

Secteur Tertiaire Informatique
Filière « Etude et développement »

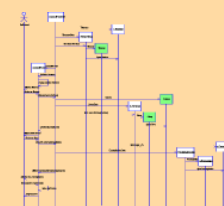
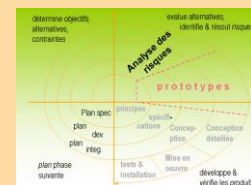
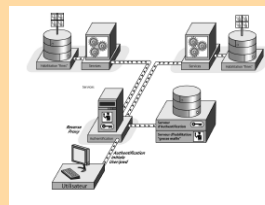
Séquence « Développer des pages web en lien
avec une base de données »

Utiliser la cryptographie et les mécanismes de
sécurité du Web

Apprentissage

Mise en situation

Evaluation



Version	Date	Auteur(s)	Action(s)
1.0	23/08/16	Lécu Régis	Création du document

1. Introduction	6
2. La cryptographie : aperçu.....	7
2.1 Vocabulaire	7
2.2 Un peu d'histoire	8
2.3 Chiffrements symétrique et asymétrique	8
2.4 Signature électronique	9
2.5 Certificats électroniques	9
3. La cryptographie en Java	12
3.1 Chiffrement d'un message avec l'algorithme symétrique AES.....	12
3.1.1 Génération de la clé secrète	12
3.1.2 Chiffrement à l'aide de la clé secrète	12
3.1.3 Déchiffrement avec la (même) clé secrète	13
3.2 <i>Wrap/unwrap</i> d'une clé avec l'algorithme symétrique AES.....	13
3.3 Chiffrement d'un message avec l'algorithme asymétrique RSA.....	14
3.3.1 Génération d'une paire de clés : publique, privée	14
3.3.2 Chiffrement avec la clé publique	15
3.3.3 Déchiffrement avec la clé privée	15
3.4 Signature d'un message en « SHA1 avec RSA »	15
3.4.1 Les étapes.....	15
3.4.2 Génération de la signature par l'émetteur	16
3.4.3 Validation de la signature par le récepteur	16
3.5 Intégrer un certificat	17
4. Configuration d'une liaison HTTPS	18
4.1 Créer un certificat auto-signé	19
4.2 Ajouter une liaison HTTPS à un site Web	19
4.3 Récupérer un certificat de test depuis une autorité de certification.....	21

Objectifs

A l'issue de cette séance, le stagiaire sera capable de :

- Connaître les notions de base de la cryptographie : objectifs, concepts usuels ;
- Utiliser la cryptographie dans le développement en faisant appel aux bibliothèques Java : par exemple, pour chiffrer le mot de passe avant l'envoi sur le réseau ;
- Comprendre et utiliser les mécanismes de sécurité du Web basés sur la cryptographie :
 - certificats, liaisons sécurisées (*SSL/TLS*, *HTTPS*) : par exemple, sécuriser la liaison avec le serveur Web, en configurant une liaison HTTPS entre le client et le serveur (optionnel) ;
 - connaître les limites d'une liaison sécurisée HTTPS (optionnel).

Pré requis

Pas de pré requis en cryptographie.

Le stagiaire doit être sensibilisé aux risques d'attaques réseau (séance « *Prendre en compte le réseau dans la sécurité du Web* ») pour bien comprendre l'intérêt des techniques de chiffrement et de signature.

Il doit connaître le langage Java pour réaliser la partie pratique : appel des API de cryptographie en Java.

Méthodologie

Ce document peut être utilisé en présentiel ou à distance.

Seule la mise en pratique optionnelle de configuration d'une liaison HTTPS requiert l'aide d'un formateur.

Mode d'emploi

Symboles utilisés :



Renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur.



Propose des exercices ou des mises en situation pratiques.



Point important qui mérite d'être souligné !

Ressources

Documents du projet CyberEdu :

- [CyberEdu_module_3_reseau_et_applicatifs.pdf](#) (3 Les bases de la cryptographie)
- [CyberEdu_crypto.odp](#)

Sur l'utilisation de la cryptographie en Java : cours Java de JM Doudoux, chapitres 29 « Sécurité », 30 « JCA » et 31 JCE

En anglais, la présentation de D. Wheeler : [Secure-Software-7-Cryptography](#)

Tutorial OWASP : sur les limites du protocole https : [Strict Transport Security.mp4](#)

Document CyberEdu pour le formateur : [Fiches-authentification-1.5.pdf](#)

- Ce document propose plusieurs scénarios pédagogiques qui utilisent la cryptographie pour le réseau, le développement, ou les deux métiers.
- Il peut donc être exploité pour aller plus loin et construire des mises en situation.

1. INTRODUCTION

L'objectif de cette séance n'est pas de s'improviser expert en cryptographie.

Mais à l'inverse, il ne serait pas raisonnable de faire l'impasse sur la cryptographie, alors que même l'utilisateur de base est confronté aux certificats dans les navigateurs Web, pour les liaisons HTTPS.

La cryptographie est utilisée dans différentes situations professionnelles qui vont de la conception au déploiement de l'application :

- en partant des besoins de sécurité de l'application identifiés dans la phase d'analyse, le concepteur peut décider de chiffrer certaines données et d'utiliser des certificats ;
- le développement s'appuiera en général sur des bibliothèques existantes : savoir qu'il ne faut pas bricoler un chiffrement mais toujours faire appel à des standards, est déjà une compétence professionnelle en développement sécurisé ;
- le déploiement pourra utiliser une liaison sécurisée.

Nous allons commencer par un aperçu de la cryptographie, en nous appuyant sur les documents CyberEdu.

Puis nous mettrons en pratique ces notions, en développement :

- en parcourant les bibliothèques de cryptographie de Java (avec le document de Jean-Michel Doudoux) ;
- en chiffrant un message pour le protéger d'une écoute réseau ;
- en le signant pour garantir son authenticité et l'identité de son émetteur ;
- en utilisant un certificat auto-généré ou issu d'une autorité de certification.

La séance se terminera par une mise en pratique optionnelle : configuration d'une liaison sécurisée HTTPS sur un serveur *Microsoft IIS*.

Ce savoir-faire relève en principe des métiers du système et du réseau, mais un développeur concerné par la sécurité de son application, doit posséder un minimum de culture technique dans ce domaine :

- pour être un interlocuteur crédible des équipes système et réseau, il doit pouvoir formuler clairement les contraintes de sécurité requises par son application ;
- dans une PME, il peut être amené à configurer lui-même une liaison HTTPS vers le serveur Web ;
- il n'y a pas de frontière nette entre le paramétrage du système et des serveurs, et l'utilisation des bibliothèques de cryptographie, en développement.

2. LA CRYPTOGRAPHIE : APERÇU

Ouvrez le document CyberEdu : [CyberEdu_module_3_reseau_et_applicatifs.pdf](#).

(3 Les bases de la cryptographie)

Si vous découvrez ce chapitre en autonomie, suivre ce



Guide de lecture

Ce document fournit une approche simple du vocabulaire et des concepts de la cryptographie¹.

2.1 VOCABULAIRE

(p. 41)

Les disciplines proches de la cryptographie :

- **Cryptanalyse** : l'attaque de procédés cryptographiques
- **Cryptologie** : Cryptographie + Cryptanalyse. C'est la science du secret.



