Compte rendu

PROJET DE CRYPTOGRAPHIE

Christophe VOLTO & Matthieu PREZUT

ISEN Toulon – M1 Cyber sécurité

09 Avril 2021

Table des matières

Apport du 2D-Doc

Le 2D-Doc est un projet de sécurisation des justificatifs qui a pour objectifs :

- La lutte contre la fraude documentaire ;
- Le développement de l'administration électronique ;
- La simplification des démarches administratives ;
- Une plus grande sécurité des services en ligne ;
- Cette solution permet de sécuriser tout type de documents aussi bien papier que numérique. Les documents ciblés concernent en particulier les justificatifs (factures eau, téléphone, EDF, quittances d'assurance et de loyer, RIB, revenus, …) utilisés par les particuliers et/ou les professionnels dans leurs relations avec les entreprises, les services de l'Administration ou les services sociaux.

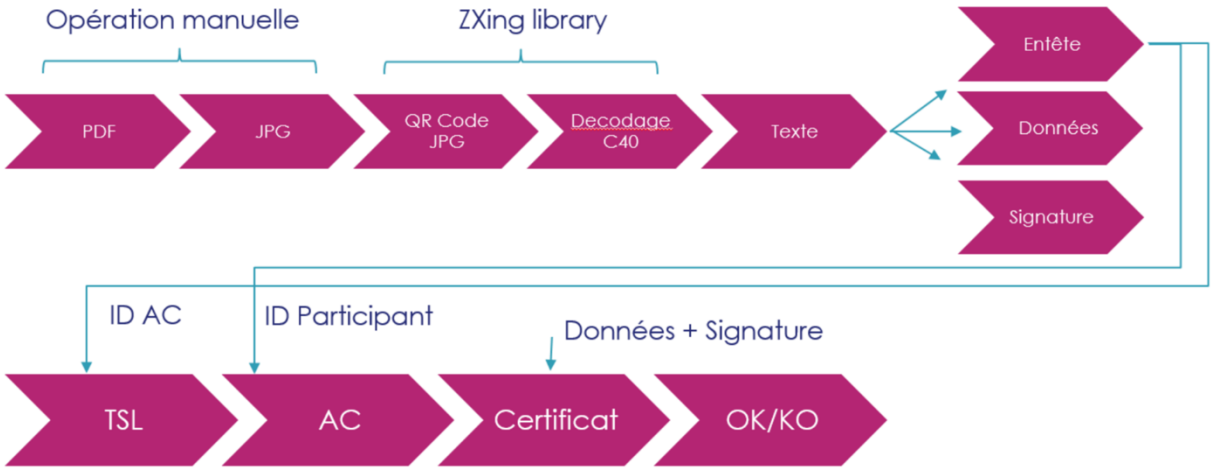
Le standard à codes-barres bidimensionnel 2D-Doc consiste à insérer un code à barres 2D emportant les informations clés du document (le type de document, le nom et le prénom de l'émetteur, la civilité, l'adresse, le numéro de facture), la date d'émission du document ou du code à barres 2D ; et ces informations sont verrouillées par une signature électronique du hash de ces données, qui garantit l'identification de l'organisme émetteur et l'intégrité du document.

Le code barre 2D est signé électroniquement par la clé privée correspondant à une clé publique placée dans un certificat du type « cachet serveur ». Ce chiffrement asymétrique permet le contrôle de la signature par tous les acteurs disposant de la clé publique du signataire émetteur.

Ce standard constitue une signature visible vérifiable uniquement par une machine (de type scanner ou lecteur de code à barres). Si la signature de ces données est exacte, ceci indique que ces données et uniquement celles-ci sont exactes.

