Version : **2022.02**.

Dernière mise-à-jour : 2023/01/07 13:51

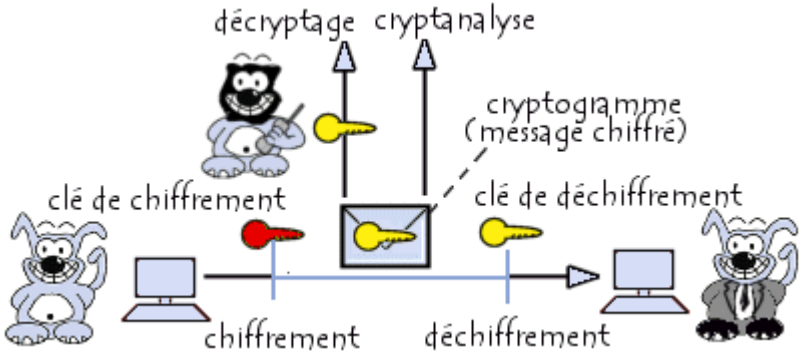
RES403 - Comprendre le Chiffrement

Contenu du Module

Introduction à la cryptologie

Définitions

- **La Cryptologie**
  - La science qui étudie les aspects scientifiques de ces techniques, c'est-à-dire qu'elle englobe la cryptographie et la cryptanalyse.
- **La Cryptanalyse**
  - Lorsque la clef de déchiffrement n'est pas connue de l'attaquant on parle alors de cryptanalyse ou cryptoanalyse (on entend souvent aussi le terme plus familier de cassage).
- **La Cryptographie**
  - Un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Les verbes crypter et chiffrer sont utilisés.
- **Le Décryptement ou Décryptage**
  - Est le fait d'essayer de déchiffrer illégitimement le message (que la clé de déchiffrement soit connue ou non de l'attaquant).



La Cryptographie

La cryptographie apporte quatre points clefs:

- La confidentialité
  - consiste à rendre l'information inintelligible à d'autres personnes que les acteurs de la transaction.

- L'intégrité
  - consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).
- L'authentification
  - consiste à assurer l'identité d'un utilisateur.
- La non-répudiation
  - est la garantie qu'aucun des correspondants ne pourra nier la transaction.

La cryptographie est basée sur l'arithmétique. Il s'agit, dans le cas d'un texte, de transformer les lettres qui composent le message en une succession de chiffres (sous forme de bits dans le cas de l'informatique), puis ensuite de faire des calculs sur ces chiffres pour:

- Procéder au chiffrement
  - Le résultat de cette modification (le message chiffré) est appelé cryptogramme (Ciphertext) par opposition au message initial, appelé message en clair (Plaintext)
- Procéder au déchiffrement

Le chiffrement se fait à l'aide d'une clef de chiffrement. Le déchiffrement nécessite une clef de déchiffrement.

On distingue deux types de clefs:

- Les clés symétriques:
  - des clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- Les clés asymétriques:
  - des clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

Le Chiffrement par Substitution

Le chiffrement par substitution consiste à remplacer dans un message une ou plusieurs entités (généralement des lettres) par une ou plusieurs autres entités. On distingue généralement plusieurs types de cryptosystèmes par substitution :

- La substitution **monoalphabétique**
  - consiste à remplacer chaque lettre du message par une autre lettre de l'alphabet
- La substitution **polyalphabétique**

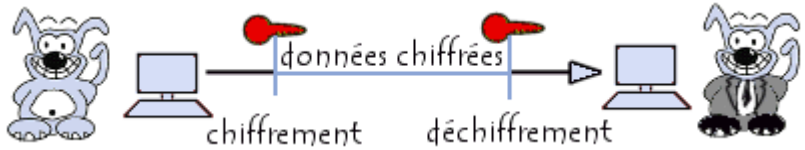
    - consiste à utiliser une suite de chiffres monoalphabétique réutilisée périodiquement
- La substitution **homophonique**
    - permet de faire correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères
- La substitution de **polygrammes**
    - consiste à substituer un groupe de caractères (polygramme) dans le message par un autre groupe de caractères

Algorithmes à clé secrète

Le Chiffrement Symétrique

Ce système est aussi appelé le système à **Clef Secrète** ou à **clef privée**.