Portail sur la cryptologie : <https://fr.wikipedia.org/wiki/Portail:Cryptologie>

La cryptographie joue un rôle essentiel pour garantir trois des critères de sécurité **DICP** (Disponibilité, Intégrité, Confidentialité, Preuve) que nous avons déjà présentés : intégrité, confidentialité et preuve.

La **Disponibilité** d'une application ou d'un serveur ne repose pas directement sur la cryptographie : un service peut être interrompu par une coupure réseau, un problème serveur ou un bogue de l'application.

Mais l'**Intégrité** contribue indirectement à la Disponibilité, en interdisant aux utilisateurs malveillants de modifier le paramétrage du serveur, de supprimer des données essentielles etc.

Sur la **Preuve** (authentification et non répudiation) :

- **l'authentification** a pour but de vérifier **l'identité** dont une entité se réclame. Généralement l'authentification est précédée d'une identification qui permet à cette entité de se faire reconnaître du système par un élément dont on l'a doté ;
- **identification** : permet de connaître l'identité, par un compte utilisateur, par une carte bancaire etc.
- **authentification** : vérifie cette identité à l'aide d'un secret, d'un objet, d'un caractère (photo, biométrie), d'un savoir faire (signature)
- **Non répudiation** : la répudiation est le fait de nier avoir participé à des échanges, totalement ou en partie.
- **Deux formes de non répudiation** : la « Non-répudiation de l'origine » garantit que l'émetteur d'informations ne peut pas nier impunément avoir envoyé les informations. La

¹ Ce commentaire s'appuie en partie sur le document ANSSI de José Araujo sur la cryptographie (fourni)
Utiliser la cryptographie et les mécanismes de sécurité du Web

« Non-répudiation de la réception » garantit que le destinataire des informations ne peut pas nier impunément avoir reçu les informations.

(p. 42)

Il faut bien distinguer le **chiffrement** de l'**encodage** : le chiffrement produit un texte non compréhensible qui ne peut être **déchiffré** que par le possesseur de la clé.

Il y a un secret partagé entre l'émetteur et le récepteur, qui ne repose pas sur l'algorithme de chiffrement (connu de tous), mais sur la possession de la clé.

L'opération d'encodage est une transformation sans clé (et sans secret) : par exemple, dans une page HTML, le caractère € sera encodé en **&euro** ;

Le schéma décrit un **chiffrement symétrique** où la même clé secrète (en jaune) sert au chiffrement et au déchiffrement du message : elle n'est connue que par les deux interlocuteurs qui partagent le secret.

(p. 43)

La signature électronique s'appuie sur un **chiffrement asymétrique** : l'émetteur signe son message avec sa clé privée (en jaune et noir) qu'il est le seul à posséder ; la signature diffère pour chaque message signé.

Le récepteur déchiffre la signature du message avec la clé publique de l'émetteur (connue de tous, en jaune sur le schéma) :

- comme une clé publique correspond à une clé privée unique, on est sûr de l'identité de l'émetteur (en supposant pour le moment que le possesseur de la clé privée soit bien celui que l'on croit !)
- comme une signature correspond à un message unique, on est sûr que le message n'a pas été altéré entre l'émetteur et le récepteur.

2.2 UN PEU D'HISTOIRE

(pp. 44-48)

La connaissance de l'historique n'est pas indispensable pour pratiquer la cryptographie actuelle, qui repose sur des algorithmes mathématiques implémentés dans des bibliothèques.

Mais nous recommandons ce chapitre pour les curieux.



On pourra lire aussi les chapitres « [Histoire et principes](#) » et « [Aspects légaux](#) » dans l'exposé ANSSI de José Araujo (document fourni [2_CyberEdu_crypto.odp](#))

2.3 CHIFFREMENTS SYMETRIQUE ET ASYMETRIQUE

(pp. 49-53)

Les slides présentent clairement les principes des chiffrements symétrique et asymétrique et leurs avantages et inconvénients respectifs.

Nous mettrons en œuvre en Java les algorithmes de chiffrement **AES** (symétrique) et **RSA** (asymétrique).



Pour aller plus loin : voir l'exposé de José Araujo : chap. « [Cryptographie symétrique](#) » et « [Cryptographie asymétrique](#) ».

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

2.4 SIGNATURE ELECTRONIQUE

(pp. 54-57)

Les slides présentent clairement le principe de la signature électronique.

Nous mettrons en œuvre en Java un algorithme de signature basé sur **RSA** pour le procédé cryptographique et **SHA** pour le condensat.



Précision sur le condensat, ou fonction de « hachage » (hash code) :

exposé de José Araujo : chapitre « [fonction de hachage](#) »

et wiki : https://fr.wikipedia.org/wiki/Fonction_de_hachage

2.5 CERTIFICATS ELECTRONIQUES

(pp. 58-63)

Les slides présentent clairement le principe des certificats électroniques.

Nous les mettrons en œuvre en Java en créant un certificat auto-signé (fichier avec l'extension .cer) et en le gérant avec les bibliothèques de cryptographie.



Pour aller plus loin, voir l'exposé de José Araujo :

(p. 54) Attaque par « l'homme du milieu » (*Man in the Middle*) :

- montre la nécessité d'utiliser des certificats issus d'une autorité de confiance, car l'utilisation d'une paire clé publique-clé privée sur un réseau public ne suffit pas à garantir la confidentialité et l'intégrité des échanges ;
- en effet, un utilisateur malveillant (*Claude*, en noir) à l'écoute sur le réseau, peut intercepter les clés publiques des vrais utilisateurs (*Alice* et *Bob*), leur envoyer des clés publiques erronées en usurpant leur identité, puis lire, voire modifier les messages chiffrés par ces clés publiques, avant de les renvoyer à leur vrai destinataire.

