");
    }
    return keys;
  }
```

Els parells de claus es representen amb la classe *KeyPair* i amb els seus mètodes **getPrivate()** i **getPublic()** obtenim la clau privada i pública continguda

Les claus privades es representen amb la classe *PrivateKey* i les públiques amb la classe *PublicKey.*

# Criptografia

## Criptografia de clau asimètrica o pública

### Xifrat RSA directe

A Java aquest xifratge també es basa en la classe *Cipher* i el padding que es fa servir és PKCS1Padding

```
public byte[] encryptData(byte[] data, PublicKey pub) {
    byte[] encryptedData = null;
    try {
      Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding","SunJCE");
      cipher.init(Cipher.ENCRYPT_MODE, pub);
      encryptedData =  cipher.doFinal(data);
    } catch (Exception  ex) {
      System.err.println("Error xifrant: " + ex);
    }
    return encryptedData;
  }
```

Per desxifrar caldrà usar la clau privada al inicialitzar Cipher en mode Cipher.DECRYPT_MODE

# Criptografia

## Criptografia de clau asimètrica o pública

### Xifrat RSA amb clau embolcallada

Una clau RSA està formada pel seu mòdul i l'exponent  i te la limitació que només es poden xifrar dades de longitud màxima = **mida clau RSA en bytes – 11**.

Per tant, una clau de 2.048 bits només pot ser usada per xifrar fins a 245 bytes

La solució de compromís per resoldre aquest problema és una aproximació híbrida, que combina xifrat asimètric amb xifrat simètric

En un sistema de clau embolcallada (wrapped key):

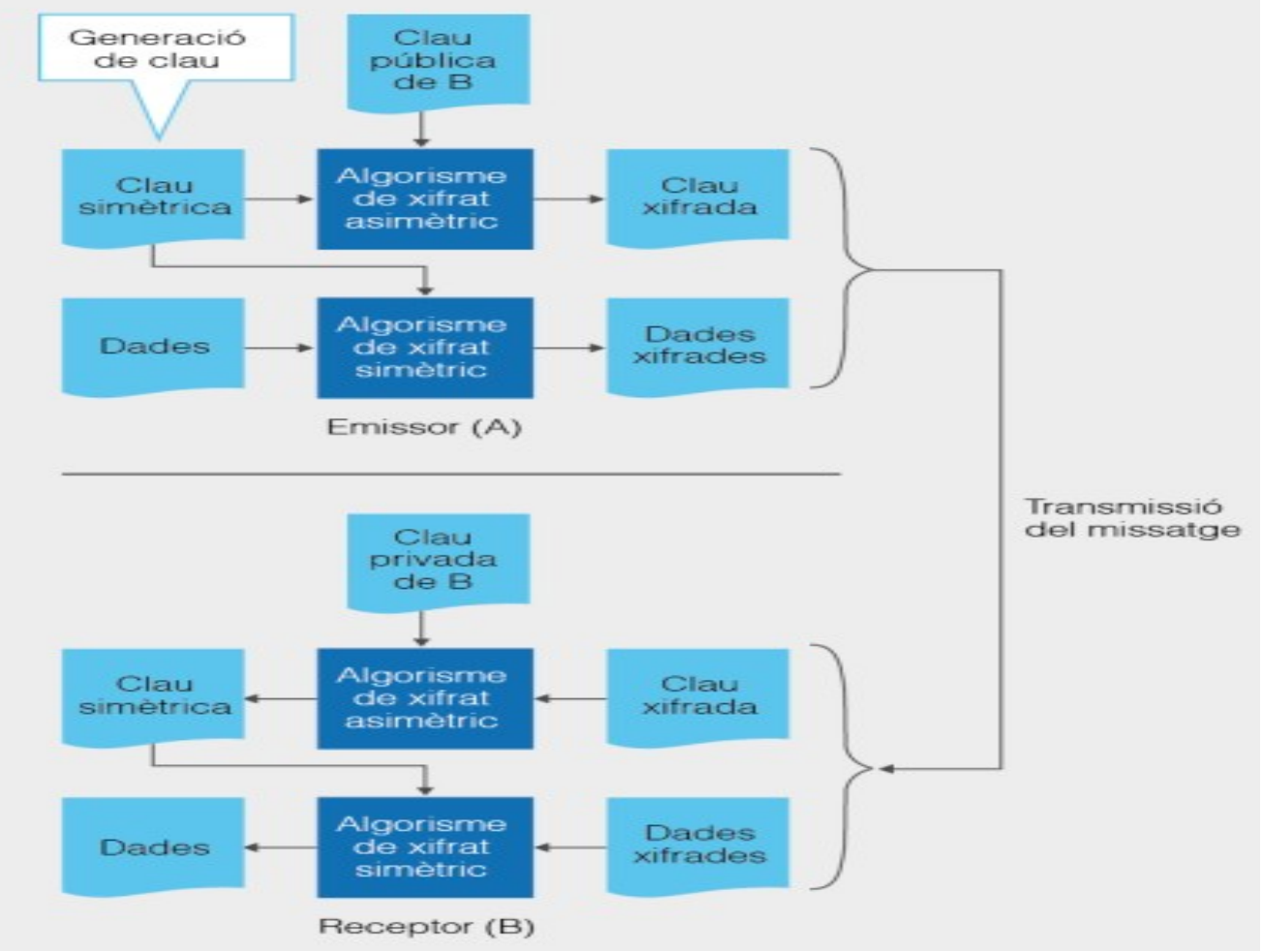
- Les dades es xifren usant una clau simètrica d'un sol ús, generada a l'atzar.