Ce système consiste à effectuer une opération de chiffrement par algorithme mais comporte un inconvénient, à savoir qu'il nécessite un canal sécurisé pour la transmission de la clef de chiffrement/déchiffrement.



Le système de Méthode du Masque Jetable (One Time Pad) fût mis au point dans les années 1920. Il utilisait une clef générée aléatoirement à usage unique.

Les algorithmes de chiffrement symétrique couramment utilisés en informatique sont:

- **Data Encryption Standard** (DES),
- **Triple DES** (3DES),
- **RC2**,
- **Blowfish**,

- **International Data Encryption Algorithm** (IDEA),
- **Advanced Encryption Standard** (AES).

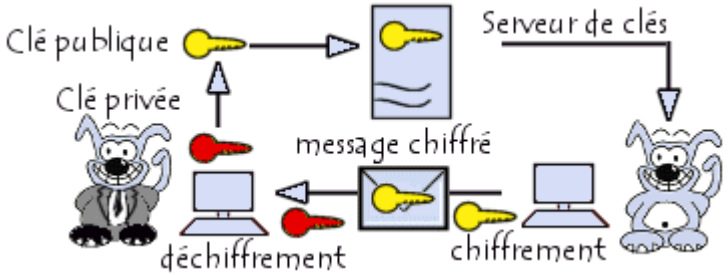
Algorithmes à clef publique

Le Chiffrement Asymétrique

Ce système est aussi appelé **Système à Clef Publique**.

Ce système consiste à avoir deux clefs appelées des **bi-clefs**:

- Une clef **publique** pour le chiffrement
- Une clef **secrète** ou **privée** pour le déchiffrement



- L'utilisateur A (celui qui déchiffre) choisit une clef privée.
- A partir de cette clef il génère plusieurs clefs publiques grâce à un algorithme.
- L'utilisateur B (celui qui chiffre) choisit une des clefs publiques à travers un canal non-sécurisé pour chiffrer les données à l'attention de l'utilisateur A.

Ce système est basé sur ce que l'on appelle une **fonction à trappe à sens unique** ou **one-way trap door**.

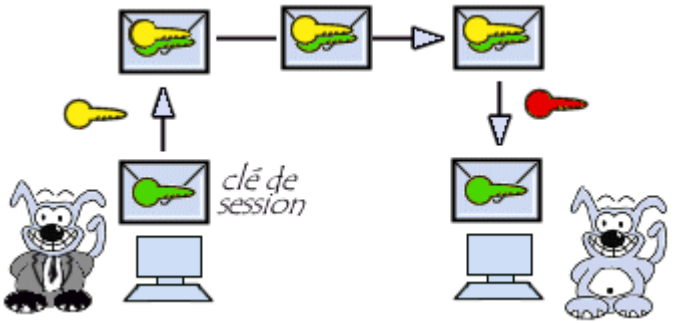
Il existe toutefois un problème – s'assurer que la clef publique récupérée est bien celle qui correspond au destinataire !

Les algorithmes de chiffrement asymétrique couramment utilisés en informatique sont:

- **Digital Signature Algorithm** (DSA)
- **Rivest, Shamir, Adleman** (RSA)

La Clef de Session

Ce système est un compromis entre le système symétrique et le système asymétrique. Il permet l'envoie de données chiffrées à l'aide d'un algorithme de chiffrement symétrique par un canal non-sécurisé et a été mis au point pour palier au problème de lenteur de déchiffrement du système asymétrique.

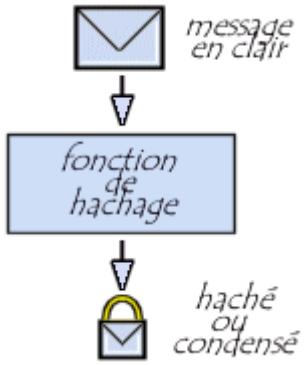


Ce système fonctionne de la façon suivante :

- L'utilisateur A chiffre une clef privée générée aléatoirement, appelée une « clef de session », en utilisant une des clefs publiques de l'utilisateur B.
- L'utilisateur A chiffre les données avec la clef de session.
- L'utilisateur B déchiffre la clef de session en utilisant sa propre clef privée.
- L'utilisateur B déchiffre les données en utilisant la clef de session.