(p. 58) Norme sur les certificats : X.509



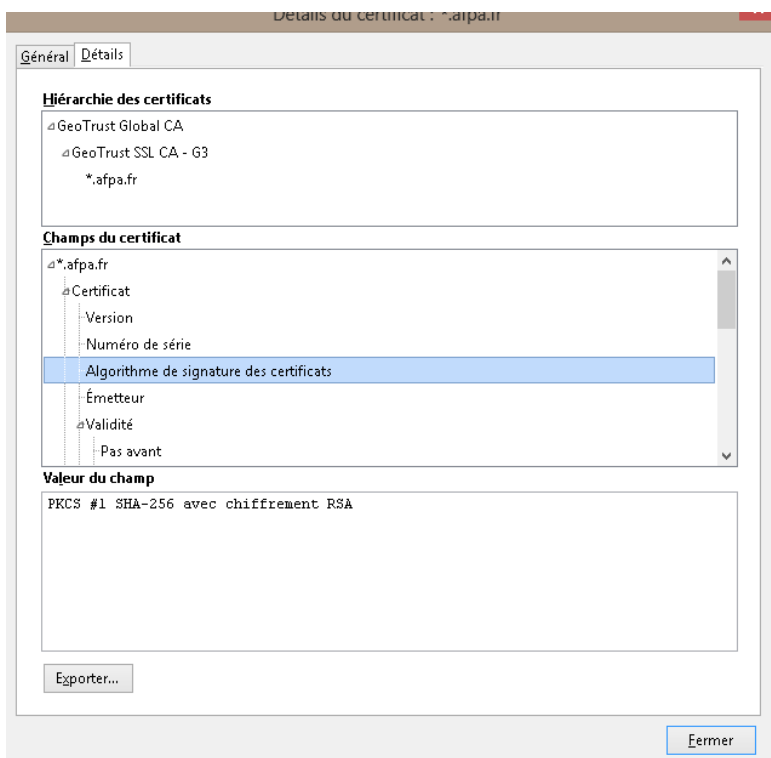
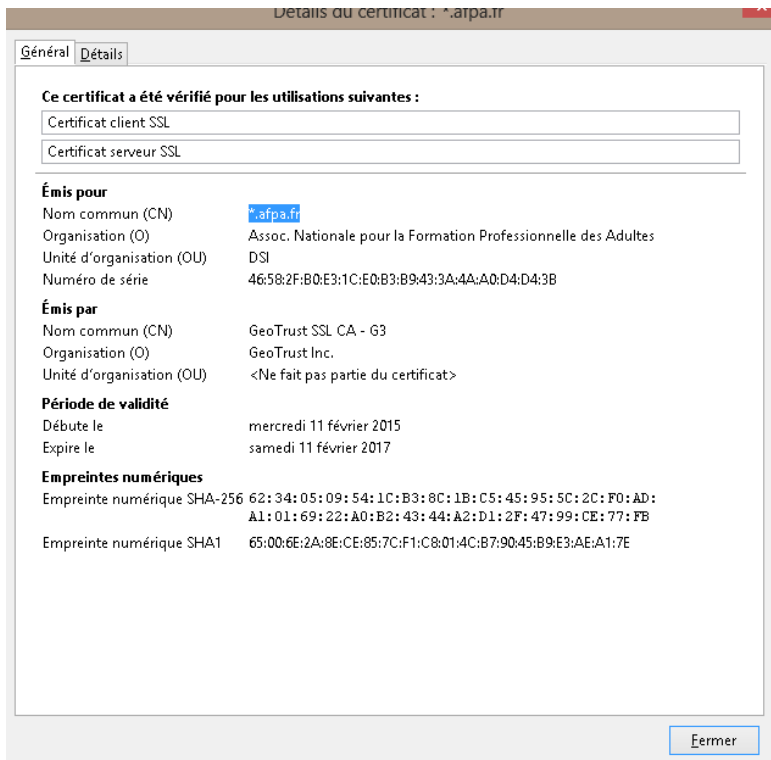
Mise en pratique

1) Affichez le certificat de la messagerie Afpa : <https://owa.afpa.fr>

- cliquez sur le cadenas de Firefox
- recherchez la signification des différents champs du certificat. On s'appuiera sur la description de la norme X. 509 par José Araujo pour identifier les champs :
 - o quelle est l'autorité de certification qui a émis le certificat de l'Afpa ?
 - o par quelle autorité de plus haut niveau, le certificat de l'autorité de certification est-il signé ?
 - o quelle est la durée de validité de ce certificat ?
 - o quel est son usage ?

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »



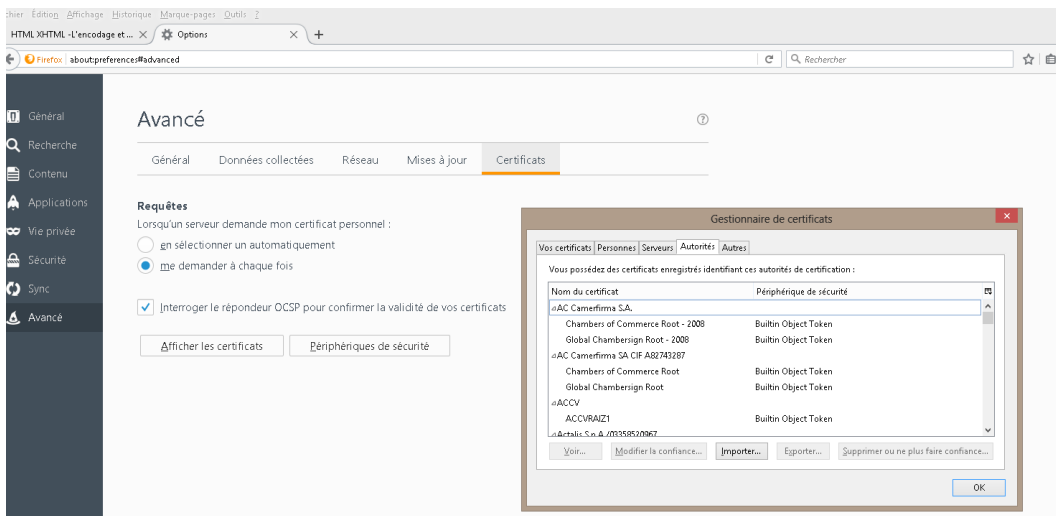
2) Visualisez les autorités de certification reconnues par votre navigateur :

Sous Firefox :

Utiliser la cryptographie et les mécanismes de sécurité du Web

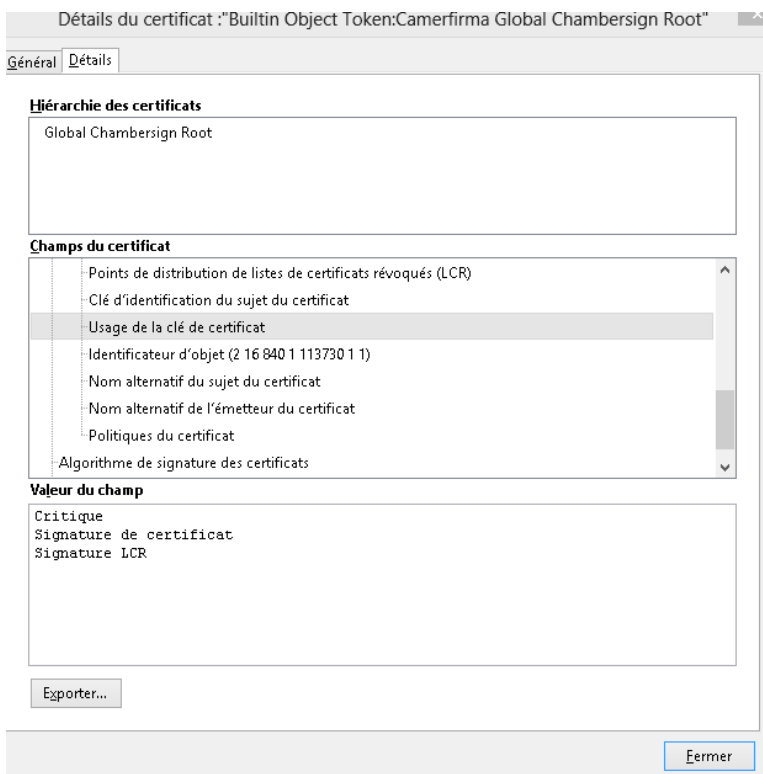
Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

- menu Outils / Option / Avancé
- onglet Certificats/ Autorités.



Exemple de certificat racine :

- l'usage du certificat est « *Signature de certificat* ».
- l'organisme est au sommet de la hiérarchie (c'est une « racine »)
- il est son propre émetteur.



Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

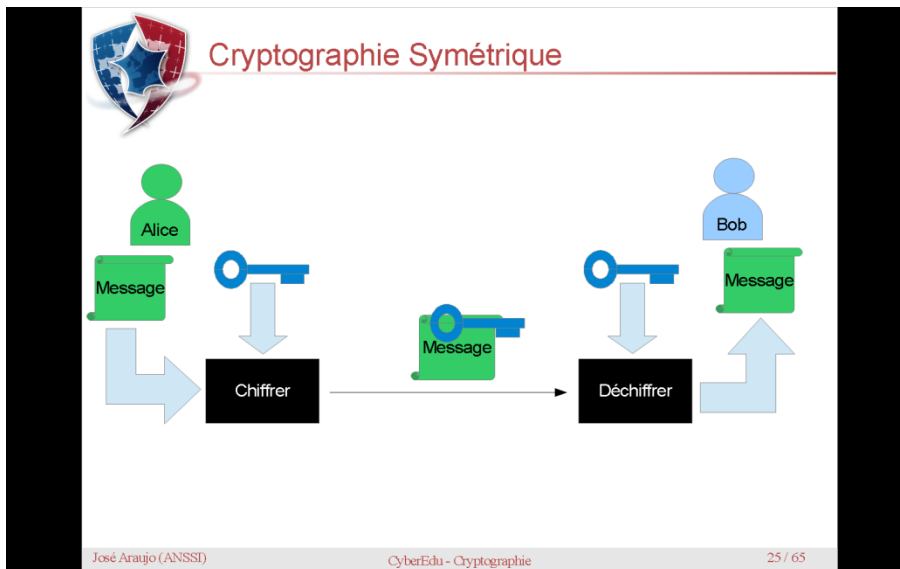
3. LA CRYPTOGRAPHIE EN JAVA

Nous allons maintenant mettre en pratique la cryptographie, en utilisant les API Java : **JCA** (*Java Cryptography Architecture*) et **JCE** (*Java Cryptography Extension*).

Nous prendrons comme documentation de référence le didacticiel de Jean-Marie Doudoux, « *Développons en Java* » : http://www.jmdoudoux.fr/accueil_java.htm (chap. 22 à 24).

3.1 CHIFFREMENT D'UN MESSAGE AVEC L'ALGORITHME SYMETRIQUE AES

Lire ou relire l'exposé de José Araujo : chap. « *Cryptographie symétrique* » :



AES est un algorithme de chiffrement par bloc plus récent que DES, et encore considéré comme sûr (en janvier 2015).

Pour les curieux, fonctionnement interne de l'algorithme sur wiki :

https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Pour la mise en œuvre en Java, lire le chap. 24 « *Des exemples d'utilisation d'algorithmes de chiffrement* », mise en œuvre de l'algorithme AES, du didacticiel « *Développons en Java* ».

Quelques repères :

3.1.1 Génération de la clé secrète

Pour générer une clé secrète, on récupère une instance du générateur de clé, pour un algorithme donné (*DES*, *AES* etc.) :

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
```

On initialise ce générateur avec la longueur de la clé secrète à générer :

```
keyGen.init(128);
```

On génère une clé secrète :

```
SecretKey secretKey = keyGen.generateKey();
```

3.1.2 Chiffrement à l'aide de la clé secrète

Pour chiffrer, on récupère une instance de la classe de chiffrement/déchiffrement (*Cipher*) pour un algorithme donné (*AES*) :

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

```
Cipher aesCipher = Cipher.getInstance("AES");
```

On initialise cet objet en mode « chiffrement » avec la clé secrète que l'on vient de générer :

```
aesCipher.init(Cipher.ENCRYPT_MODE, secretKey);
```

On chiffre le message (et on le stocke, l'envoie sur un réseau non sûr etc.) :

```
byte[] byteCipherText = aesCipher.doFinal("Mon beau message si secret!".getBytes());
```

3.1.3 Déchiffrement avec la (même) clé secrète

Le seul changement est le mode d'initialisation de l'objet : « déchiffrement »

```
aesCipher.init(Cipher.DECRYPT_MODE, secretKey);
```



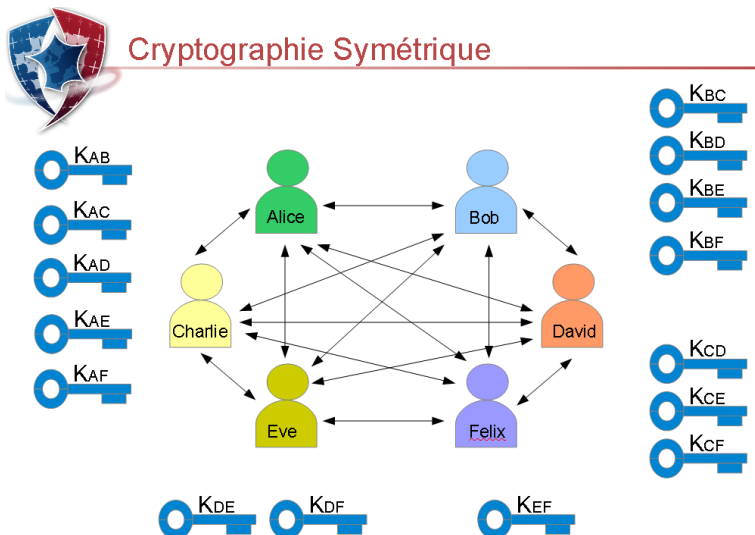
Remaniez le code fourni par le didacticiel en trois fonctionnalités indépendantes et réutilisables :

- La génération de la clé secrète
- Le chiffrement par l'émetteur du message (qui dispose de la clé secrète)
- Le déchiffrement par le récepteur du message (qui dispose aussi de la clé secrète)
- Testez votre code avec un jeu d'essai correct : vraie clé secrète et un message chiffré non altéré ;
- Testez votre code avec des tentatives d'attaques : message chiffré avec une autre clé secrète, vraie clé secrète avec un message altéré. On constate que le déchiffrement ne lève pas d'exception sur un message altéré : il aboutit à un message en clair incorrect.

Proposition de corrigé : [TestChiffrementAES.java](#)

3.2 WRAP/UNWRAP D'UNE CLE AVEC L'ALGORITHME SYMETRIQUE AES

Un problème classique des algorithmes symétriques est le partage des clés : il faut distribuer les clés aux différentes paires d'interlocuteur, en conservant le secret, ce qui devient vite lourd :



Une solution consiste à s'appuyer sur un secret qui est déjà partagé entre deux interlocuteurs (même clé secrète) pour en communiquer un nouveau : c'est la technique de « wrap » où l'on chiffre une clé privée par une autre qui est déjà connue du destinataire, avant de l'envoyer dans un environnement non sûr.

Pour la mise en œuvre en Java, lire le chap. 24 « *Les modes wrap et unwrap* » du didacticiel.

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

On utilise à nouveau une instance de la classe *Cipher*, en l'initialisant dans les modes *wrap* puis *unwrap*.

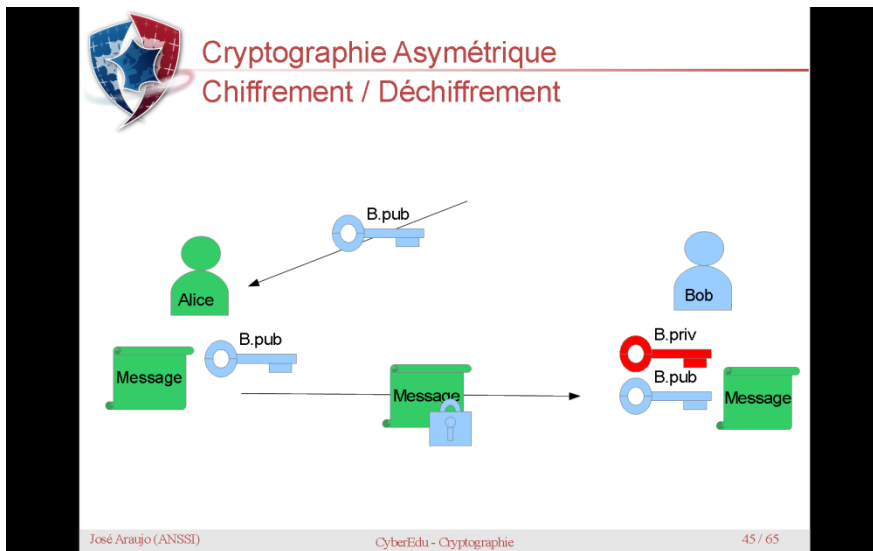


Exécutez et appropriiez-vous l'exemple fourni

Proposition de corrigé : [TestKeyWrap.java](#)

3.3 CHIFFREMENT D'UN MESSAGE AVEC L'ALGORITHME ASYMETRIQUE RSA

Lire ou relire l'exposé de José Araujo : chap. « [Cryptographie asymétrique](#) ».



La cryptographie asymétrique évite le problème de la distribution des clés secrètes dans la cryptographie symétrique. Elle permet de chiffrer/déchiffrer des messages sans avoir à communiquer un secret au destinataire puisque :

- l'émetteur (*Alice*) chiffre avec la clé publique de son destinataire (*Bob*). Une clé publique ne pose pas de problème de diffusion, puisqu'elle est destinée à être connue de tous ;
- le récepteur (*Bob*) déchiffre avec sa clé privée. Une clé privée ne pose pas non plus de problème de diffusion, puisqu'elle ne doit surtout pas être diffusée !

Nous allons décrire les principales étapes de la mise en œuvre en JAVA.

3.3.1 Génération d'une paire de clés : publique, privée

On commence par récupérer une instance du générateur de paire de clés, pour un algorithme donné (**RSA**) :

