Qualité Sécurité Environnement

Introduction à la cryptographie

et

Application à la sécurité informatique

Toulouse, 22 & 23 septembre 2014

Je me présente...

- Jean-Baptiste ANGELELLI
- Ingénieur (École Centrale de Paris 2005)
- Architecte logiciel au Grand Port Maritime de Marseille

Au menu :

- Les bases de la cryptographie
- Les fonctions de hachage
- Les certificats électroniques
- Utilisation des certificats électroniques
- Les mathématiques de la cryptographie
- L'environnement de la sécurité informatique

Qu'est ce que la cryptographie ?

L'art et la science du chiffrement

Une petite mais cruciale partie de la sécurité informatique

La cryptographie à la Jules César

- Essayons de chiffrer “julius caesar”

j	u	l	i	u	s		c	a	e	s	a	r

- On utilise une table de correspondance

A	C	E	I	J	L	R	S	U	space
K	Z	U	O	B	space	C	L	R	H

- Et on remplace la lettre par la lettre qui lui correspond

b	r		o	r	l	h	z	k	u	l	k	c

- Et on obtient “br orlhzkulkc” !

Principes de bases

- **L'algorithme**, c'est le mécanisme à éxecuter pour chiffrer et déchiffrer.

- **La clé**, c'est l'élément à fournir à l'algorithme en plus du clair pour obtenir le chiffré, ou en plus du chiffré pour obtenir le clair

Le vocabulaire de la cryptographie

- Essayons de chiffrer “julius caesar”

j	u	l	i	u	s		c	a	e	s	a	r

- On utilise une table de correspondance

A	C	E	I	J	L	R	S	U	space
K	Z	U	O	B	space	C	L	R	H

- Et on remplace la lettre par la lettre qui lui correspond

b	r		o	r	l	h	z	k	u	l	k	c

- Et on obtient “br orlhzkulkc” !

Le clair

La clé

L'algorithme

Le chiffré

Les attaques

- Le modèle *Chiffré seul*
- Le modèle *Clair connu*
- Le modèle *Clair choisi*
- Le modèle *Chiffré choisi*

Nos amis

Alice

Bob

Sauf que...

Charlie

Alice

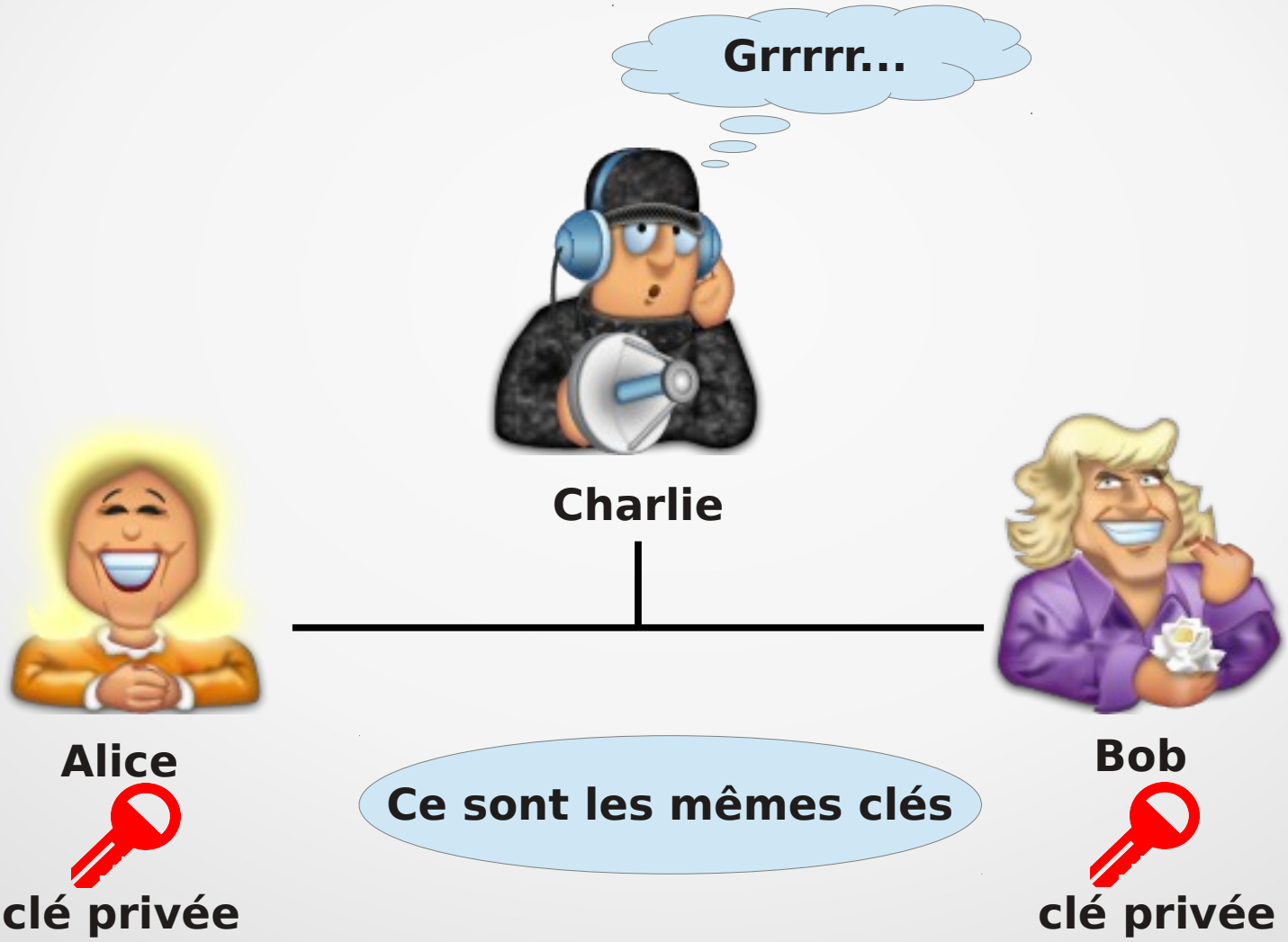
Bob

Principes de base

- En cryptographie, on part du principe que toutes les communications sont interceptées.
- Empêcher Charlie d'accéder à la ligne n'est pas l'objet de la cryptographie.

Cryptographie symétrique

- Également appelée “à clé privée”



Cryptographie symétrique

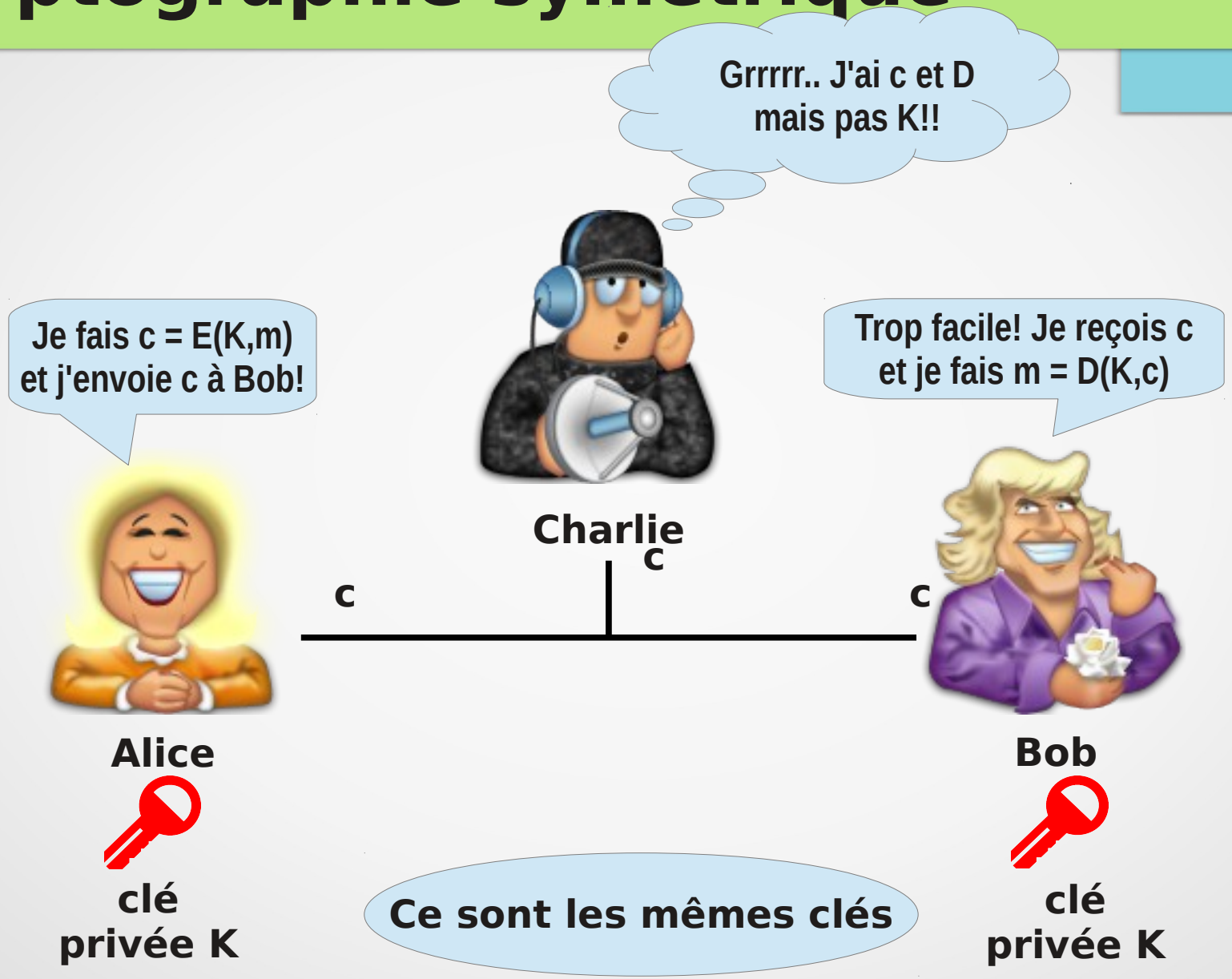
- Soit ***m*** le message ***clair***
- Soit ***c*** le message ***chiffré***
- Soit ***K*** la *clé*
- Soit ***E*** la fonction de chiffrement, on a :

$$c = E(K, m)$$

- Soit ***D*** la fonction de déchiffrement, on a **facilement** : $m = D(K, c)$

  mais **difficilement** : $m = D(c)$

Cryptographie symétrique



Cryptographie symétrique

Quelques algorithmes de cryptographie symétrique :

- Advanced Encryption System (AES)
- Triple Data Encryption Algorithm (3DES)
- One time pad
- Twofish

Cryptographie symétrique

Moi, je vois quelques problèmes dans ce processus..

Lesquels ??

Principes de base

<u>Principe de Kerckhoff</u> :

La clé est secrète, pas l'algorithme, ni son implémentation !!

Principes de base

L'algorithme doit être public car :

- premièrement, il n'est pas possible de le garder secret, donc pas question de baser une partie de la sécurité là-dessus.
- deuxièmement, la publication permet la revue critique : plus il y a de revues critiques, plus l'algorithme est sûr !

Principes de base

L'implémentation doit être publique car :

- premièrement, pour être sûr que le logiciel fait ce qu'il est supposé faire.
- deuxièmement, la publication permet la revue critique : plus il y a de revues critiques, plus l'implémentation est sûre !

Cryptographie asymétrique “à clé publique”

- Le plus souvent appelée cryptographie à clé publique

Charlie

Alice

Bob

clé publique de Bob

clé privée de Bob

Cryptographie asymétrique “à clé publique”

- Les messages sont ***chiffrés*** au moyen de la clé ***publique***.
- Les messages sont ***déchiffrés*** au moyen de la clé ***privée***.

Cryptographie asymétrique “à clé publique”

Génial !! Je distribue ma clé publique à tout le monde! Ils l'utilisent pour chiffrer les messages qu'ils m'envoient, et je suis le seul à pouvoir les déchiffrer avec ma clé privée!

Bob

clé publique de Bob

clé privée de Bob

Cryptographie asymétrique “à clé publique”

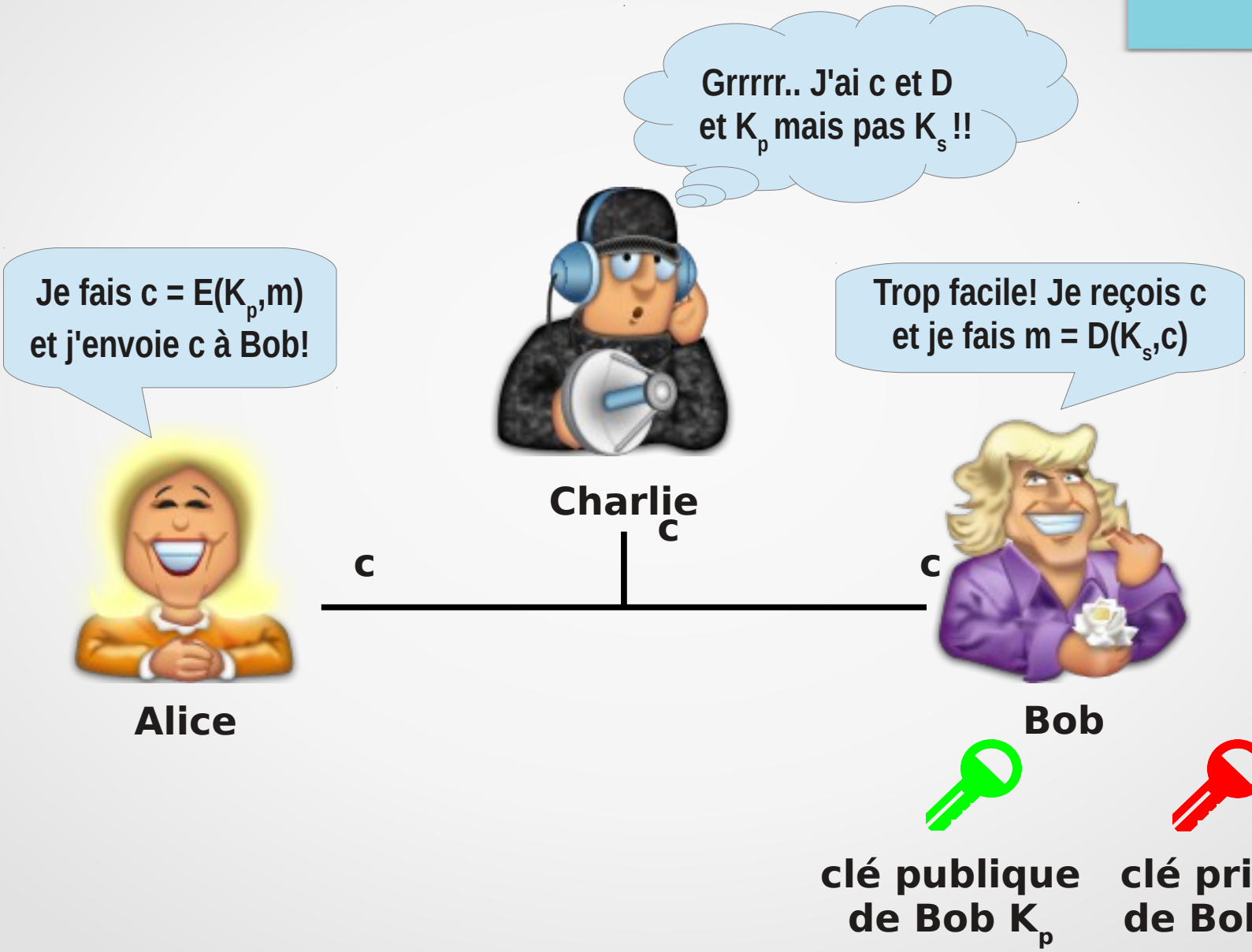
- Soit ***m*** le message ***clair***
- Soit ***c*** le message ***chiffré***
- Soit K_p la ***clé publique***
- Soit K_s la ***clé privée***
- Soit ***E*** la fonction de chiffrement, on a :

$$c = E(K_p, m)$$

- Soit ***D*** la fonction de déchiffrement, on a **facilement** : $m = D(K_s, c)$

  mais **difficilement** : $m = D(K_p, c)$

Cryptographie asymétrique “à clé publique”



Cryptographie asymétrique “à clé publique”

Quelques algorithmes de cryptographie à clé publique :

- RSA Algorithm
- Digital Signature Algorithm
- Cramer – Shoup

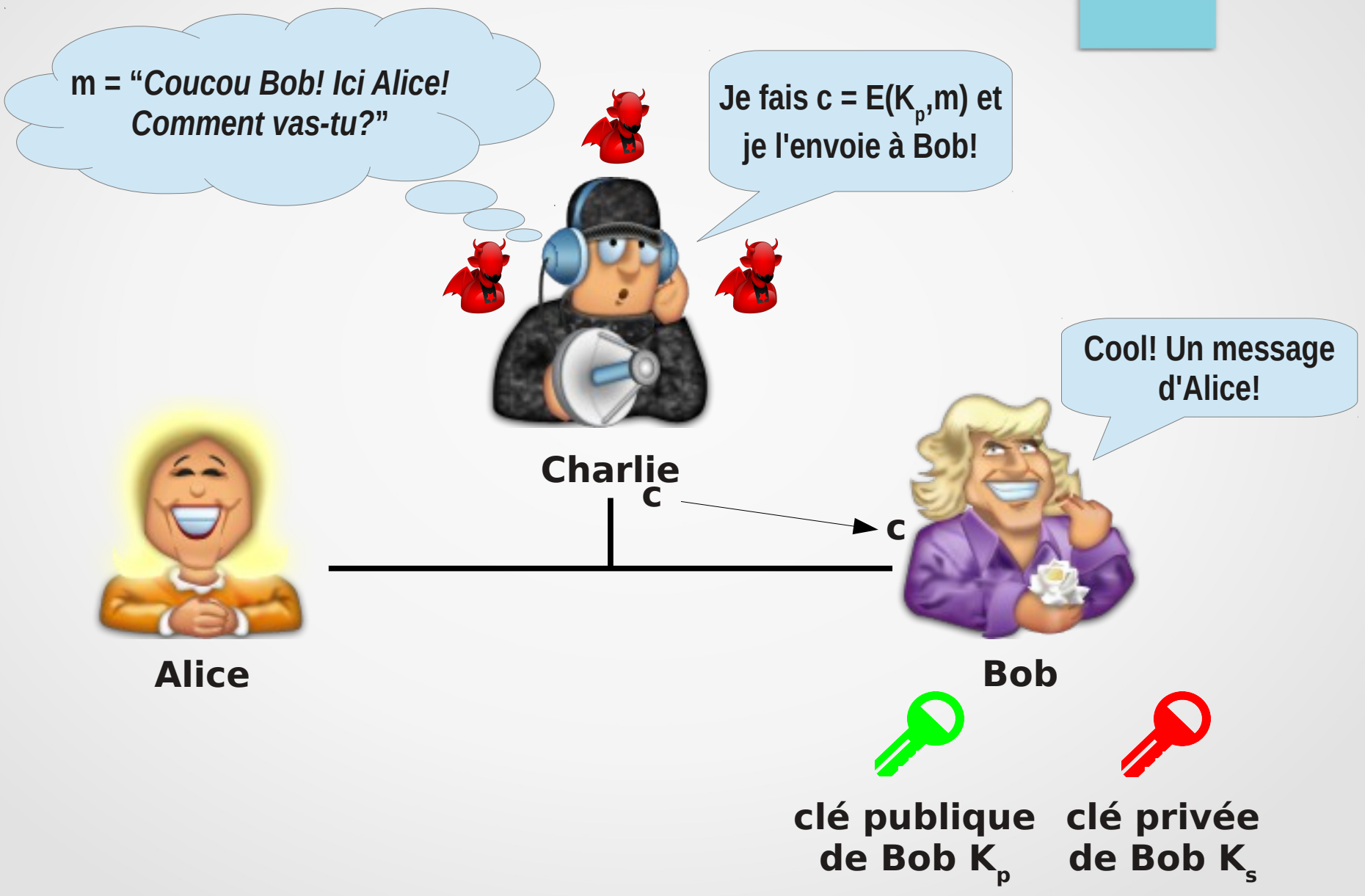
Cryptographie asymétrique “à clé publique”

Moi, je vois quelques problèmes dans ce processus..

Lesquels ??

Cryptographie asymétrique : signature numérique

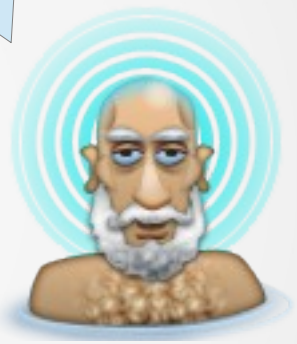
Cryptographie asymétrique : signature numérique



Cryptographie asymétrique :
signature numérique

Tout le monde peut m'envoyer des messages chiffrés maintenant, comment je peux être sûr que celui-ci vient bien d'Alice ???

Demande lui de *signer* ses messages



Comment faire ?

Fonctions de hachage

- Une fonction de hachage prend en entrée un objet, par exemple un texte, et produit en sortie une *empreinte* ou *condensat* (ou *digest*, ou *hash*) de cet objet.
- Les objets en entrée sont de tailles variables mais les empreintes ont toutes la même longueur.

Exemples de fonctions de hachage : MD5

- MD5 (Message Digest 5)
- Publiée en avril 1992 (Ron Rivest)
- Produit des empreintes de 128 bits, typiquement présentées sous la forme d'une série de 32 caractères héxadecimaux

Exemples de fonctions de hachage : SHA-1

- SHA-1 (Secure Hashing Algorithm)
- Publiée en 1995 (National Security Agency)
- Produit des empreintes de 160 bits, typiquement présentées sous la forme d'une série de 40 caractères héxadecimaux

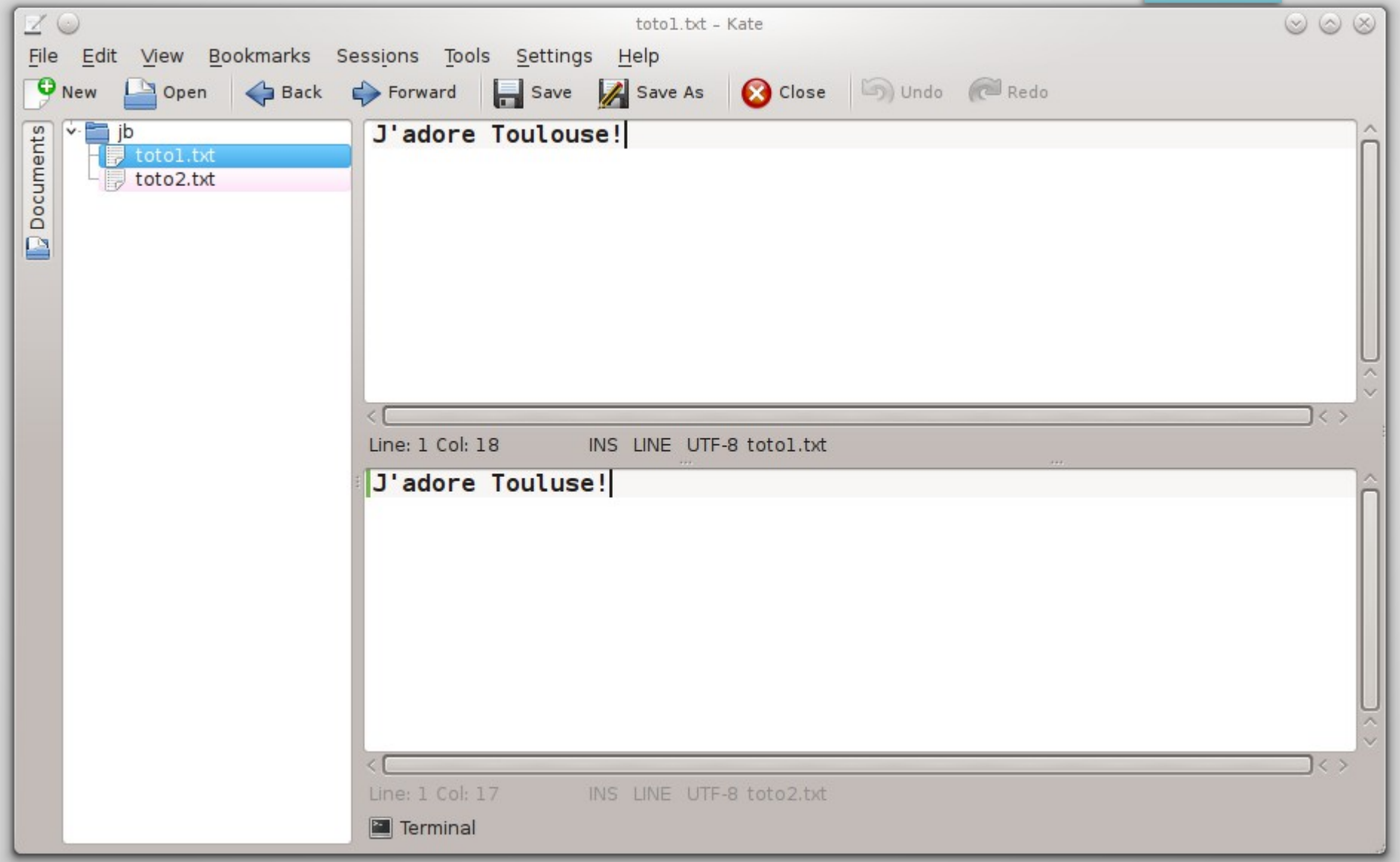
Exemples de fonctions de hachage : SHA-256

- SHA-256 (famille SHA-2)
- Publiée en 2001 (National Security Agency)
- Produit des empreintes de 256 bits, typiquement présentées sous la forme d'une série de 64 caractères héxadecimaux

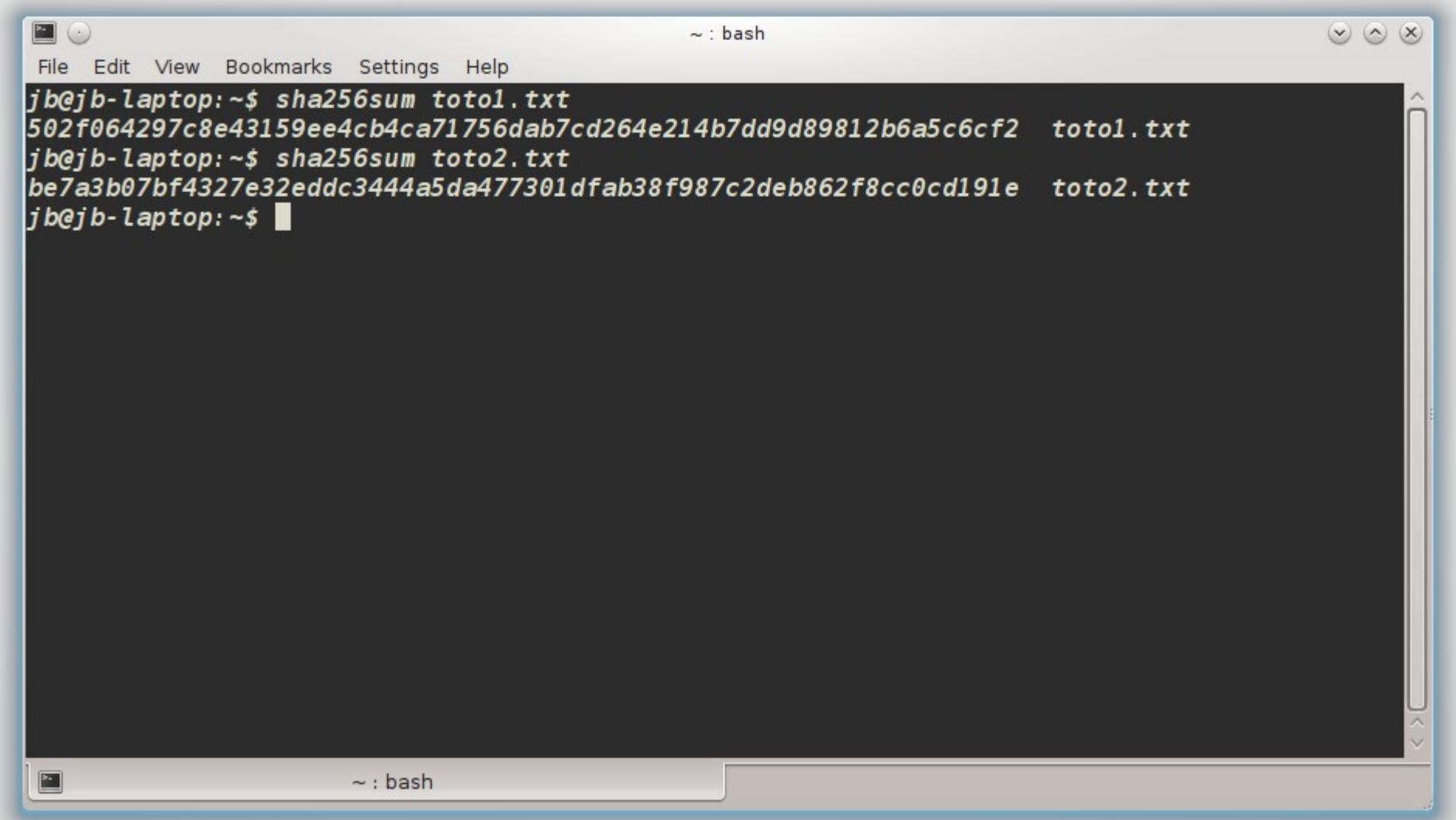
Exemples de fonctions de hachage : SHA-3

- SHA-3 (également connu comme Keccak)
- Publiée en 2012 (Bertoni, Daemen, Peeters, Van Assche)
- Vainqueur d'un concours du NIST
- Produit des empreintes de 1600 bits

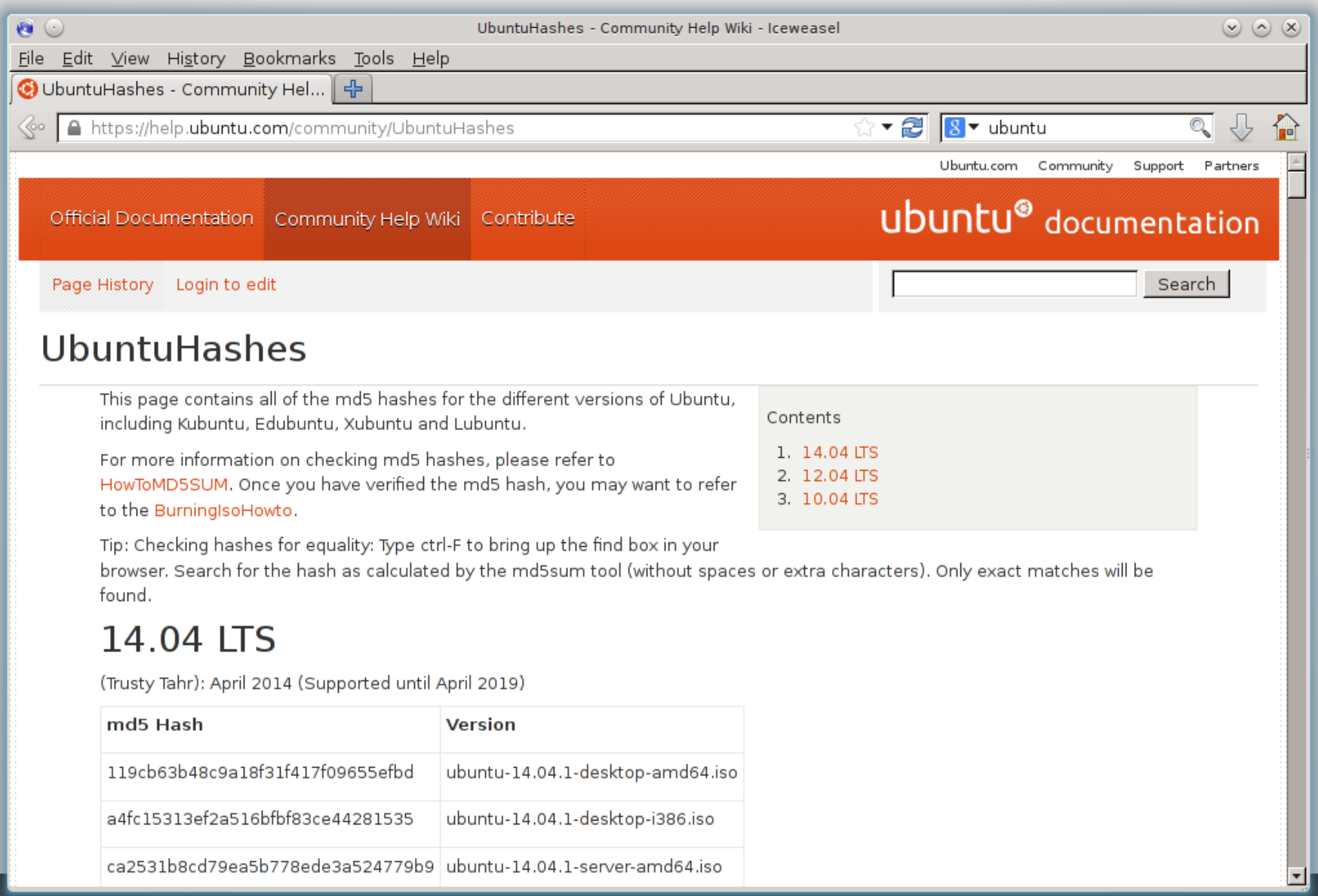
Exemples de fonctions de hachage : SHA-256



Exemples de fonctions de hachage : SHA-256



Exemples de fonctions de hachage : MD-5



Fonctions de hachage

Une bonne fonction de hachage pour la cryptographie doit avoir les propriétés suivantes :

- Un changement minime sur l'objet en entrée résulte en une empreinte très différente
- Il est très difficile d'obtenir des informations sur l'objet en entrée à partir de l'empreinte
- MD-5 et SHA-1 sont considérées non fiables
- Aucune attaque connue à ce jour sur SHA-256

Cryptographie asymétrique :
signature numérique

Le principe de la signature est de prouver qu'un message a été émis par une personne et n'a pas été modifié après avoir été signé.