Fonctions de Hachage

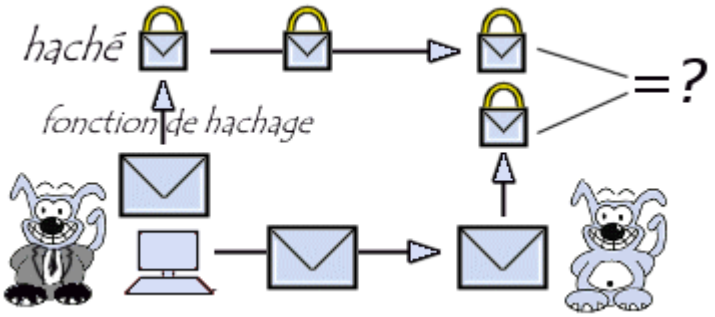
La fonction de **hachage**, aussi appelée une fonction de **condensation**, est à **sens unique** (one way function). Il « condense » un message en clair et produit un haché unique.



Les deux algorithmes de hachage utilisés sont:

- **Message Digest 5** (MD5)
- **Secure Hash Algorithm** (SHA)

Lors de son envoie, le message est accompagné de son haché et il est donc possible de garantir son intégrité:

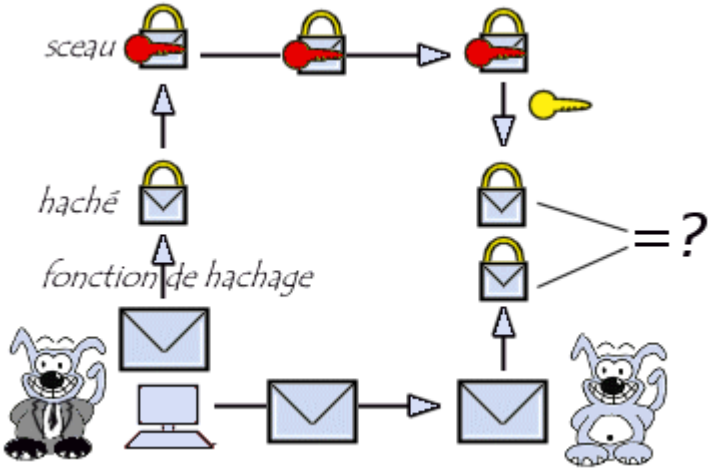


- A la réception du message, le destinataire ou l'utilisateur B calcule le haché du message reçu et le compare avec le haché accompagnant le document.
- Si le message ou le haché a été falsifié durant la communication, les deux empreintes ne correspondront pas.

Ce système permet de vérifier que l'empreinte correspond bien au message reçu, mais ne permet pas de prouver que le message a bien été envoyé par l'utilisateur A.

Signature Numérique

Pour garantir l'authentification du message l'utilisateur A va chiffrer ou **signer** le haché à l'aide de sa clé privée. Le haché signé est appelé un **sceau**.



- L'utilisateur A envoie le sceau au destinataire.
- A la réception du message L'utilisateur B déchiffre le sceau avec la clé publique de l'utilisateur A.
- Il compare le haché obtenu au haché reçu en pièce jointe.

Ce mécanisme de création de sceau est appelé **scellement**.

Ce mécanisme est identique au procédé utilisé par SSH lors d'une connexion

Utilisation de GnuPG

Présentation

GNU Privacy Guard permet aux utilisateurs de transférer des messages chiffrés et/ou signés.