```
KeyPairGenerator gen = KeyPairGenerator.getInstance("RSA");
```

On initialise ce générateur avec la taille de la clé en bit (2048) et une source d'aléa fiable en utilisant la classe `java.security.SecureRandom` :

```
gen.initialize(2048, new SecureRandom());
```

On génère une paire clé privée, clé publique :

```
KeyPair cles = gen.generateKeyPair();
```

On accède aux clés publique et privée d'une instance de classe `KeyPair` par les méthodes `getPublic` et `getPrivate`

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

3.3.2 Chiffrement avec la clé publique

On récupère un objet de chiffrement avec l'algorithme RSA :

```
Cipher cipher = Cipher.getInstance("RSA");
```

On initialise cet objet en mode « chiffrement » avec la clé publique :

```
cipher.init(Cipher.ENCRYPT_MODE, clePublique);
```

Et on chiffre le texte :

```
texteChiffré = cipher.doFinal(text.getBytes());
```

3.3.3 Déchiffrement avec la clé privée

On récupère l'objet de chiffrement de la même manière, mais on l'initialise en mode « déchiffrement » avec la clé privée :

```
cipher.init(Cipher.DECRYPT_MODE, clePrivee);
```

Et on déchiffre le texte :

```
texteClair = cipher.doFinal(texteChiffré);
```



Remaniez le code fourni par le didacticiel en trois fonctionnalités indépendantes et réutilisables :

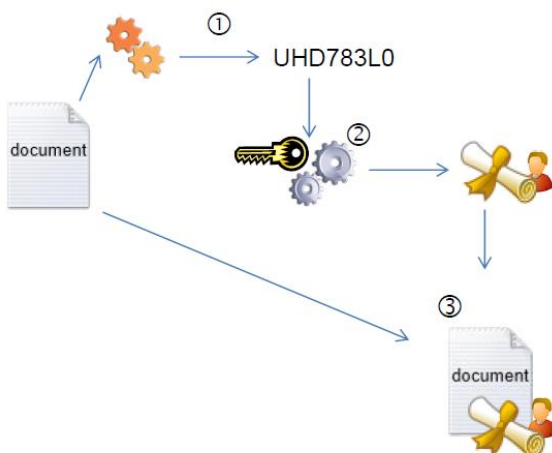
- La génération de la paire de clés, publique, privée
- Le chiffrement par l'émetteur du message (avec la clé publique du destinataire)
- Le déchiffrement par le récepteur du message (avec sa clé privée)
- Testez votre code avec un jeu d'essai correct : vraie clé secrète et un message chiffré non altéré ;
- Testez votre code avec des tentatives d'attaques : tentative de déchiffrement avec une clé incorrecte, avec une autre clé privée, avec un message chiffré altéré. On constate que le déchiffrement lève des exceptions dans tous les cas d'erreurs.

Proposition de corrigé : [TestChiffrementRSA.java](#)

3.4 SIGNATURE D'UN MESSAGE EN « SHA1 AVEC RSA »

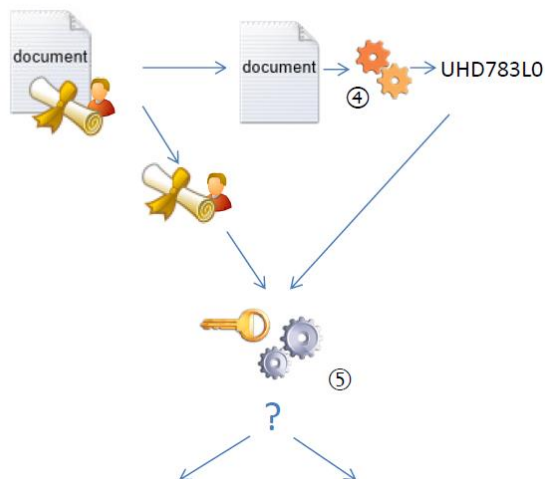
3.4.1 Les étapes

Relire le chapitre sur la signature électronique dans le document principal.



Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »



Pour la mise en œuvre en Java, lire le chap. 23 « *La classe `java.security.Signature`* » du didacticiel.

3.4.2 Génération de la signature par l'émetteur

On commence par récupérer une instance du générateur de signature, pour un algorithme donné (**SHA1withRSA**) :

```
Signature signature = Signature.getInstance("SHA1withRSA");
```

SHA1 est l'algorithme de *hash-code* (le condensat *UHD783L0*, dans l'étape 1 sur les schémas précédents) qui va garantir l'intégrité du message.

RSA est l'algorithme qui va permettre de chiffrer le condensat avec une clé privée, pour garantir l'identité de son émetteur (étape 2 sur les schémas précédents) :

On initialise le générateur de signature avec la clé privée de l'émetteur et une source d'aléa fiable :

```
signature.initSign(keyPair.getPrivate(), new SecureRandom());
```

On particularise le générateur de signature pour le message à signer :

```
signature.update(message);
```

On génère la signature : crée le condensat à partir du message et le chiffre avec la clé privée de l'émetteur :