Étapes de vérification

Principe général de vérification d'une signature de 2D-Doc



Principe de vérification d'une signature de 2D-Doc

Dans un premier temps, il est nécessaire d'extraire le QR code de façon manuelle par une simple capture d'écran. Le décodage du QR code (y compris le décodage C40) est effectué à l'aide de la librairie Java ZXing. Le texte obtenu est décomposé en trois parties : l'entête, les données et la signature. L'entête fournit un identifiant d'autorité de certification (AC) qui permet de retrouver les données de l'AC (dont l'URL de son annuaire et son certificat) dans la TSL de l'ANTS. Puis en consultant l'annuaire de l'AC à l'aide de l'identifiant du participant il est possible d'obtenir son certificat. Ces deux étapes peuvent être, dans un premier temps, réalisées manuellement mais on a décidé de les réaliser directement en automatique. On utilise alors la clé publique contenue dans le certificat participant pour vérifier la signature (attention, cette signature est encodée en Base32 et il faut donc la décoder).

Principe de vérification du certificat du Participant

Tout d'abord, on découpe le 2D-Doc décodé en 3 parties : l'en-tête, le message et la signature. Une fois ceci fait, on peut récupérer les données essentielles de l'en-tête comme la version du 2D-Doc, l'ID de l'AC, l'ID du Participant. En fonction de la version du 2D-Doc on peut déterminer la taille de l'en-tête. Puis, on parcourt le fichier TSL.xml afin de trouver l'AC correspondante à l'ID trouvé dans l'en-tête du 2D-Doc. On récupère l'URL associée sur laquelle on applique la modification suivante : l'ajout de « name=identifiant de participant 2D Doc ». Identifiant de participant 2D Doc qu'on retrouve dans l'en-tête du 2D Doc et que l'on va conserver dans une variable. Cette URL nouvellement modifiée nous permet de récupérer le certificat du certificat du participant sur lequel on appliquera la fonction *IsRevoked* qui nous renverra *« true »* ou *« false »* en fonction de si oui ou non le certificat de l'AC a été révoqué. Cette fonction va dans un premier temps vérifier que ce certificat ne se trouve pas dans une CRL (Certificate Revocation List) et ensuite il va vérifier sa date de validité en faisant une comparaison toute simple entre la date de fin de certificat et la date d'aujourd'hui.

Principe de vérification du certificat de l'AC

En premier lieu, il nous sera nécessaire de récupérer le certificat de l'AC qui se trouve dans le fichier XML entre les balises *<tsl:X509Certificate></tsl:X509Certificate>*. Une fois celui-ci récupéré dans un String et transformé en objet X509Certificate, on peut appliquer la fonction *isRevoked* qui nous renverra *« true »* ou *« false »* en fonction de si oui ou non le certificat de l'AC a été révoqué. Cette fonction va dans un premier temps vérifier que ce certificat ne se trouve pas dans une CRL (Certificate Revocation List) et ensuite il va vérifier sa date de validité en faisant une comparaison toute simple entre la date de fin de certificat et la date d'aujourd'hui.

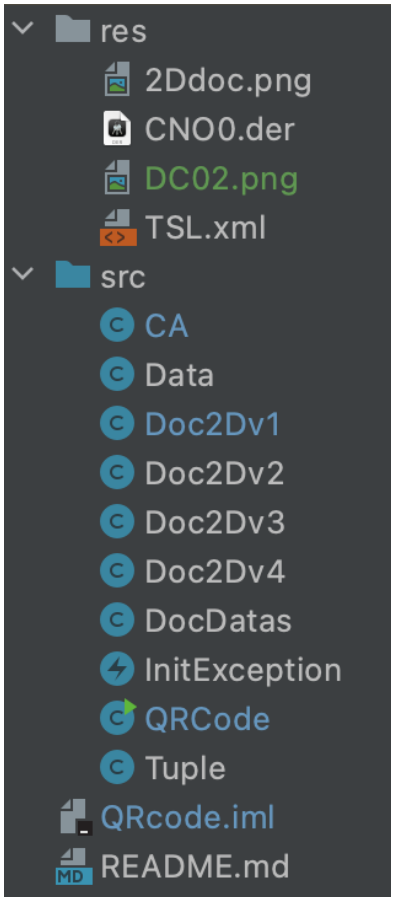
Principe de vérification du certificat de la TSL

Premièrement, on a besoin du certificat de la TSL qui se trouve entre les balises *<ds:Signature>* du fichier XML. Une fois le certificat récupéré, on peut appliquer notre fonction *isRevoked*. Cette fonction va dans un premier temps vérifier que ce certificat ne se trouve pas dans une CRL (Certificate Revocation List) et ensuite il va vérifier sa date de validité en faisant une comparaison toute simple entre la date de fin de certificat et la date d'aujourd'hui. En fonction de la CRL et de la date de validité, la fonction nous renverra *« true »* si le certificat n'est plus valide ou *« false »* si celui-ci est toujours valide.