Cryptographie asymétrique : signature numérique

Alice

clé publique d'Alice K_p^{Alice}

clé privée d'Alice K_s^{Alice}

- Pour ***signer*** le message Alice va :
  - générer une empreinte de son message : h = sha256(m)
  - chiffrer l'empreinte avec sa ***clé privée*** : $S = E(K_s^{Alice}, h)$
  - envoyer S à Bob en même temps que son message chiffré (ou non)

S est la signature !

Cryptographie asymétrique : signature numérique

Alice

clé publique d'Alice K_p^{Alice}

clé privée d'Alice K_s^{Alice}

- Pour ***signer*** le message Alice va :
  - générer une empreinte de son message : h = sha256(m)
  - chiffrer l'empreinte avec sa ***clé privée*** : $S = E(K_s^{Alice}, h)$
  - envoyer S à Bob en même temps que son message chiffré (ou non)

S est la signature !

Cryptographie asymétrique : signature numérique

Bob

- Pour ***vérifier*** la signature Bob va :
  - récupérer le message m' (en le déchiffrant si nécéssaire)
  - récupérer la signature S
  - générer une empreinte de son message : h' = sha256(m')
  - déchiffrer la signature avec la ***clé publique d'Alice*** : $V = D(K_p^{Alice}, S)$
  - comparer h' et V. Si h' = V, c'est à dire

    $sha256(m') = D(K_p^{Alice}, E(K_s^{Alice}, sha256(m)))$,

    alors Bob est *en pratique* sûr que m' = m

Cryptographie asymétrique : signature numérique

- Il se pourrait que m ≠ m' et sha256(m) = sha256(m') mais la probabilité est très faible.
- Avec une bonne fonction de hachage cryptographique et sans la clé privée d'Alice, il est impossible à Charlie de remplacer à la fois m par m' et S par $S'=E(K_s^{Alice}, sha256(m'))$.
- Connaissant K_p^{Alice}, Charlie peut retrouver facilement sha256(m) en faisant $D(K_p^{Alice}, S)$, mais il ne peut remonter à m.

Cryptographie asymétrique :
signature numérique

La clé privée étant nécéssaire pour signer un message, Charlie peut toujours modifier le message, mais il ne peut pas le ***resigner*** !

Cryptographie asymétrique :
signature numérique

C'est ce mécanisme qui est utilisé pour réaliser ***l'authentification***

Cryptographie asymétrique :
En résumé

Pour écrire un message chiffré à Bob, Alice utilise :

la clé publique de Bob

Pour déchiffrer un message reçu d'Alice, Bob utilise :

la clé privée de Bob

Normal, je veux que tout le monde puisse m'écrire mais être le seul à déchiffrer mes messages !

Cryptographie asymétrique :
En résumé

Pour signer un message, Alice utilise :

la clé privée d'Alice (+ hachage)

Pour vérifier une signature d'Alice, Bob utilise:

la clé publique d'Alice (+ hachage)

Normal, je signe avec quelque chose qui n'appartient qu'à moi seule, mais tout le monde doit pouvoir vérifier ma signature !

Cryptographie asymétrique “à clé publique”

Moi, je vois toujours quelques problèmes dans ce processus..

Lesquels ??

Cryptographie asymétrique “à clé publique”

Je chiffre mes messages à Bob avec sa clé publique K_p^{Bob}, mais est-ce vraiment la sienne?

Alice

Je vérifie les messages signés d'Alice avec sa clé publique K_p^{Alice}, mais est-ce vraiment la sienne?

Bob

Cryptographie asymétrique “à clé publique”

Je vais envoyer ma clé publique $K_p^{Charlie}$ à Alice et Bob en leur faisant croire à chacun que c'est la clé publique de l'autre.
Je pourrais donc déchiffrer leur messages et leur envoyer des faux messages signés avec ma clé privée $K_s^{Charlie}$!

C'est l'attaque de l'homme du milieu!!!

Les certificats électroniques

Les certificats électroniques permettent d'associer de façon fiable une ***clé publique*** à une ***entité*** (personne, entreprise, serveur...)

Les certificats électroniques

Un certificat électronique est un fichier signé qui contient essentiellement :

- des informations sur l'entité
- la clé publique de l'entité

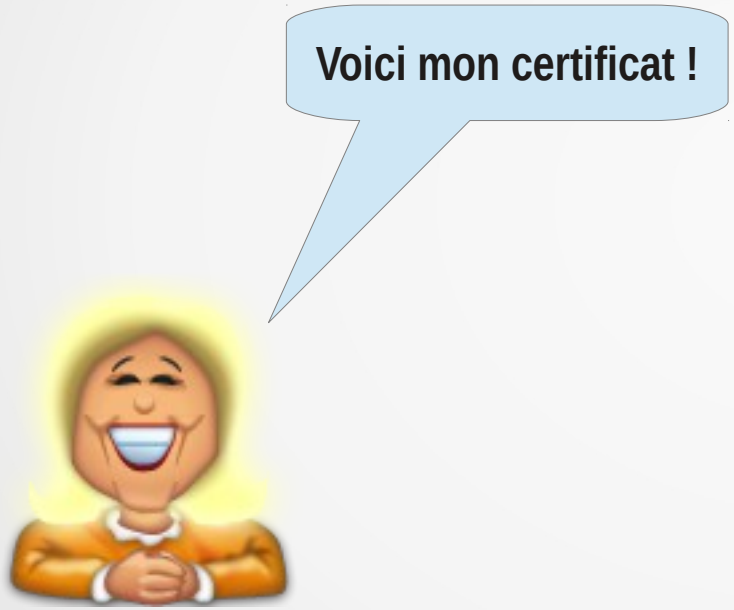
plus :

- l'identifiant de l'émetteur
- une date de validité
- un numéro de série

et surtout :

- la signature de l'émetteur

Les certificats électroniques



Alice

Nom : Alice

Clé publique : 24E1F2893A

Emis par : SuperSign Corp.

Valide jusqu'au : 23/09/2024

N° de série : 56882442

Signature :

SuperSign Corp.

Les certificats électroniques

Une minute! J'ai un certificat certifiant la clé publique d'Alice, signé par la clé privée de SuperSign Corp.

Je vais vérifier cette signature avec la clé publique de SuperSign Corp. Mais le problème recommence!

Qui me garantit que j'ai bien la clé publique de SuperSign Corp. ?
Il me faut le certificat de SuperSign Corp !!!

Les certificats électroniques

Nom : SuperSign Corp.

Clé publique : 962BF283

Emis par : SuperSign Corp.

Valide jusqu'au : 23/09/2024

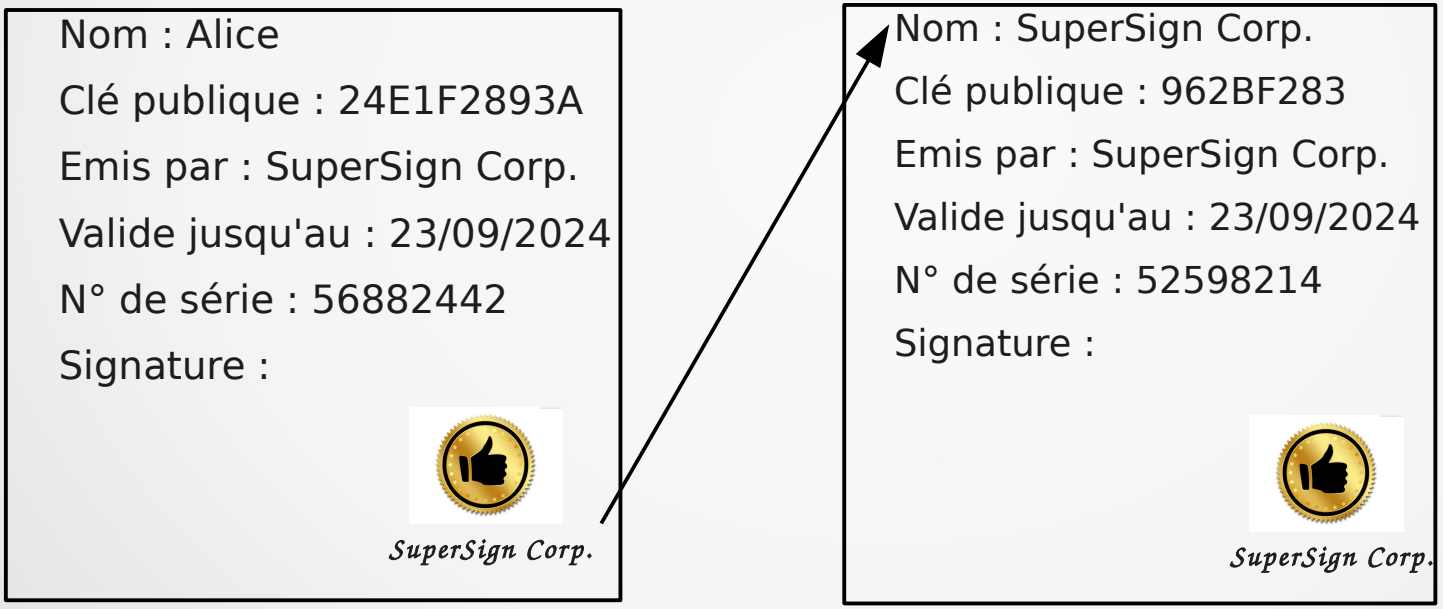
N° de série : 52598214

Signature :

SuperSign Corp.

Les certificats électroniques

- On obtient un ***chaîne de certificats*** !
- Le certificat de SuperSign Corp. est ***auto-signé*** !



Les certificats électroniques

SuperSign Corp. auto-signé ??
Ca veut dire que je suis supposé lui faire confiance ?
Sur parole ?

Les certificats électroniques

SuperSign Corp. auto-signé ??
Ca veut dire que je suis supposé lui faire confiance ?
Sur parole ?

Enchanté ! Je travaille pour SuperSign Corp.
Nous sommes une *autorité de certification*.
Nous sommes connus et fiables,
tellement fiables que nous figurons
par défaut dans tous les navigateurs web!

SuperSign Corp.

Les certificats électroniques : en résumé

Un certificat électronique certifie une clé publique, il est signé par (la clé privée de) l'émetteur, typiquement une autorité de certification de confiance.

Les certificats électroniques : en résumé

Une clé publique est sans aucune utilité si elle n'est pas associée de façon sûre à son propriétaire par un certificat.

En pratique, on ne rencontre jamais une clé publique sans son certificat.

Les certificats électroniques : en résumé

Les clés publiques sont tellement fortement associées au certificats qu'on dit souvent :

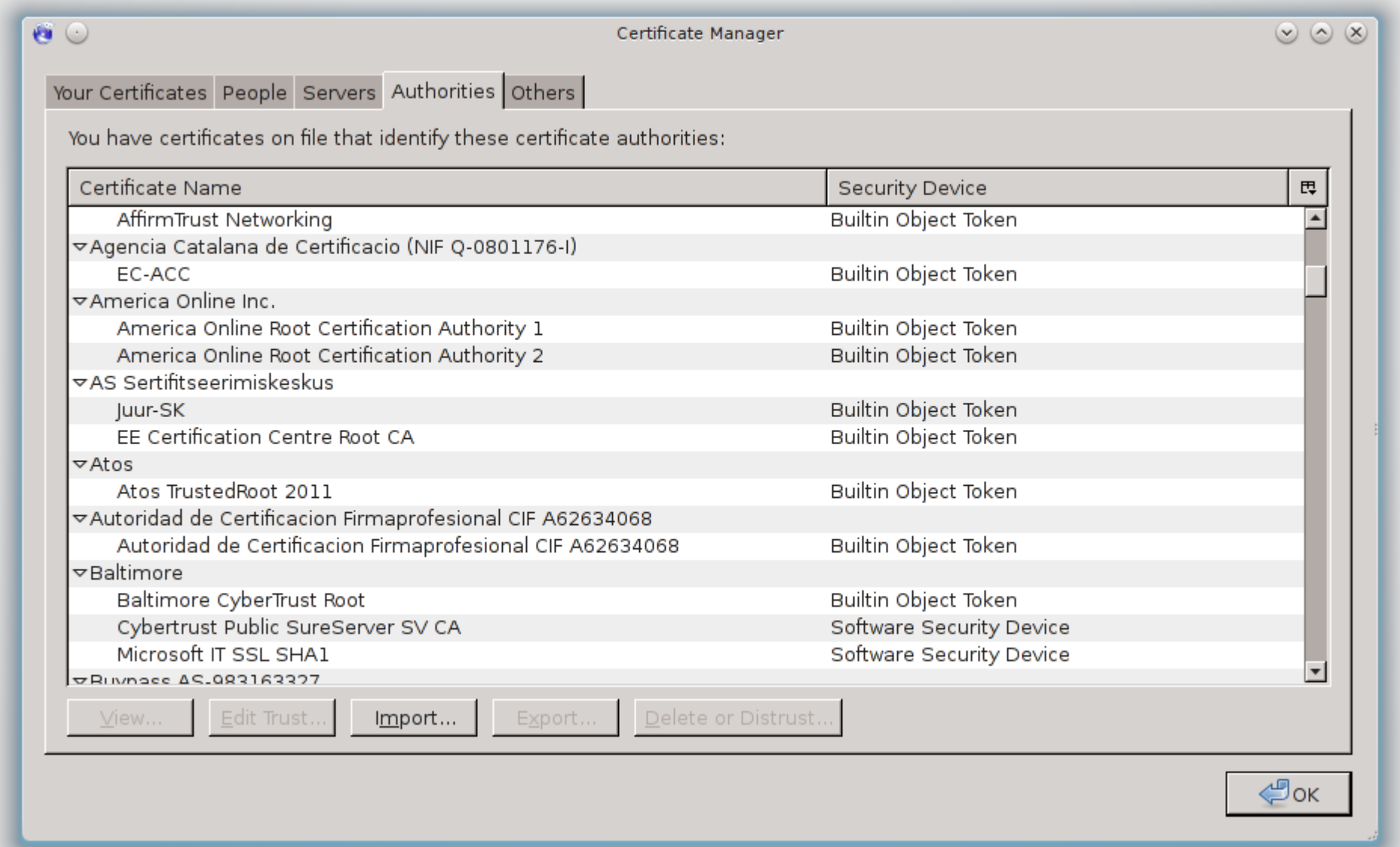
- *“Chiffré avec le certificat de Bob”* pour dire *“Chiffré avec la clé publique qui se trouve dans le certificat de Bob”*.
- *“Déchiffré avec le certificat de Bob”* pour dire *“Déchiffré avec la clé privée associée à la clé publique qui se trouve dans le certificat de Bob”*.

Les certificats électroniques : en résumé

Les clés publiques sont tellement fortement associées au certificats qu'on dit souvent :

- *“Signé avec le certificat d'Alice”* pour dire *“Signé avec la clé privée associée à la clé publique qui se trouve dans le certificat d'Alice”*.
- *“Signature vérifiée avec le certificat d'Alice”* pour dire *“Signature vérifiée avec la clé publique qui se trouve dans le certificat d'Alice”*.

Les certificats électroniques

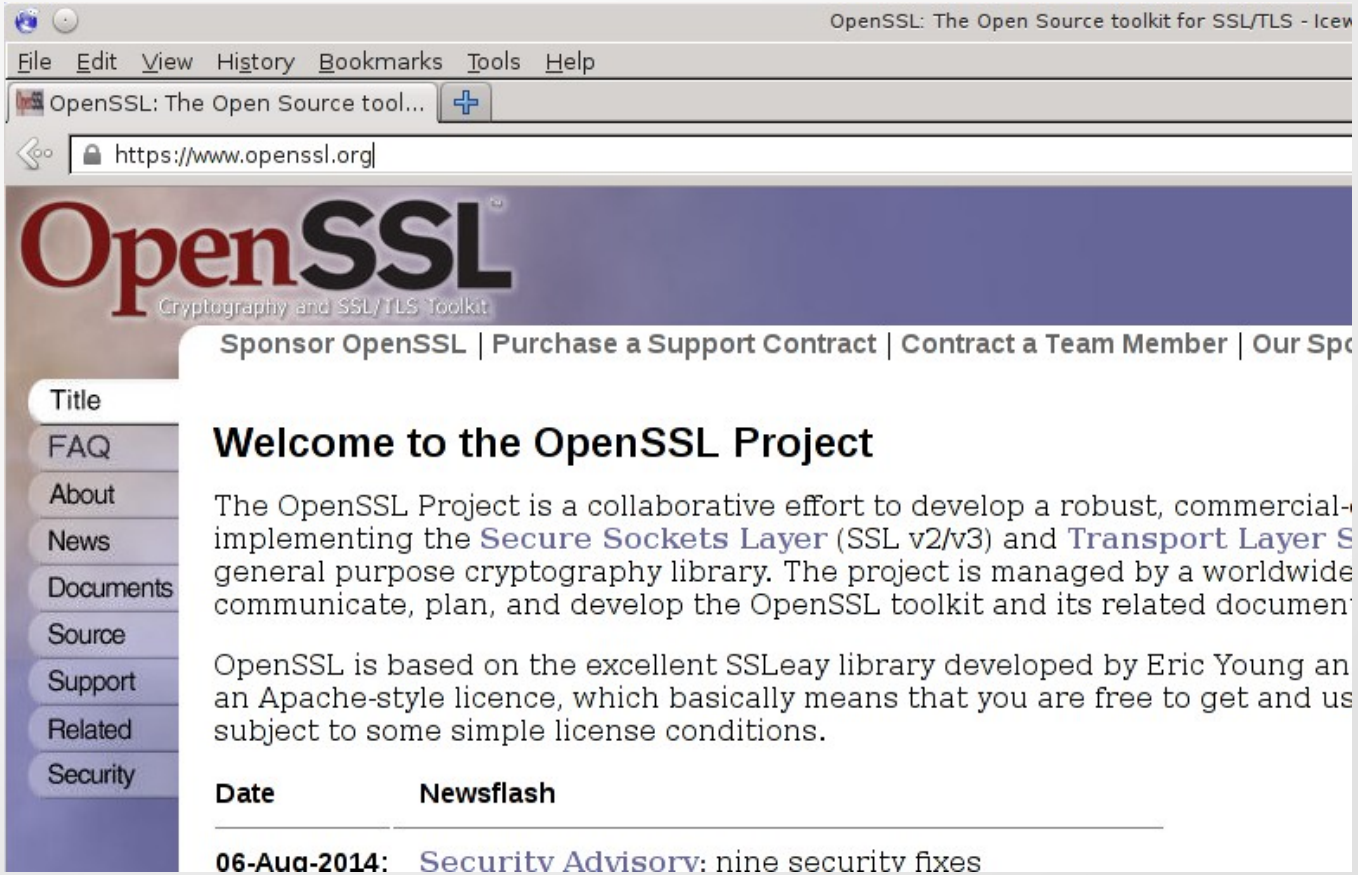


Les certificats électroniques

En pratique, l'autorité de certification sépare souvent la fonction racine (*root)* de la fonction signante (*signing)*

Création d'une autorité de certification

- Nous utilisons l'outil OpenSSL
- https://www.openssl.org/



Création d'une autorité de certification

- La première étape consiste à :
  - Générer la clé privée de l'autorité racine
  - Générer la clé publique de l'autorité racine
  - Générer la *Certificate Signing Request* de l'autorité racine

Création d'une autorité de certification

rootCA.conf – Kate

File Edit View Bookmarks Sessions Tools Settings Help

New Open Back Forward Save Save As Close Undo Redo

Documents

rootCA
rootCA.conf

```
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extens

[ ca_dn ]
0.domainComponent       = "home.lan"
1.domainComponent       = "supersign"
organizationName        = "SuperSign Corp."
commonName              = "SuperSign Root Certificate Authority"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash

# The remainder of the configuration file is used by the openssl ca comm
# The CA section defines the locations of CA assets, as well as the poli
# applying to the CA.

[ ca ]
default_ca              = root_ca               # The default CA

[ root_ca ]
certificate             = $dir/$ca/$ca.crt      # The CA cert
private_key             = $dir/$ca/private/$ca.key  # CA private key
new_certs_dir           = $dir/$ca/certs        # Certificate ar
serial                  = $dir/$ca/db/crt_serial  # Serial number
```

Line: 1 Col: 1 INS LINE UTF-8 rootCA.conf

Terminal

Création d'une autorité de certification

- Cela se fait en une seule étape avec OpenSSL

~/superSignCA/rootCA : bash

File Edit View Bookmarks Settings Help

```
jb@jb-laptop:~/superSignCA/rootCA$ openssl req -new -config rootCA.conf -out rootCA.csr
 -keyout private/rootCA.key
Generating a 2048 bit RSA private key
............+++
...+++
writing new private key to 'private/rootCA.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
jb@jb-laptop:~/superSignCA/rootCA$
```

~/superSignCA/rootCA : bash

Création d'une autorité de certification

- On obtient ainsi la CSR rootCA.csr ainsi que la clé privée rootCA.key (2048 bits)

rootCA.key – Kate

File Edit View Bookmarks Sessions Tools Settings Help

New Open Back Forward Save Save As Close Undo Redo

Documents

private

rootCA.key

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFDjBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQI6f0eui/oAJwCAggA
MBQGCCqGSIb3DQMHBAiqVPJn8llE9QSCBMij9WP4Lw+NAy1bKKL++nEvvvjAAu2A
SsoV4pt3xPmtGIfw9GZymcWBcNwZXTdM2EJ6PH7TltLDqsnwKyi3RftvCs6hWrIM
kWetqlrHwleCNOQL3mW8+dPSeMERHu/tOQjgSB+EpTxfvlgTbPmkdaj1rZMb+PLg
SHO3o6EjaILVSvINskEPxJ42g6t/wFvCPKqlmNqcelr3cZkYCqPUjykvb/Xzy2SK
pjdVyWZhmZEfphbjtFxovtiDB23uypKCG+pK/o97RMQE95SNpmiD4n+CvTAUDngq
u6XbX1WYcF22Fo+CSZR0STZTwXaf4TCmlnIyvjP5d8tLVUliHtcGasXVkNWID3yb
6otwR9rqM67LpgxLntkzxnEIIfC1YptdAgooULg/qMYMzzZ8z02JSHE/KIb/FOVc
+VHJRDlYUglwjgcYmuCeaYRW0WtPAGR93k1/Y3IYKs3hW5dO7zumPxxbNU5yQRd2
ZK6ql5NbZvN8UmdCLAbTR+cj0u2N+1ESVrTAIW5HjKKenPTjlc0UqaRas2bk3Bhg
zI0mAVfFhRtJNkA4Un+0umW7Dy50TFT2PEbYPAHISV93qWN+b/fgwuZn2ndKtxkb
AmQiyVrxnyxD0L239br14zdNVy0Aapbvq6eQZFa/H0C7c3STfr9h50aFKGNVlqTw
M91vv2sKrOul2shHvV4u0Jd8vrNa5jhR5dvla7LG2LApFtWqy4VhmjPeouFTBrLT
lr3zCADC2LIYXT3GMt6nycb+aUQy9dgkUghMXCAAD9Cv1yiICUxC8mxSCRliHqst
T42MHxDPCvX7Z2RVL2yN5eq9u3w8GQBqpj9/qSDfEXKkiUUxwFklu82FNqag5KYY
tCK593o4U96o15cc8ExHoMyZZ7ZeAzDzdnZTGaSKKSKLxPqRDUp8FrGMdbsNO7g1
3sE3bplAL6qT4jx9Bfjsm/sn1kgDdHpNsQnw8B0USgRoOdn5FY89nbbYRpZ5D6dF
YfA+nGvWMu7KUchsr3t9LugYYkKZMcZ/2QP5MWu+iv1ZsfJRYSFic0ni4PGpkDfI
Y150HSJOeC42UfDGpHb5tQTfWDd2ef8/uqTperM/QE0amljinTbFnDC930kBOzu3
NbtE7kVAxl2a+drRMuM//RF7qSshqHLXimyO9xwo2JWu/NIDr9t73TJOZTvGjr8U
iEVunYWUX2ytYTWY/9xdh73ha+kC0BDC1rnlBKatHDcPFA4NXLNWCEM7giN33DPf
+hVTacf+cAwqNkTHs0WS+dcndwFqYGMke8yWLNgzThyGCvv0qsjaqL8rIX3vZiRS
Ng8K6f4Z16dIy/FKHB5Z55FHvJl79yxM0wsMZQ+QaGYGGemEeTl7Z7qH22q0rwQ0
0D/+6IJ3VU67P8k6drMnpk8PeY44599lxHbkqVBWLwQOn9BvnbcQ9V3a+FCu2xNn
np3h27bsZcihwj9eYVKii83J54llLnpM3XcD1xxMlYxhmeQQS8IY/T6VUNHqLrrh
hHzZeIJPya8K45UtuzHF2bHh4EaocTgNFbfj/Pi8IDGKjH/0C18TlZq3WouMOraK
m2PauqLgCyoM4vyx5BFeId4+J+rpO9Qe+LD8pj6leCVDRJonZXmS3rArt6nELXqM
QtE=
-----END ENCRYPTED PRIVATE KEY-----
```

Line: 1 Col: 1 INS LINE UTF-8 rootCA.key

Terminal

Création d'une autorité de certification

- Maintenant nous allons signer la CSR au moyen de la clé privée rootCA.key

~/superSignCA/rootCA : bash

File Edit View Bookmarks Settings Help

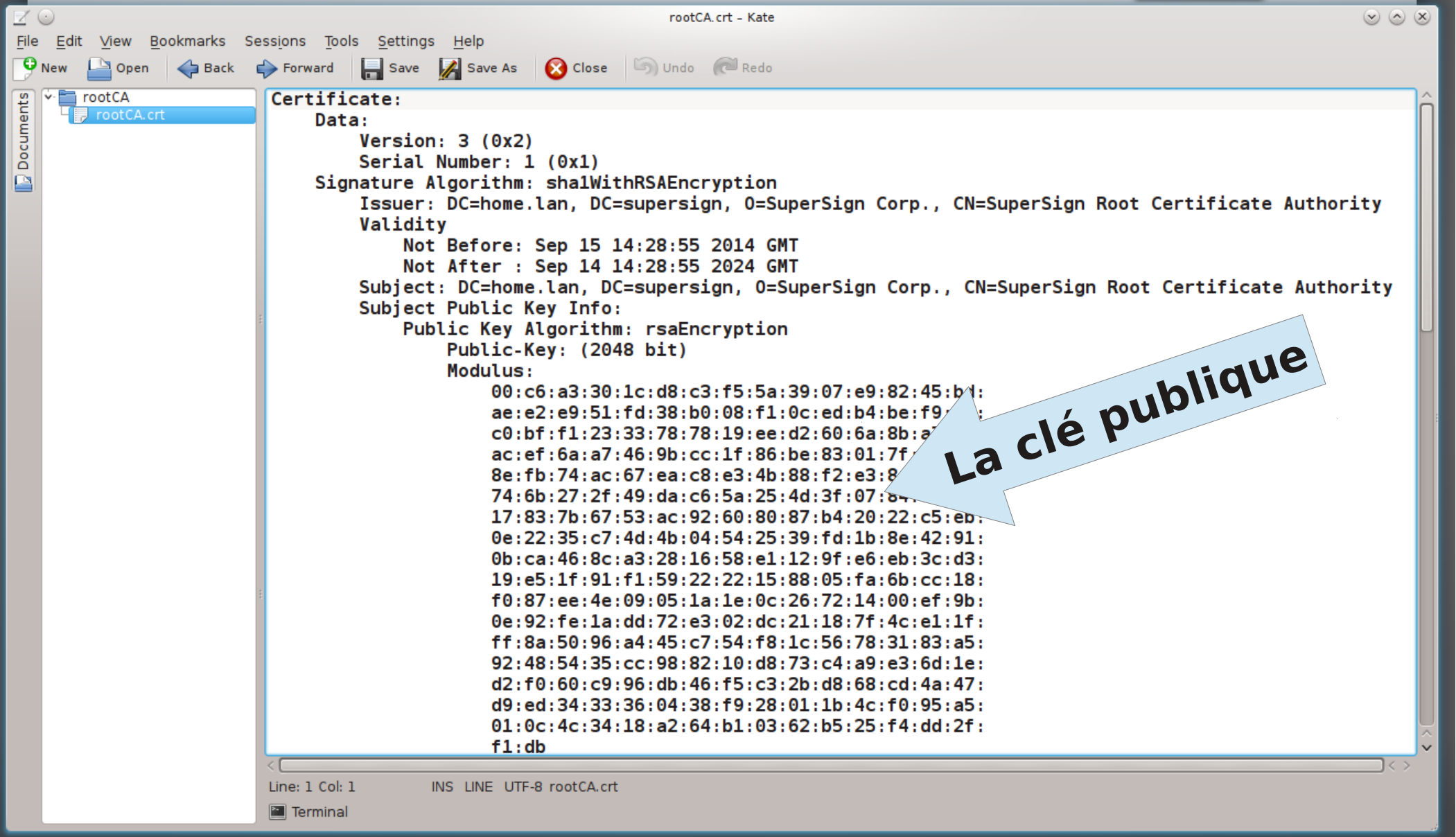
```
jb@jb-laptop:~/superSignCA/rootCA$ openssl ca -selfsign -config rootCA.conf -in rootCA.csr
-out rootCA.crt -extensions root_ca_ext
Using configuration from rootCA.conf
Enter pass phrase for /home/jb/superSignCA/rootCA/private/rootCA.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Sep 15 14:28:55 2014 GMT
            Not After : Sep 14 14:28:55 2024 GMT
        Subject:
            domainComponent           = home.lan
            domainComponent           = supersign
            organizationName          = SuperSign Corp.
            commonName                = SuperSign Root Certificate Authority
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
```

~/superSignCA/rootCA : bash

Création d'une autorité de certification

- Et on obtient notre certificat d'autorité racine auto-signé!

Création d'une autorité de certification



Création d'une autorité de certification

Création d'une autorité de certification

- La deuxième étape consiste à :
  - Générer la clé privée de l'autorité signante
  - Générer la clé publique de l'autorité signante
  - Générer la *Certificate Signing Request* de l'autorité signante

Création d'une autorité de certification

signingCA.conf – Kate

File Edit View Bookmarks Sessions Tools Settings Help

New Open Back Forward Save Save As Close Undo Redo

Documents

signingCA
signingCA.conf

```
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensic

[ ca_dn ]
0.domainComponent       = "home.lan"
1.domainComponent       = "supersign"
organizationName        = "SuperSign Corp."
commonName              = "SuperSign Signing Certificate Authority

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash

# The remainder of the configuration file is used by the openssl c
# The CA section defines the locations of CA assets, as well as th
# applying to the CA.