```
byte[] signatureBytes = signature.sign();
```

3.4.3 Validation de la signature par le récepteur

On commence par récupérer une instance du générateur de signature, pour l'algorithme SHA1withRSA et on l'initialise avec la clé publique de l'expéditeur :

```
Signature signature = Signature.getInstance("SHA1withRSA");
signature.initVerify(keyPair.getPublic());
```

On particularise le générateur de signature pour le message à valider :

```
signature.update(message);
```

On vérifie la signature (identité de l'émetteur, et authenticité du message) :

```
ok = signature.verify(signatureBytes);
```



Faîtes une maquette de la signature avec trois fonctionnalités :

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

- La génération de la paire de clés, publique, privée
- La signature du message par l'émetteur (avec la clé privée de l'émetteur)
- La vérification de la signature par le récepteur (avec la clé publique de l'émetteur)
- Testez votre code avec un jeu d'essai correct : bonne clé publique, bonne signature et message non altéré ;
- Testez votre code avec des tentatives d'attaques : mauvaise clé publique, signature altérée, message altéré. On constate que toutes ces erreurs sont détectées par le mécanisme de signature.

Proposition de corrigé : [TestSignature.java](#)

3.5 INTEGRER UN CERTIFICAT

Les API de sécurité de Java permettent d'intégrer des certificats, à partir de leur fichier de description (extension **.cer**).

L'extrait de code ci-dessous crée une instance de la classe *Certificate* à partir du fichier de certificat ("**moncertificat.cer**"). C'est un certificat auto-signé, créé pour la démonstration :