Installation

Sous RHEL/CentOS 8, le paquet gnupg est installé par défaut :

```
root@debian8:~# whereis gpg
gpg: /usr/bin/gpg /usr/share/man/man1/gpg.1.gz
```

Configuration

Pour initialiser GnuPG, saisissez la commande suivante :

```
root@debian8:~# whereis gpg
gpg: /usr/bin/gpg /usr/share/man/man1/gpg.1.gz
root@debian8:~# gpg
gpg: directory `/root/.gnupg' created
gpg: new configuration file `/root/.gnupg/gpg.conf' created
gpg: WARNING: options in `/root/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/root/.gnupg/secring.gpg' created
gpg: keyring `/root/.gnupg/pubring.gpg' created
gpg: Go ahead and type your message ...
^C
gpg: Interrupt caught ... exiting
```

Pour aider gpg dans la génération des clefs, utilisez **rngd** pour fournir suffisament d“Entropy” au noyau :

```
root@debian8:~# apt-get install rng-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  rng-tools
0 upgraded, 1 newly installed, 0 to remove and 95 not upgraded.
```

```
Need to get 46.8 kB of archives.
After this operation, 209 kB of additional disk space will be used.
Get:1 http://ftp.fr.debian.org/debian/ jessie/main rng-tools amd64 2-unofficial-mt.14-1 [46.8 kB]
Fetched 46.8 kB in 0s (146 kB/s)
Selecting previously unselected package rng-tools.
(Reading database ... 82472 files and directories currently installed.)
Preparing to unpack .../rng-tools_2-unofficial-mt.14-1_amd64.deb ...
Unpacking rng-tools (2-unofficial-mt.14-1) ...
Processing triggers for systemd (215-17+deb8u4) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up rng-tools (2-unofficial-mt.14-1) ...
Job for rng-tools.service failed. See 'systemctl status rng-tools.service' and 'journalctl -xn' for details.
invoke-rc.d: initscript rng-tools, action "start" failed.
Processing triggers for systemd (215-17+deb8u4) ...

root@debian8:~# rngd -f -r /dev/urandom
rngd 2-unofficial-mt.14 starting up...
entropy feed to the kernel ready
^Cstats: bits received from HRNG source: 60064
stats: bits sent to kernel pool: 4096
stats: entropy added to kernel pool: 4096
stats: FIPS 140-2 successes: 3
stats: FIPS 140-2 failures: 0
stats: FIPS 140-2(2001-10-10) Monobit: 0
stats: FIPS 140-2(2001-10-10) Poker: 0
stats: FIPS 140-2(2001-10-10) Runs: 0
stats: FIPS 140-2(2001-10-10) Long run: 0
stats: FIPS 140-2(2001-10-10) Continuous run: 0
stats: HRNG source speed: (min=64.005; avg=64.949; max=65.998)Mibits/s
stats: FIPS tests speed: (min=99.861; avg=111.541; max=124.663)Mibits/s
stats: Lowest ready-buffers level: 2
stats: Entropy starvations: 0
stats: Time spent starving for entropy: (min=0; avg=0.000; max=0)us
Exiting...
```

```
root@debian8:~# cat /proc/sys/kernel/random/entropy_avail
2022
```

Pour générer les clefs, saisissez la commande suivante :

```
root@debian8:~# gpg --gen-key
gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: I2TCH
```

```
Email address: infos@i2tch.eu
Comment: Test Key
You selected this USER-ID:
    "I2TCH (Test Key) <infos@i2tch.eu>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 158 more bytes)
.......+++++
.+++++
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
+++++
..+++++
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 324951F4 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/324951F4 2016-08-07
      Key fingerprint = 293A 4AB0 C917 DEAD 1838  FAE6 B112 5F50 3249 51F4
uid                  I2TCH (Test Key) <infos@i2tch.eu>
sub   2048R/315943DA 2016-08-07
```

La liste de clefs peut être visualisée avec la commande suivante :

```
root@debian8:~# gpg --list-keys
/root/.gnupg/pubring.gpg
------------------------
pub   2048R/324951F4 2016-08-07
uid                  I2TCH (Test Key) <infos@i2tch.eu>
sub   2048R/315943DA 2016-08-07
```

Pour importer la clef d'un correspondant dans sa trousse de clefs il convient d'utiliser la commande suivante :

```
# gpg --import la-clef.asc
```