Principe de vérification de la signature de la TSL

On peut par la suite vérifier sa signature comme nous l'avons fait au début avec la signature du 2D Doc. On va donc pour cela avoir besoin du certificat X509 se trouvant *<ds:X509Certificate></ds:X509Certificate>* qui nous fournira la clé publique ainsi que des données constituants le fichier XML soit toutes les informations entre les balises *<tsl:TrustServiceProviderList></tsl:TrustServiceProviderList>*. On créé donc un objet *Signature* qu'on instancie avec *SHA256withRSA*, qu'on initialise avec la clé publique du certificat X509 et qu'on update avec les données précédemment récupérées. Ensuite, on peut vérifier la signature en lui passant la signature récupérée dans entre les balises *<ds:SignatureValue></ds:SignatureValue>* du fichier XML.

Structure du programme



Avant toute chose, on a décidé de créer 2 dossiers afin de ranger d'un côté les ressources statiques tel que les images, les fichiers XML, les certificats ou encore les PDFs et de l'autre le code avec les classes Java.

Dans le dossier, il est important de noter qu'on a stocké en dur le fichier XML de la TSL ainsi que les images des 2D Doc qu'on a manuellement récupérés depuis les PDFs.

QRCode.java :
Classe principale contenant la fonction main.

DocDatas.java :
Classe utilitaire contenant tous les identifiants de données du périmètre données C40 '01' avec les informations correspondantes (nom, taille min, taille max).

Data.java :
Classe ayant pour but de stocker une donnée.

Doc2Dv**X**.java :
Classe représentant un 2D Doc de version **X**. Si un 2D Doc de version différente est chargée dans cette classe, une exception de type InitException est levée.

CA.java :
Classe représentant une CA, contient toutes les informations relatives à une CA telles que, son certificat, le certificat de la TSL ainsi que la liste des URLs nous permettant plus tard de récupérer le certificat du participant. La classe CA contient aussi la fonction IsRevoked qui permet de vérifier si un certificat X509 est révoqué ou non.

InitException.java :
Classe utilitaire héritant de la classe Exception.