- La clau simètrica es xifra usant la clau pública del destinatari del missatge
- Finalment, s'envia al destinatari el missatge i la clau xifrades

El destinatari per desxifrar ha de desxifrar la clau simètrica amb la seva clau privada i llavors, un cop la té, usar-la per desxifrar el missatge original

# Criptografia

## Criptografia de clau asimètrica o pública

### Xifrat RSA amb clau embolcallada



# Criptografia

## Criptografia de clau asimètrica o pública

### Xifrat RSA amb clau embolcallada

```
public byte[][] encryptWrappedData(byte[] data, PublicKey pub) {
    byte[][] encWrappedData = new byte[2][];
    try {
      KeyGenerator kgen = KeyGenerator.getInstance("AES");
      kgen.init(128);
      SecretKey sKey = kgen.generateKey();
      Cipher cipher = Cipher.getInstance("AES");
      cipher.init(Cipher.ENCRYPT_MODE, sKey);
      byte[] encMsg = cipher.doFinal(data);
      cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
      cipher.init(Cipher.WRAP_MODE, pub);
      byte[] encKey = cipher.wrap(sKey);
      encWrappedData[0] = encMsg;
      encWrappedData[1] = encKey;
    } catch (Exception  ex) {
      System.err.println("Ha succeït un error xifrant: " + ex);
    }
    return encWrappedData;
  }
```

Cal anar amb compte a la inicialització i ús de la classe Cipher a l'embolcallar o desembolcallar la clau simètrica:

- El mode d'inicialització passa a ser WRAP_MODE o UNWRAP_MODE
- Els mètodes per xifrar i desxifrar la clau que cal usar són wrap i unwrap

Per desxifrar cal primer desxifrar la clau simètrica amb la clau privada i després desxifrar el missatge amb la clau simètrica

# Criptografia

## Firma digital

El xifratge per si sol només garanteix la privadesa del missatge

Si volem fer transaccions, tràmits administratius o venda de productes necessitem d'altres serveis de seguretat

- **Integritat**. Poder identificar si un document ha estat manipulat encara que no es pugui evitar la manipulació
- **Autenticació**. Poder garantir quina és la identitat de l'autor del document, evitant que sigui suplantat.