[ ca ]
default_ca              = signing_ca            # The default CA s

[ signing_ca ]
certificate             = $dir/$ca/$ca.crt       # The CA cert
private_key             = $dir/$ca/private/$ca.key # CA private ke
new_certs_dir           = $dir/$ca/certs           # Certificate a
serial                  = $dir/$ca/db/crt_serial # Serial number f
```

Line: 29 Col: 67 INS LINE UTF-8 signingCA.conf

Terminal

Création d'une autorité de certification

- Cela se fait en une seule étape avec OpenSSL

~/superSignCA/signingCA : bash

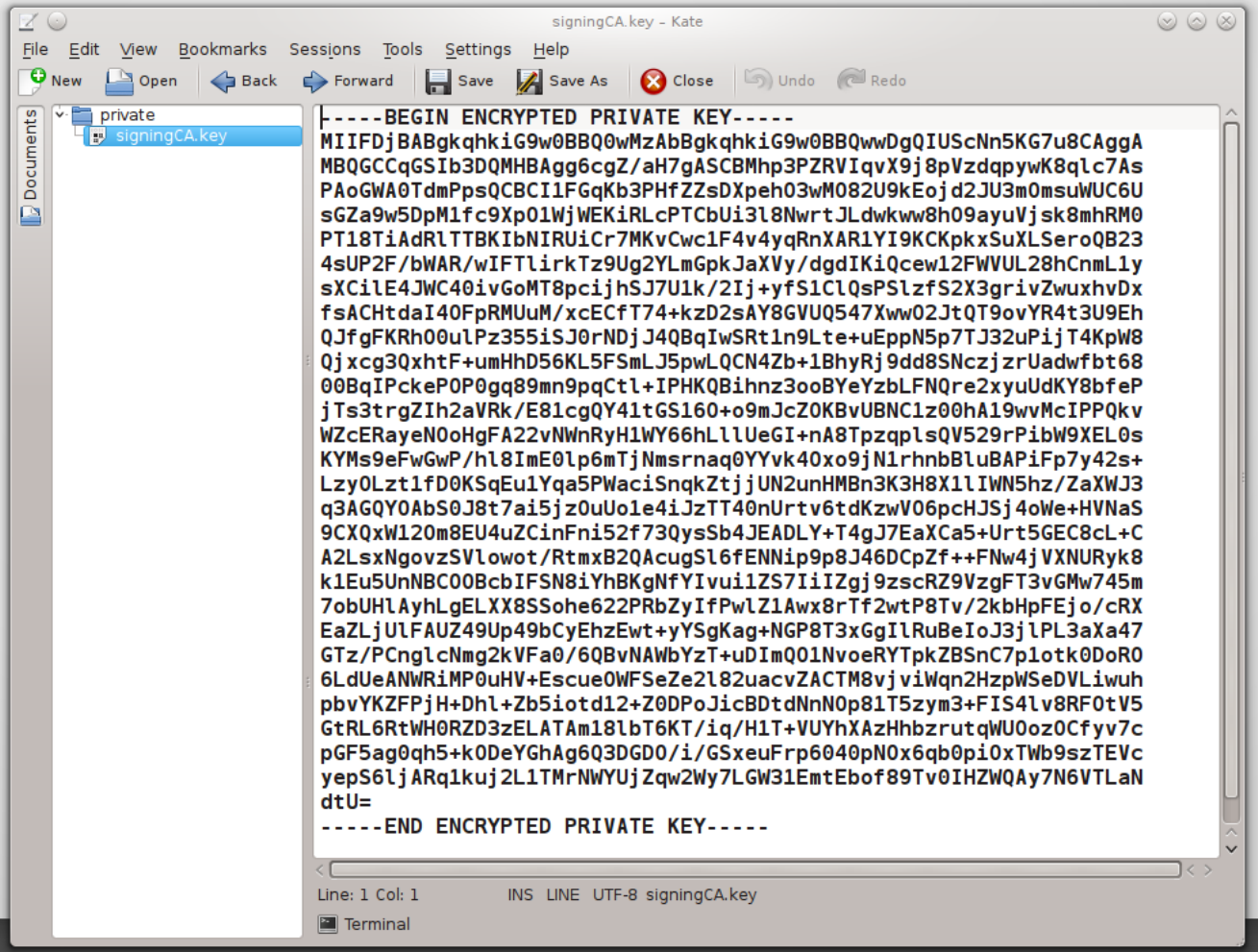
File Edit View Bookmarks Settings Help

```
jb@jb-laptop:~/superSignCA/signingCA$ openssl req -new -config signingCA.conf -out signingCA.csr
 -keyout private/signingCA.key
Generating a 2048 bit RSA private key
....+++
.............+++
writing new private key to 'private/signingCA.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
jb@jb-laptop:~/superSignCA/signingCA$
```

~/superSignCA/signingCA : bash

Creation d'une autorité de certification

- On obtient ainsi la CSR signingCA.csr ainsi que la clé privée signingCA.key (2048 bits)



Creation d'une autorité de certification

- Maintenant nous allons signer la CSR au moyen de la clé privée rootCA.key

~/superSignCA/signingCA : bash

File Edit View Bookmarks Settings Help

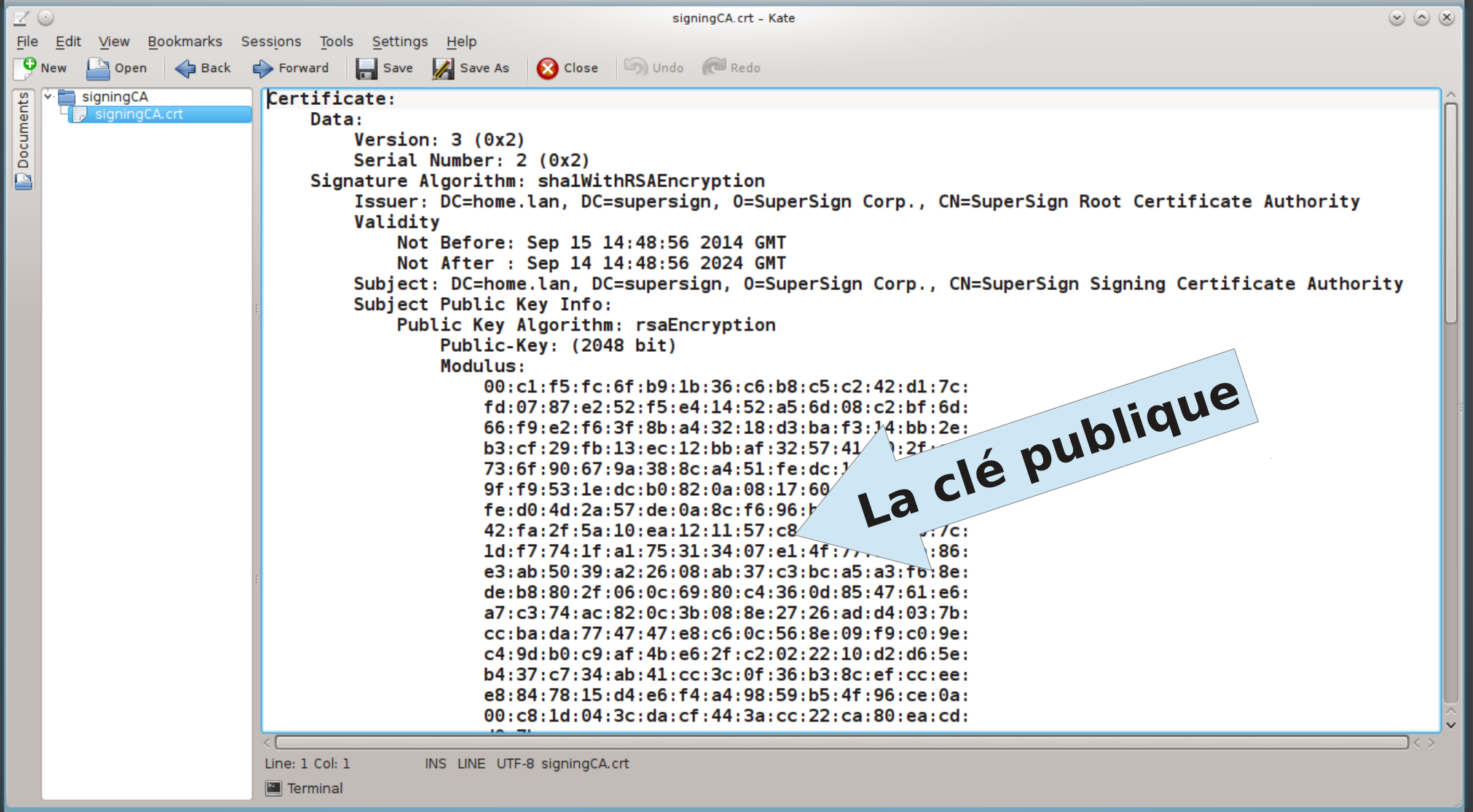
```
jb@jb-laptop:~/superSignCA/signingCA$ openssl ca -config ../rootCA/rootCA.conf -in signingCA.csr
 -out signingCA.crt -extensions signing_ca_ext
Using configuration from ../rootCA/rootCA.conf
Enter pass phrase for /home/jb/superSignCA/rootCA/private/rootCA.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 2 (0x2)
        Validity
            Not Before: Sep 15 14:48:56 2014 GMT
            Not After : Sep 14 14:48:56 2024 GMT
        Subject:
            domainComponent           = home.lan
            domainComponent           = supersign
            organizationName          = SuperSign Corp.
            commonName                = SuperSign Signing Certificate Authority
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
```

~/superSignCA/signingCA : bash

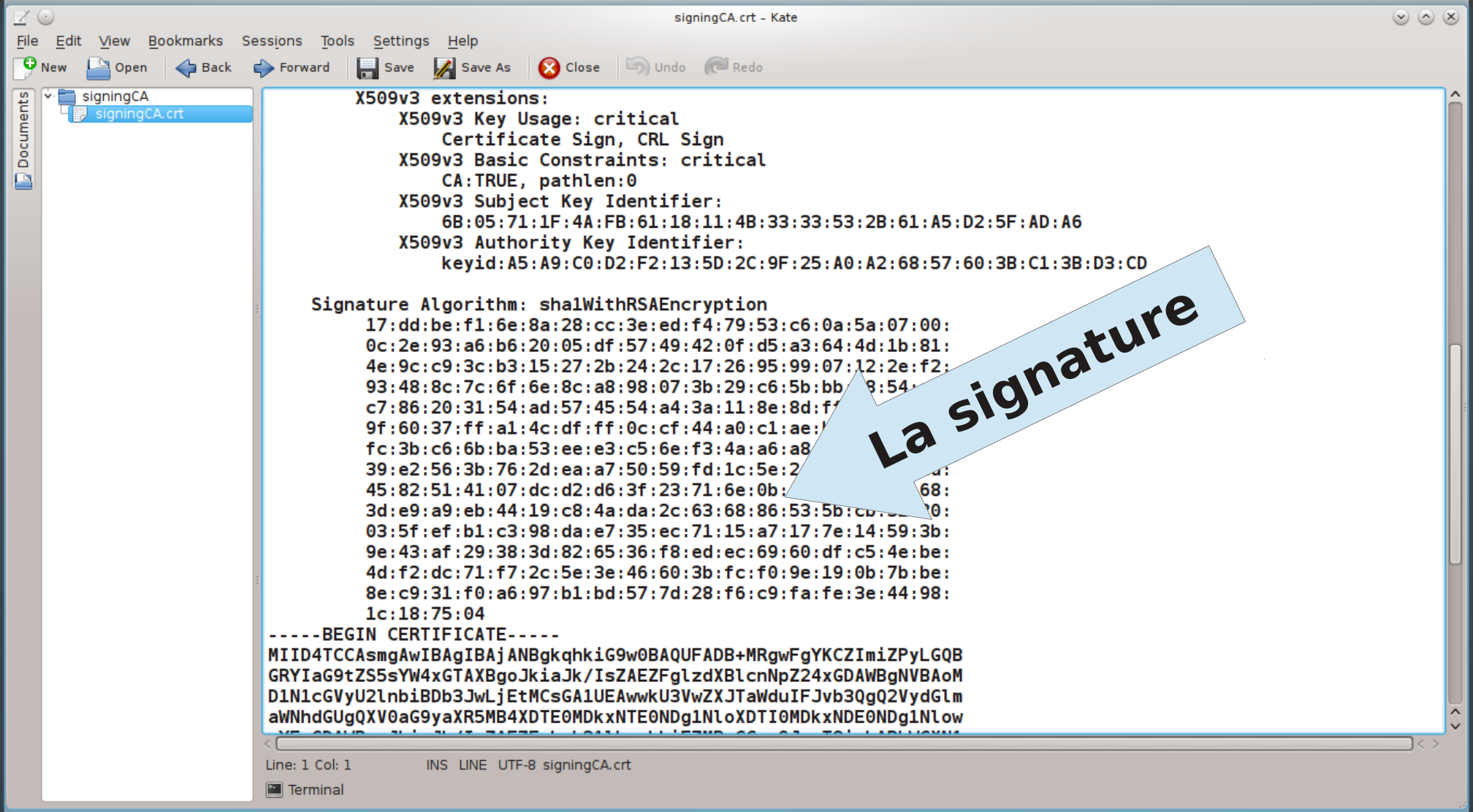
Création d'une autorité de certification

- Et on obtient notre certificat d'autorité signante signé par l'autorité racine (c'est-à-dire par la clé privée de l'autorité racine) !

Création d'une autorité de certification



Création d'une autorité de certification



Création d'une autorité de certification

- A présent nous allons :
  - Générer la clé privée de monsiteweb.home.lan
  - Générer la clé publique de monsiteweb.home.lan
  - Générer la *Certificate Signing Request* de monsiteweb.home.lan

Création d'une autorité de certification

- Cela se fait en une seule étape avec OpenSSL

~/superSignCA/server : bash

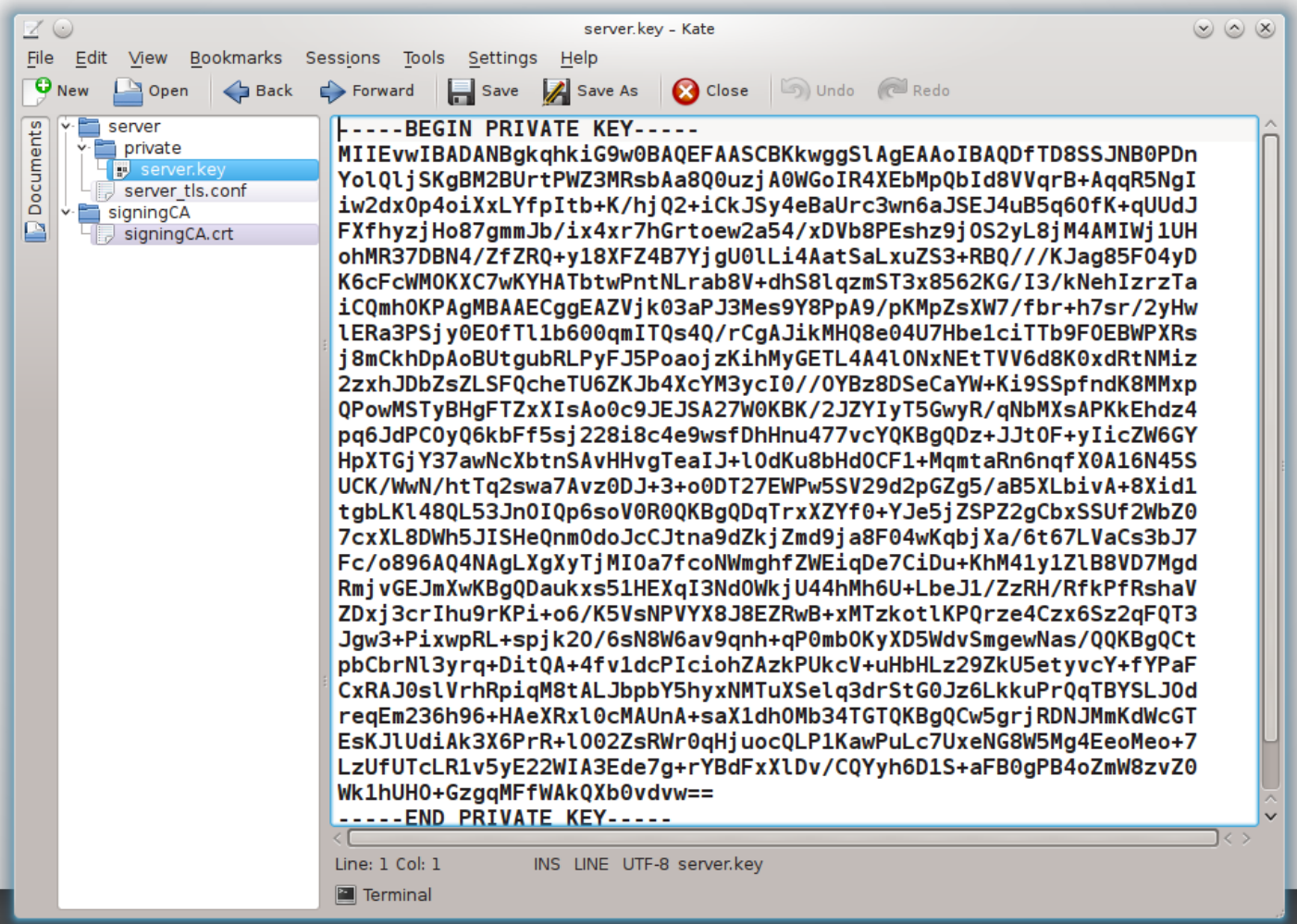
File Edit View Bookmarks Settings Help

```
jb@jb-laptop:~/superSignCA/server$ openssl req -new -config server_tls.conf -out requests/server.csr
-keyout private/server.key
Generating a 2048 bit RSA private key
.....................................................................................................
.................+++
.....................................................+++
writing new private key to 'private/server.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
1. Domain Component         (eg, com)       []:home.lan
2. Domain Component         (eg, company)   []:supersign
4. Organization Name        (eg, company)   []:SuperSign Corp.
6. Common Name              (eg, full name) []:monsiteweb.home.lan
jb@jb-laptop:~/superSignCA/server$
```

~/superSignCA/server : bash

Création d'une autorité de certification

- On obtient ainsi la CSR server.csr ainsi que la clé privée server.key (2048 bits)



Création d'une autorité de certification

- Maintenant nous allons signer la CSR au moyen de la clé privée signingCA.key

~/superSignCA/server : bash

File Edit View Bookmarks Settings Help

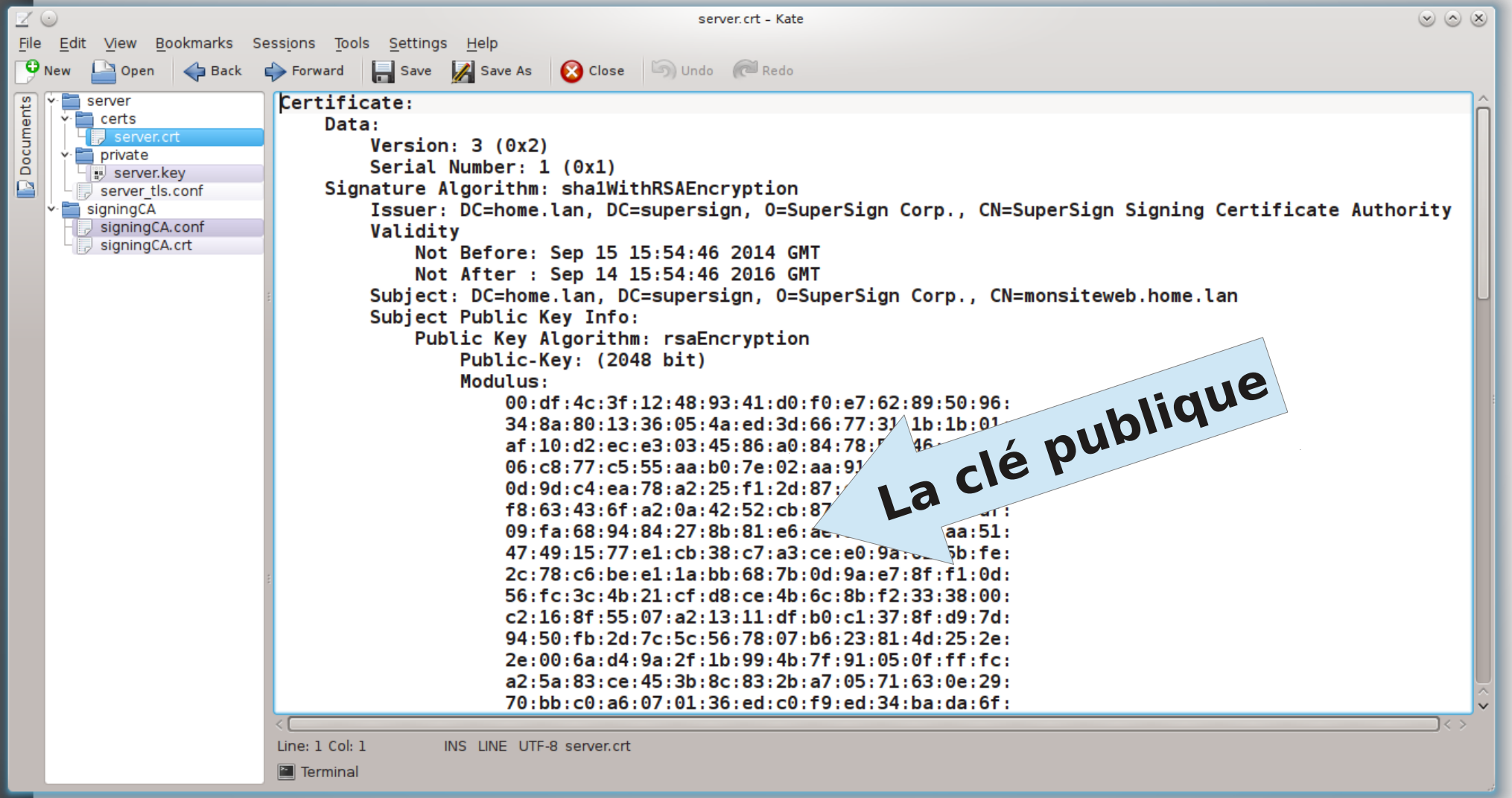
```
jb@jb-laptop:~/superSignCA/server$ openssl ca -config ../signingCA/signingCA.conf -in requests/server.csr
 -out certs/server.crt -extensions server_ext
Using configuration from ../signingCA/signingCA.conf
Enter pass phrase for /home/jb/superSignCA/signingCA/private/signingCA.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Sep 15 15:54:46 2014 GMT
            Not After : Sep 14 15:54:46 2016 GMT
        Subject:
            domainComponent           = home.lan
            domainComponent           = supersign
            organizationName          = SuperSign Corp.
            commonName                = monsiteweb.home.lan
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Extended Key Usage:
```

~/superSignCA/server : bash

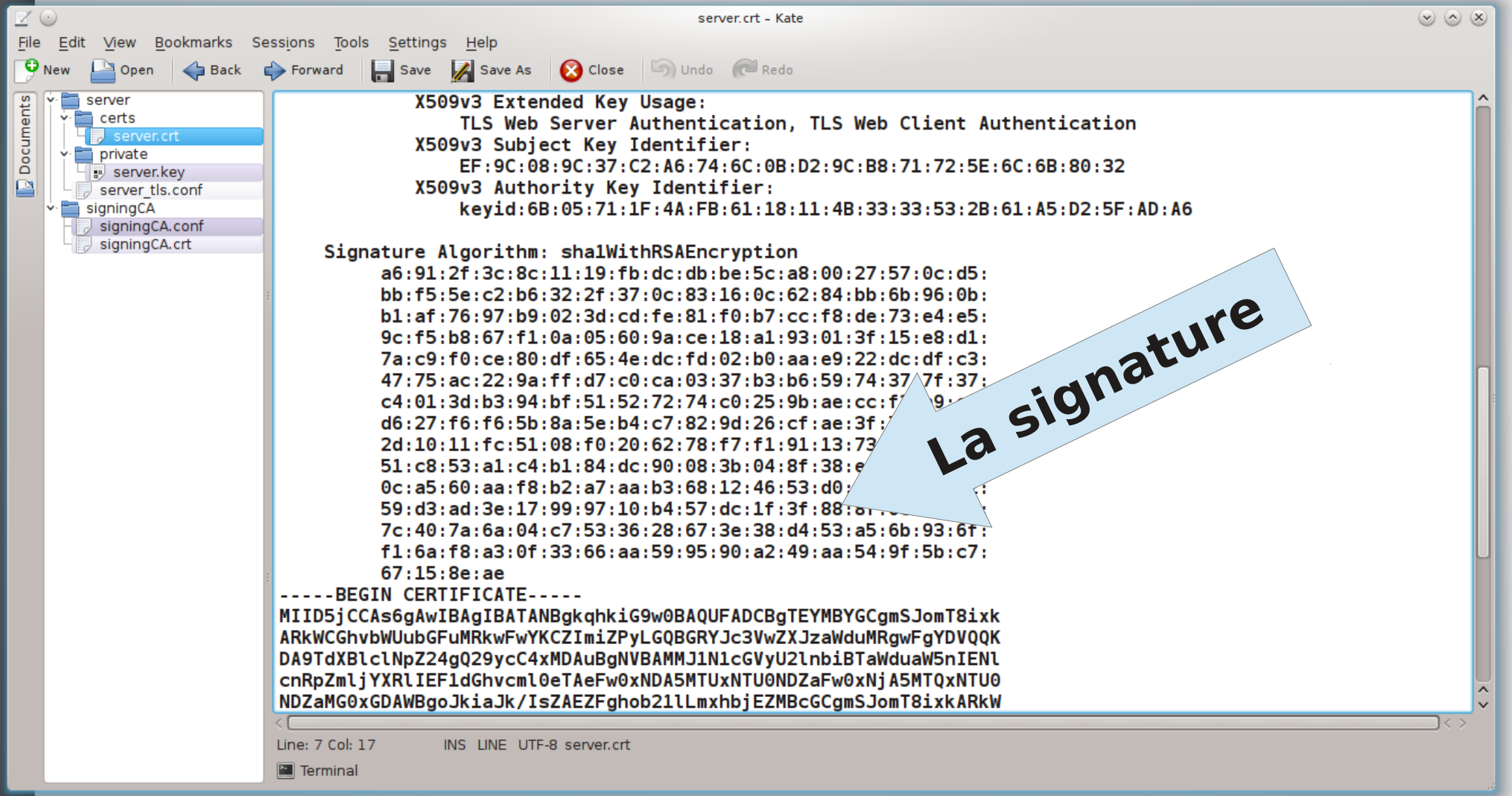
Création d'une autorité de certification

- Et on obtient notre certificat serveur pour monsiteweb.home.lan !

Création d'une autorité de certification



Création d'une autorité de certification



Création d'une autorité de certification

- Et pour finir nous allons :
  - Générer la clé privée d'Alice
  - Générer la clé publique d'Alice
  - Générer la *Certificate Signing Request* d'Alice

Création d'une autorité de certification

- Cela se fait en une seule étape avec OpenSSL

~/superSignCA/user : bash

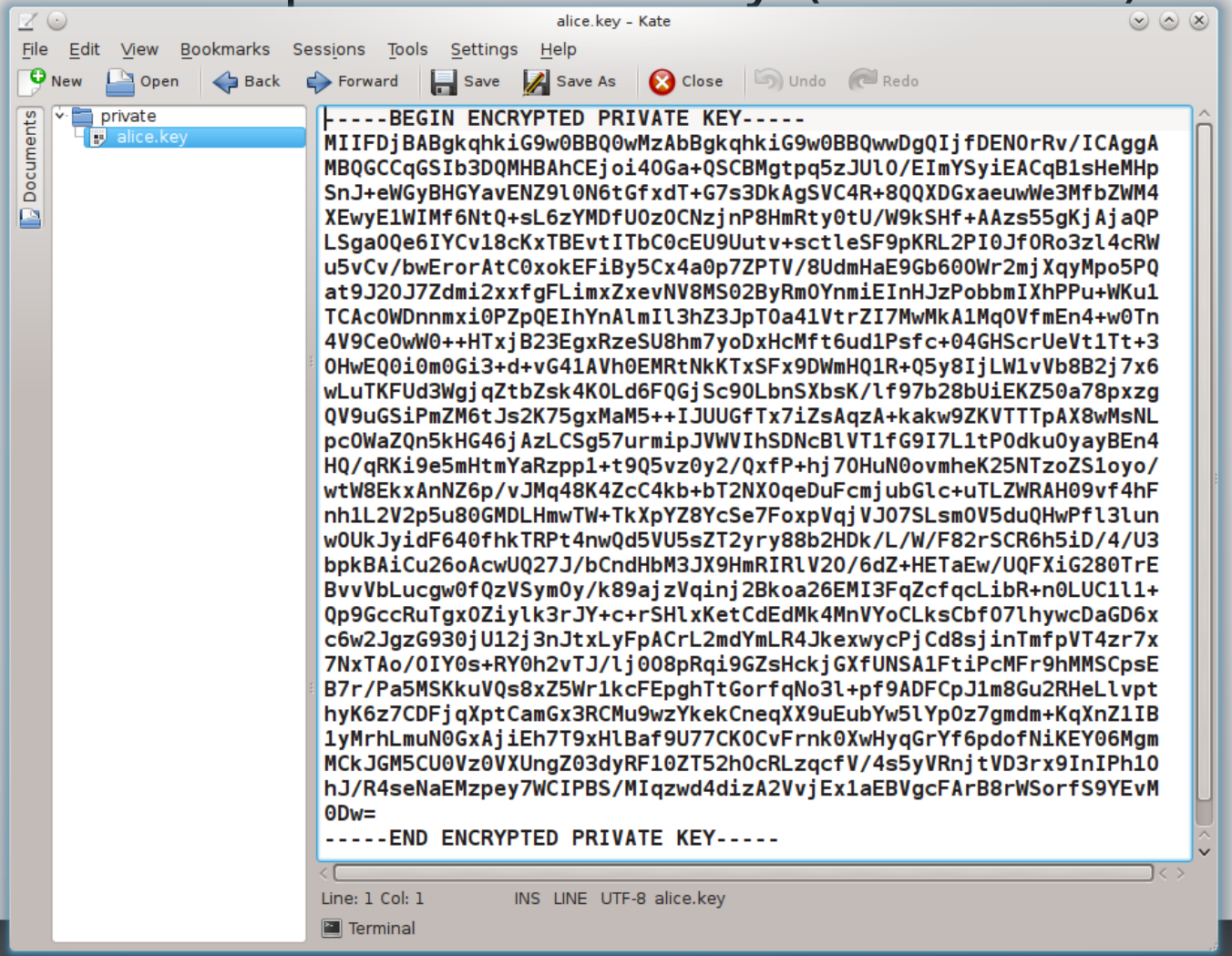
File Edit View Bookmarks Settings Help