```
// création d'une fabrique de certificats au format X.509
CertificateFactory cf = CertificateFactory.getInstance("X.509");
// récupération d'un certificat autosigné
FileInputStream in = new FileInputStream("c:/test/moncertificat.cer");
// création du certificat en Java
Certificate cert = cf.generateCertificate(in);
```

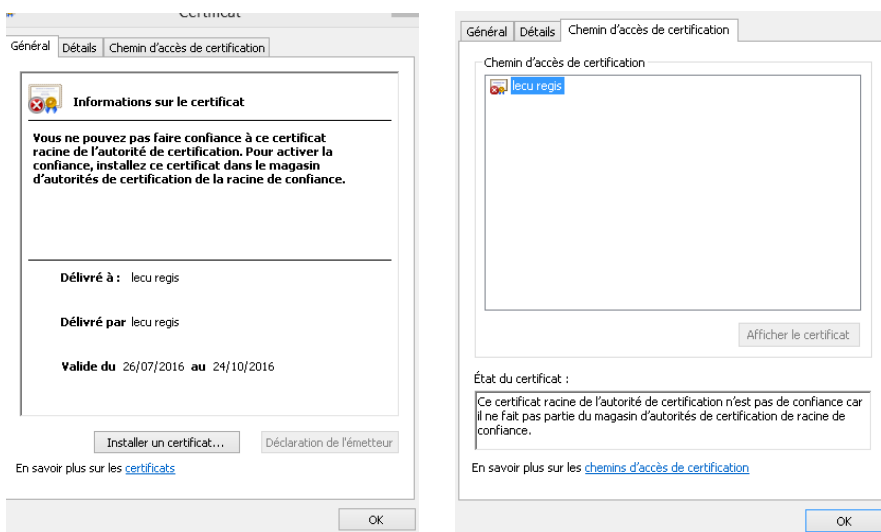
L'objet **cert** contient la clé publique de l'émetteur du certificat, qui va permettre à l'application Java de chiffrer ses informations.



Coder la création du certificat

Le fichier de certificat est fourni dans corrigés : **moncertificat.cer**.

Cliquez sur ce fichier et vérifiez qu'il s'agit d'un certificat auto-signé, qui n'est pas délivré par un tiers de confiance (pas d'autorité racine) et ne devrait pas être accepté par Windows ou un Navigateur web :



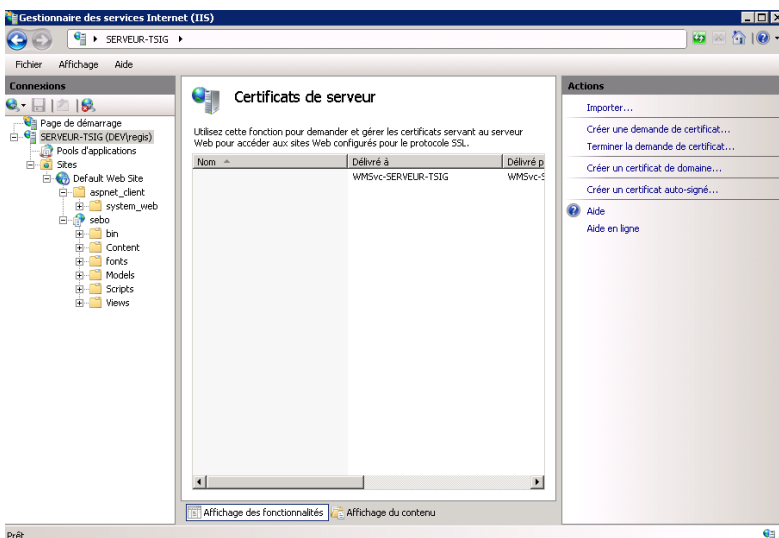
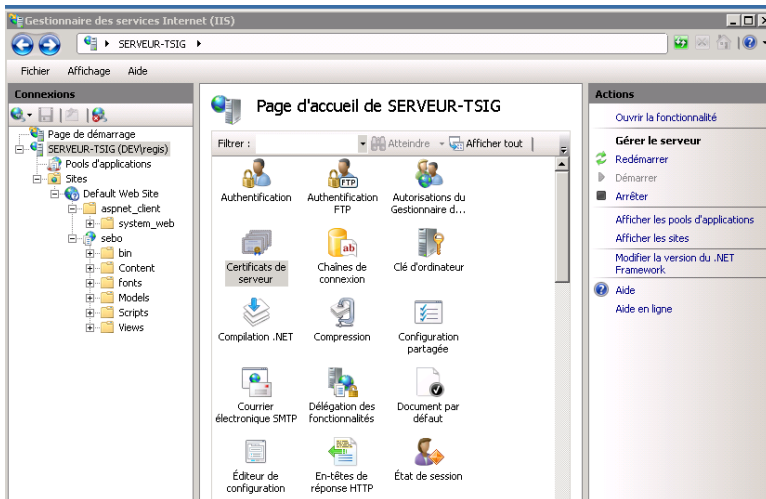
Proposition de corrigé : [TestCertificats.java](#)

Utiliser la cryptographie et les mécanismes de sécurité du Web

4. CONFIGURATION D'UNE LIAISON HTTPS

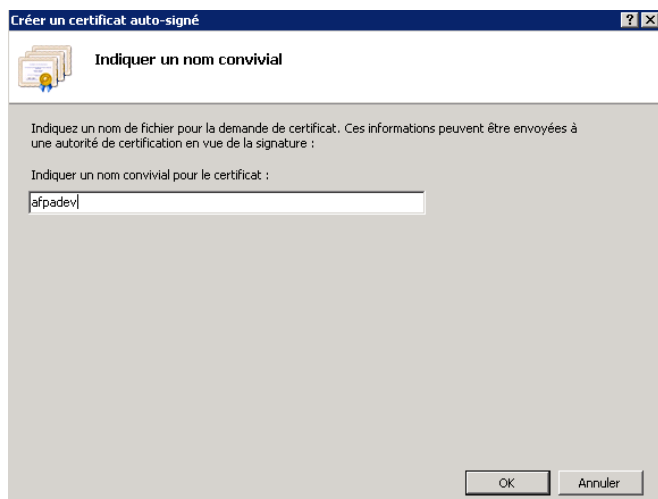
Le principe est le même sur tous les serveurs Web, avec des interfaces graphiques plus ou moins élaborées. Pour des raisons de simplicité, on utilisera un serveur Microsoft IIS7 correctement configuré en HTTP et on lui rajoutera une liaison sécurisée HTTPS.

Lancez le « *Gestionnaire des services Internet (IIS)* » et cliquez sur « *Certificats de serveur* » :



Utiliser la cryptographie et les mécanismes de sécurité du Web

4.1 CREER UN CERTIFICAT AUTO-SIGNE



Cette option permet de créer un certificat local à notre entreprise, qui n'est pas signé par une autorité de certification (comme celui que nous venons de manipuler en Java).

Ce certificat n'a donc pas de valeur (puisque tout le monde pourrait le faire) et ne devrait servir qu'à des tests internes à l'entreprise.

Les navigateurs Web reconnaissent les certificats auto-signés et lèvent des alarmes de sécurité.

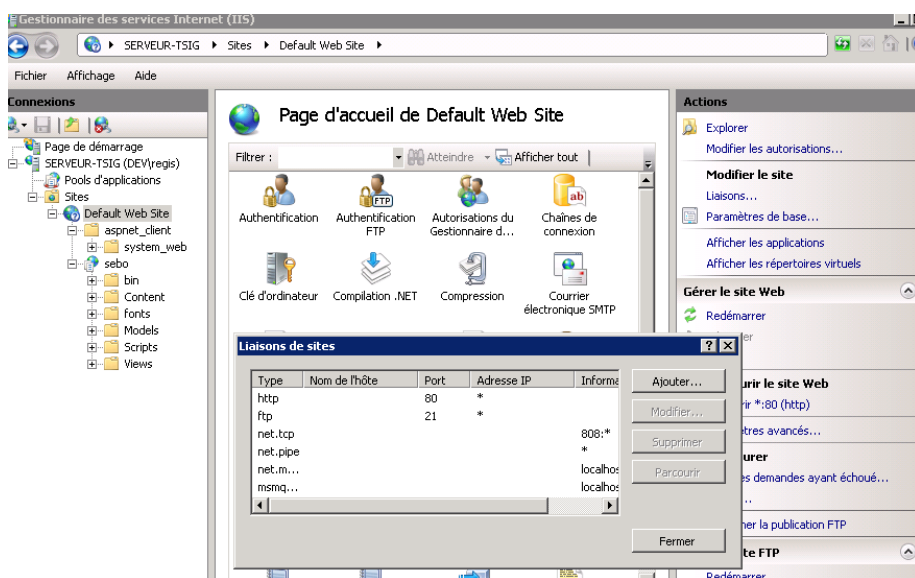


Sur le danger des certificats auto-signés :

<https://www.globalsign.fr/fr/centre-information-ssl/dangers-certificats-auto-signes/>

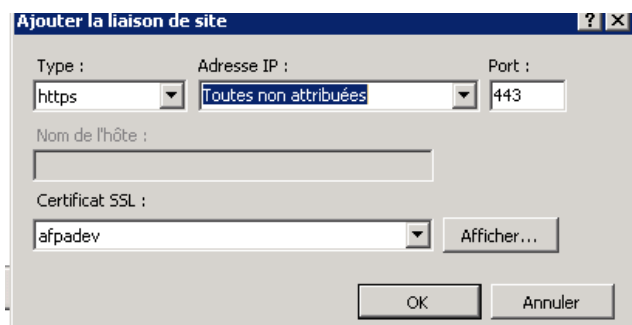
4.2 AJOUTER UNE LIAISON HTTPS A UN SITE WEB

Sélectionnez le site Web que vous voulez sécuriser, et cliquez sur « *Modifier les liaisons...* » avec le bouton droit de la souris :

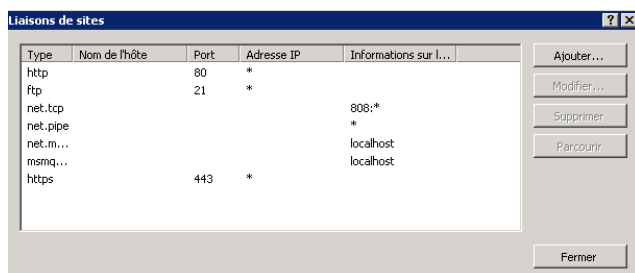


Dans la fenêtre « *Liaisons de sites* », ajoutez le protocole *HTTPS*, sur le port par défaut *443*, et associez-lui le certificat *SSL* auto-signé que nous venons de créer (*afpadev*) :

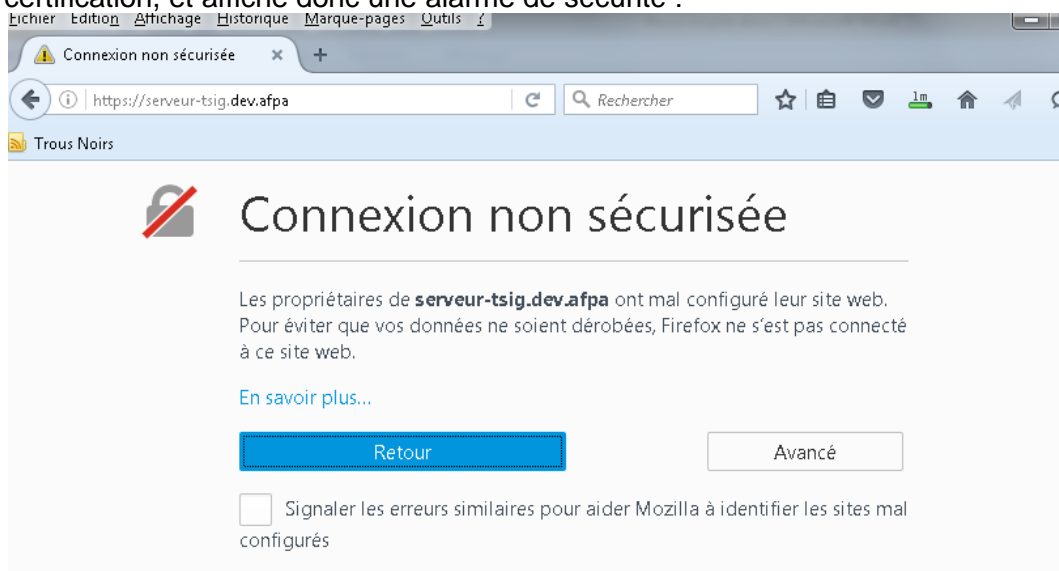
Utiliser la cryptographie et les mécanismes de sécurité du Web



Le nouveau protocole apparaît dans la liste :



Essayez l'URL de votre site avec le protocole HTTPS : le navigateur ne trouve pas d'autorité de certification, et affiche donc une alarme de sécurité :

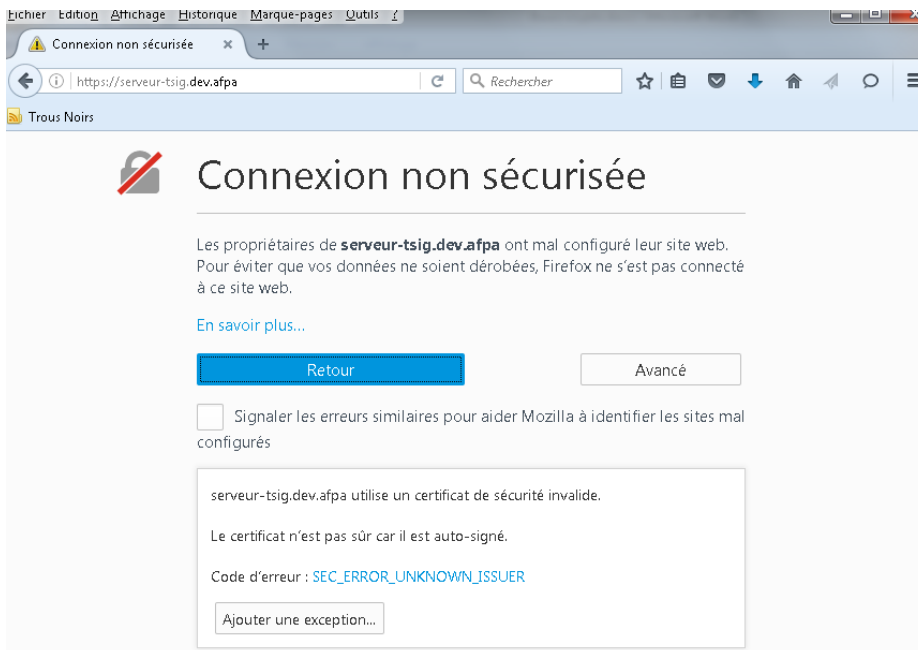


Dans la « vraie vie », il vaut mieux s'arrêter là, en ne faisant pas confiance à n'importe qui.

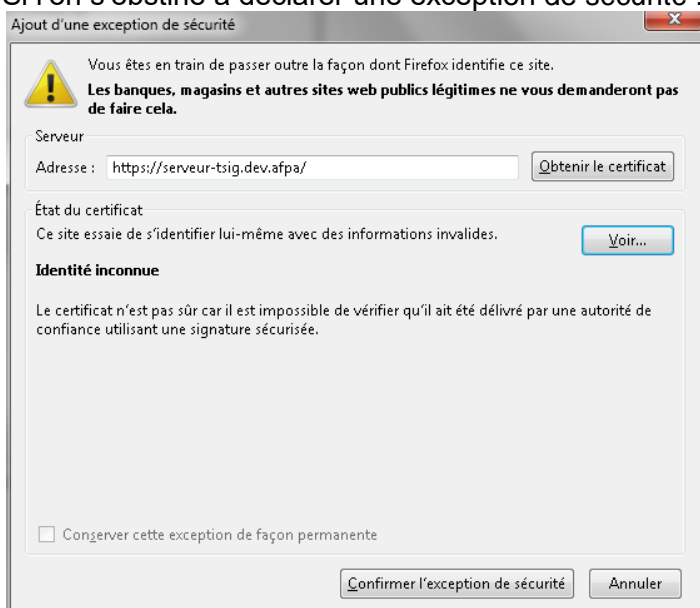
Pour notre test, nous allons cliquer sur « **Avancé** » et déclarer notre connexion comme une exception de sécurité :

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »



Si l'on s'obstine à déclarer une exception de sécurité :



4.3 RECUPERER UN CERTIFICAT DE TEST DEPUIS UNE AUTORITE DE CERTIFICATION

Pour éviter les inconvénients des certificats auto-signés, il faut demander un certificat à une autorité de certification qui va garantir l'identité de notre entreprise.

Cette demande n'est pas gratuite, mais il est toutefois possible d'obtenir gratuitement des certificats temporaires pour effectuer des tests.

Dans la fenêtre « *Certificats de serveur* », cliquez sur « *Créer une demande de certificat* » et remplissez le formulaire :

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

Demande de certificat

Propriétés du nom unique

Indiquez les informations requises pour le certificat. Lorsque vous entrez le département ou région et la ville/localité, utilisez des noms complets et officiels, et n'employez aucune abréviation.

Nom commun :

Organisation :

Unité d'organisation :

Ville :

Département/région :

Pays/région :

Précédent Suivant Terminer Annuler

Cliquez sur *Suivant*, et choisissez 2048 comme longueur de clé :

Demande de certificat

Propriétés du fournisseur de services de chiffrement

Sélectionnez un fournisseur de services de chiffrement et une longueur en bits. La longueur en bits de la clé de chiffrement détermine l'efficacité du chiffrement du certificat. Avec une clé de chiffrement plus longue, la sécurité est accrue. Toutefois une longueur trop importante peut nuire aux performances.

Fournisseur de services de chiffrement :

Longueur en bits :

Précédent Suivant Terminer Annuler

Cliquez sur *Suivant* et rentrez le nom du fichier de demande de certificat.

Ce fichier texte a le format suivant :