- **No-repudi**. Evitar que l'autor pugui negar que ell ha generat el document

Quan els documents es feiem per escrit això s'aconseguia mitjançant la signatura escrita. Però això no es pot fer directament de manera electrònica

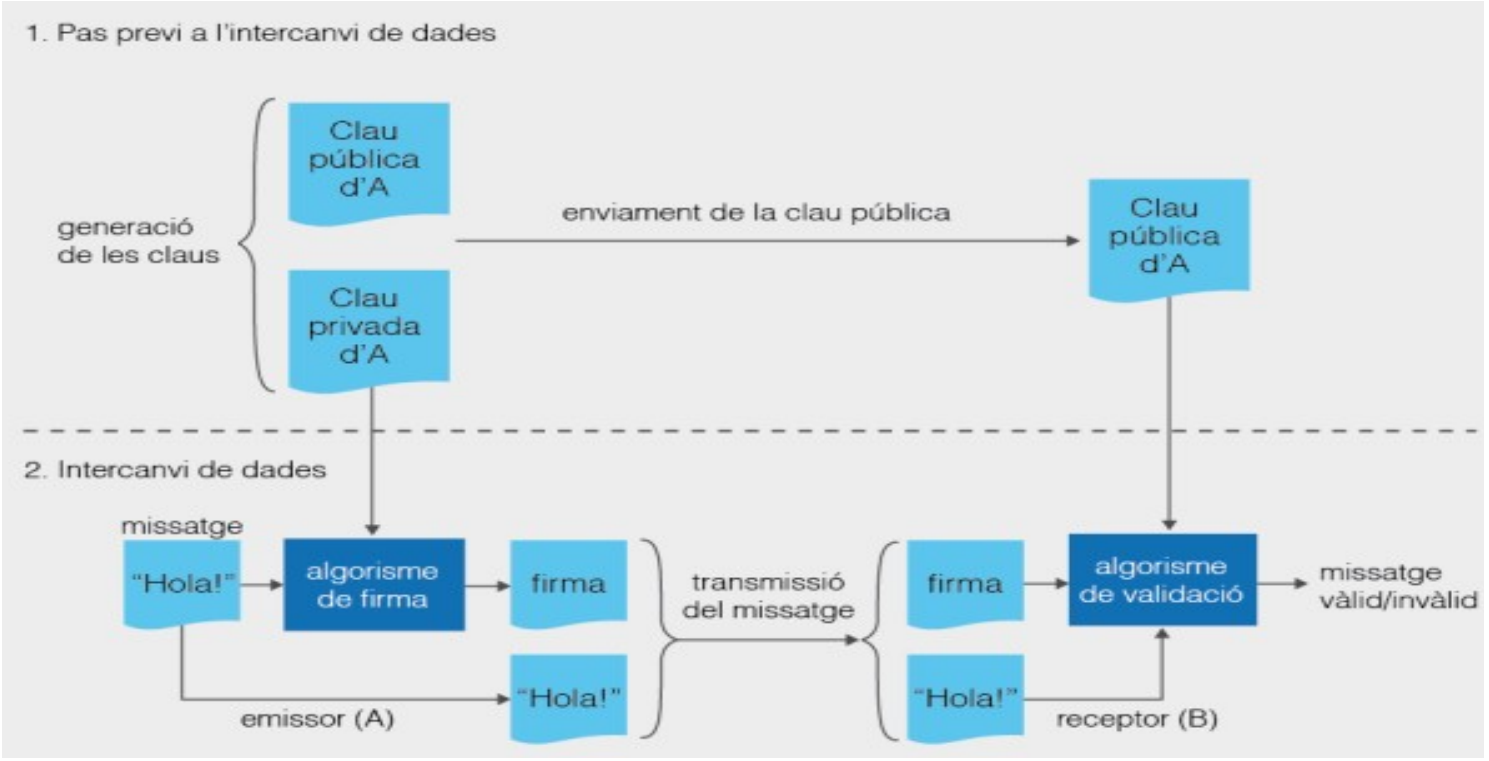
La **firma digital** és el mecanisme criptogràfic que trasllada totes les propietats de la firma manuscrita del món físic a la informació en format digital

Quan firmem digitalment la firma no modifica el document signat sinò que s'envia al receptor el document original i la firma

# Criptografia

## Firma digital

Esquema de firma digital basat en clau asimètrica




L'emisor per signar farà servir la seva clau privada i el receptor validarà la firma fent servir la clau pública de l'emissor

# Criptografia

## Firma digital

Perquè la validació de la firma sigui correcta a l'extrem del receptor, s'han de complir les següents condicions:

- Les dades rebudes han de ser exactament les mateixes que les que es van usar a l'executar l'algorisme de firma
- La firma pròpiament tampoc ha estat modificada
- La clau pública usada per la validació ha de ser exactament l'associada a la clau privada usada en la generació de la firma. Una firma vàlida només es pot haver generat amb una clau privada concreta, cosa que identifica unívocament al seu propietari com el signant

existeixen diferents algorismes de firma digital, nosaltres farem servir el sistema de firma RSA

# Criptografia

## Firma digital

### Generació de firma digital RSA

La generació de claus asimètriques per la firma digital és exactament igual que per generar claus asimètriques pel xifrat (Pàg. 19)

La classe responsable de gestionar una firma digital és *Signature*

```
public byte[] signData(byte[] data, PrivateKey priv) {
    byte[] signature = null;

    try {
      Signature signer = Signature.getInstance("SHA1withRSA");
      signer.initSign(priv);
      signer.update(data);
      signature = signer.sign();
    } catch (Exception ex) {
      System.err.println("Error signant les dades: " + ex);
    }
    return signature;
  }
```

# Criptografia

## Firma digital

### Validació de firma digital RSA

També farem servir la classe *Signature*

```
public boolean validateSignature(byte[] data, byte[] signature, PublicKey pub)
    {
    boolean isValid = false;
    try {
      Signature signer = Signature.getInstance("SHA1withRSA");
      signer.initVerify(pub);
      signer.update(data);
      isValid = signer.verify(signature);
    } catch (Exception ex) {
      System.err.println("Error validant les dades: " + ex);
    }
    return isValid;
  }
```

# Criptografia

## Firma digital

### Certificats digitals

Per gestionar claus públiques cal distingir clarament i de manera senzilla, la identitat del seu propietari.