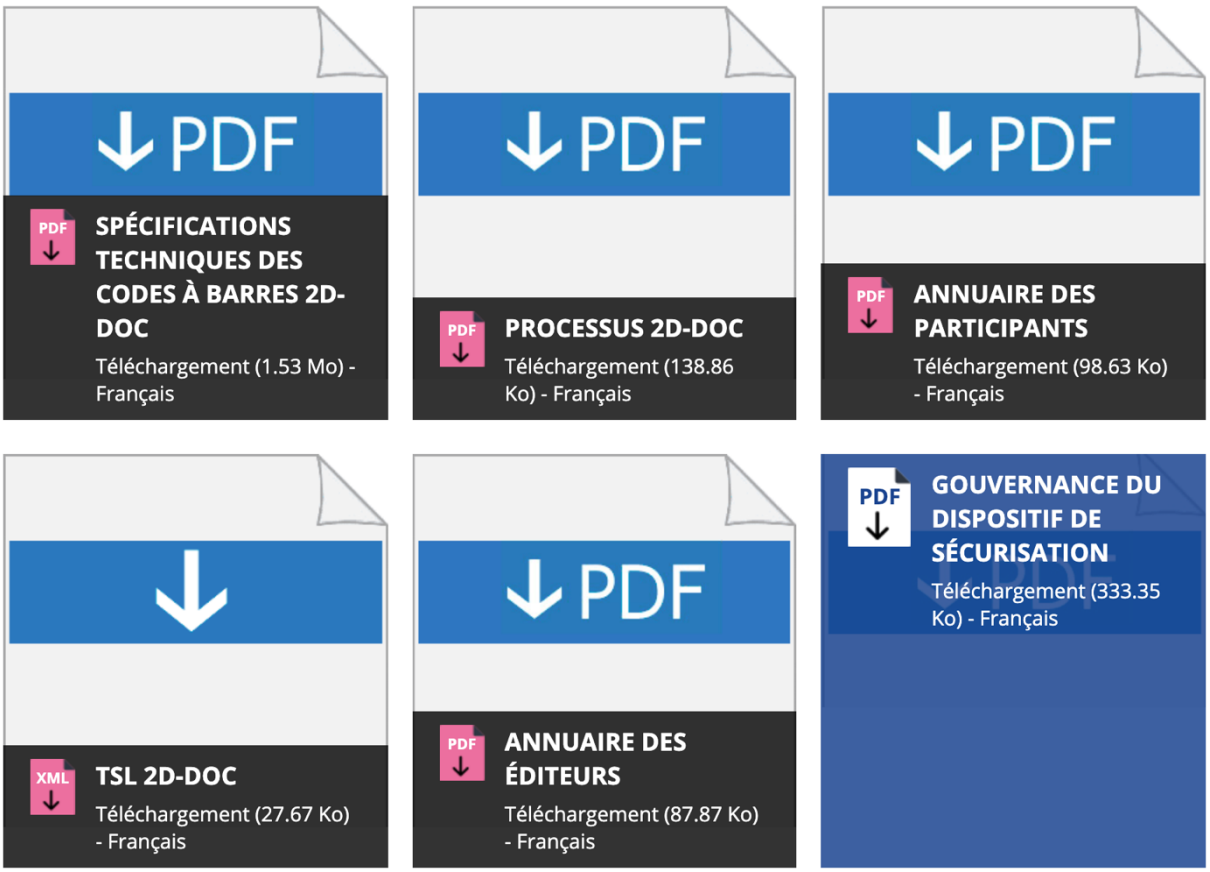
Tuple :
Classe utilitaire permettant de créer un tuple.

A côté de ça, nous utilisons plusieurs librairies telles que la librairie « bouncy castle » pour l'ASN.1, « Zxing » pour le décodage du 2D Doc, « commons-codec » pour le décodage du Base32, « Spire.PDF » pour l'extraction des 2D Docs à partir d'un PDF et « chilkat ».

Documents utilisés

Voici donc la liste des documents utilisés, publiés par l'Agence Nationale des Titres Sécurisés (ANTS) :

- Le fond documentaire : document de processus 2D-Doc et spécifications techniques des codes à barres 2D-Doc
- L'annuaire des participants
- L'annuaire des autorités de certification dit TSL
- L'annuaire des éditeurs de Codes à Barres 2D-Doc
- La gouvernance du dispositif de sécurisation des justificatifs de domicile par codes à barres 2D-Doc



La Trusted Service List ou encore TSL est un fichier au format xml qui sera stocké en dur dans le dossier *« /res/ »* de notre projet. Cette liste nous permettra de récupérer le certificat X509 de l'AC ainsi que les différents certificats des participants.

Contenu des QR Codes décodés

QR Code trouvé sur Internet

```
Comparer les informations suivantes au document papier:
Type de document: Justificatif académique - Attestation de Versement de la Contribution à la Vie Etudiante
Informations:
Data{id='B0', name='Liste des prénoms', value='MANON HELENE MARIA'}
Data{id='B2', name='Nom patronymique', value='CHILLON'}
Data{id='B7', name='Date de naissance ', value='26072000'}
Data{id='BB', name='Numéro ou code d'identification de l'étudiant ', value='071028484FH'}
Data{id='BK', name='Numérode l'Attestation de versement dela CVE', value='NCY8-NRHKFW-79'}
```