Pour exporter sa clef publique, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --export --armor I2TCH > ~/I2TCH.asc
root@debian8:~# cat I2TCH.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQENBFem/AUBCADD1ExMqUny634CUdpqY8zvkaaPdg3JdV3dSmUEMOhhqyaftp++
RLqhi7F7+/uSTNv1I611xsVQ1zgPO8YTIwM2c/glIKa6pRTfPFO7qzPczFMhwNrO
a2jnwJjlqg4/BB8N0xWry2TTOjn6lTcwtWcFzjy0jmjFDQLut0/85mnGNB2aml9r
iaSMF2XwvjSIz6nPv0EVvfzzLUdebKBDBrqiZebCbCfeTzR3yyMr9rA8QWdfCHuW
yQaI/FO9BvHdv6CTmBXlKcfu+nTuBfhUfWlWIYn2549Fy47KIdLS+rB7b91FlUw+
8kidI4AVIUm2WlmpLzTJN/0m1PuBoHgi8TLzABEBAAG0IUkyVENIIChUZXN0IEtl
eSkgPGluZm9zQGkydGNoLmV1PokBOAQTAQIAIgUCV6b8BQIbAwYLCQgHAwIGFQgC
CQoLBBYCAwECHgECF4AACgkQsRJfUDJJUfSz3ggAwsH18xuOjidLNST2OM+8Sf/6
954Ajp3x7zBSQCihhHnuVL/vwlrIJ8ScHudQDPq42+zghfH+37O1PXU7cv43hsXT
+cLtSRjGtVE3uvScIUodJONGJJM6oOf3doyLsYHfA1511IWViryTGylS5sBBfjcN
```

```
/ljiSjE9K/IRqJ6g1iKU8reC5DU+FmjpOJsnTTef2Je91iHfuR+DPZUzgimulI5G
0Nrq4pqOD/o1j55N+AKM3fqlqRlyUuozsT+CfBGC/xN9gj1FTeeof3bwNoWuiD75
8nYA97eX0jQooxyrHq+9HHU6kvFtOVVpUEbgHyZzenQzdbAuYabJTETE7vgMfbkB
DQRXpvwFAQgArXPkbmoLS/B7eBUm1cTVvkcJET/RG4AcVs4aZNZ24ve8/qNva7Ec
d4A3kG1t3rHKlFlHnsGm8tHw3Jjg+/6WFFAzG4mzm8QrwA+vnmcHmSrhVgCaA0NS
vqOIWCys96bKcwLIJuYDK9kLuUDRRniMcaA2sl44BaVDl9V8HEm3PS3jYbewwCZa
Z7vtiiK39cyn6AatZHBw7ubeYupmtTUc34dfSym5K7jKgg3VOhaVGsDF1OPogBit
w1tLdNHRyxYkIkhV9xtBIaUMSDsfulmMVsQXXCwY27m3EQMc25C7xuQ5K/+TRLsv
DKzcYcPcHd0cT8B7Ym7+Xetq+CdQD1jOSQARAQABiQEfBBgBAgAJBQJXpvwFAhsM
AAoJELESX1AySVH0/PsIAKwbeV1fwcucq3X+afa/DtEmFWlRS50XwqVGb/ADu10R
7XsftBUSRBPMTvKvFNLSOklggKKP2OFz3ZmBttjdJXL+24U48LtLplG2cfXpAzjD
7rMVuKICgJGHlLWr+sT3t2uH/Mw7Rn3aw0a1MpV5GoqrBvKfdcYbTcbp9HSzCaJL
eVYbXTwXRLlyhcPVy2BZl8OVQlizqLkuzonAmzOMUpYAXU983MZlzEKhMp8R/otC
vx/Y+7kCHiWBqZzpD1MktjeSxUWt+jC7z4pb13t7DOFUzysiBdAYhWtCWAIbJdO2
Pi6GtIZkpKnV1Q7Ct6x3K7HJKfpis3YjIZSFq+Gr09o=
=bhYQ
-----END PGP PUBLIC KEY BLOCK-----
```

Cette clef peut ensuite être jointe à des messages électroniques ou bien déposée sur un serveur de clefs tel http://www.keyserver.net.

Signer un message

Créez maintenant un message à signer :