Un **certificat digital** és un document electrònic que estableix la identitat del propietari d'una clau pública i está compost per una estructura de dades que, a part de la clau, conté un conjunt d'informació addicional

Els certificats digitals més usats segueixen l'anomenat estàndard X.509

Els continguts més importants d'un certificat digital són els següents:

- La clau pública del seu propietari
- La identitat del propietari (Subject)
- La identitat de l'emissor del certificat (Issuer)
- Les seves dates de generació i d'expiració
- Una firma digital, generada per l'emissor, de tota la informació continguda

# Criptografia

## Firma digital

### Certificats digitals

Les identitats, tant del propietari com de l'emissor, es codifiquen mitjançant un **nom distingit**

Components d'un nom distingit (DN, Distinguised Name)

| Acrònim | Nom llarg | Descripció |
|---|---|---|
| CN | *Common Name* | Nom i cognoms si és una persona. Nom DNS si és una màquina. |
| O | *Organization* | Nom de l'organització |
| OU | *Organizational Unit* | Nom de la unitat estructural |
| L | *Localty* | Localitat i ciutat |
| ST | *State* | Província o estat (dels EEUU) |
| CO | *Country* | Dues lletres inicials del país (en anglès) |

# Criptografia

## Firma digital

### Emissió de certificats

A l'hora de distribuir la nostra clau pública usant certificats digitals, una pregunta molt important és Qui serà l'emissor?

Els certificats en els quals la identitat del propietari i l'emissor és la mateixa s'anomenen **autosignats.** El nom distingit del propietari (Subject) i de l'emissor (Issuer) és exactament el mateix. Així mateix, el certificat està signat amb la clau privada de la pròpia clau pública continguda

Com que si et crees tu mateix el certificat pots possar les dades que vulguis, aquests no es poden fer servir per signar documents ja que el receptor no pot confiar en que li ha donat el certificat

La majoria de certificats digitals usats en entorns reals han estat signats per una tercera entitat, una **autoritat de certificació** (CA, Certification Authority)

# Criptografia

## Firma digital

### L'autoritat de certificació

És una entitat que actua com a notari digital, tothom implicat en un intercanvi de claus confia en què mai emetrà un certificat amb dades incorrectes

Quan algú vol generar un certificat:

- Envia una petició amb la seva clau pública i la seva identitat a l'autoritat
- L'autoritat comprova d'alguna manera que les dades del sol·licitant siguin reals. Per exemple fen-te anar amb el DNI a una de les seves oficines
- Llavors genera el certificat, signant-lo amb la seva clau privada

Per poder fer aquest servei la CA necessita una infraestructura que s'anomena infraestructura de clau pública (o PKI, Public Key Infrastructure) que li permet desplegar un mecanisme segur d'intercanvi de claus públiques

Són exemples de CA l'Agència Catalana de Certificació (CATCert) o la Fábrica Nacional de Moneda i Timbre (FNMT), que generen certificats vàlids per fer la declaració de la renda per Internet

# Criptografia

## Firma digital

### Validació del certificat digital

Que vol dir veure si realment ha estat emès per l'autoritat de certificació que posa al seu camp de l'emissor

Cal fer un seguit de comprovacions:

- Comprovar si la firma digital del certificat és correcta, usant la clau pública de la CA
- Comprovar si el certificat és vigent en la data actual, validant la seva data d'emissió i d'expiració
- Comprovar si està o no a la llista de certificats revocats (donats de baixa), que proporciona la mateixa CA