```
jb@jb-laptop:~/superSignCA/user$ openssl req -new -config user.conf -out requests/alice.csr
-keyout private/alice.key
Generating a 2048 bit RSA private key
.....................................................+++
.....................................................................+++
writing new private key to 'private/alice.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
1. Domain Component         (eg, home.lan)       []:home.lan
2. Domain Component         (eg, excelsior)   []:supersign
4. Organization Name        (eg, Excelsior)   []:SuperSign Corp.
6. Common Name              (eg, John Smith) []:Alice
7. Email Address            (eg, john.smith@foobar.com) []:alice@machintruc.com
```

~/superSignCA/user : bash

Création d'une autorité de certification

- On obtient ainsi la CSR alice.csr ainsi que la clé privée alice.key (2048 bits)



Création d'une autorité de certification

- Maintenant nous allons signer la CSR au moyen de la clé privée signingCA.key

~/superSignCA/user : bash

File Edit View Bookmarks Settings Help

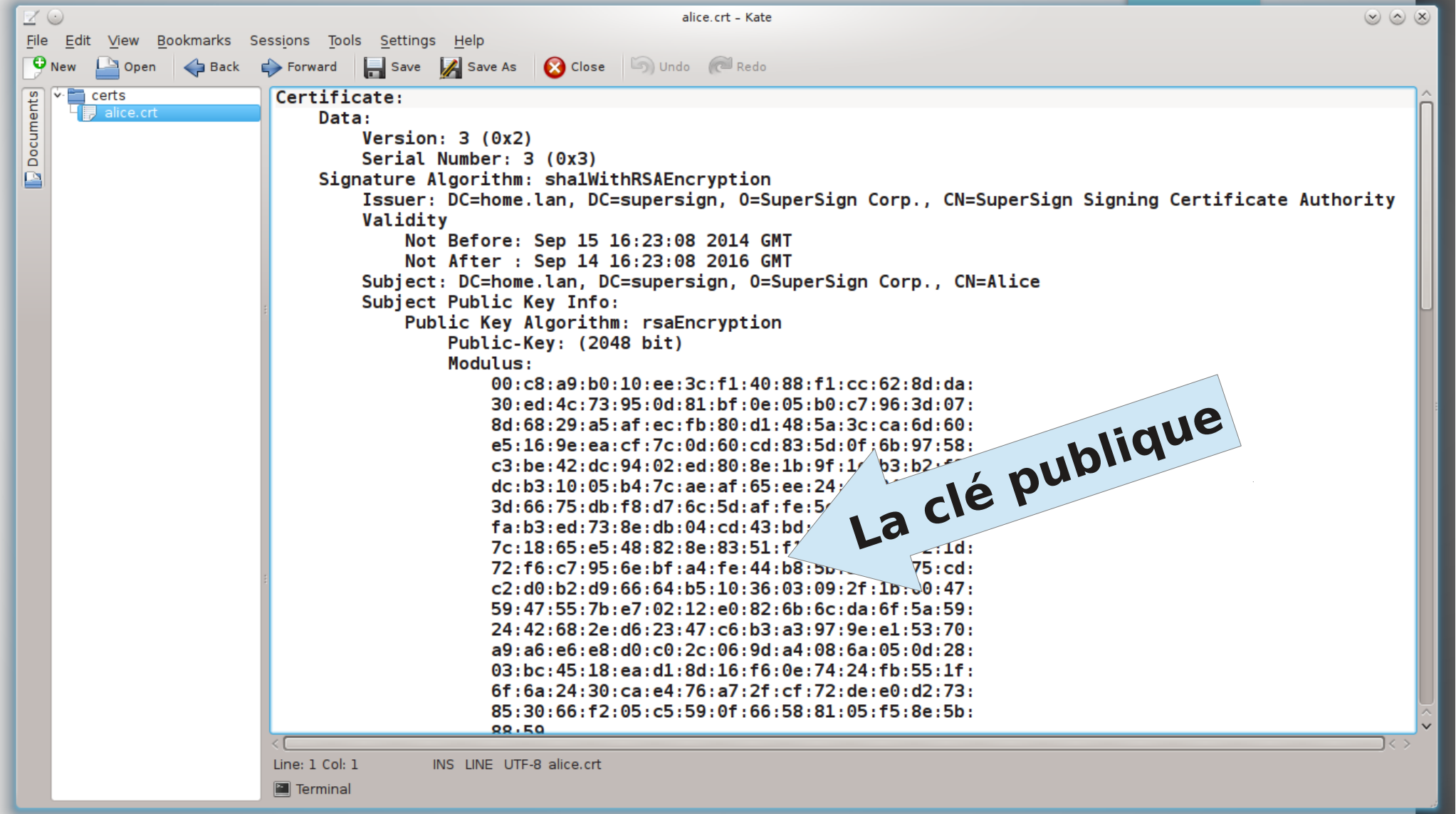
```
jb@jb-laptop:~/superSignCA/user$ openssl ca -config ../signingCA/signingCA.conf -in requests/alice.csr
 -out certs/alice.crt -extensions email_ext
Using configuration from ../signingCA/signingCA.conf
Enter pass phrase for /home/jb/superSignCA/signingCA/private/signingCA.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 3 (0x3)
        Validity
            Not Before: Sep 15 16:23:08 2014 GMT
            Not After : Sep 14 16:23:08 2016 GMT
        Subject:
            domainComponent           = home.lan
            domainComponent           = supersign
            organizationName          = SuperSign Corp.
            commonName                = Alice
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints:
                CA:FALSE
```

~/superSignCA/user : bash

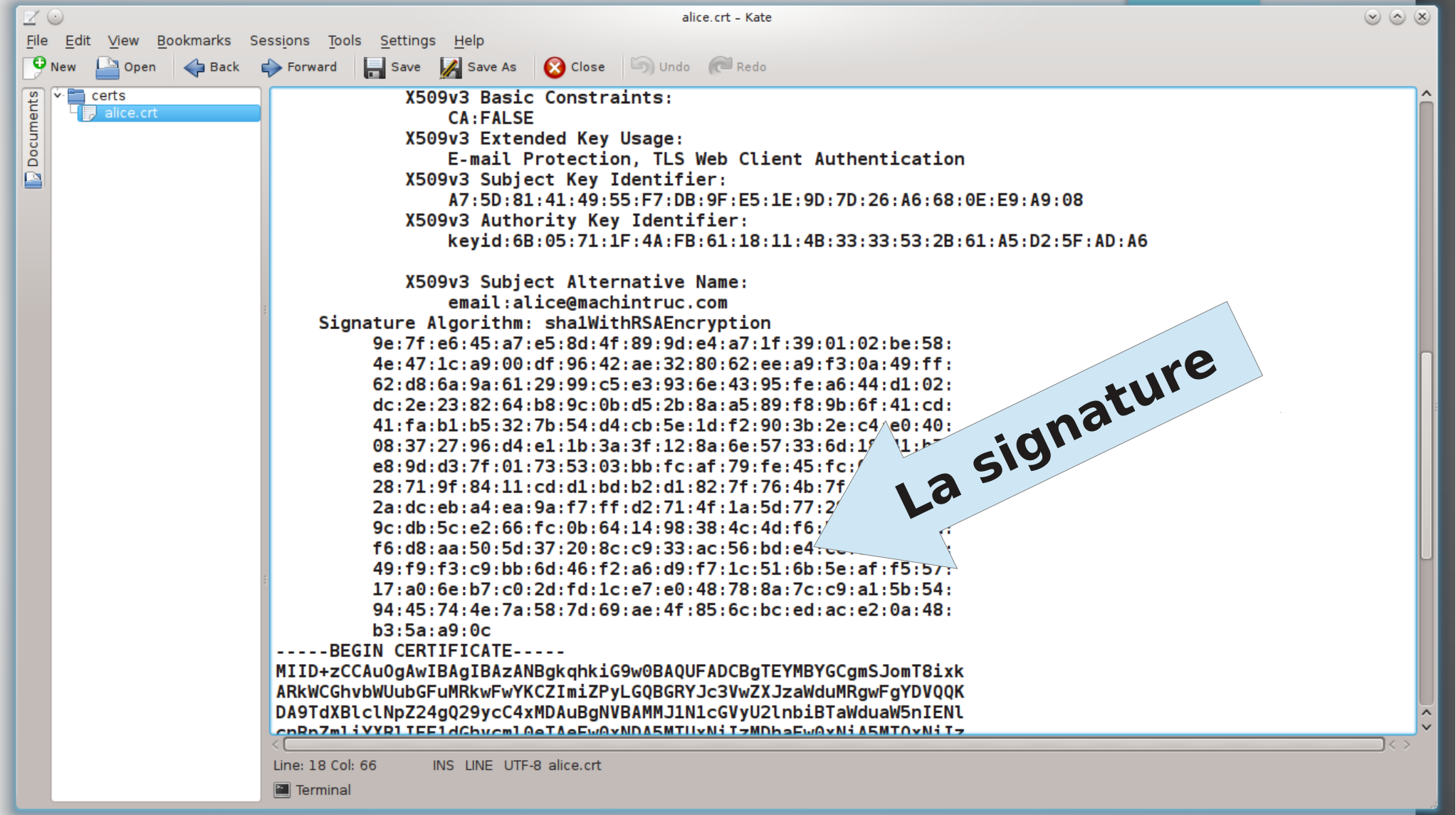
Création d'une autorité de certification

- Et on obtient le certificat de Alice!

Création d'une autorité de certification



Création d'une autorité de certification



Création d'une autorité de certification