Type de document : Justificatif académique - Attestation de Versement de la Contribution à la Vie Étudiante
Informations :
Data {id='B0', name='Liste des prénoms', value='MANON HELENE MARIA'}
Data {id='B2', name='Nom patronymique', value='CHILLON'}
Data {id='B7', name='Date de naissance ', value='26072000'}
Data {id='BB', name='Numéro ou code d'identification de l'étudiant ', value='071028484FH'}
Data {id='BK', name='Numéro de l'Attestation de versement de la CVE', value='NCY8-NRHKFW-79'}

QR Code fourni par notre professeur Pierre GIRARD

```
Comparer les informations suivantes au document papier:
Type de document: Justificatif de domicile - Factures de fournisseur d'énergie | Factures de téléphonie| Factures defournisseur d'accès internet | Factures de fournisseur d'eau
Informations:
Data{id='26', name='Pays de service des prestations', value='FR'}
Data{id='24', name='Code postal ou code cedex du point de service des prestations', value='13015'}
Data{id='10', name='Ligne 1 de la norme adresse postale du bénéficiaire de la prestation', value='M/BOUDJEDRI/AMMAR'}
Data{id='22', name='Ligne 4 de la norme adresse postale du point de service des prestations', value='309 RUE DE LYON'}
Data{id='25', name='Localité de destination ou libellé cedex du point de service des prestations', value='MARSEILLE'}
Data{id='18', name='Numéro de la facture', value='1048469311'}
```

Type de document : Justificatif de domicile - Factures de fournisseur d'énergie | Factures de téléphonie | Factures de fournisseur d'accès internet | Factures de fournisseur d'eau
Informations :
Data {id='26', name='Pays de service des prestations', value='FR'}
Data {id='24', name='Code postal ou code cedex du point de service des prestations', value='13015'}
Data {id='10', name='Ligne 1 de la norme adresse postale du bénéficiaire de la prestation', value='M/BOUDJEDRI/AMMAR'}
Data {id='22', name='Ligne 4 de la norme adresse postale du point de service des prestations', value='309 RUE DE LYON'}
Data {id='25', name='Localité de destination ou libellé cedex du point de service des prestations', value='MARSEILLE'}
Data {id='18', name='Numéro de la facture', value='1048469311'}

Difficultés rencontrées

1- Les URLs des AC

L'une des difficultés rencontrées fut celle concernant les URLS des différentes Autorités de Certification :

- Dans le cas de la FR03 (Certigna), le moyen de récupérer le certificat du participant est plutôt simple : il suffit de récupérer l'URL et d'y ajouter *« name= »* ainsi que l'ID du Participant.
- Dans le cas de la FR02, le lien ne marche pas.
- Et enfin, dans le cas de la FR01 et de la FR04, les URLs nous fournissent un fichier avec comme extension *« .2ddoc »*. Ne connaissant pas cette extension, il est donc nécessaire de faire des recherches mais qui s'avèrent compliquées car on trouve plus d'info sur le 2D-Doc façon QR Code plutôt que l'extension.

2- Extraction du 2D Doc à partir du PDF et conversion en image

Une autre difficulté fut celle de rendre automatique l'extraction des 2D Docs à partir de PDFs. Il fallait trouver une librairie et vérifier qu'elle fonctionne. Après de nombreuses recherches j'ai trouvé Spire.PDF qui a comme fonctionnalité l'extraction d'images d'un PDF. Cette librairie prend en compte le nombre de page du PDF ainsi que les différentes images d'un PDF. C'est top cependant je dois faire le tri parmi les images extraites car on a de tout : les logos de la marque, le 2D Doc … Il nous fallait donc trouver un moyen de comparer les images afin de garder la seule qui nous intéressait. N'ayant pas réussi à trouver un moyen efficace dans le temps imparti sachant que ce n'était pas une fonctionnalité vitale, la comparaison d'images a été abandonnée. En plus, si l'image se trouve en fin de page, elle va écraser celles qui se trouvent avant et on va donc pouvoir garder la seule image qui nous intéresse soit le 2D Doc.