```

---BEGIN NEW CERTIFICATE REQUEST-----
MIIEtTCCAzUCAQAwYTELMAkGA1UEBhMCR1IxZzANBgNVBAGMBmlzw6hyZTERMA8G
A1UEBwwIZ3Jlbm9ibGUxDTALBgNVBAoMBGFmcGExDDAKBgNVBAsMA2RldjERMA8G
A1UEAwIZGV2LmFmcGEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDB
ojsbf0HowgTelcz0ANiT1C8HEgmS3xagN/2kDuN7NjPykY9E3o3Iem9oL3RqD95i
GNGzjCiv/LSJWmj0fp9euIqPKuDNidWmyakg2c4Hg2YxVJXBwFLAHACwXXf2vQc0
oFDuo1tOP71XeuN8M0CmZ2501hTnwSd3PBaGezgV4nhhken6BKYgWhGsZW4TPzkf
fXBgfBG6kTpSnhWcnPIM/fMob/ch7ABiAR6QFTc8uhRJJOfzhmoPcqB7ebk8ghfh
dw==
-----END NEW CERTIFICATE REQUEST-----

```



Documentation Microsoft pour demander un certificat sous IIS 7 :

<https://technet.microsoft.com/en-us/library/d780d806-e8a8-4bc5-8d7a-9f045d1f3e22>

Utiliser la cryptographie et les mécanismes de sécurité du Web

Afpa © 2016 – Section Tertiaire Informatique – Filière « Etude et développement »

Il faut ensuite envoyer notre demande à l'autorité de certification pour obtenir le certificat à intégrer dans IIS.



Cette étape est à faire sous la conduite d'un formateur, de préférence réseau. La plupart des autorités de certification proposent des certificats temporaires gratuits. Par exemple : https://www.tbs-certificats.com/certificats_test.html.fr

Il faut ensuite intégrer la réponse de l'autorité de certification dans IIS, pour terminer la demande :

On ajoutera la connexion HTTPS de la même manière que précédemment.

CRÉDITS

OEUVRE COLLECTIVE DE L'AFPA

Sous le pilotage de la DIIP
et du centre sectoriel Tertiaire

EQUIPE DE CONCEPTION

Chantal PERRACHON – IF Neuilly-sur-Marne
Régis Lécu – Formateur AFPA Pont de Claix