```
root@debian8:~# vi ~/message.txt
root@debian8:~# cat ~/message.txt
This is a test message for gpg
```

Pour signer ce message en format binaire, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --default-key I2TCH --detach-sign message.txt
```

```
You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 324951F4, created 2016-08-07

root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.sig
□□□□□□□□W□□□
    □□_P2IQ□E□□□>□□)7□Lt□□□□□C□+□vC□0□□cX□□□□□1N□Z□6□;□□□□□□R□□□□Y□□>□gЧ□□□c□□□□T□iO|&□□TF□□□h□□□□u□□□□□V□
                                _ă□□Tf□□9ct□□□ě□□Q□i>□□ж□□□"K□c□□)□□□~□□□□*□u□f%□□DykheG/u□□_□K□'□□}G□H
                        □R^i□tЛь□y□`□[]a□_1□ժP)□a□□\□□□□ɘFO$m□□l
                                                                    3z□□□я□{□ҷe□*H□□w□H□N   □□□□x□ڳ□□
```

Pour signer ce message en format ascii, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --default-key I2TCH --armor --detach-sign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 324951F4, created 2016-08-07

root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      473 Aug 10 11:29 message.txt.asc
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.asc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAABAgAGBQJXqwICAAoJELESX1AySVH0tqUH/igOKaMwaEIgPCwUpv8rJLIy
```

```
2cCj2stliojf437f7eZDyHTNiB/ghmLc6GCogV3uA+jJg2O5vidI5HTD7M2qnnix
CTFRMwH0v61ssZTU/nB+KyO2S3NWRGV1T/dGXJGjXf2QBcSowWoTeQBjbVGDnB3O
twDpWH8NFdW7yGNoKdnPkCdrKEMuUCthLz05W1yCngFDpPwkd8MM9wFq7UdQ+EN9
G2iiyIQbArOI18v67BH571z+4UlOfJlUB+O2C9D8tobJBU7KNSPLWnYAJck6YrrC
3RVW5M74LcHtfQypSeCJAqxiZf2SMtNU99zPnDqMzwX8tNCpYrlL3IX6DdduOJ4=
=/wL9
-----END PGP SIGNATURE-----
```

Pour vérifier la signature d'un message signé en mode ascii, il convient d'utiliser la commande :

```
root@debian8:~# gpg --verify message.txt.asc
gpg: assuming signed data in `message.txt'
gpg: Signature made Wed 10 Aug 2016 11:34:35 BST using RSA key ID 324951F4
gpg: Good signature from "I2TCH (Test Key) <infos@i2tch.eu>"
```

Pour vérifier la signature d'un message signé en mode ascii et produit en dehors du message lui-même, il convient d'utiliser la commande :

```
# gpg --verify message.txt.asc message.txt
```

Pour signer ce message **dans le message lui-même** en format ascii, il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --default-key I2TCH --clearsign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 324951F4, created 2016-08-07

File `message.txt.asc' exists. Overwrite? (y/N) y
root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
```

```
-rw-r--r-- 1 root root      551 Aug 10 11:30 message.txt.asc
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig
root@debian8:~# cat message.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

This is a test message for gpg
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAEBAgAGBQJXqwJHAAoJELESX1AySVH0VEMIAJc7NP2v8s/mmllpRi3Kj9W1
oTS721z/XSbM4iQaFM7QnoJmlfCaAbOB01WNqiAVL4A1LGFntttknsoMF7lERU6k
hPCeMtdcWTF3/KwlLHBZ3jjNJeVS4BfJjiW0qStgbMRuzaVHV0iONACA80mnYEOx
TkVnk/IMkO0fsCBocbFeJ/DdR/P9o4u4yqhkwin2+cKPoEWUYBODhIOtHzLMuq1m
582UmrGUQq5z6CC7kiZzifbOtm54pT5MioVfHpYwt6+zlfvYhgVn8VQ62eKAg0zs
IaaRTYVmlD1XUbWxswqvBA9RwIRck6A5Oi5YAoH8jUHaZjvVK9KaEXDQ7Ga/Nk4=
=i5f6
-----END PGP SIGNATURE-----
```