3- Vérification de la signature de la TSL

```
Signature sign = Signature.getInstance("SHA256withRSA");
sign.initVerify(TSLCert.getPublicKey());
File myXMLFile = new File( pathname: "./res/TSL.xml");
byte[] ByteXML = Files.readAllBytes(myXMLFile.toPath());
String XML = new String(ByteXML);
String data = XML.substring(XML.indexOf("<tsl:SchemeInformation>"), XML.indexOf("</tsl:TrustServiceProviderList>"));
sign.update(data.getBytes());
boolean isSignatureOk = sign.verify(SignTSL.getBytes());
System.out.println("La TSL est intègre ? : " + isSignatureOk);
```

Afin de vérifier l'intégrité de la TSL, il nous a fallu récupéré son certificat qui se trouve à la fin du fichier XML dans les balises *<ds:x509Certificate></ds:x509Certificate>.* Cependant, pour cela nous avons aussi besoin de sa signature qui se situe entre les balises *<ds:Signature></ds:Signature>*. Il faut par la suite récupérer les données du XML afin de pouvoir vérifier sa signature. Cependant, nous nous confrontons à des erreurs du style *« No Key Found ! »* ou encore *« No Key Value »*. Nous n'avons donc pas réussi à vérifier la signature de la TSL.

Améliorations possibles

1- Récupération automatique d'un 2D Doc à partir d'un PDF

L'idéal serait de fournir un PDF entier et le programme se chargerait d'identifier le 2D Doc sur celui-ci tandis qu'actuellement on est obligé de faire cette manipulation à la main et donc de faire une capture d'écran de ce 2D Doc sur le PDF et d'en faire une image afin de le fournir au programme.

2- L'affichage des données

```
Is CA revoked (CRL + Validity period) ? : false
Is Participant revoked (CRL + Validity period) ? : true
Is TSL revoked (CRL + Validity period) ? : true
Le 2D Doc est intègre ? : true

Comparer les informations suivantes au document papier:
Type de document: Justificatif académique - Attestation de Versement de la Contribution à la Vie Etudiante
Informations:
Data{id='B0', name='Liste des prénoms', value='MANON HELENE MARIA'}
Data{id='B2', name='Nom patronymique', value='CHILLON'}
Data{id='B7', name='Date de naissance ', value='26072000'}
Data{id='BB', name='Numéro ou code d'identification de l'étudiant ', value='071028484FH'}
Data{id='BK', name='Numérode l'Attestation de versement dela CVE', value='NCY8-NRHKFW-79'}
```

Notre affichage étant très basique, une amélioration possible reste un meilleur affichage comme par exemple dans un tableau et une meilleure gestion des données comme la précision sur la révocation par CRL ou par date.

3- La vérification de la signature de la TSL

Malgré le fait que la TSL.xml soit en dur dans notre projet, rien ne nous confirme qu'elle est intègre et qu'elle n'a pas été modifié lors du téléchargement de celle-ci. Donc, la vérification de la TSL est nécessaire afin de s'assurer de l'intégrité de la totalité de notre programme.

Conclusion

Notre projet permet de vérifier si un 2D Doc est intègre ou non et il affiche aussi ses données. A savoir qu'on limite le nombre de récupération de données en ligne avec par exemple la TSL.xml qui se trouve en dur dans le dossier *« res »* car si ce dossier est compromis via une attaque Man In The Middle (MITM) ou autre impossible de savoir si un 2D Doc est réellement intègre ou non. La seule récupération en ligne que l'on fait est celle du certificat du participant via une requête RFC4387 :

Dans notre cas la seule information disponible sur le certificat est le *CommonName*. Une requête chez une AC sera donc de la forme *name=<identifiant de participant 2D Doc>*.

Exemple de http_URL complète pour la facture SFR d'exemple du projet (AC = AriadNEXT, identifiant du participant = SFR2) :

http://cert.pki-2ddoc.ariadnext.fr/pki-2ddoc.der?name=SFR2

Et si le certificat que l'on récupère a été modifié ou échangé via une personne malveillante, on le saura vite car le document sera marqué comme non intègre avec notre programme.