- Dans la pratique, les certificats “utilisateurs” sont souvent associés à la clé privée dans un “bundle” ou enveloppe. La norme la plus utilisée est PKCS 12.

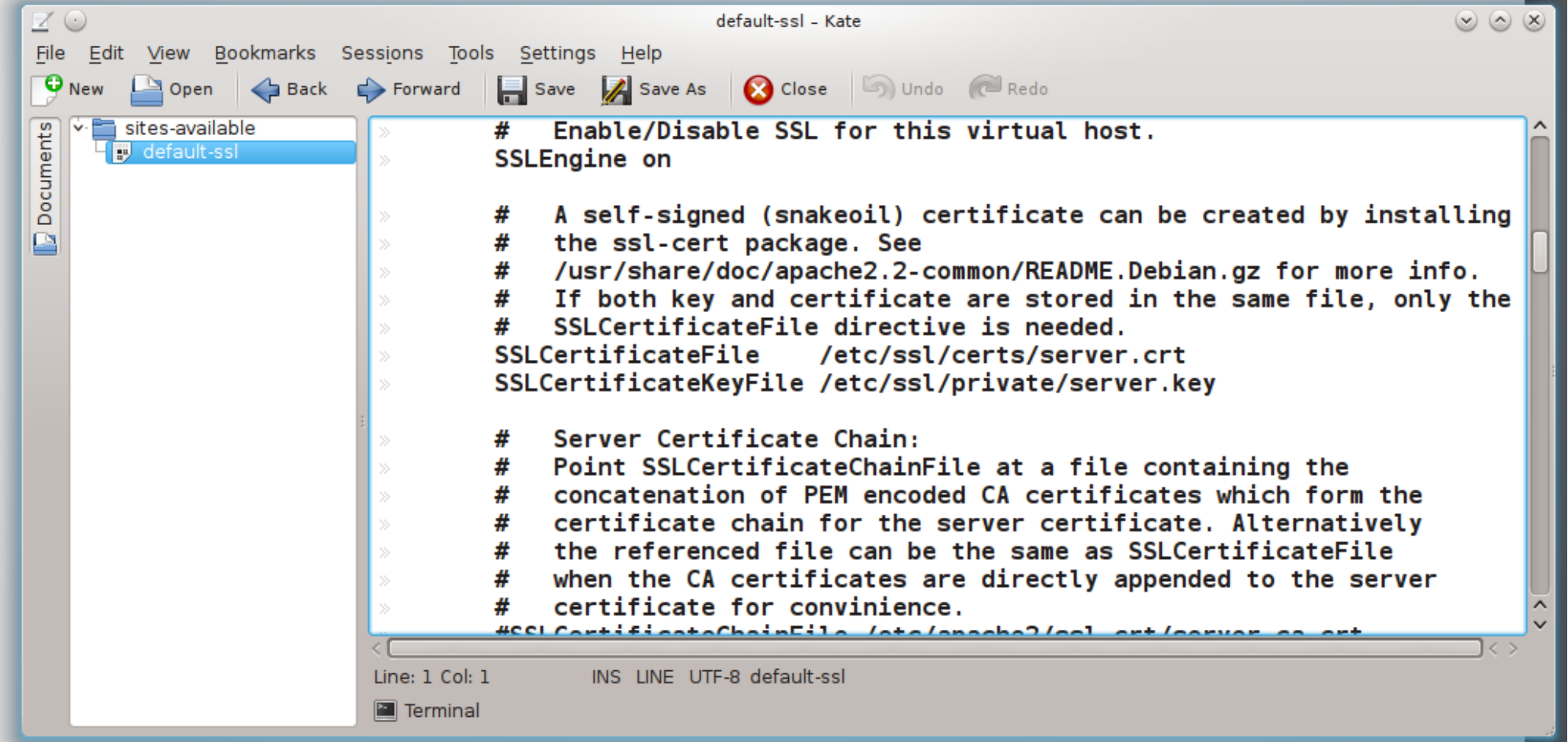
~/superSignCA/user : bash

File Edit View Bookmarks Settings Help

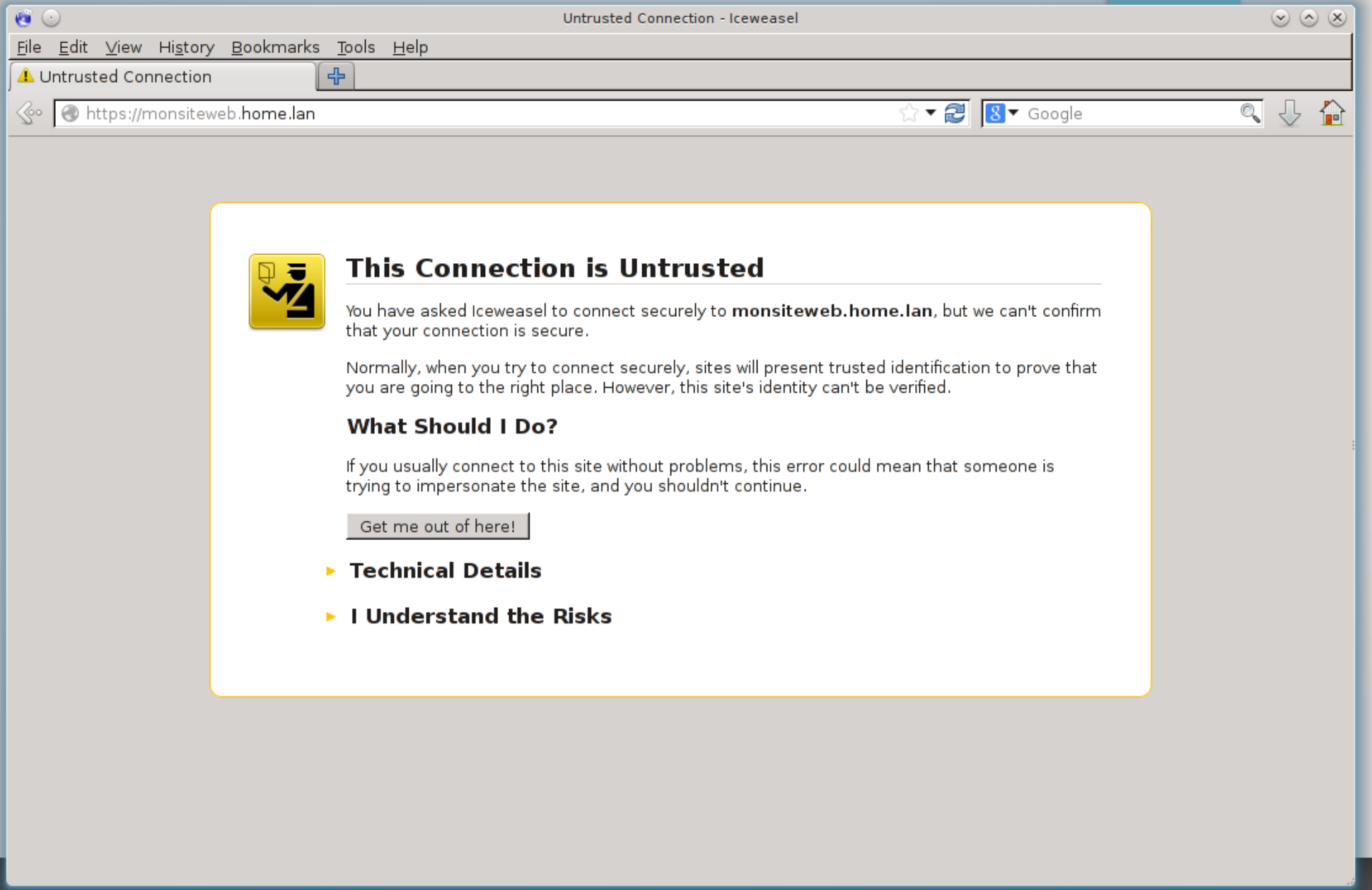
```
jb@jb-laptop:~/superSignCA/user$ openssl pkcs12 -export -name "Alice" -inkey private/alice.key
-in certs/alice.crt -out bundles/alice.p12
Enter pass phrase for private/alice.key:
Enter Export Password:
Verifying - Enter Export Password:
jb@jb-laptop:~/superSignCA/user$
```

~/superSignCA/user : bash

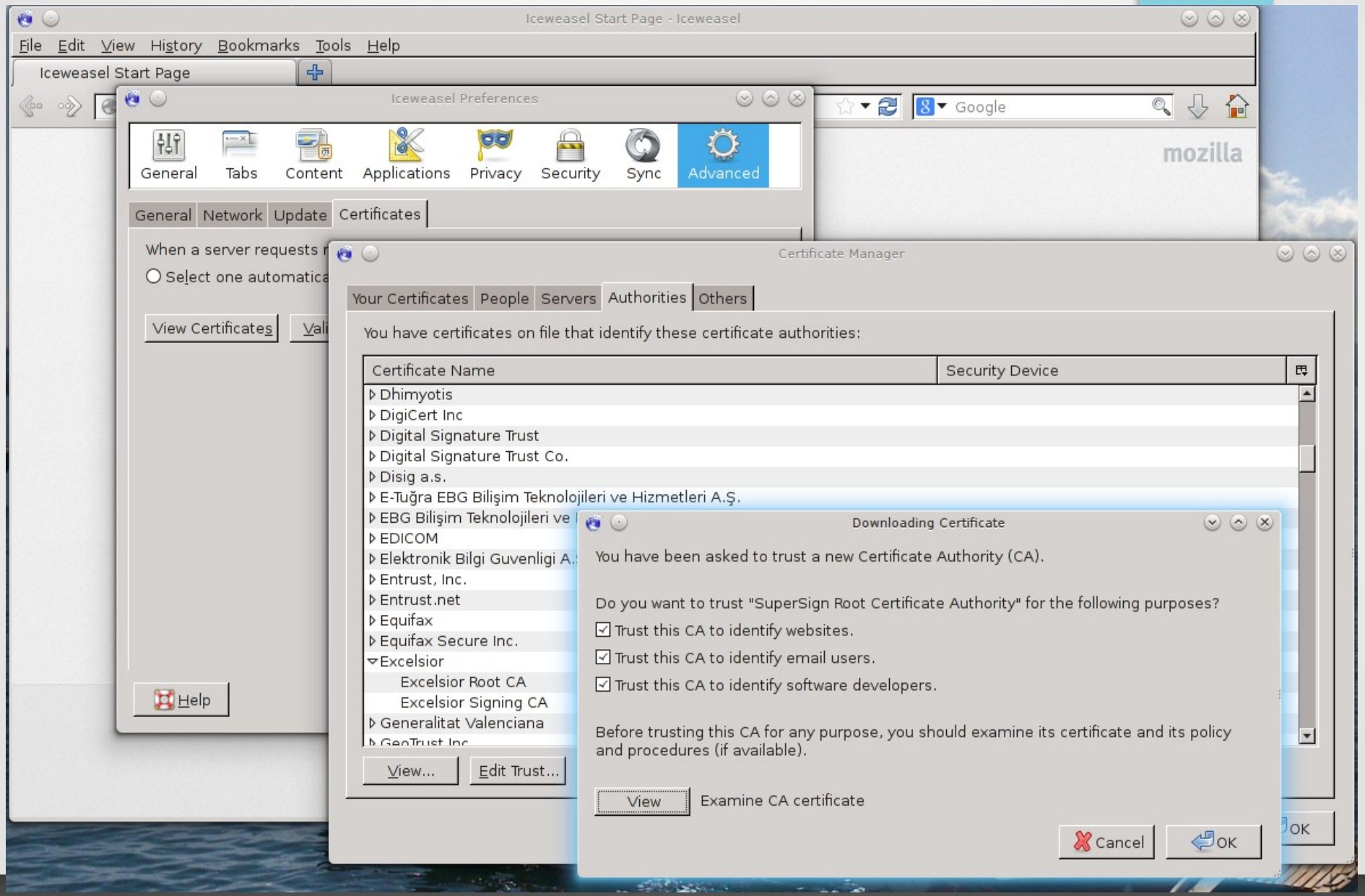
Déploiement du certificat serveur



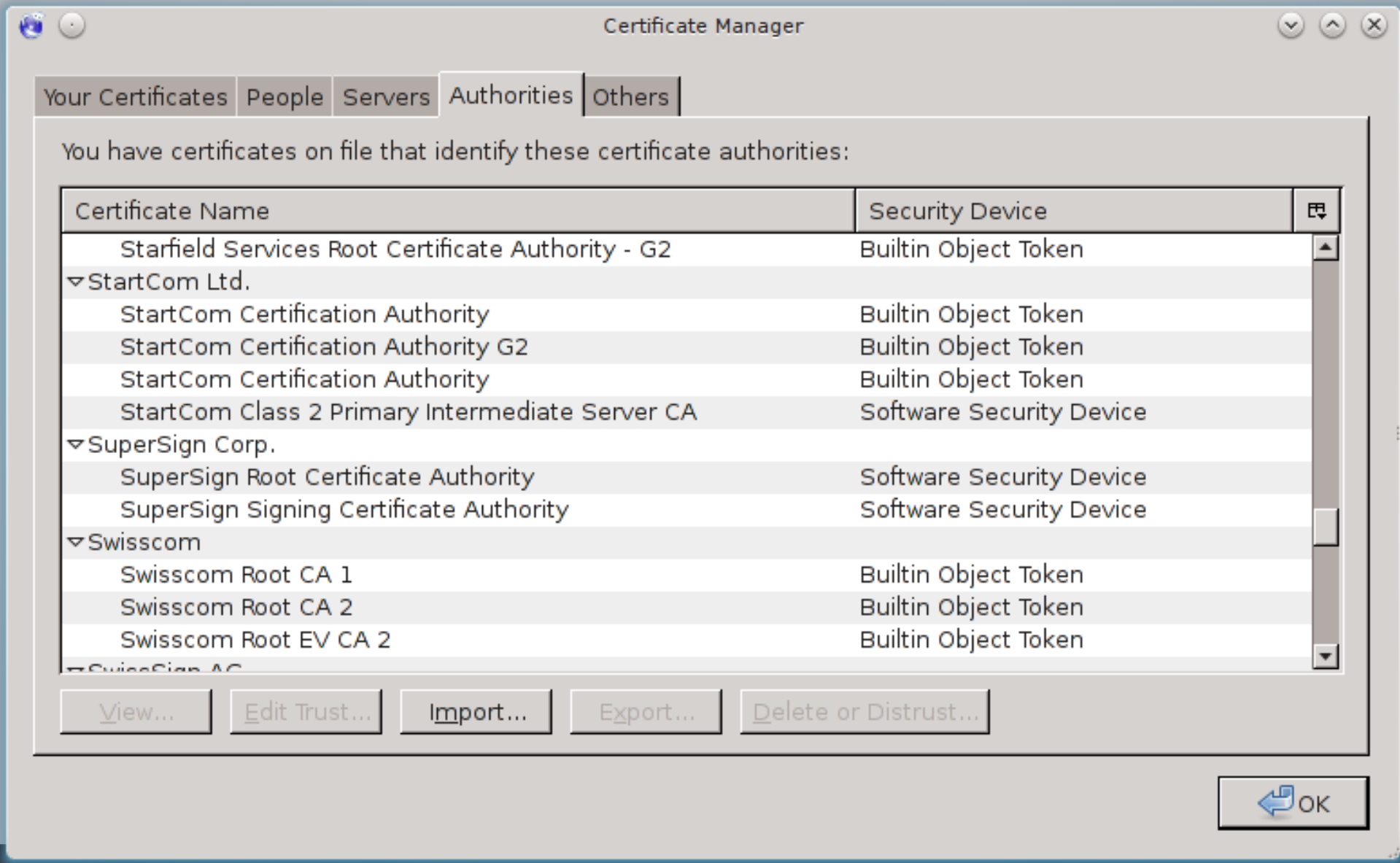
Déploiement du certificat serveur



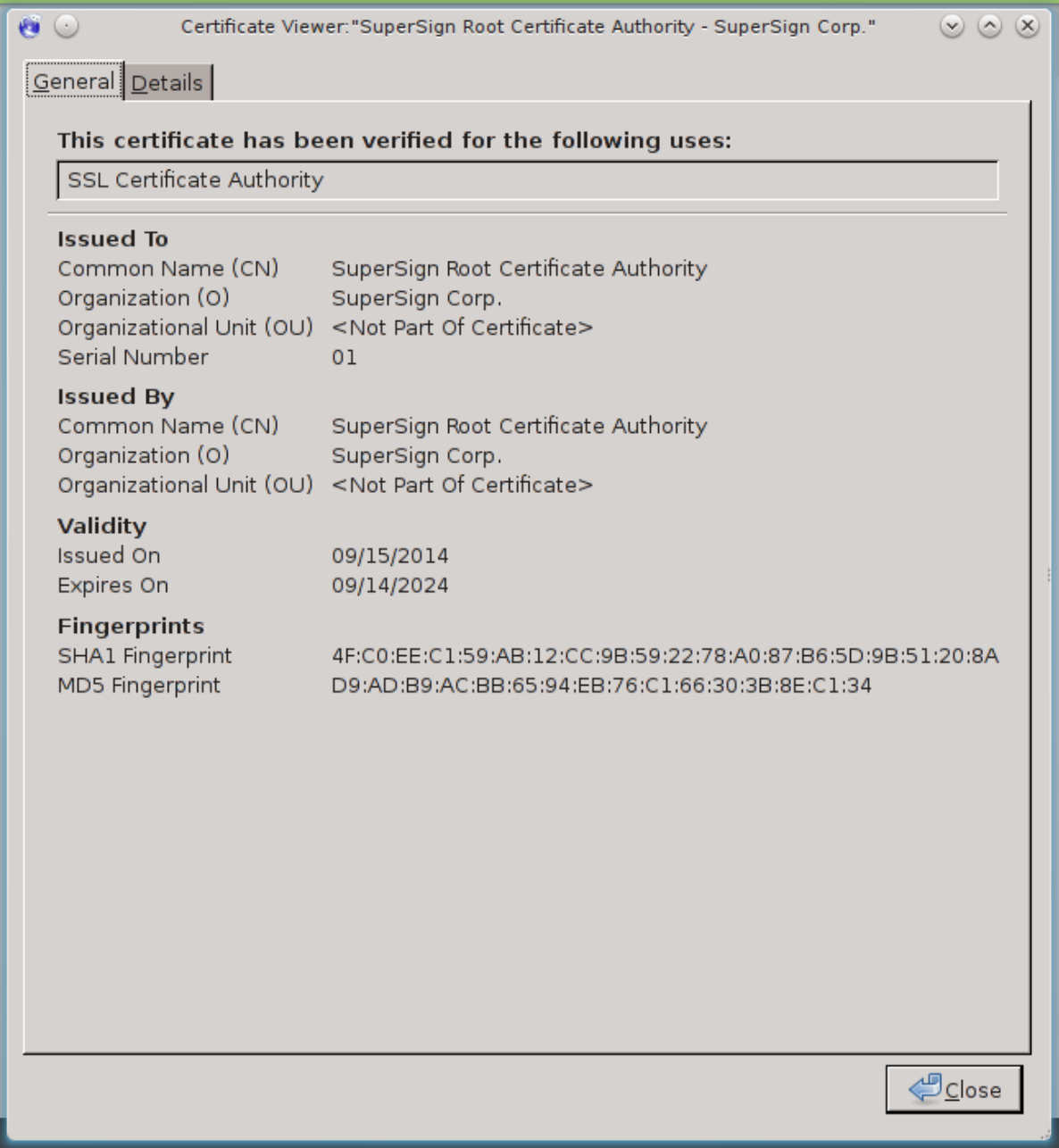
Déploiement du certificat serveur



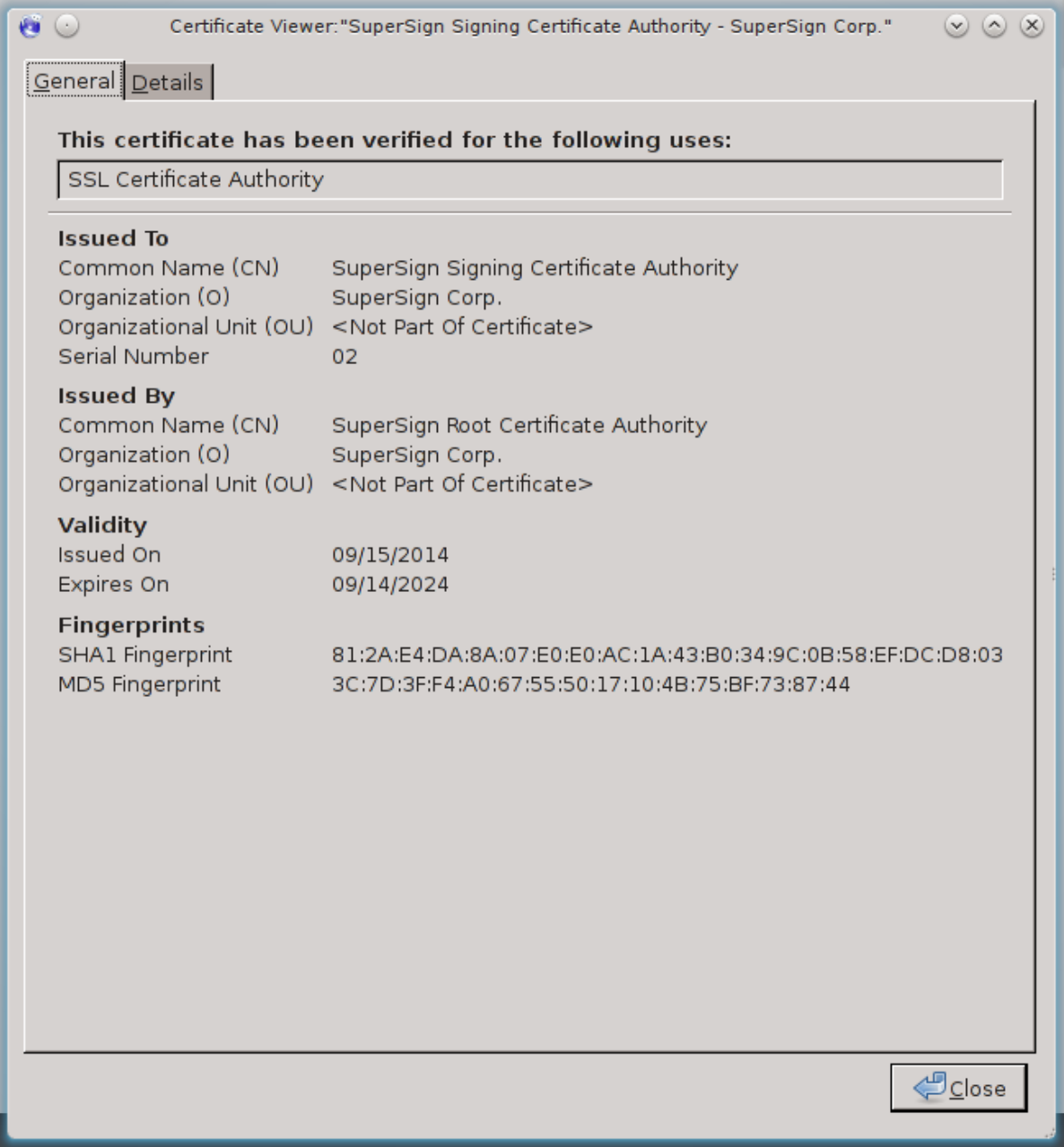
Déploiement du certificat serveur



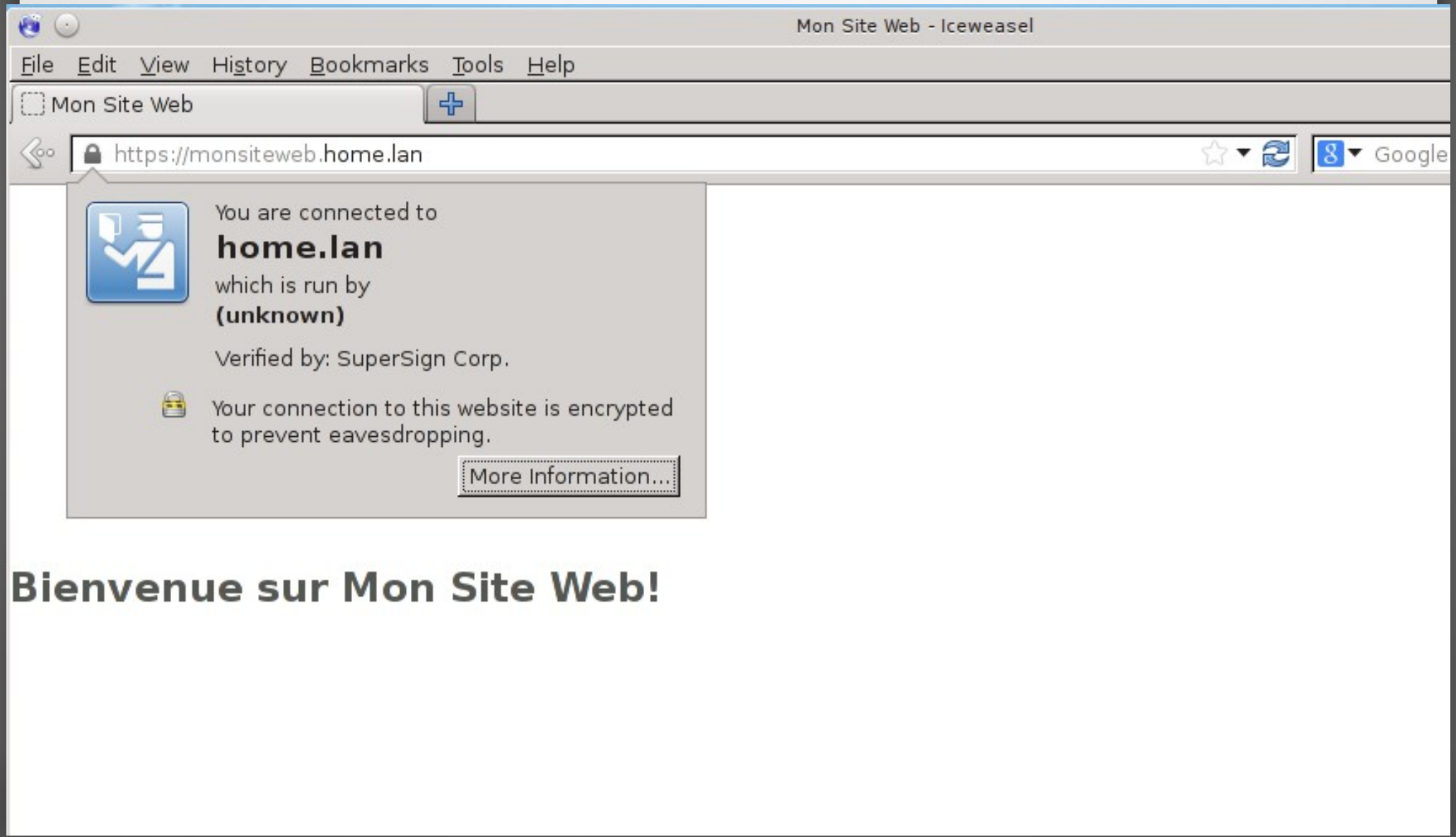
Les certificats électroniques



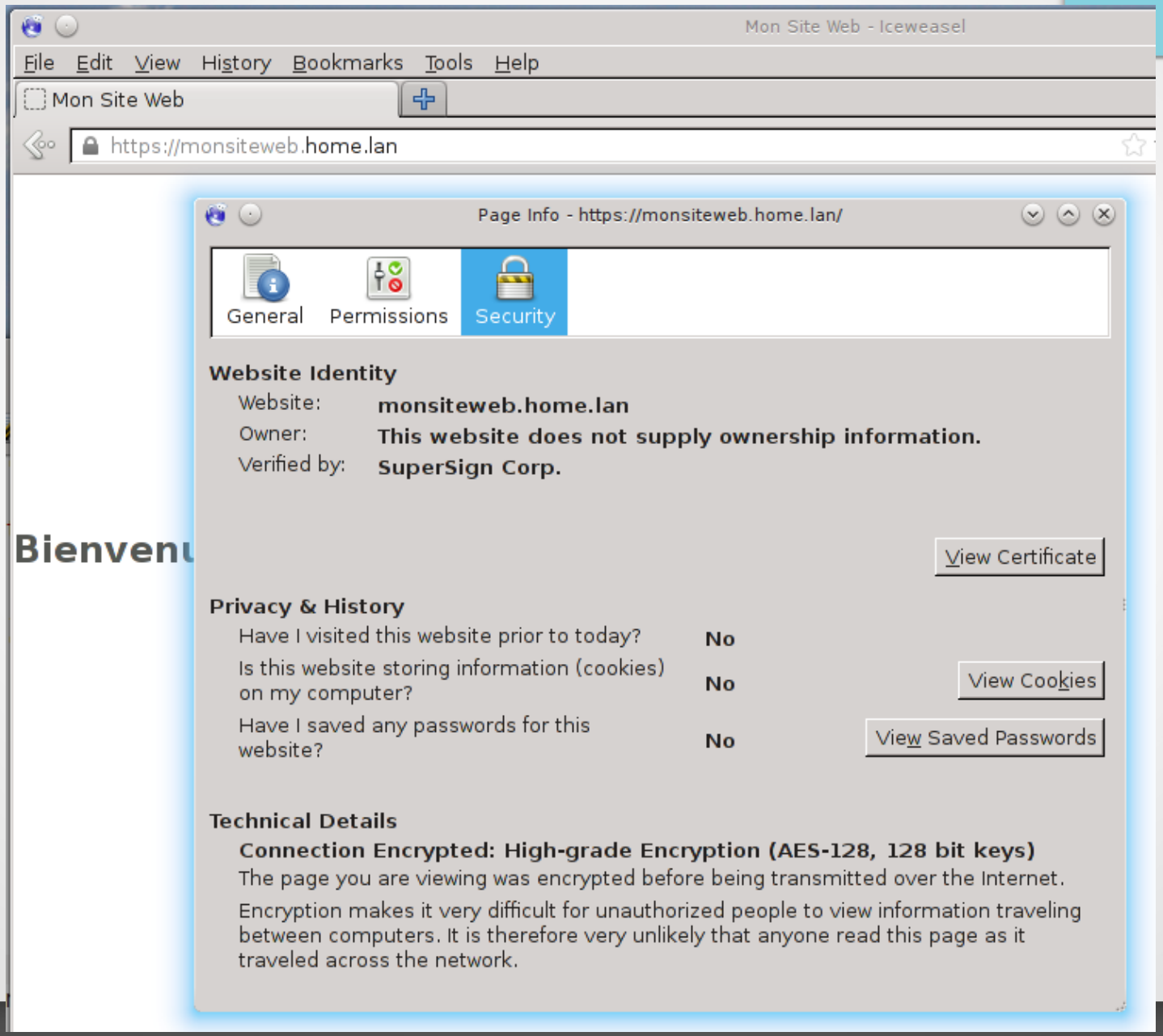
Déploiement du certificat serveur



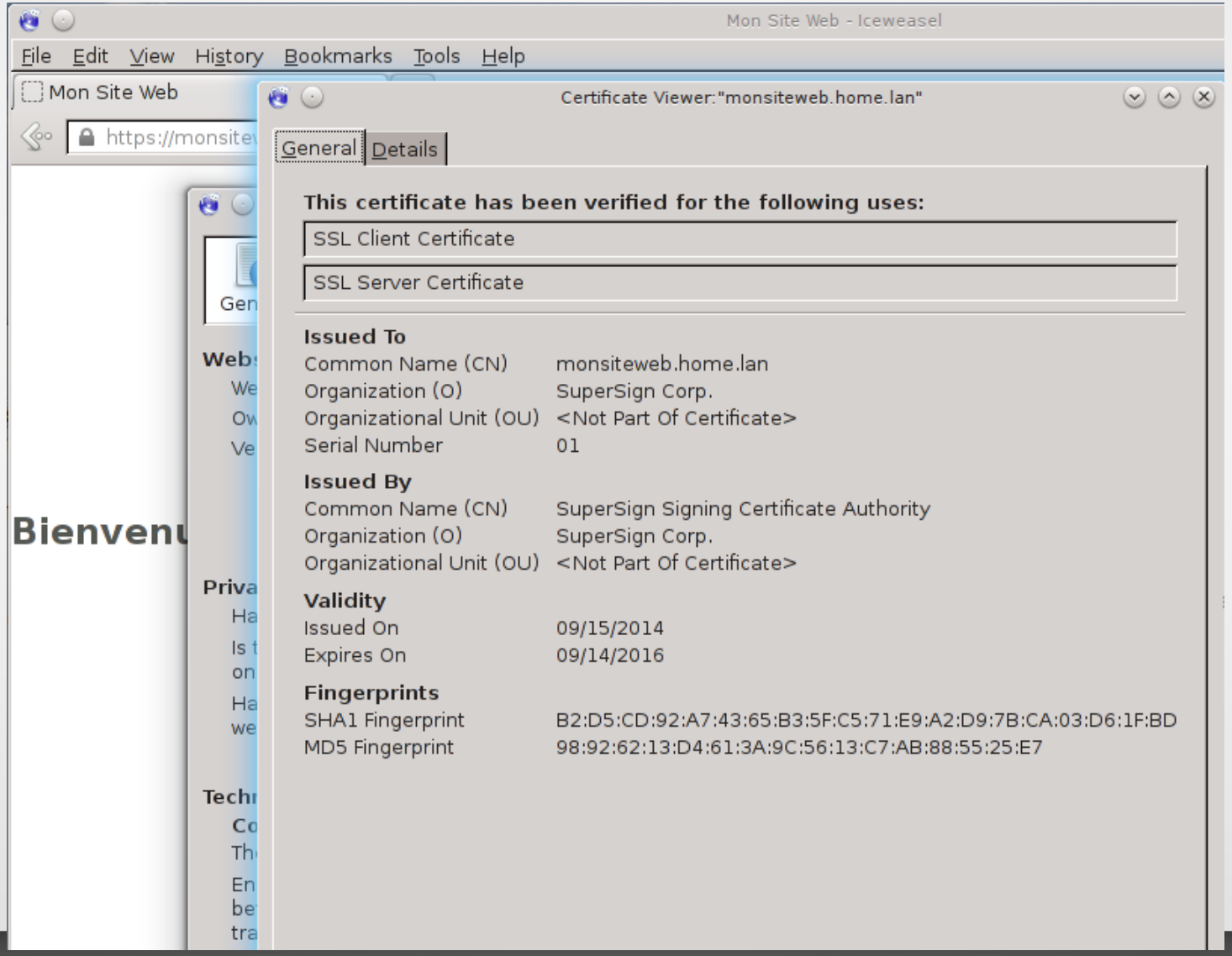
Déploiement du certificat serveur



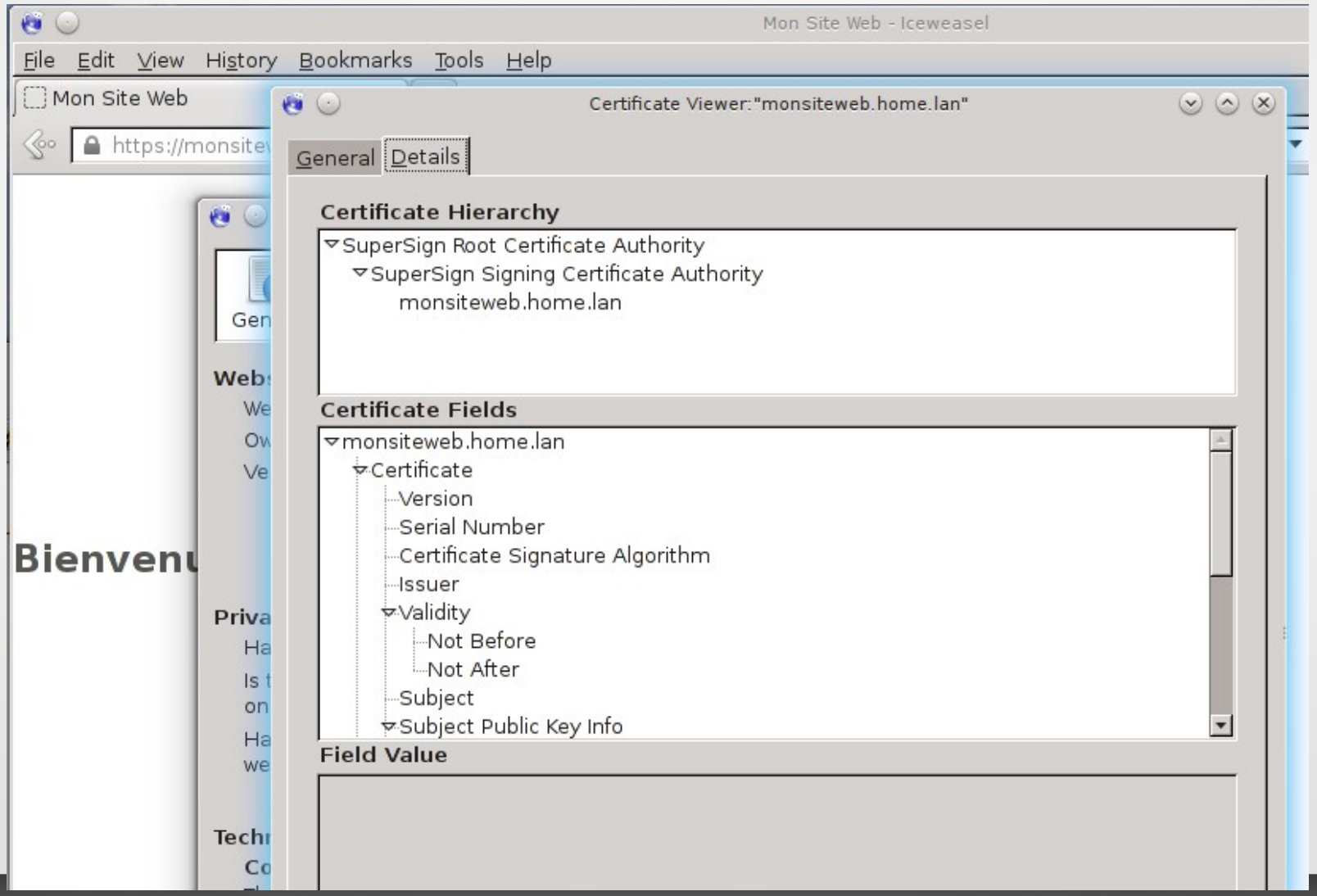
Déploiement du certificat serveur



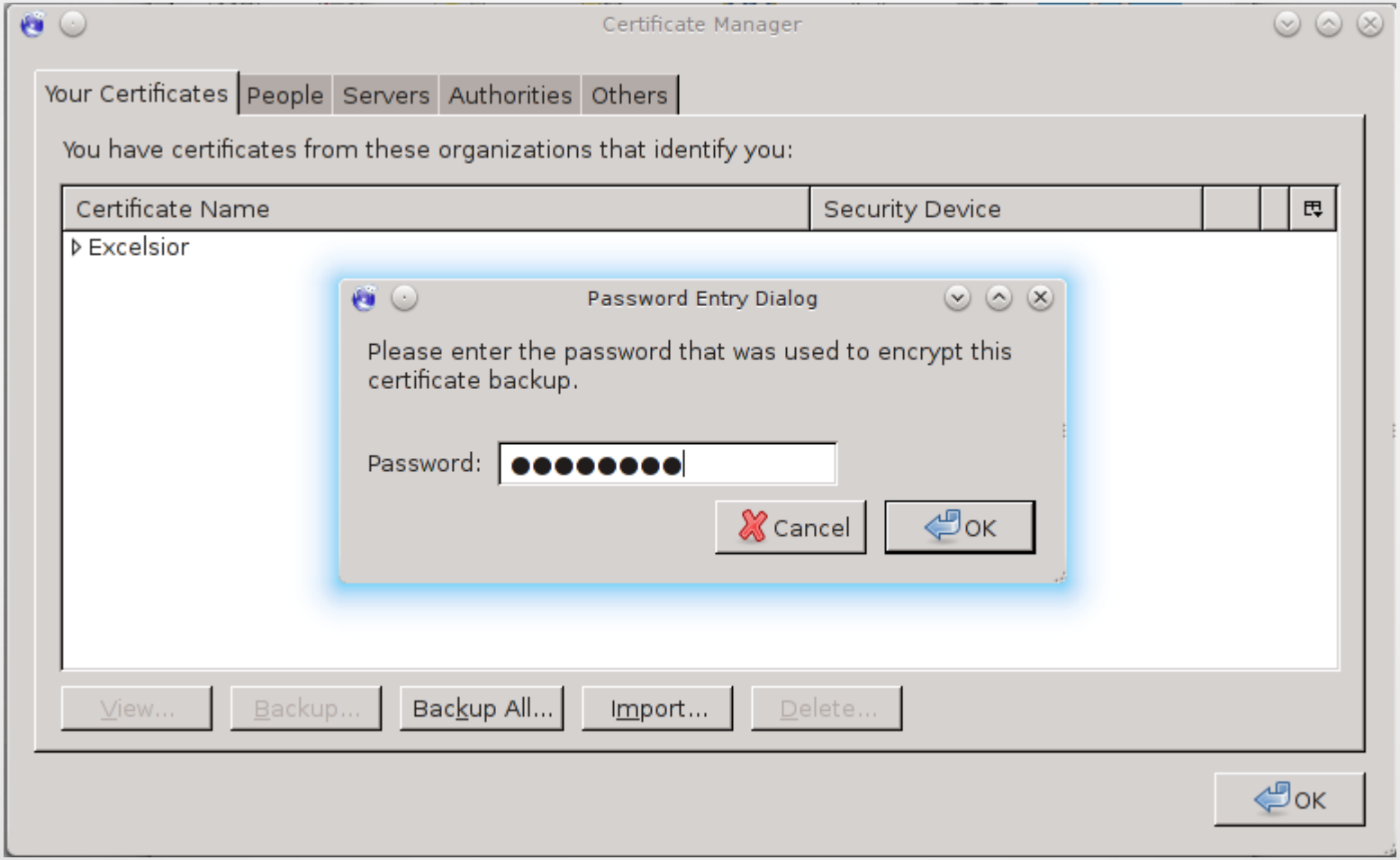
Déploiement du certificat serveur



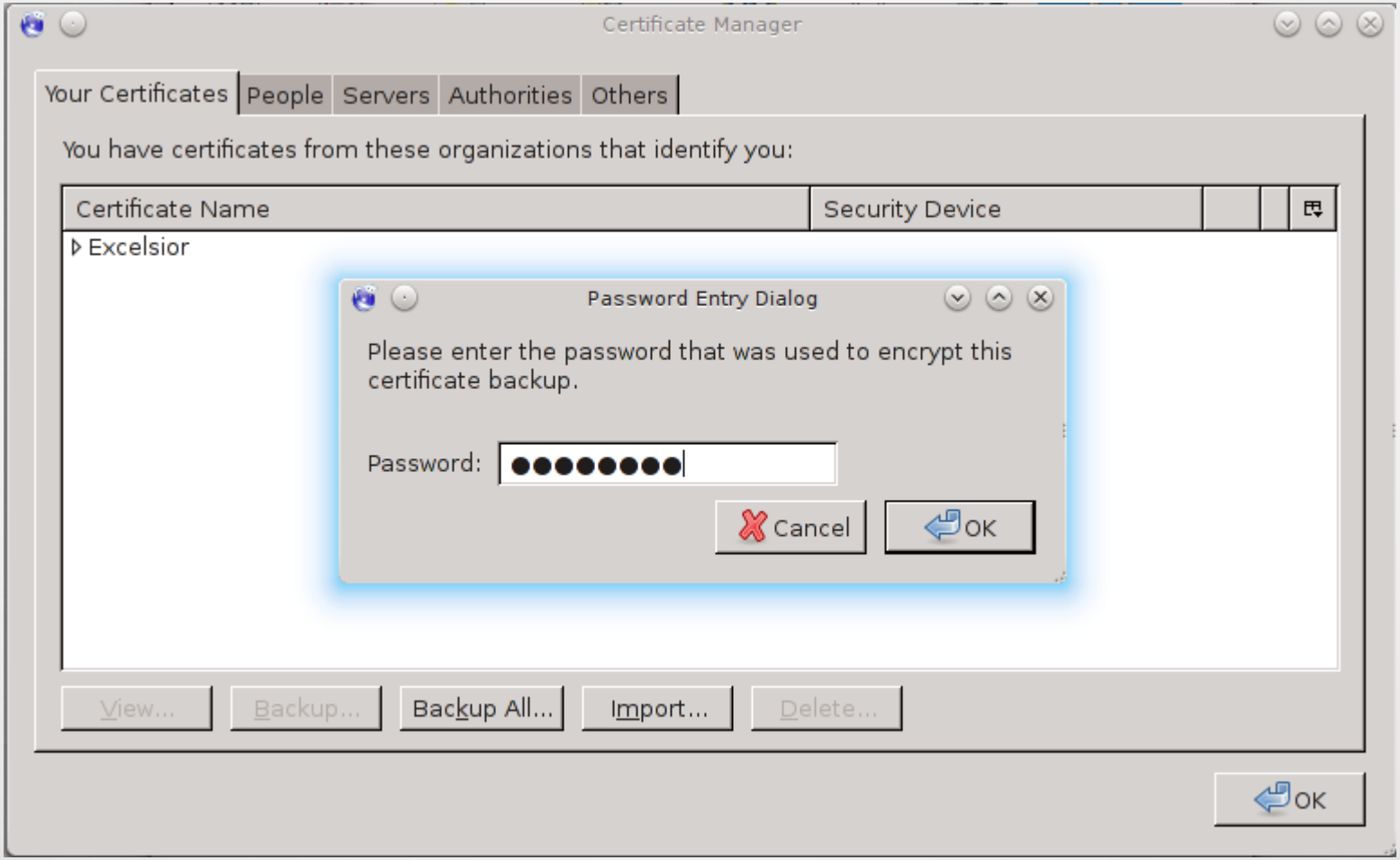
Déploiement du certificat serveur



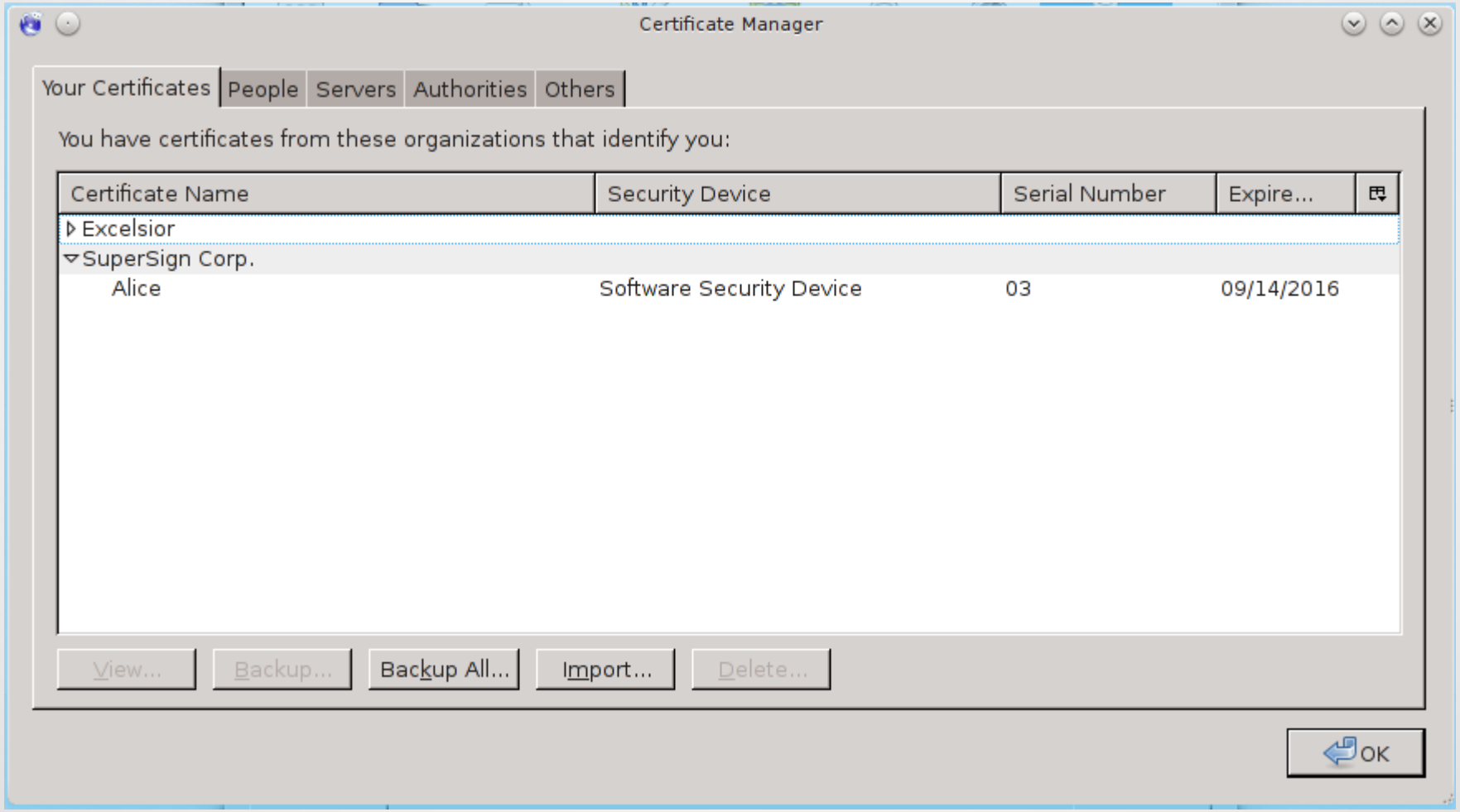
Déploiement du certificat d'Alice



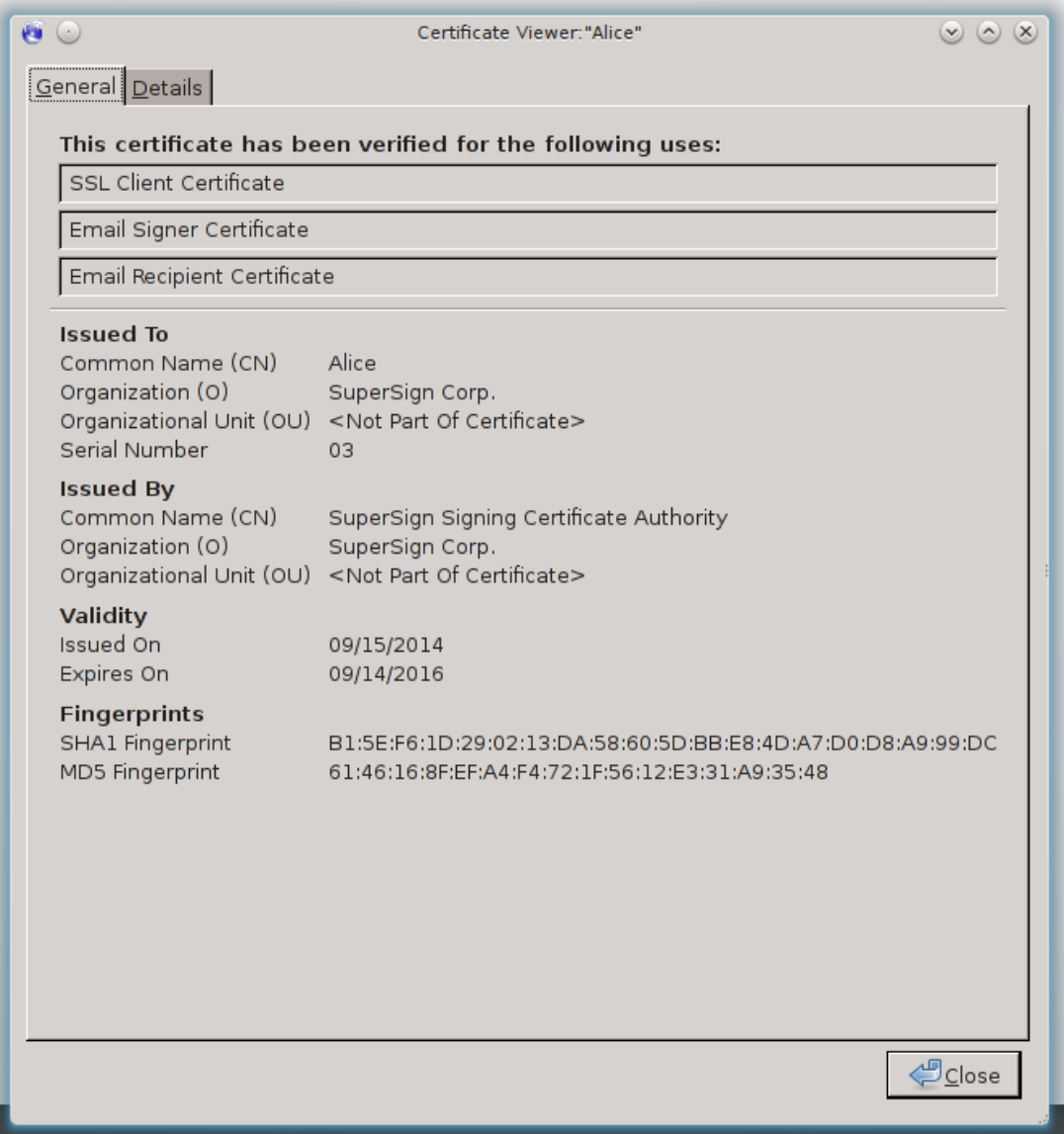
Déploiement du certificat d'Alice



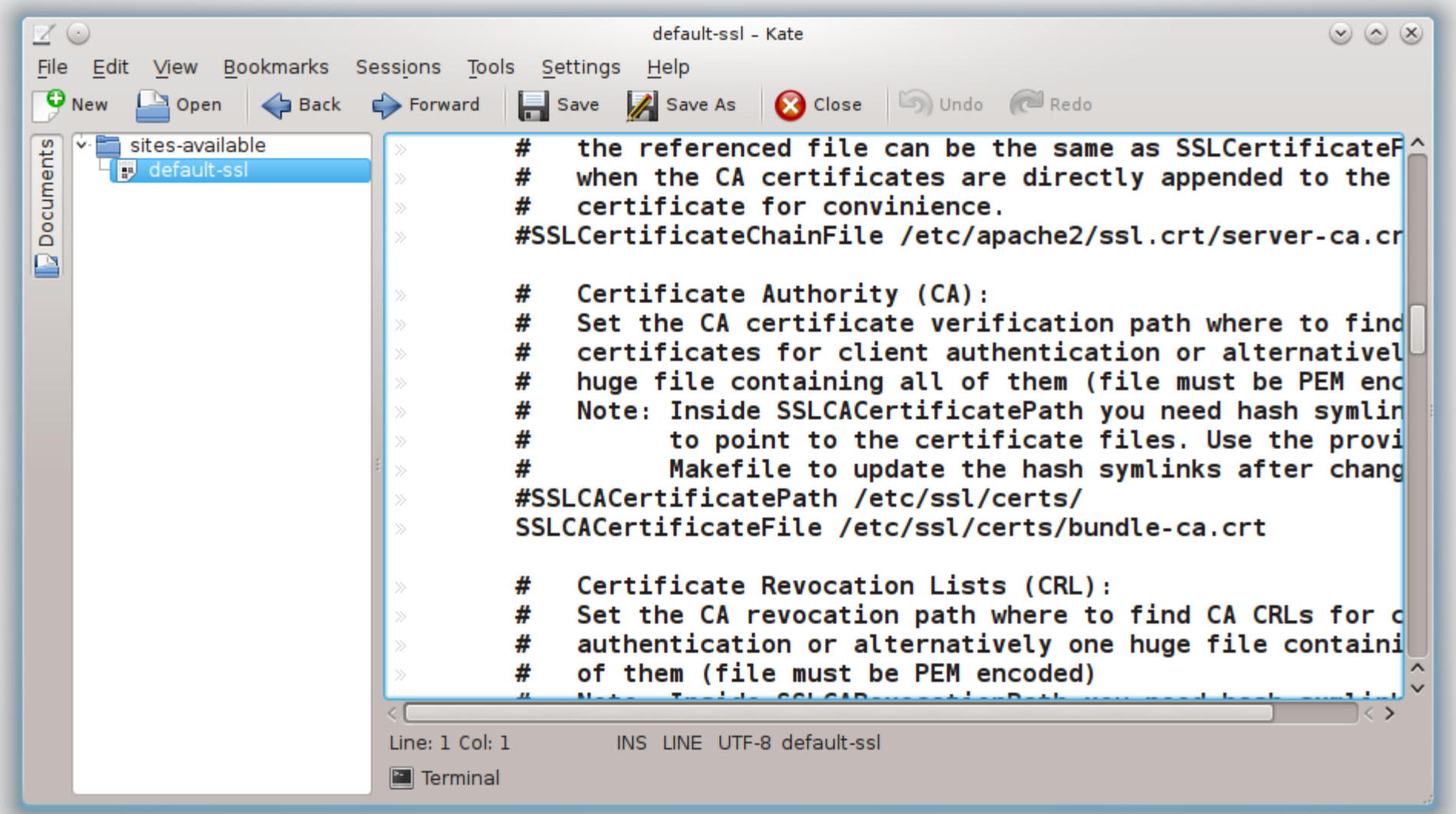
Déploiement du certificat d'Alice



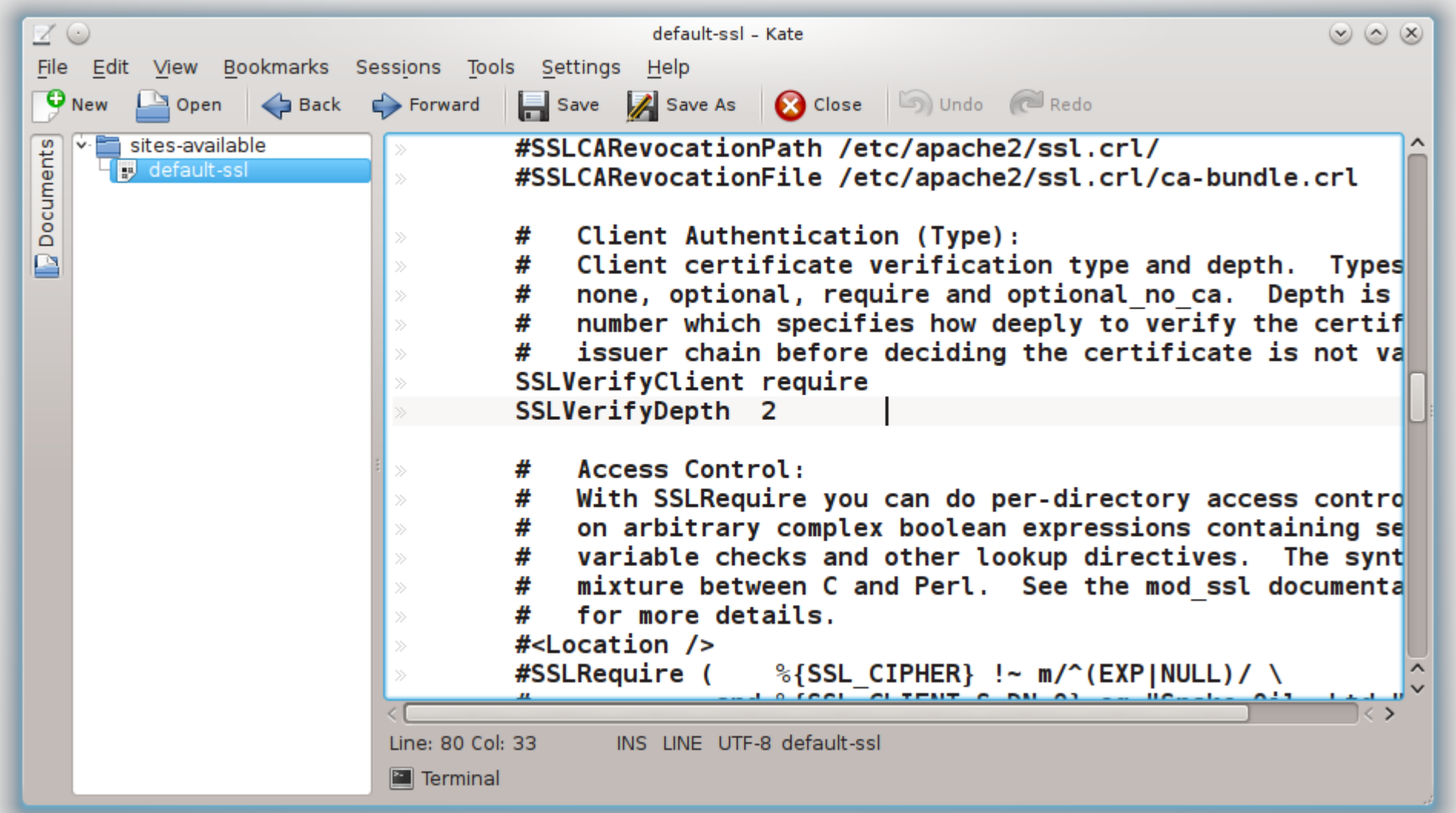
Déploiement du certificat d'Alice



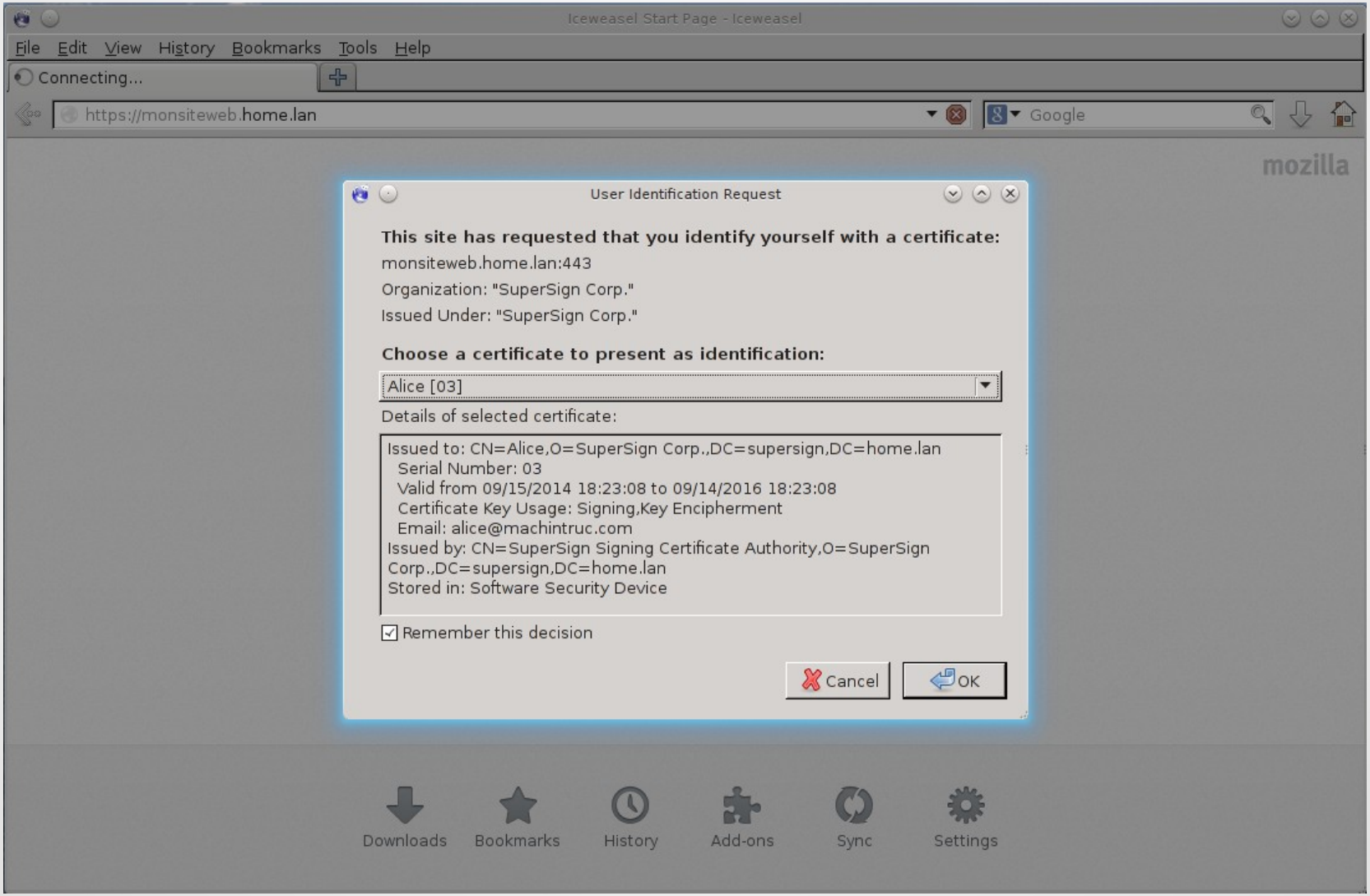
Déploiement du certificat d'autorité



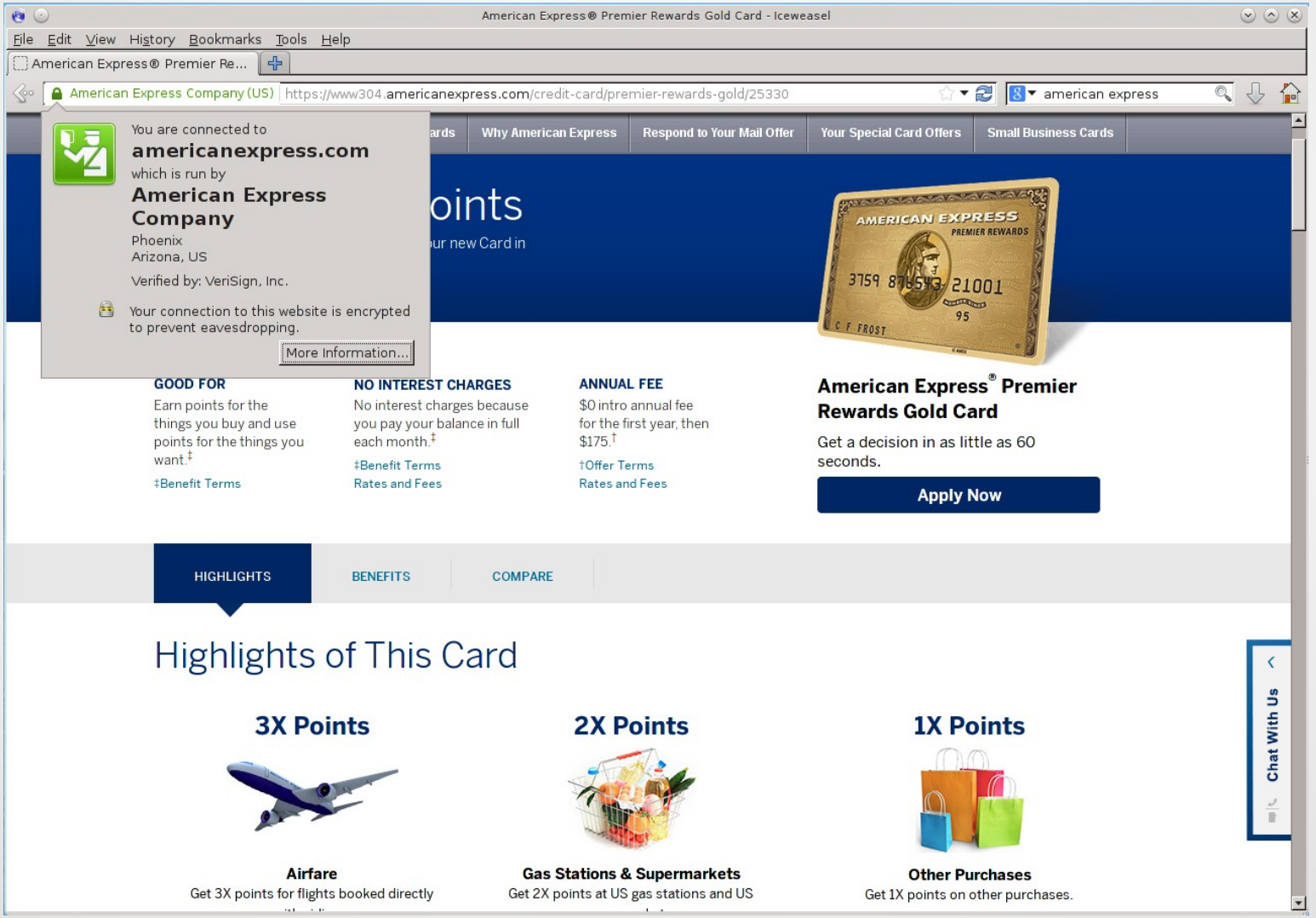
Déploiement du certificat d'autorité



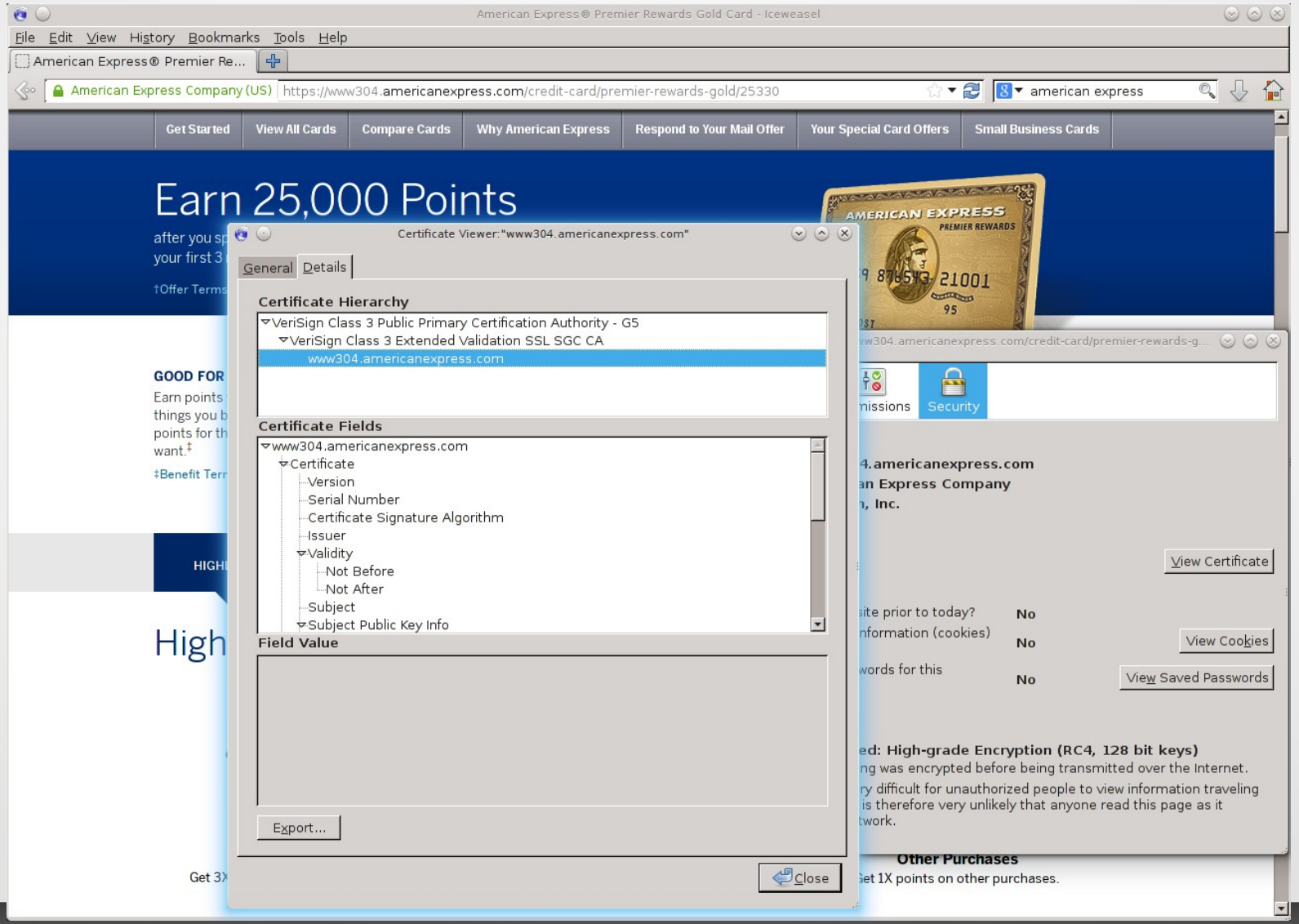
Déploiement du certificat d'autorité



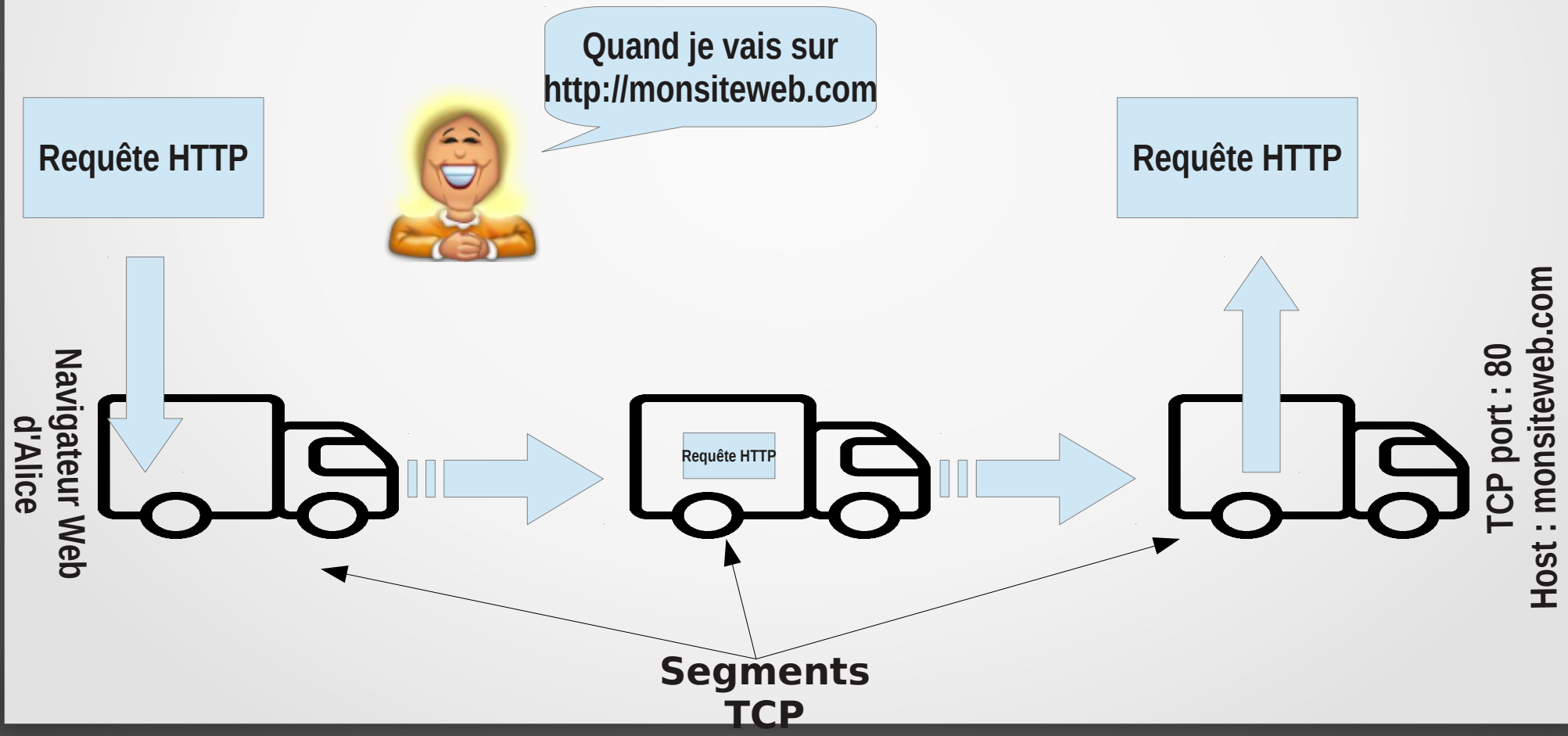
Les certificats électroniques



Les certificats électroniques

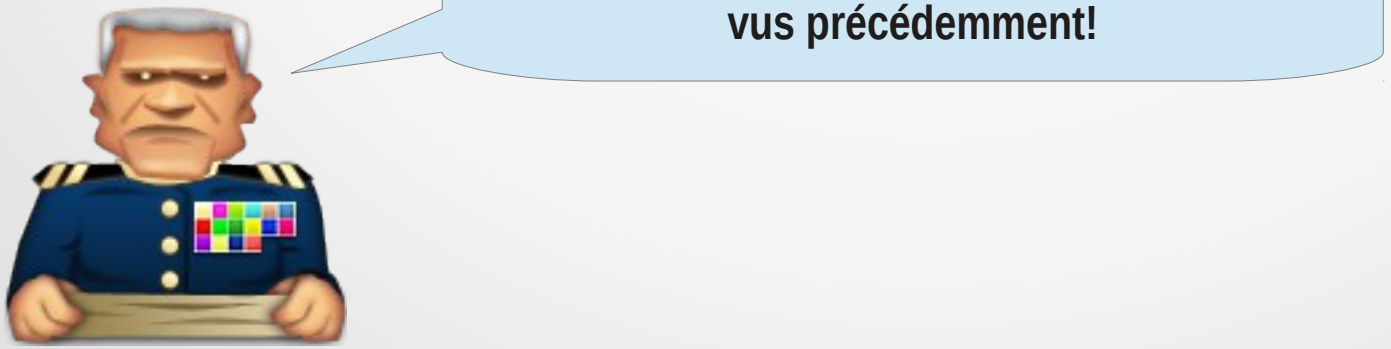


La suite Internet ultra-simplifiée



Secure Socket Layer (SSL/TLS)

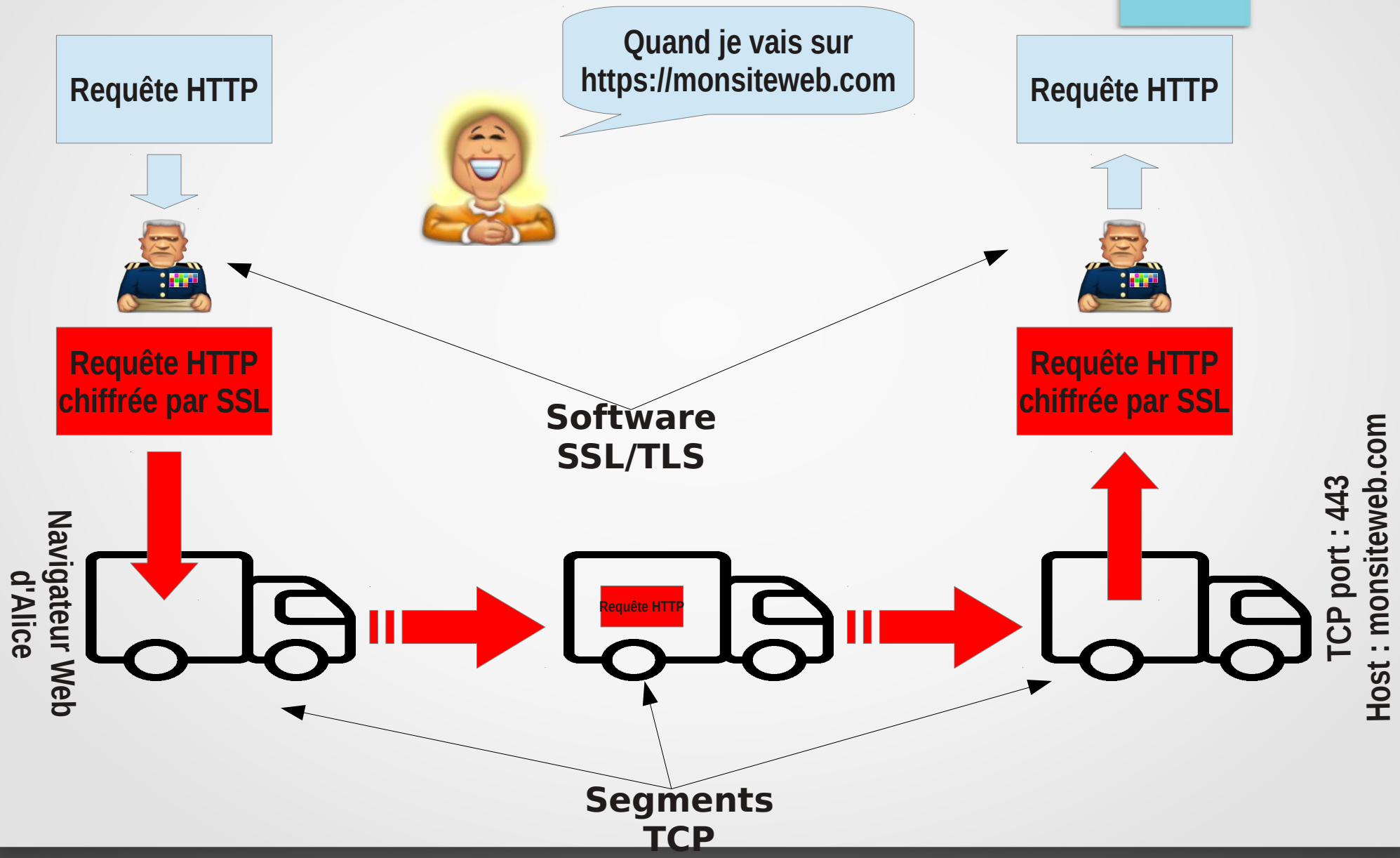
- Le nom actuel est Transport Layer Security mais l'ancien nom persiste
- C'est un ensemble de protocoles qui appliquent la cryptographie à clé publique
- On le retrouve partout ! (web, e-mail, messagerie instantannée, VoIP, ...)



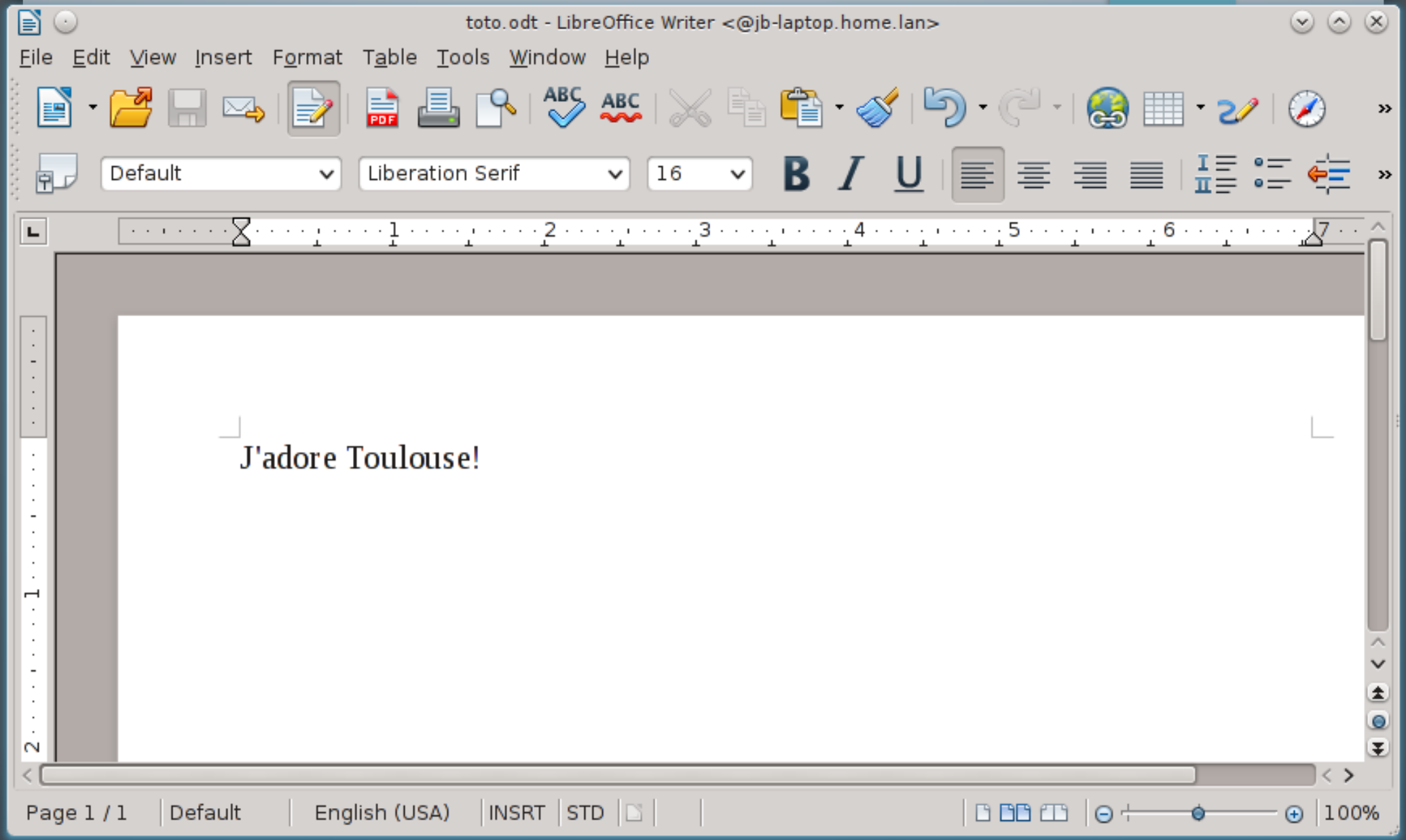
Secure Socket Layer (SSL/TLS)

- Assure l'authentification des deux parties : *handshake*
- Procède à un échange de clé symétriques temporaires
- Les messages sont ensuite chiffrés, signés et échangés par cryptographie symétrique

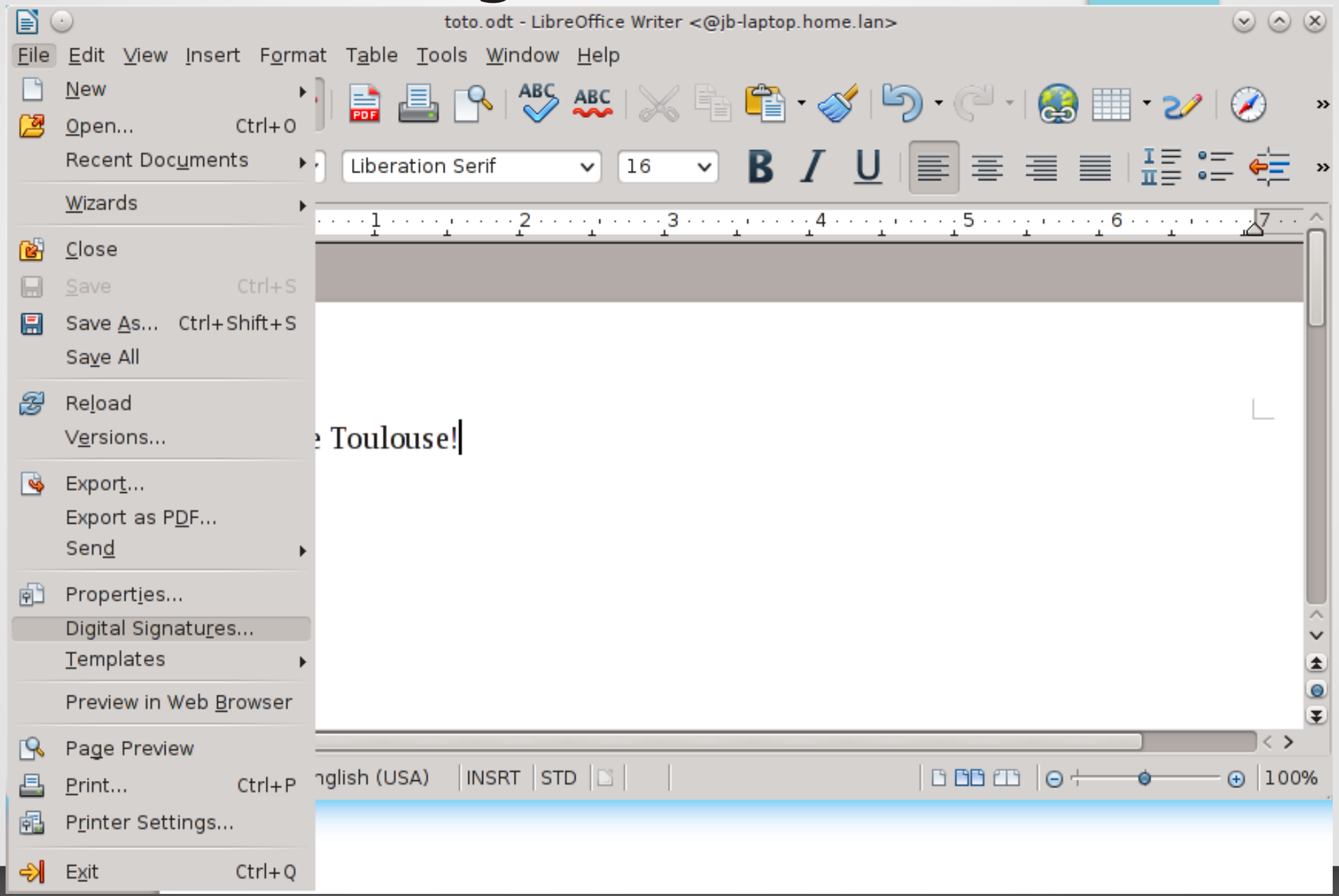
Le protocole HTTPS



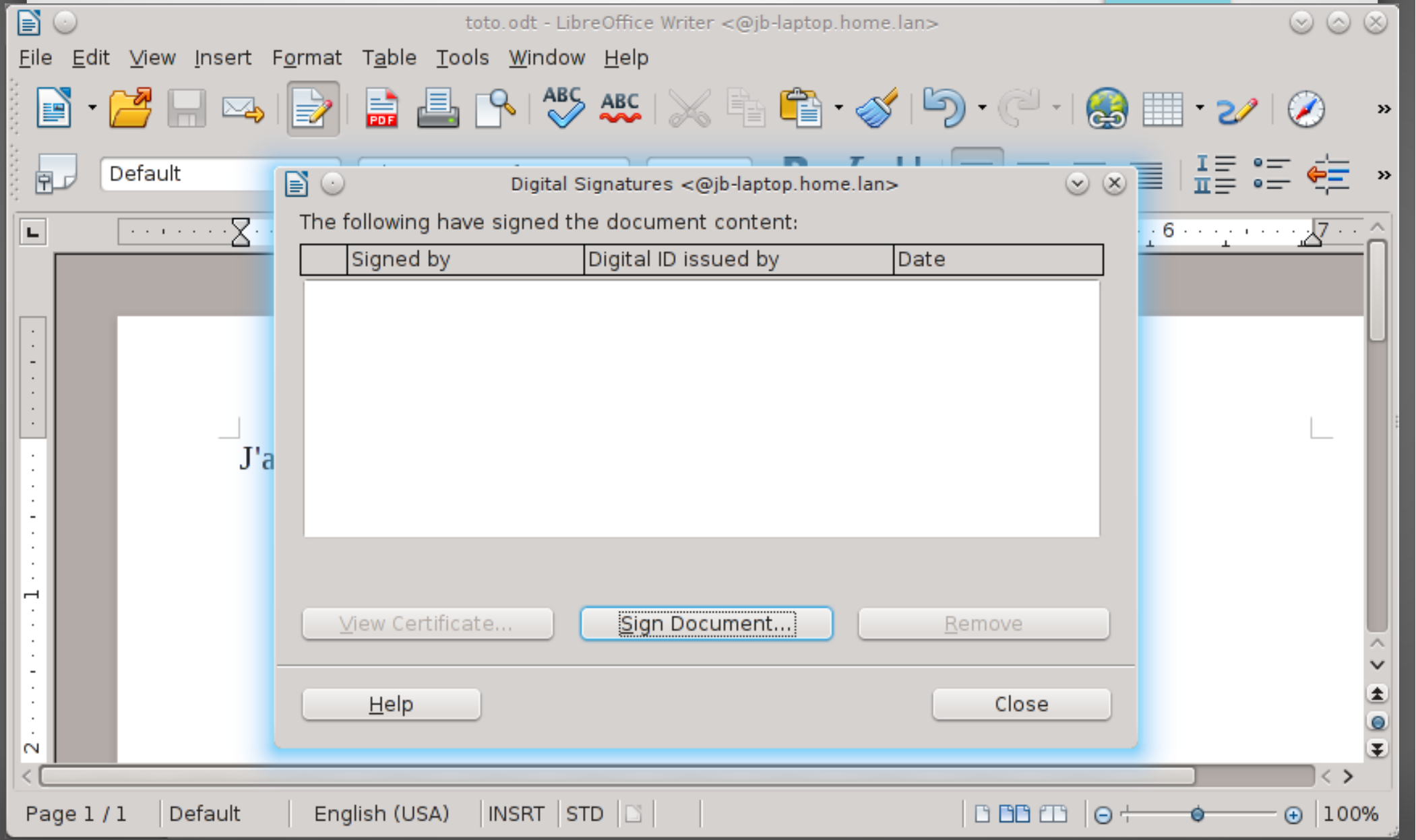
Signer un document avec un utilitaire intégré



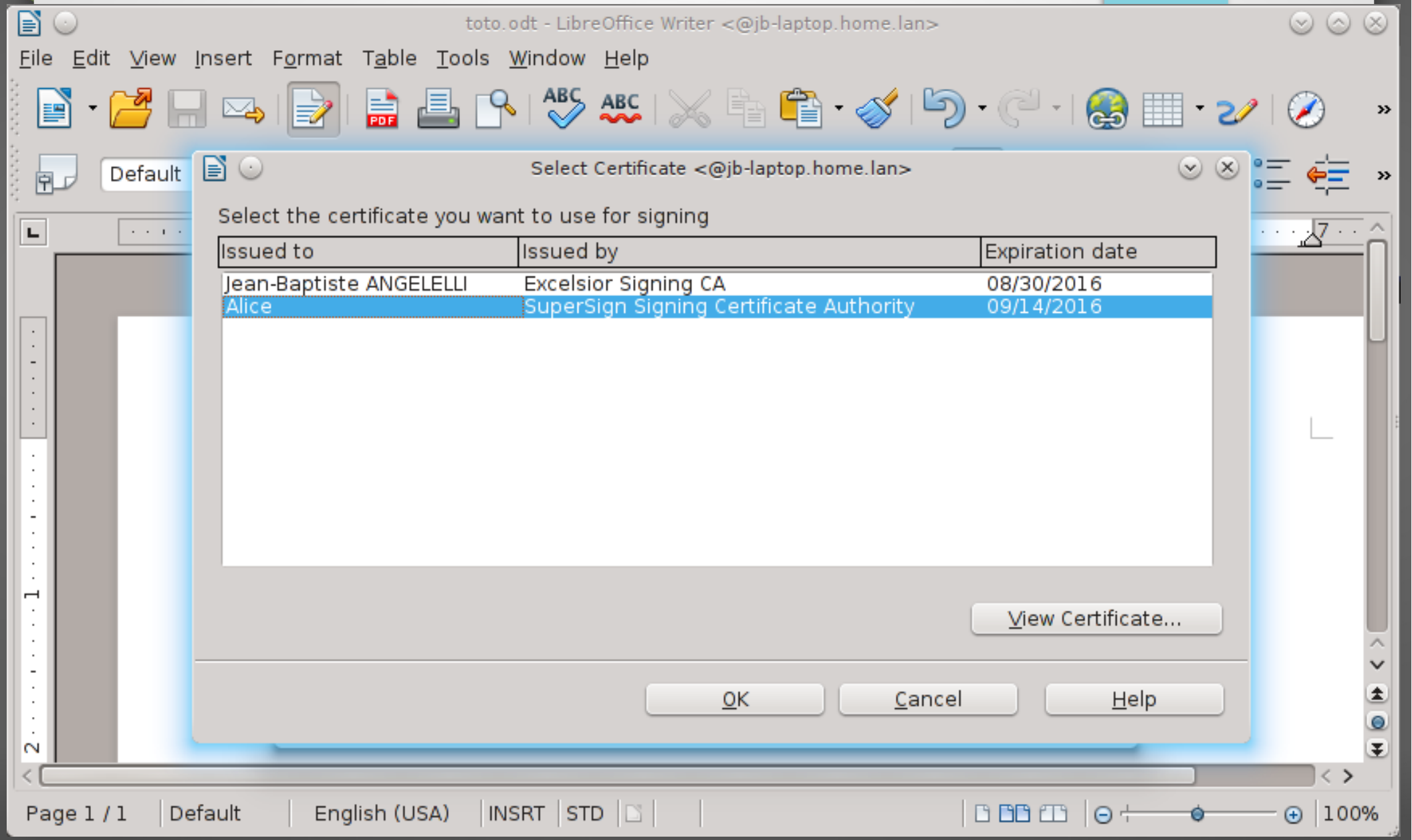
Signer un document avec un utilitaire intégré



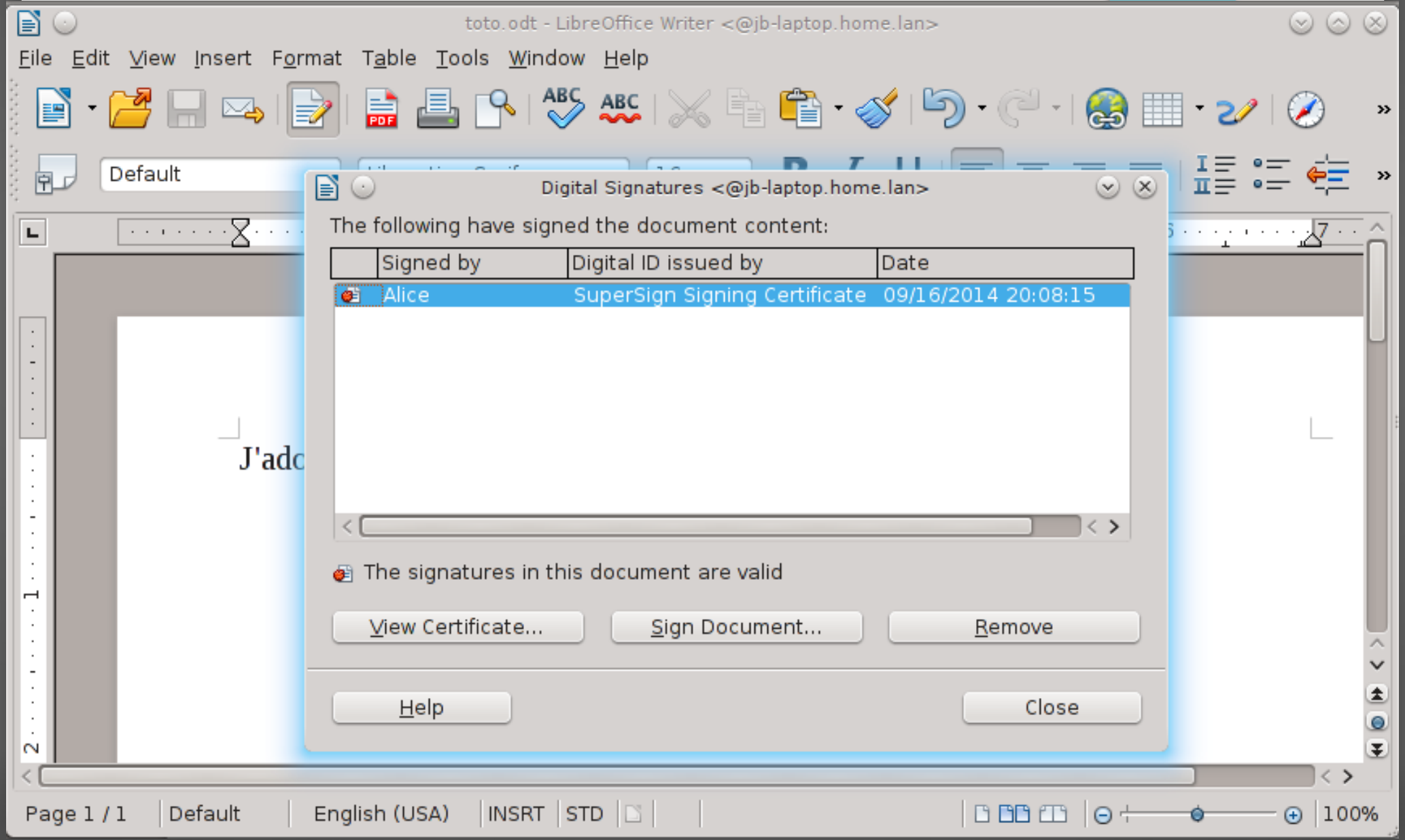
Signer un document avec un utilitaire intégré



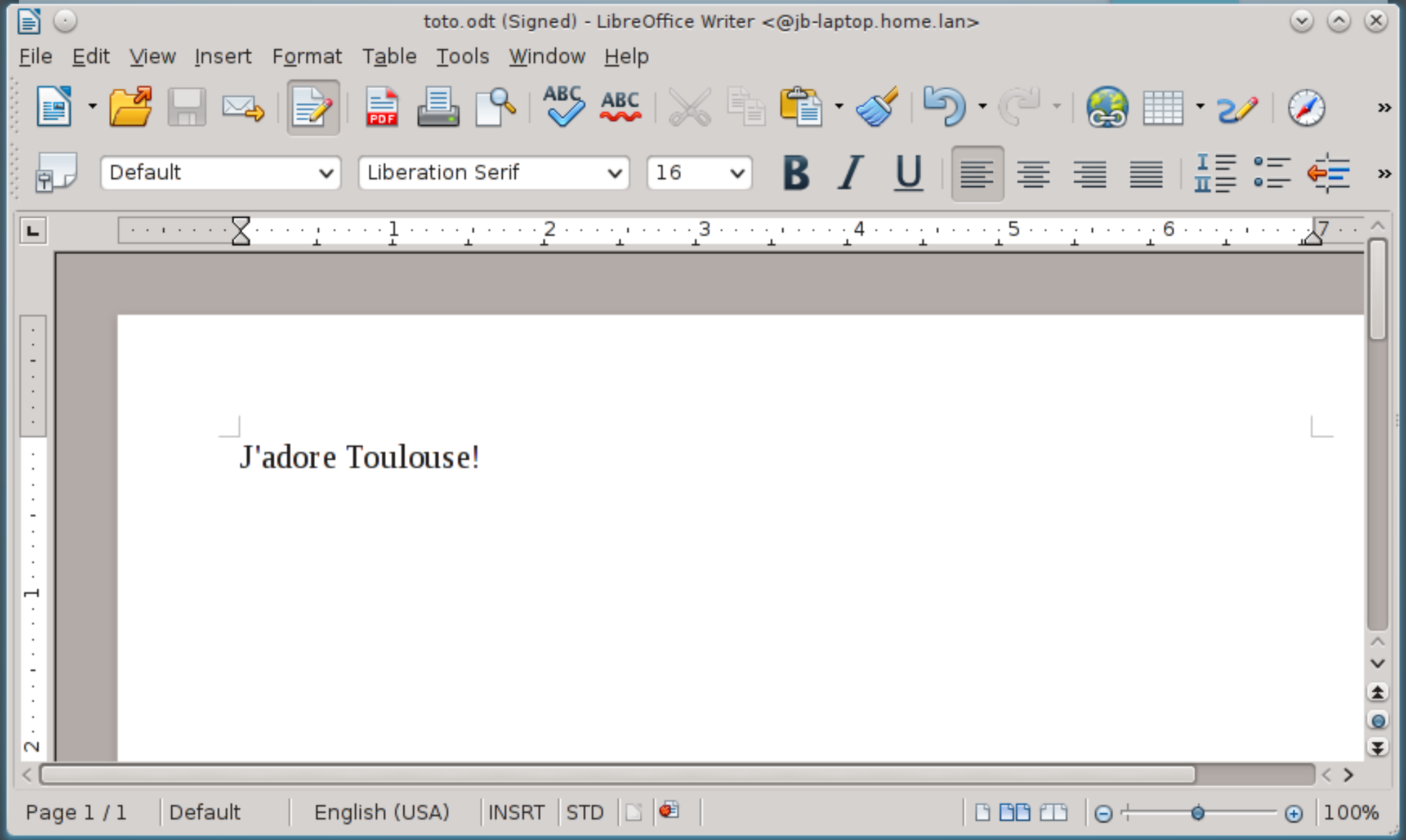
Signer un document avec un utilitaire intégré



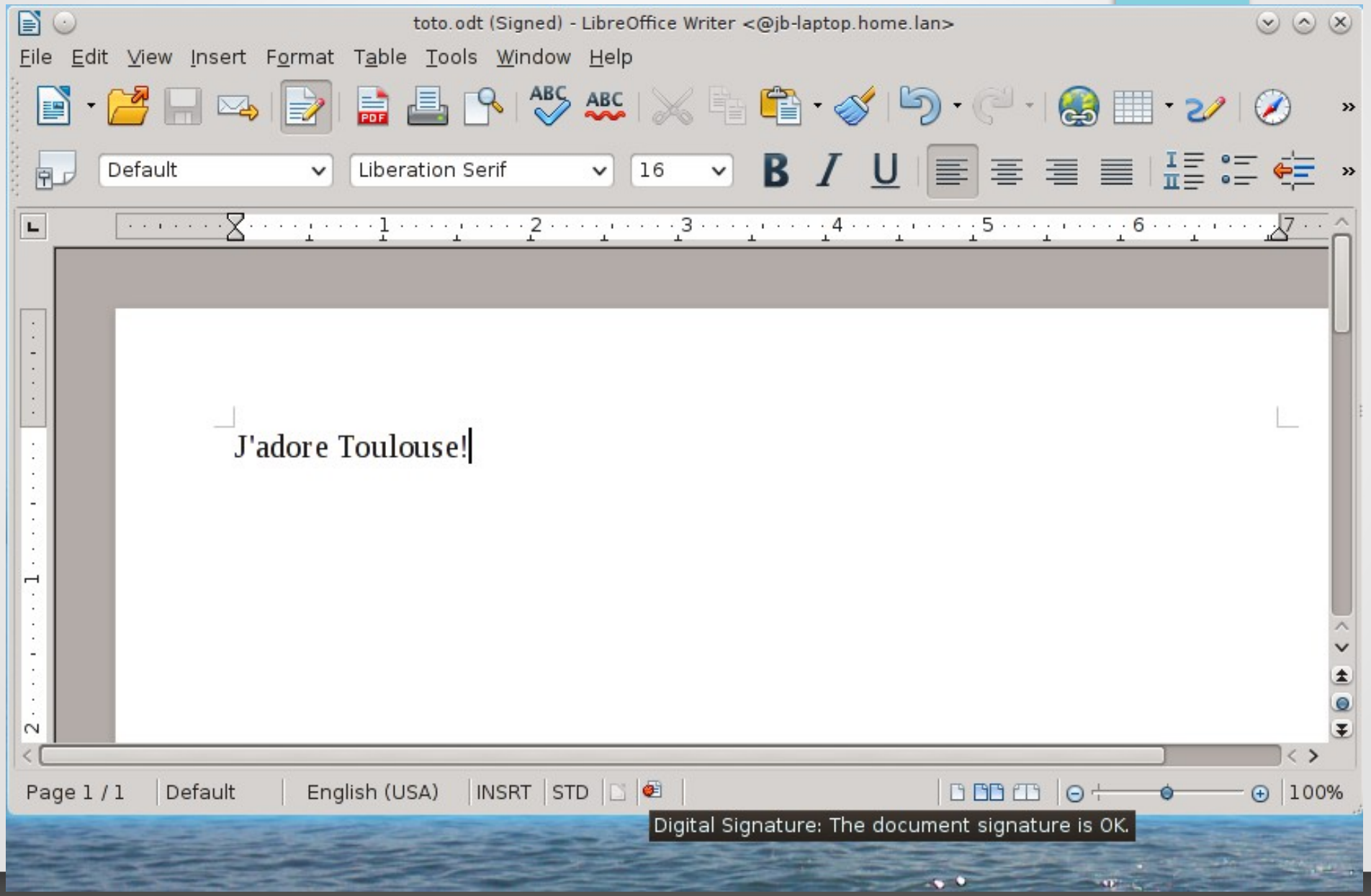
Signer un document avec un utilitaire intégré



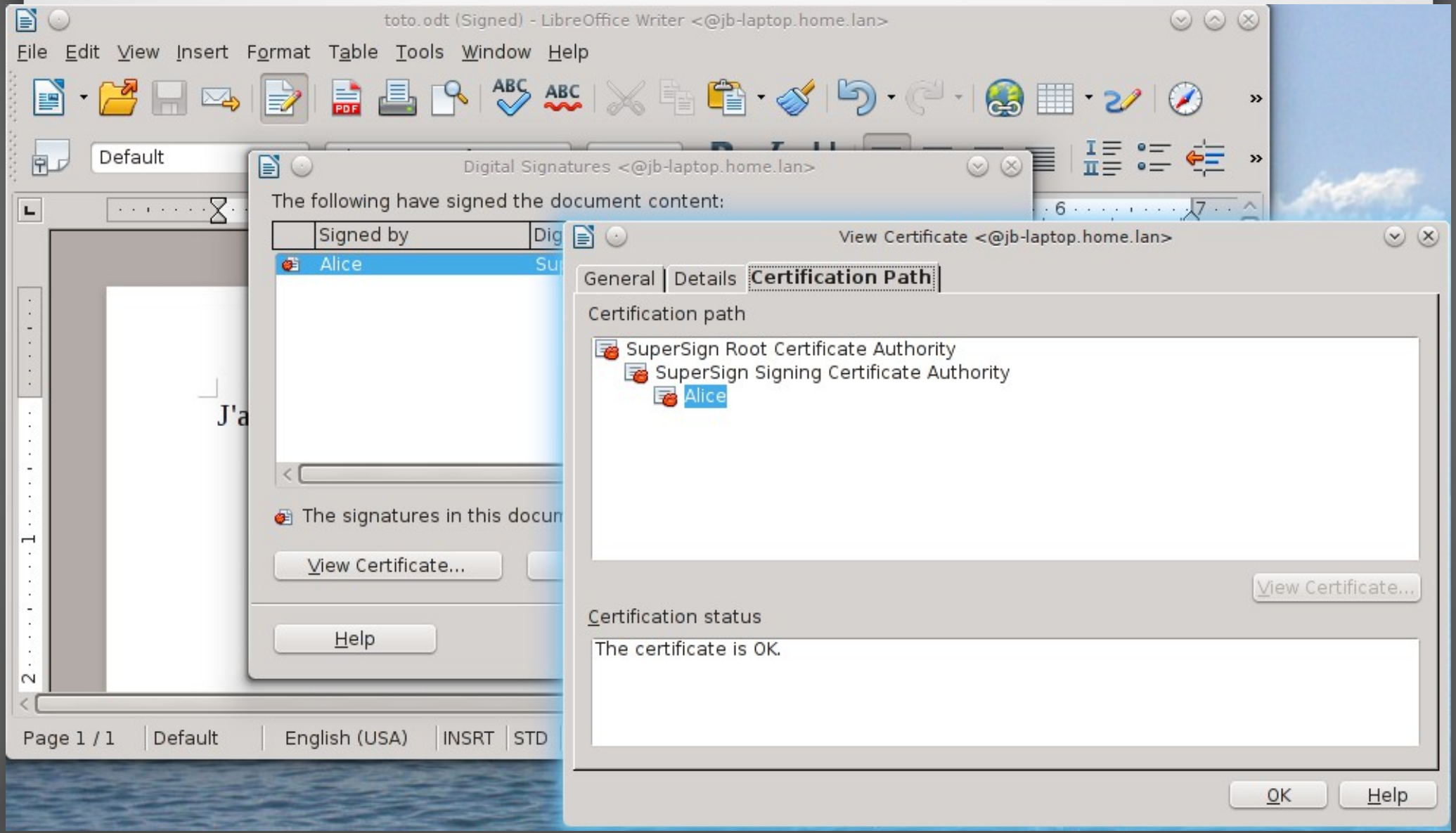
Signer un document avec un utilitaire intégré



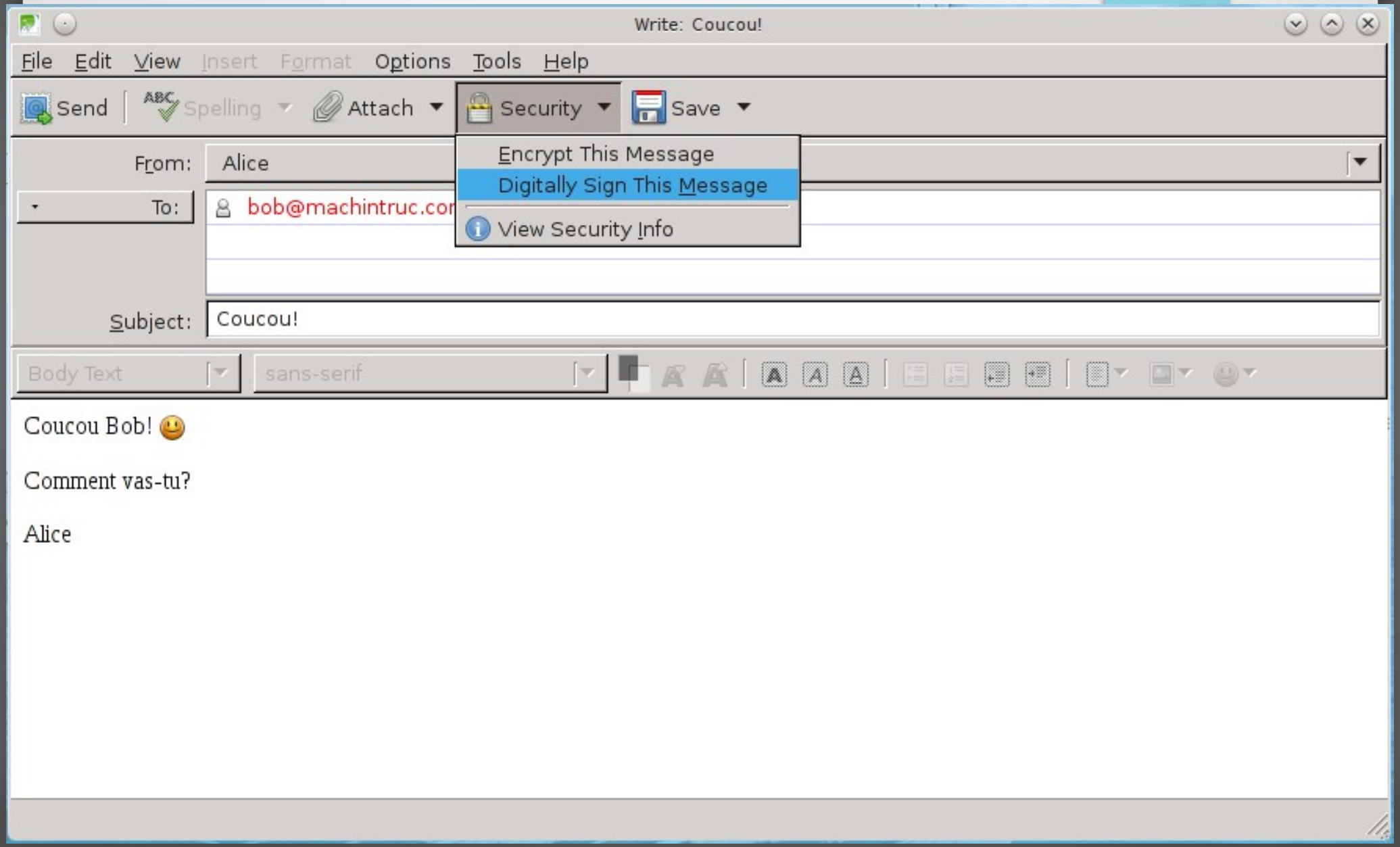
Signer un document avec un utilitaire intégré



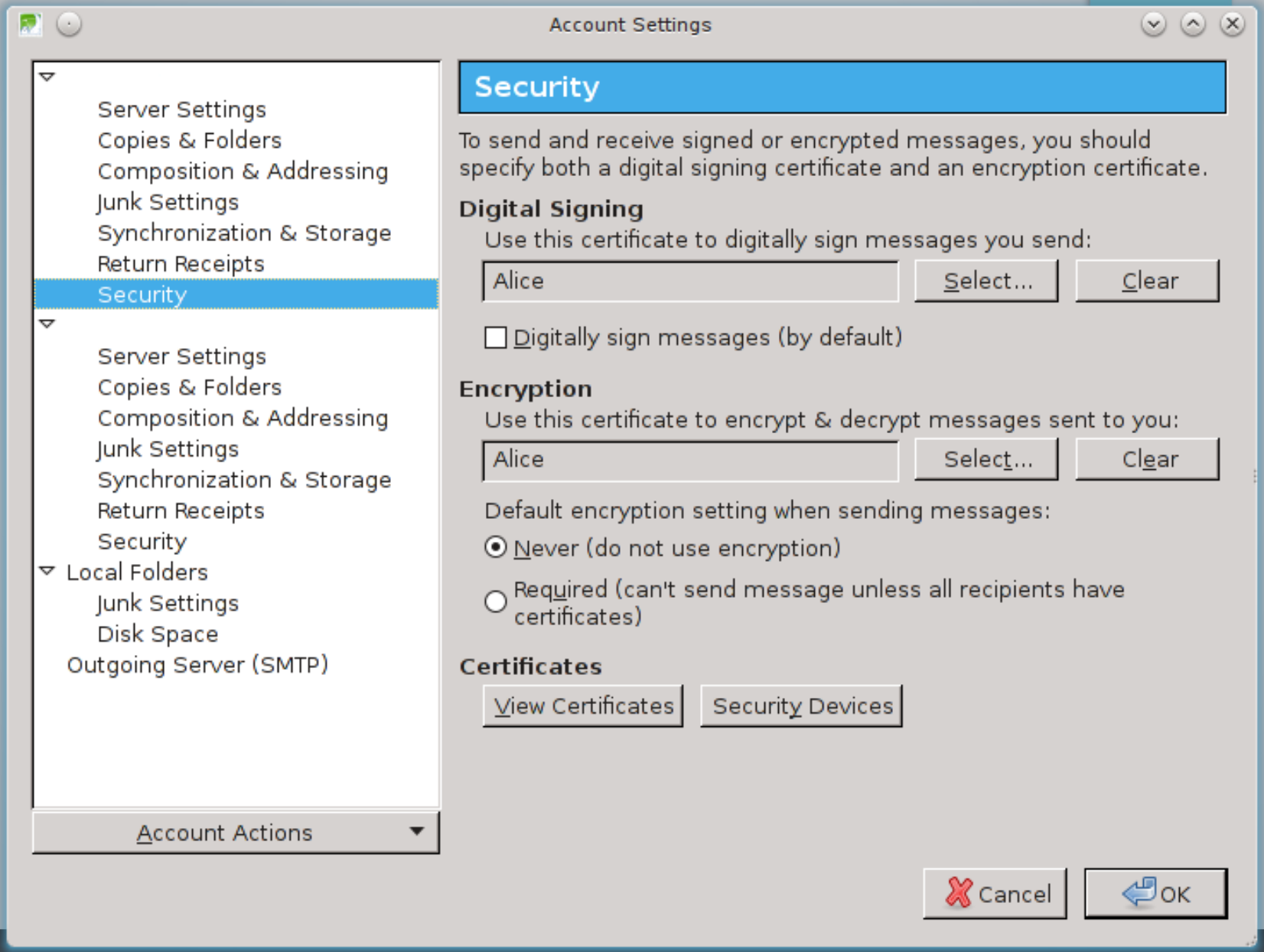
Signer un document avec un utilitaire intégré



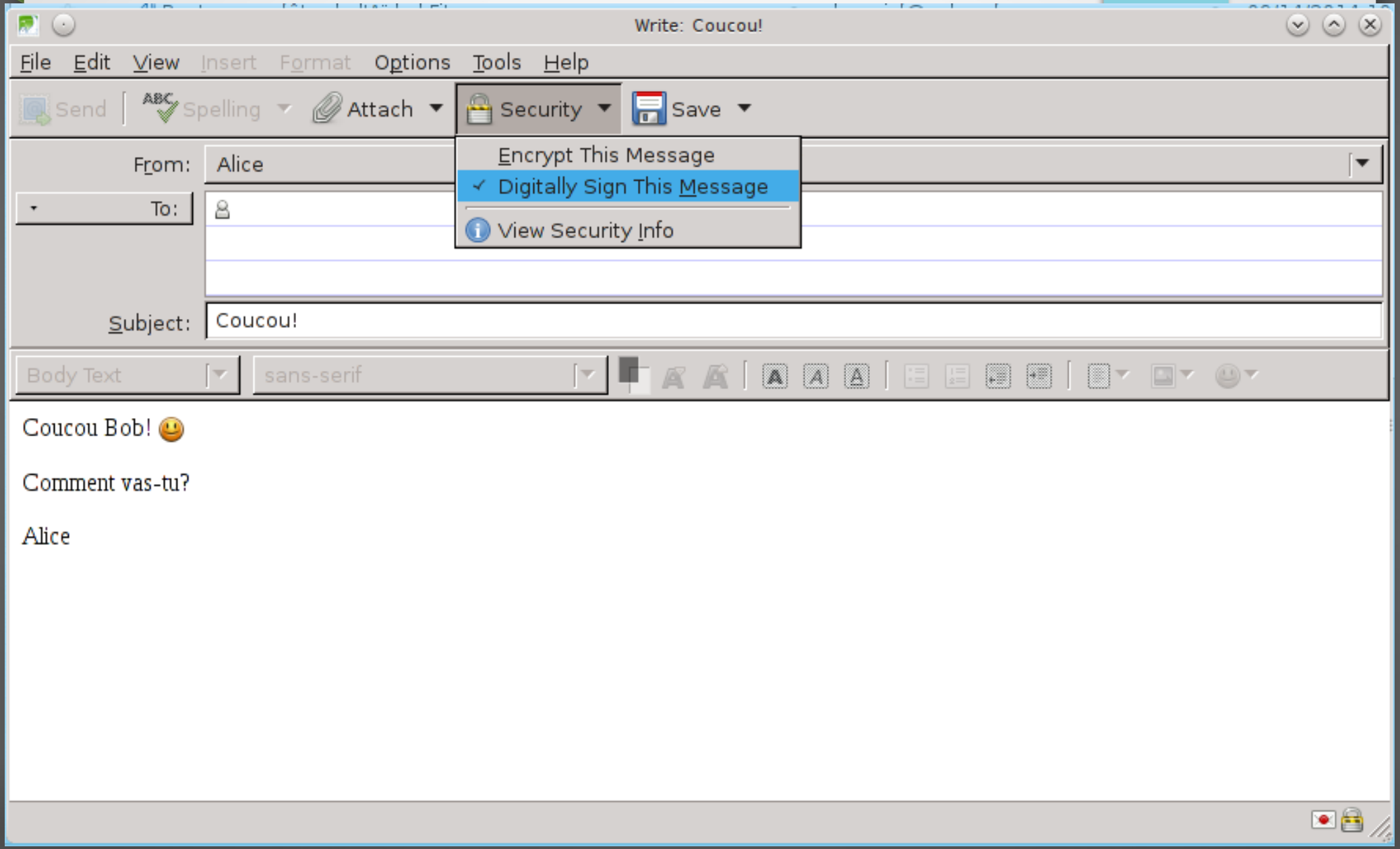
La signature d'e-mail



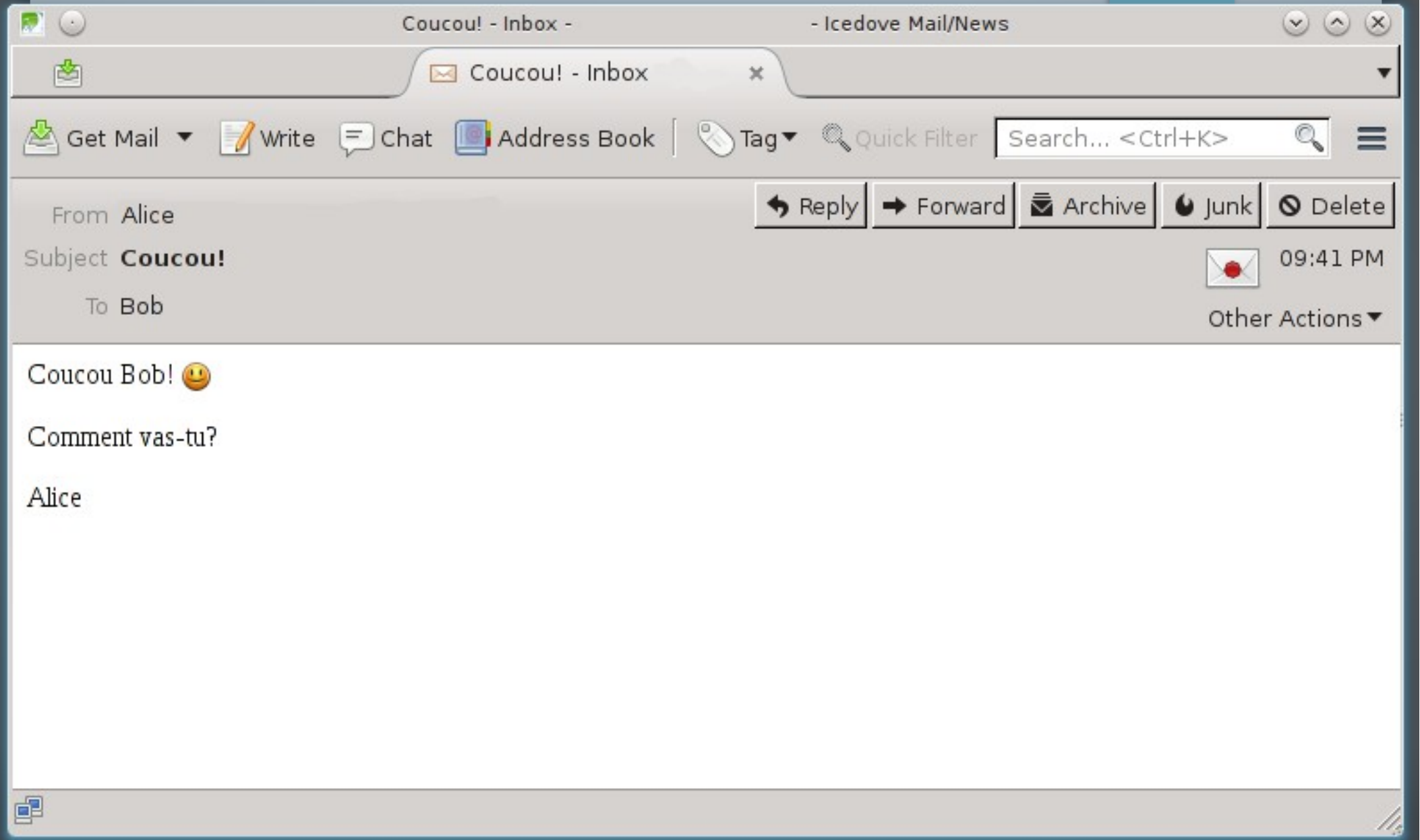
La signature d'e-mail



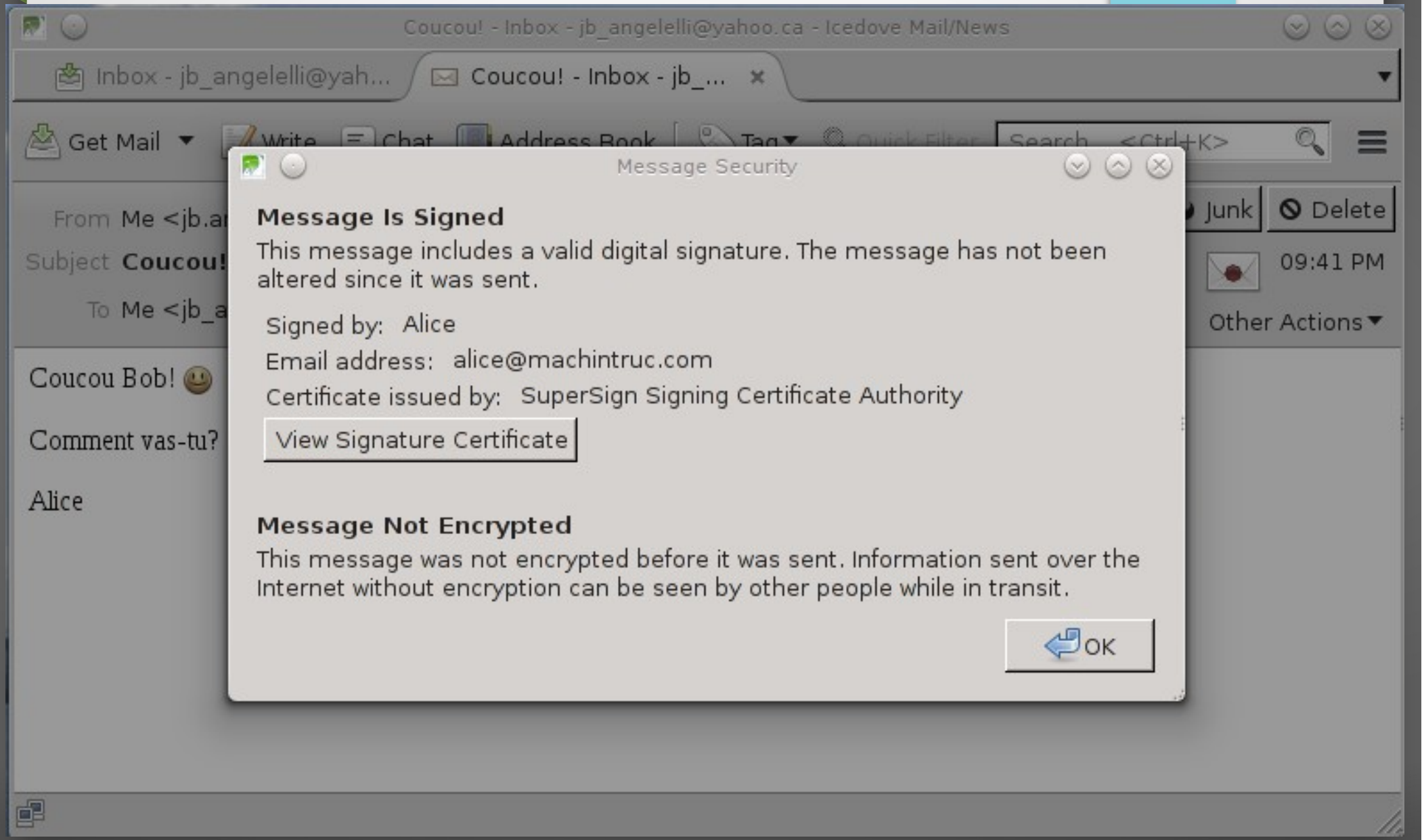
La signature d'e-mail



La signature d'e-mail



La signature d'e-mail



Les mathématiques de la cryptographie

- Factorisons en nombres premiers 15 =

Les mathématiques de la cryptographie

- Factorisons en nombres premiers 15 = 3 * 5

Les mathématiques de la cryptographie

- Factorisons en nombres premiers 15 = 3 * 5
- Factorisons en nombres premiers 143 =

Les mathématiques de la cryptographie

- Factorisons en nombres premiers 15 = 3 * 5
- Factorisons en nombres premiers 143 =11 * 13

Les mathématiques de la cryptographie

- Factorisons en nombres premiers 15 = 3 * 5
- Factorisons en nombres premiers 143 =11 * 13
- Factorisons en nombres premiers 1517 =

Les mathématiques de la cryptographie

- Factorisons en nombres premiers 15 = 3 * 5
- Factorisons en nombres premiers 143 = 11 * 13
- Factorisons en nombres premiers 1517 = 37 * 41

Les mathématiques de la cryptographie

- Factorisons en nombres premiers 15 = 3 * 5
- Factorisons en nombres premiers 143 = 11 * 13
- Factorisons en nombres premiers 1517 = 37 * 41

L'idée est qu'il est très difficile de factoriser des grands nombres.

En fait, c'est un problème NP-difficile, on ne connaît pas de méthode polynomiale.

L'échange de clés de Diffie-Helmann

TLS/SSL fait ça !

Charlie

Alice

Bob

On veut arriver à :

Clés privées symétriques

L'échange de clés de Diffie-Helmann

- Alice et Bob choisissent un nombre premier p et un nombre g (appelé base).
- Alice choisit un nombre au hasard a et fait $A = g^a \pmod p$, elle l'envoie à Bob.
- Bob choisit un nombre au hasard b et fait $B = g^b \pmod p$, il l'envoie à Alice.
- Bob fait A^b et il obtient K ($= (g^a)^b = g^{ab} \pmod p$)
- Alice fait B^a et elle obtient K aussi !

  ($= (g^b)^a = g^{ab} \pmod p$)

L'algorithme RSA

- Algorithme à clé publique
- Rivest, Shamir et Adleman
- Publié pour la première fois en 1977
- Inventé secrètement par Clifford Cocks en 1973 pour le compte du GCHQ

L'algorithme RSA

- Soit p,q deux (grands) nombres premiers
- $n = pq$, puis $\varphi(n) = (p-1)(q-1)$
- Soit e tel que $1 < e < \varphi(n)$ avec $pgcd(e, \varphi(n)) = 1$
- Soit $d = e^{-1} \pmod{\varphi(n)}$

n et e forment la clé publique

d est la clé privée

L'algorithme RSA

- Soit m l'entier clair
- Pour chiffrer, on fait $c = m^e \pmod n$
- Pour déchiffrer, on fait $m = c^d \pmod n$

- La force de l'algorithme vient du fait qu'il est très difficile de retrouver p et q à partir de n!

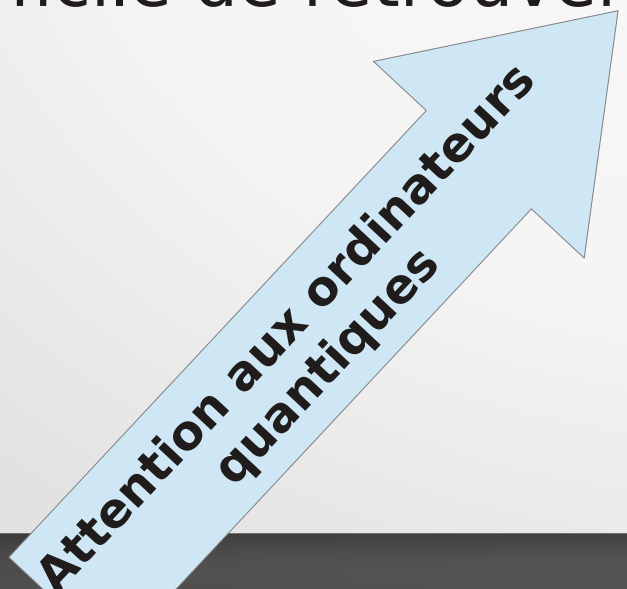
n et e forment la clé publique

d est la clé privée

L'algorithme RSA

- Soit m l'entier clair
- Pour chiffrer, on fait $c = m^e \pmod n$
- Pour déchiffrer, on fait $m = c^d \pmod n$

- La force de l'algorithme vient du fait qu'il est très difficile de retrouver p et q à partir de n!



Chiffrement par blocs

Bon ok ! Je sais chiffrer un nombre..
Maintenant comment je fais pour chiffrer un document entier?

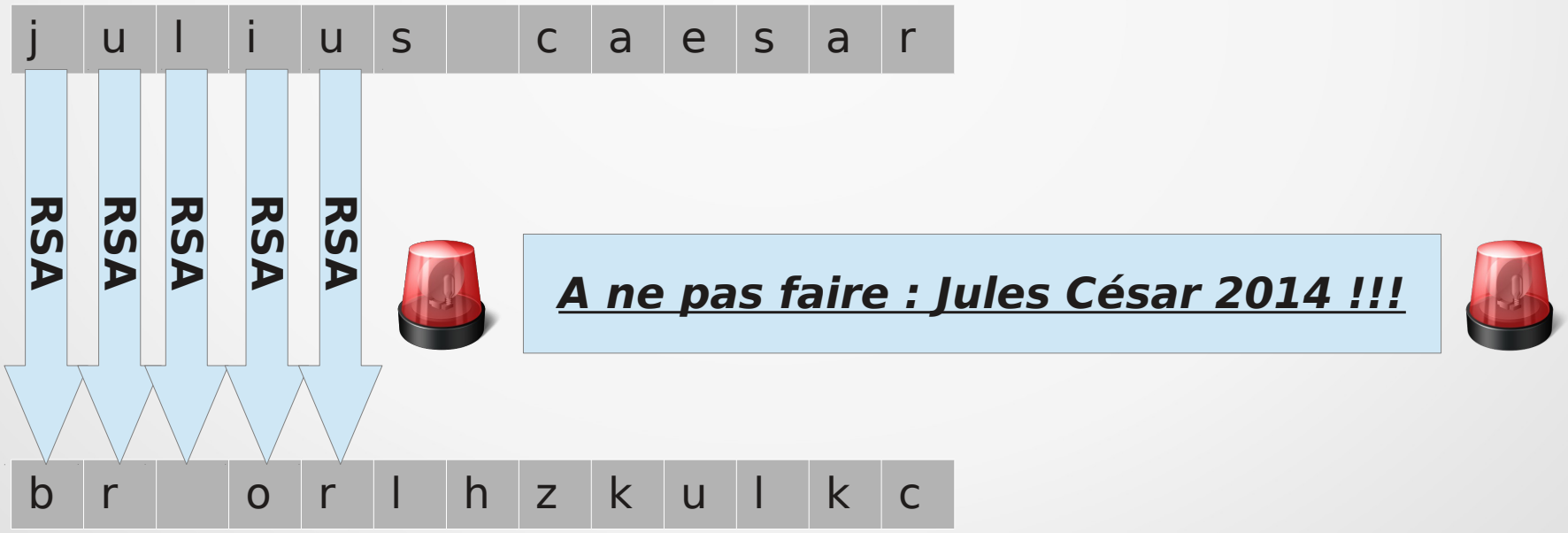
Il te faut un ***chiffrement par blocs***...

Chiffrement par blocs

- Une des deux grandes catégories de chiffrement (l'autre est le ***chiffrement par flot***)
- Découpe l'objet à chiffrer en blocs de taille fixée (par exemple 128 bits)
- Un chiffrement par blocs est tout simplement une fonction bijective qui à un bloc de 128 bits clair associe un bloc de 128 bits chiffré

Chiffrement par blocs : attention aux utilisations trop naïves !

Blocs trop petits...

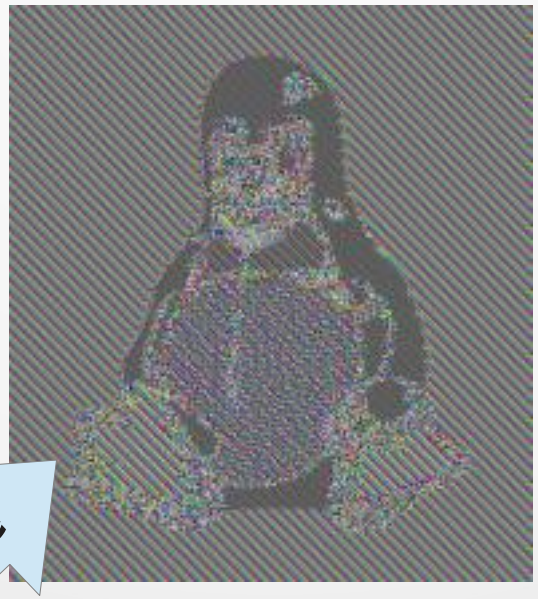


Chiffrement par blocs : attention aux utilisations trop naïves

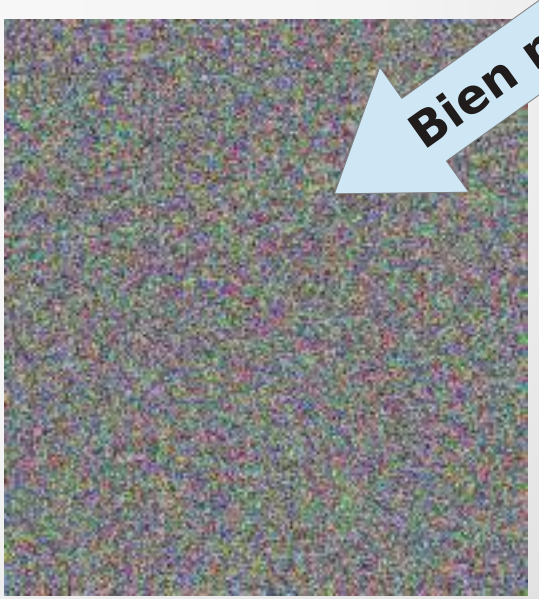
Electronic Codebook (ECB) : Blocs chiffrés indépendement les uns des autres.

Chain block cipher (CBC) : On combine (par XOR) le clair avec le chiffré du bloc précédent avant chiffrement

Larry Ewing
lewing@isc.tamu.edu
The Gimp



Avec ECB



Avec CBC

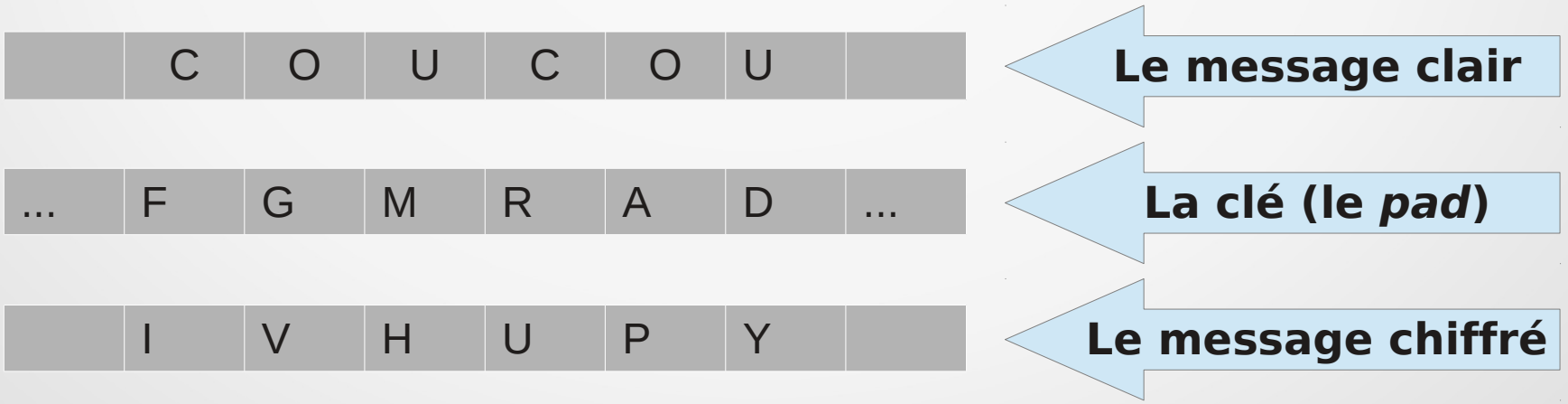
Pas terrible

Bien mieux

Le One Time Pad

On obtient le chiffré en additionnant le clair et la clé (modulo quelquechose)

Dans cet exemple, on additionne les lettres modulo 26



Le One Time Pad

- Si la clé est une suite complètement aléatoire, alors le système est parfaitement (mathématiquement) sûr (démontré par Claude Shannon).
- L'idée est que si la clé est inconnue mais rigoureusement aléatoire, alors pour un chiffré donné, tous les clairs possibles sont équi-probables.
- Le système a une importante utilisation historique au XX^{ème} siècle.
- La génération d'une clé aléatoire est une importante limitation en pratique.

La Commission Nationale Informatique et Libertés

- Commission de l'état
- S'assurer que les systèmes informatiques ne portent pas atteinte aux droits de l'hommes, aux libertés individuelles et publiques, à la vie privée
- Droit d'information, droit d'accès, droit de rectification ou de radiation, droit d'opposition, droit d'accès indirect

Remerciements pour les icônes

- www.iconarchive.com
- http://www.afterglow.ie/icons.html

- http://www.designcontest.com/

- http://icons8.com/

- http://www.aha-soft.com/

- http://www.psdgraphics.com

Et voilà! Questions?

Merci de votre attention !

jean-baptiste.angelelli@centraliens.net