Chiffrer un message

Pour chiffrer un message, il faut disposer de la clef publique du destinataire du message. Ce dernier utilisera ensuite sa clef privée pour déchiffrer le message. Il convient de préciser le destinataire du message, ou plus précisément la clef publique à utiliser, lors d'un chiffrement :

```
  gpg --recipient <destinataire> --encrypt <message>
```

- <*destinataire*> représente toute information permettant de distinguer sans ambigüité une clef publique dans votre trousseau. Cette information peut-être le nom ou l'adresse email associé à la clef publique que vous voulez utiliser,
- <*message*> représente le message à chiffrer.

Par exemple pour chiffrer un message en mode binaire, il convient de saisir la commande suivante :

```
root@debian8:~# gpg --recipient I2TCH --encrypt message.txt
```

```
root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      473 Aug 10 11:34 message.txt.asc
-rw-r--r-- 1 root root      367 Aug 10 11:37 message.txt.gpg
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.sig
□□□□□□□□W□□□
    □□_P2IQ□E□□□>□□)7□Lt□□□□C□+□vC□0□□cX□□□□□1N□Z□6□;□□□□□R□□□□Y□□>□gЧ□□□c□□□□T□iO|&□□TF□□□h□□□□u□□□□□V□
                              _ă□□Tf□□9ct□□□ě□□Q□i>□□ж□□□"K□c□□)□□□~□□□□*□u□f%□□DykheG/u□□_□K□'□□}G□H
                      □R^i□tЛь□y□`□[]a□_1□дP)□a□□\□□□□ɘFO$m□□l
                                                                  3z□□□я□{□ҷe□*H□□w□H□N  □□□□x□¿□□
```

Et pour chiffrer un message en mode ascii, il convient de saisir la commande suivante :

```
root@debian8:~# gpg --recipient I2TCH --armor --encrypt message.txt
File `message.txt.asc' exists. Overwrite? (y/N) y

root@debian8:~# ls -l | grep message
-rw-r--r-- 1 root root       31 Aug  7 10:18 message.txt
-rw-r--r-- 1 root root      579 Aug 10 11:38 message.txt.asc
-rw-r--r-- 1 root root      367 Aug 10 11:37 message.txt.gpg
-rw-r--r-- 1 root root      287 Aug  7 10:18 message.txt.sig

root@debian8:~# cat message.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1

hQEMA9hdNoMxWUPaAQgAk8S4GyQNVeB36uoLY0icuC/WXoxL5P7cJHXVnQuPQiBU
XJX3rHW0RCavc6mLTJvUGYUysubcVisQFWm3LTZ0ZD796S671EdLNNGgsfHNdR90
ext6f6UihR2ep5Y++8eTSat7YQc8nLiF5yexbnOCBuyxGI7gP2lLiZAX6sifmY61
pMegrrhA4BN2Wupbwm2A7WX4NcU4mdZtNxW+zC1Rtp+NOr6ad5JftEes2yPdUcrD
WW2gZaL1DJx50OmBc/tmUt1ea2VEtVDr1WDosD6dEWUbffDBA6wzr1DzUW488D0s
```

```
mYoVwu1bYSSSzmNGGvlFDa6EE/nHwVMhvgld1SB63tJeAbYgXKgEyKTfgIe/Byss
EyofLf5p+DVtJs1MK3OdJ87GV5n2XamtD3Qp3O2EvF/YW9e9aRmt6rF9jLSbIBiS
m4Ka3BM8p2yK9PQQbAvQIIyUg9TjP31c1bzdRa/VNQ==
=laaP
-----END PGP MESSAGE-----
```

Pour décrypter un message il convient d'utiliser la commande suivante :

```
root@debian8:~# gpg --decrypt message.txt.asc

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 315943DA, created 2016-08-07 (main key ID 324951F4)

gpg: encrypted with 2048-bit RSA key, ID 315943DA, created 2016-08-07
      "I2TCH (Test Key) <infos@i2tch.eu>"
This is a test message for gpg
```

Public Key Infrastructure

On appelle **PKI** (Public Key Infrastucture, ou en français **infrastructure à clé publique (ICP)**, parfois **infrastructure de gestion de clés (IGC)**) l'ensemble des solutions techniques basées sur la cryptographie à clé publique.

Les cryptosystèmes à clés publiques permettent de s'affranchir de la nécessité d'avoir recours systématiquement à un canal sécurisé pour s'échanger les clés. En revanche, la publication de la clé publique à grande échelle doit se faire en toute confiance pour assurer que :

- La clé publique est bien celle de son propriétaire ;
- Le propriétaire de la clé est digne de confiance ;
- La clé est toujours valide.

Ainsi, il est nécessaire d'associer au bi-clé (ensemble clé publique / clé privée) un certificat délivré par un **tiers de confiance** : l'infrastructure de gestion de clés.

Le tiers de confiance est une entité appelée communément autorité de certification (ou en anglais Certification authority, abrégé CA) chargée d'assurer la véracité des informations contenues dans le certificat de clé publique et de sa validité.

Pour ce faire, l'autorité signe le certificat de clé publique à l'aide de sa propre clé en utilisant le principe de signature numérique.

Le rôle de l'infrastructure de clés publiques est multiple et couvre notamment les champs suivants :

- enregistrer des demandes de clés en vérifiant l'identité des demandeurs ;
- générer les paires de clés (clé privée / clé publique) ;
- garantir la confidentialité des clés privées correspondant aux clés publiques ;
- certifier l'association entre chaque utilisateurs et sa clé publique ;
- révoquer des clés (en cas de perte par son propriétaire, d'expiration de sa date de validité ou de compromission).

Une infrastructure à clé publique est en règle générale composée de trois entités distinctes :

- L'autorité d'enregistrement (AE ou RA pour Recording authority), chargée des formalité administratives telles que la vérification de l'identité des demandeurs, le suivi et la gestion des demandes, etc.) ;
- L'autorité de certification (AC ou CA pour Certification Authority), chargée des tâches techniques de création de certificats. L'autorité de certification est ainsi chargée de la signature des demandes de certificat (CSR pour Certificate Signing Request, parfois appelées PKCS#10, nom du format correspondant). L'autorité de certification a également pour mission la signature des listes de révocations (CRL pour Certificate Revocation List) ;
- L'Autorité de dépôt (Repository) dont la mission est de conserver en sécurité les certificats.

Certificats X509

Pour palier aux problèmes liés à des clefs publiques piratées, un système de certificats a été mis en place.

Le certificat permet d’associer la clef publique à une entité ou une personne. Les certificats sont délivrés par des Organismes de Certification.

Les certificats sont des fichiers divisés en deux parties :

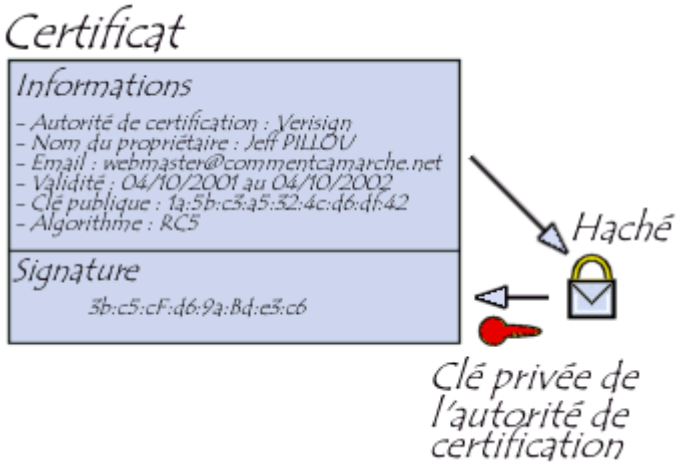
- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification

La structure des certificats est normalisée par le standard **X.509** de l'**Union internationale des télécommunications**.

Elle contient :

- Le nom de l'autorité de certification
- Le nom du propriétaire du certificat
- La date de validité du certificat
- L'algorithme de chiffrement utilisé
- La clé publique du propriétaire

Le Certificat est signé par l'autorité de certification:



La vérification se passe ainsi: