

COMPTE RENDU DU PROJET CRYPTOGRAPHIQUE

2DOC CHECKING

18 / 04 / 2021



RÉALISÉ PAR :
FILI THÉRÈSE – STOELTZLEN ALEXIS

SOMMAIRE

Introduction.....	3
2D-Doc et fraude administrative.....	5
Étapes de vérification.....	7
Structure du programme.....	13
2D-Doc vérifiés	17
Difficultés rencontrées.....	19
Améliorations possibles	21
Conclusion	23

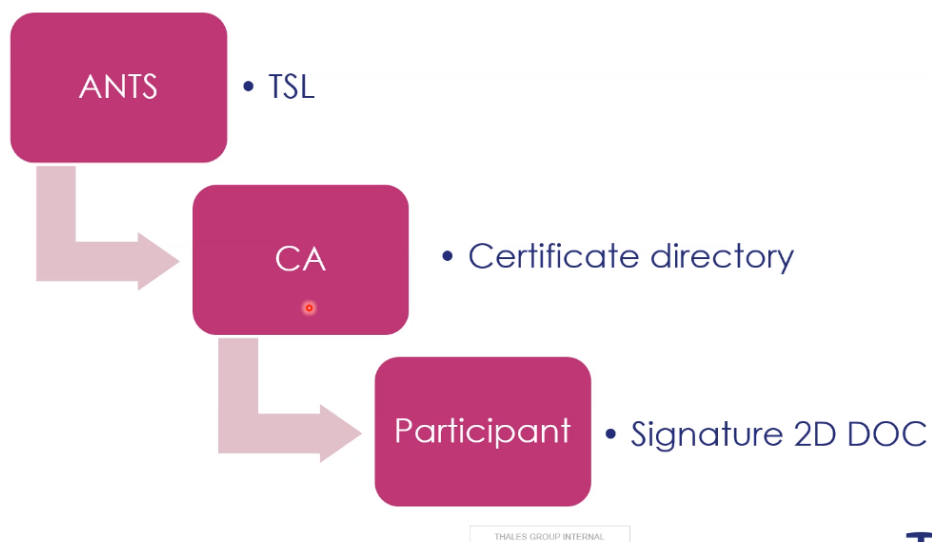
INTRODUCTION

Contexte

Pour clôturer notre cours de Cryptographie avec monsieur Pierre GIRARD nous avons comme projet d'analyser la chaîne de confiance autour du système 2D-Doc et de comprendre dans quel cas nous l'utilisons.

Objectif du projet

Chaîne de confiance



Le but va être de vérifier l'ensemble des entités de la chaîne de confiance du 2D-Doc en 6 étapes :

- Première étape : Récupération des informations contenues dans le 2D Doc en décodant le code à barres. Cela permet ainsi d'obtenir l'identifiant de l'AC et l'identifiant du certificat du participant qui a signé avec sa clé privée le 2D Doc.
- Deuxième étape : Grâce à une TSL publiée sur le site de l'ANTS sous un format XML, nous pourrions vérifier que l'identifiant de l'AC (Autorité de Certification) existe, puis nous récupérerons le certificat de l'AC correspondante, ainsi que l'adresse de l'annuaire où l'AC publie ses certificats.

- Troisième étape : On récupère dans l'annuaire de l'AC le certificat du participant correspondant grâce à l'identifiant du participant récupéré dans la première étape.
- Quatrième étape : Nous pouvons donc vérifier l'intégrité de la signature numérique du 2D Doc à l'aide du certificat du Participant.
- Cinquième étape : On récupère la CRL (Certification Revocation List) de l'AC au point de distribution de la CRL qui est indiqué dans le certificat du participant puis on vérifie que le certificat du participant et de l'AC ne sont pas révoqués. On vérifie également la période de validité des deux certificats
- Dernière étape : on vérifie les signatures des certificats et de la TSL

Objectifs supplémentaires

Les objectifs bonus sont la vérification du certificat participant, du certificat de l'AC et de la TSL. Nous devons supporter au minimum la version DC02 des 2D-doc, et donc optionnellement ajouter les versions ultérieures. Enfin, nous pouvons aussi automatiser les tâches réalisées manuellement.

Les moyens imposés

Nous avons réalisé un code en Java permettant d'effectuer les opérations décrites dans les étapes précédentes. Il permet de mettre en forme toutes les informations provenant de la TLS ainsi que les opérations menées sur le 2D Doc et les certificats.

2D-DOC ET FRAUDE ADMINISTRATIVE

Objectif de 2D-Doc

La solution 2D-Doc a été mise en place pour lutter contre la fraude administrative et sécuriser les données échangées sous forme papier entre l'utilisateur et l'administration avec la collaboration d'entités privées et publiques tel que l'Agence Nationale des titres sécurisés mandatée par le ministère de l'Intérieur.

Cette solution a pour objectif la sécurisation des justificatifs tel que :

- 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

La solution permet donc de sécuriser tout type de documents aussi bien papier (en impression par exemple) que numérique. Les documents ciblés concernent en particulier les justificatifs (factures eau, téléphone, EDF, quittances d'assurance et de loyer, RIB, revenus, ...) utilisés par des particuliers et/ou les professionnels dans leurs relations avec les entreprises, les services de l'administration ou les services sociaux.

Principe de fonctionnement

2D-Doc utilise le standard à codes-barres bidimensionnel

Elle 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

Toutes ces informations sont verrouillées par une signature électronique du hash de ces données, qui permet de garantir l'identification de l'organisme émetteur et l'intégrité du document.

Cette signature électronique est réalisée par une clé privée correspondant à la 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.

ÉTAPES DE VÉRIFICATION

Voici une description précise de toutes les étapes nécessaires pour vérifier complètement la signature ainsi que la chaîne de certificats en précisant le but de chaque étape. Nous avons également indiqué quelles étapes ont été effectivement implémentées.

I. Récupération des données du 2D-doc

- Décoder le C40
- Extraire l'en-tête
- Récupérer le numéro d'AC
- Récupérer le numéro du certificat

II. Vérification de la signature 2D-doc

- Chercher l'AC dans le TSL pour récupérer l'URL des certificats de l'AC
- Utiliser cet URL avec le numéro de certificat du participant pour récupérer le certificat
- Vérifier la signature 2D-doc avec la clé publique du participant

III. Vérifier le certificat du participant

- Vérifier les informations
- Vérifier la signature
- Récupérer l'URL de CRL et vérifier la révocation du certificat
- Vérifier la date du certificat

IV. Vérifier le certificat de l'AC

- Vérifier la signature
- Récupérer l'URL de CRL et vérifier la révocation du certificat
- Vérifier la date du certificat

V. Vérifier la TSL

I. Récupération des données du 2D-doc

Décoder le C40

Lorsqu'on lit un 2D-doc, le résultat récupéré est encodé en C40 ou en binaire. L'encodage C40 permet d'optimiser la taille des données et donc d'un code 2D-doc, mais également d'être lisible par un grand nombre de lecteurs de codes à barres. Dans le cadre de notre projet, nous nous sommes uniquement intéressés au format C40. La toute première étape consiste donc à décoder ce qu'on récupère lors de la lecture du 2D-doc. Pour cela, il est possible d'utiliser des librairies Java.

Une fois décodé, le 2D-doc se décompose en 3 parties :

- L'en-tête
- Les données
- La signature

C'est dans la première partie que se trouve bon nombre d'informations utiles pour la suite de la vérification.

Extraire l'en-tête

Extraire l'en-tête du 2D-doc est une étape importante car elle permet de récupérer des informations essentielles pour la suite des opérations. La 1^{ère} chose à faire est de trouver la version du 2D-doc, lisible au caractère 4 de l'en-tête. Il y a 4 versions disponibles : 01, 02, 03 et 04. Selon cette version, la taille de l'en-tête change. C'est pour cela qu'il est crucial de trouver la version pour extraire l'en-tête.

Pour l'extraire, il suffit de découper la chaîne de données au bon endroit (22, 24, 26 caractères pour les versions 01 et 02, 03, 04). Une fois ceci fait, on peut passer à l'étape suivante.

Récupérer le numéro d'AC

Dans l'en-tête du 2D-doc, les principales informations que l'on peut retrouver sont

- Le marqueur DC
- La version 04
- L'identifiant de l'Autorité de Certification FR0A
- L'identifiant du participant XT4A
- La date d'émission 0E84
- La date de signature 0E8A

L'étape consiste à récupérer le numéro de l'Autorité de Certification, pour pouvoir aller chercher dans la TSL des informations à son sujet. Notamment l'URL qui contient tous les certificats qu'elle a émis, et potentiellement l'URL pour la CRL, afin de vérifier plus tard la validité du certificat de l'AC.

Cet identifiant est composé de 4 caractères situés juste après le marqueur et la version du 2D-doc.

Récupérer le numéro du certificat

Comme pour l'AC, il va falloir récupérer l'identifiant du participant, dans le but de trouver son certificat lors d'une requête HTTP. Ce numéro est composé de 4 caractères situés juste après ceux de l'identifiant de l'AC.

II. Vérification de la signature 2D-doc

Chercher l'AC dans le TSL pour récupérer l'URL des certificats de l'AC

Une fois en possession du numéro de l'AC et de celui du certificat du participant, on va d'abord chercher cette AC dans l'annuaire des autorités de certification dit TSL. Il s'agit simplement de parcourir le fichier XML jusqu'à trouver l'identifiant correspondant à celui de l'AC récupéré. Il faut savoir que chaque autorité de certification conserve tous les 2D-doc qu'elle a signés. Par conséquent, il est intéressant de retrouver cet URL qui est contenu dans le XML entre les balises *< tsl:TSPInformationURI >*.

Avec cet URL, il est possible de retrouver n'importe quel certificat participant signé par cette autorité grâce à l'identifiant du certificat participant, lui-même récupéré dans l'en-tête du 2D-doc.

Utiliser cet URL avec le numéro de certificat du participant pour récupérer le certificat

Afin d'avoir accès aux certificats X.509 des participants 2D-DOC pour les différentes autorités de certification, il faut utiliser une requête RFC4387. Elle nécessite de faire une requête HTTP GET en utilisant l'URL précédemment trouvé. Le schéma à utiliser est le suivant :

http_URL = "http:" "://" host [":" port] [abs_path ["?" query]]

Avant le point d'interrogation on retrouve donc l'URL qui est récupéré, puis pour les paramètres il faut indiquer un attribue et une valeur qui permettent de retrouver le certificat souhaité.

query : attribute = value

Dans la liste des attributs proposés, on retrouve le « *ComonName* » que l'on a récupéré dans l'en-tête du 2D-doc. La requête sera donc de la forme *name=IDparticipant*. On aura par exemple l'URL suivant pour récupérer le certificat de la facture SFR donnée en test (AC = AriadNEXT, identifiant du participant = SFR2) :

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

Vérifier la signature 2D-doc avec la clé publique du participant

Une fois le certificat du participant en notre possession, il faut ensuite récupérer la clé publique contenue dans le certificat pour procéder à la vérification de la signature 2D-doc. Ici nous avons converti le certificat récupéré en certificat X509, puis utilisé la méthode *getPublicKey()*.

Pour vérifier la signature il faut tout d'abord la décoder car elle est en base 32. Puis, il faut suivre plusieurs étapes afin de mener à bien la vérification. La première chose à faire est de créer une instance de signature avec le bon type, ici « *SHA256withECDSA* », puis de l'initialiser avec la clé publique du participant. Ensuite, il faut récupérer l'en-tête et la zone de message afin de les injecter dans l'instance de signature. Puis finalement vérifier la signature.

Pour vérifier une signature, il faut en fait la déchiffrer au préalable en utilisant la clé publique du signataire pour obtenir le hash des données du 2D-doc. Les différentes étapes énumérées ci-dessus permettent en fait de recalculer le hash des données (header + message) pour le comparer avec le hash déchiffré.

III. Vérifier le certificat du participant

Vérifier les informations

Lorsqu'on vérifie un certificat, il faut d'abord voir si le nom du sujet correspond bien au nom indiqué sur le certificat. Il faut aussi vérifier les key usages et le nom de l'émetteur. Cette étape n'a pas été implémentée dans le code.

Vérifier la signature

Afin de vérifier la signature du certificat du participant, il faut dans un premier temps récupérer la clé publique de l'autorité de certification car c'est cette dernière qui l'a signé avec sa clé privée. Ce certificat est disponible dans le TSL entre les balises `<tsl:X509Certificate>` au format PEM. Ce dernier est facilement convertissable en version X509 qui possède la méthode `getPublicKey()` avec laquelle on peut ainsi extraire la clé publique de l'AC.

Une fois la clé de l'AC récupérée, il suffit d'utiliser la méthode `certificatParticipant.verify(certificatCA.getPublicKey())`.

Récupérer l'URL de CRL et vérifier la révocation du certificat

Une des étapes de vérification des certificats est de s'assurer que ce dernier n'est pas référencé dans la CRL, en d'autres termes, qu'il a été révoqué. Pour cela il faut récupérer le lien de la CRL au point de distribution qui se trouve dans le certificat en question. Une fois ceci fait, on vérifie que le numéro du certificat ne se trouve pas dans cette liste. Si celui-ci y est, c'est qu'il a été révoqué.

Vérifier la date du certificat

Une autre étape de vérification des certificats est de vérifier la période de validité. En effet, un certificat peut avoir dépassé sa date limite mais ne pas se trouver dans la CRL. Il faut donc vérifier cette date en la récupérant sur le certificat en question et en la comparant avec la date du jour.

IV. Vérifier le certificat de l'AC

De même que pour le certificat du participant, il faut vérifier le nom du sujet, le nom de l'émetteur, les key usages, la période de validité, la révocation et la signature. Les étapes qui ont été implémentées sont la vérification de la signature, de la période de validité et de la révocation.

Vérifier la signature

Pour vérifier la signature des 4 autorités de certification, il a d'abord fallu jeter un œil à leurs certificats. Après analyse, on remarque que seule FR01 est une autorité de certification racine. On peut donc vérifier la signature de son certificat avec sa propre clé publique. Pour les autres, nous sommes allés chercher les certificats des autorités de certification racines qui se trouvent sur les sites Internet des AC concernées. Nous les avons mis en dur dans notre code pour faciliter la vérification. Il ne restait qu'à récupérer la clé publique de ces AC root pour vérifier la signature des certificats de FR02, FR03 et FR04.

Récupérer l'URL de CRL et vérifier la révocation du certificat

De même que pour les certificats des participants, nous avons vérifié si les certificats des AC étaient révoqués.

Vérifier la date du certificat

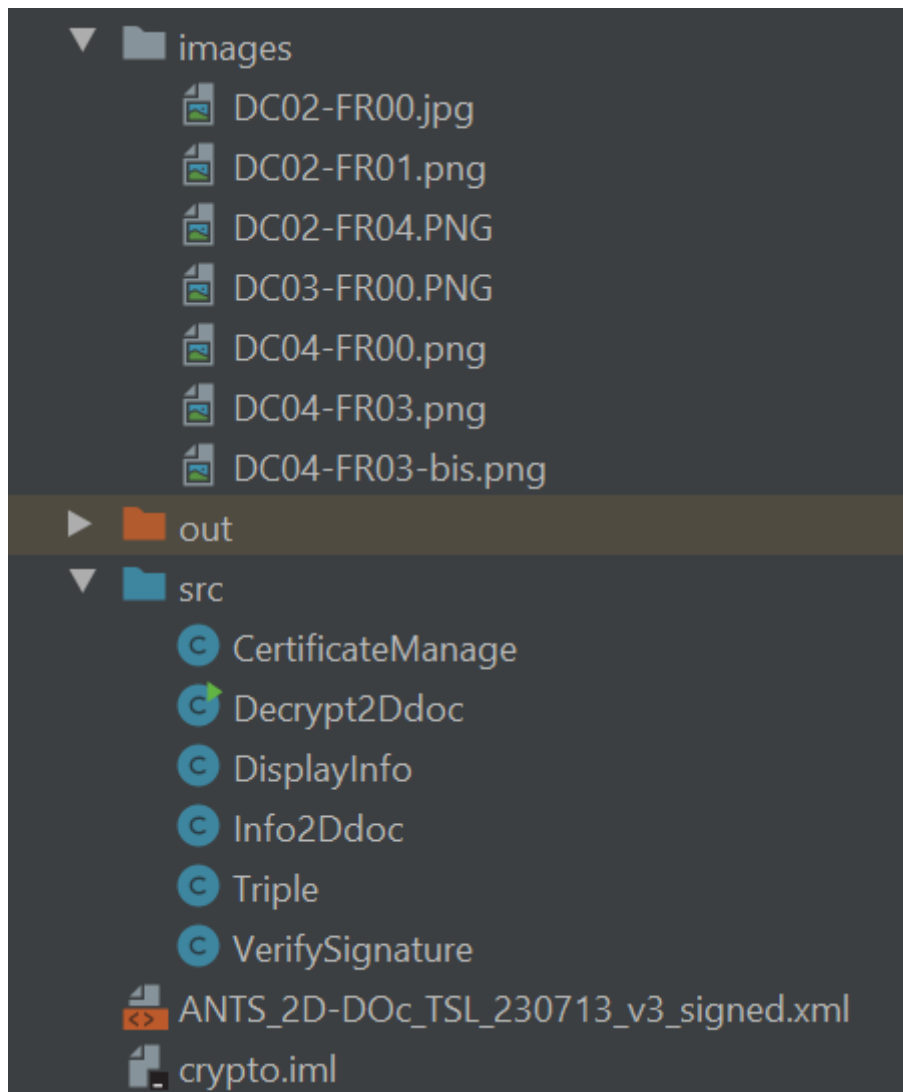
Comme pour les certificats des participants, nous avons également vérifié si les périodes de validité étaient encore d'actualité ou si les certificats avaient expiré.

V. Vérifier la TSL

La dernière étape consiste à vérifier le haut de la chaîne de confiance et donc la TSL. C'est un document XML signé. Le principe est donc de récupérer la signature qui se trouve en bas du document ainsi que le certificat qui est renseigné. Ce certificat contient la clé publique qui permet de déchiffrer la signature du document XML. Ensuite, il faut recalculer le hash des données signées pour le comparer avec la signature déchiffrée. Cette étape n'a pas été implémentée car nous avons rencontrés des erreurs que nous n'avons pas pu corriger.

STRUCTURE DU PROGRAMME

Présentation générale de la structure



Les deux dossiers importants pour le projet sont

- Images : contient tous les 2D-doc testés
- Src : contient toutes les classes utilisées pour le projet

On retrouve également le XML de la TSL à la racine du projet.

Classe Decrypt2Ddoc

Decrypt2Ddoc est la classe qui contient la fonction main.

Les étapes suivies dans le main sont :

- Récupération de l'information contenue dans l'image 2D-doc en String
- Récupération de l'en-tête de la String
- Récupération de la zone message
- Déchiffrement et affichage des informations de la zone message
- Récupération de la signature 2D-doc
- Récupération de l'ID de l'AC
- Récupération de l'ID du certificat participant
- Récupération de l'URL qui contient tous les certificats signés par l'AC
- Récupération du certificat du participant à partir de cette URL
- Récupération du certificat de l'AC
- Vérification de la signature 2D-doc
- Vérification de la signature du certificat participant
- Vérification de la période de validité et de la révocation du certificat participant
- Vérification de la période de validité et de la révocation du certificat CA
- Vérification de la signature du certificat de l'AC
- Vérification de la TSL

La classe Decrypt2Ddoc contient aussi 4 fonctions :

- Une fonction de conversion d'un certificat en String en un certificat X509
- Une fonction de conversion d'une signature en byte en une signature au format DER
- La fonction de vérification de la signature 2D-doc
- La fonction de vérification de la signature de la TSL

Classe CertificateManage

Cette classe contient les fonctions relatives aux opérations sur les certificats :

- Les fonctions de conversion, décodage et affichage : String en X509, afficher des byte en hexadécimal, conversion de byte en X509, affichage des informations du certificat.

- Les fonctions de récupérations : récupération du certificat participant, récupération du certificat de l'AC depuis la TSL
- Les fonctions de récupération de la liste de révocation
- Les fonctions de vérification de signature des certificats : vérification de la signature du participant et vérification de la signature de l'AC
- Les fonction de vérification de la révocation
- Les fonctions de récupération des informations de la TSL : récupération du certificat de la TSL et récupération de la signature de la TSL

Classe DisplayInfo

Cette classe contient les informations et fonctions nécessaires au décodage de la zone message du 2D-doc.

On a donc tout d'abord une liste de triplets qui contiennent

- Un ID
- Le nom de la catégorie correspondante
- La taille de la donnée

Si la taille de la donnée est à null, c'est qu'elle n'est pas fixe et qu'il va falloir parcourir la donnée jusqu'à tomber sur une balise RS.

Il y a ensuite deux fonctions :

- `printMessageInfo()` : elle permet d'afficher le type de la donnée et la donnée elle-même
- `getData()` : elle permet de récupérer la donnée en fonction de sa taille ou en parcourant la zone de message jusqu'à trouver une balise RS.

Classe Info2Ddoc

Ici on regroupe toutes les fonctions nécessaires au découpage des informations du 2D-doc. On a également la fonction de décodage C40.

On a donc les fonctions de récupération du header, de l'ID de l'AC, de l'ID du certificat participant, du type de document, de l'URL où sont stockés les certificats participants, de récupération de la zone message du 2D-doc et de récupération de la signature.

Classe Triple

Elle est utilisée pour la liste de triplets de la classe DisplayInfo.

Classe VerifySignature

Cette classe est réservée à la vérification de la signature de la TSL. Elle possède 2 fonctions :

- `getXmlDocument()` : qui permet de récupérer le document XML de la TSL, d'en faire un type Document et de le parser
- `isXmlDigitalSignatureValid()` : fonction qui vérifie la signature de la TSL en récupérant la clé publique se trouvant dans le certificat de la TSL, la signature, et toute la donnée qui doit être hashée pour vérifier l'intégrité de ce document


2D-DOC VÉRIFIÉS

Vous trouverez ci-dessous les 2D-doc dont les signatures et la chaîne de confiance ont été vérifiés.

DC04-FR03

```
===== Start retrieving informations =====

[HEADER] DC04FR03CN001A811A81B101FR
[DATA] B0MANON HELENE MARIA0B2CHILLON0B726072000BB071028484FH0BKNCY8-NRHKFW-79
[2D doc Information]
  - Liste des prénoms : MANON HELENE MARIA
  - Nom patronymique : CHILLON
  - Date de naissance : 26072000
  - Numéro ou code d'identification de l'étudiant : 071028484FH
  - Numéro de l'Attestation de versement de la CVE : NCY8-NRHKFW-79
  - Type de document : Justificatif académique - Attestation de Versement de la Contribution à la Vie Etudiante
[SIGNATURE] OWM33XZAE6PDHTZHJUQP4V65MUV26CYGVZDTXAJQD3Z6XWHAU3Q0G2HAN3PVYMP6UG6HNG5N5YI67KJXKJZ2RRHPV5PPF6YEBJJJ66I
```

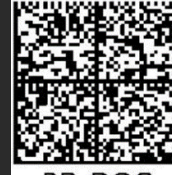


2D-Doc

DC02-FR00

```
===== Start retrieving informations =====

[HEADER] DC02FR000001125E125A02
[DATA] 26FR244400010M/ECHANTILLON/ARTHUR025NANTES02242 SQUARE DES ECHANTILLONS0
[2D doc Information]
  - Pays de service des prestations : FR
  - Code postal ou code cedex du point de service des prestations : 44000
  - Ligne 1 de la norme adresse postale du bénéficiaire de la prestation : M/ECHANTILLON/ARTHUR
  - Localité de destination ou libellé cedex du point de service des prestations : NANTES
  - Ligne 4 de la norme adresse postale du point de service des prestations : 42 SQUARE DES ECHANTILLONS
  - Type de document : Justificatif de domicile - Avis de taxe d'habitation
[SIGNATURE] UGYXWRVJY54QKC56SYE7X43DU4AN02SPHTYKGPURU050D44RSV6DA4REFUMTNBNCIC50MHZ34P4SODPIABD4YBH33T5QTWFXEZNDYQQ
```




2D-DOC

DC02-FR01

```
===== Start retrieving informations =====

[HEADER] DC02FR01SFR215A015A101
[DATA] 26FR241301510M/BOUDJEDRI/AMMAR022309 RUE DE LYON025MARSEILLE01810484693110
[2D doc Information]
  - Pays de service des prestations : FR
  - Code postal ou code cedex du point de service des prestations : 13015
  - Ligne 1 de la norme adresse postale du bénéficiaire de la prestation : M/BOUDJEDRI/AMMAR
  - Ligne 4 de la norme adresse postale du point de service des prestations : 309 RUE DE LYON
  - Localité de destination ou libellé cedex du point de service des prestations : MARSEILLE
  - Numéro de la facture : 1048469311
  - Type de document : Justificatif de domicile - Factures de fournisseur d'énergie | Factures de téléphonie | Fact
[SIGNATURE] V6F73UOWHLSLPJUCUULFDORWP6QHMKBMKURFWH7F3SK7JE4DWMVCP5AXCKIQ2HW7GAARSET6I5EDBUI4IU5HL4FGEY7A4NSYKA704Q
```




2D-DOC

DC04-FR00

```
===== Start retrieving informations =====

[HEADER] DC04FR0000011CE91D231301FR
[DATA] 6J26K92012020040123591236L010420206PAUTORISE A TRAVAILLER06Q750312052162SPECIMEN063BERTHIER SPECIMEN060NATA
[2D doc Information]
- Type de document étranger : 2
- Numéro de la demande document étranger : 9201202004012359123
- Date de dépôt de la demande : 01042020
- Autorisation : AUTORISE A TRAVAILLER
- Numéro d'étranger : 750312052162SPECIMEN
- Nom d'usage : BERTHIER SPECIMEN
- Liste des prénoms : NATACHA/CORINNE
- Genre : F
- Date de naissance : 12071973
- Lieu de naissance : BUENOS AIRES
- Pays de naissance : AR
- Nationalité : AR
- Ligne 4 de l'adresse postale du domicile : 145 AVENUE DES SPECIMENS
- Code postal ou code cedex de l'adresse postale du domicile : 75000
- Commune de l'adresse postale du domicile : PARIS
- Catégorie du titre : ENFANT ENTRE PAR REGROUPEMENT FAMILIAL
- Date de début de validité : 01052020
- Date de fin de validité : 30062020
- Code pays de l'adresse postale du domicile : FR
- Type de document : Justificatif d'identité - Document étranger
[SIGNATURE] PJIX6XX3TPFYJTHI6IEIPDU5C5D2X4VBHZEIC24WPW4I3XETZ0GETCMU2J2DKLEGSJDDKYJXBTVBK KY5MPAKE0L2A4RZNIUFTHW44A
```




2D-DOC

DC03-FR00

```
===== Start retrieving informations =====

[HEADER] DC03FR0000011242163604
[DATA] 10M/IMPOSABLE/FRANCOIS04012345678901234110428760
[2D doc Information]
- Ligne 1 de la norme adresse postale du bénéficiaire de la prestation : M/IMPOSABLE/FRANCOIS
- Numéro fiscal : 1234567890123
- Revenu fiscal de référence : 1042876
- Type de document : Justificatif de ressources - Avis d'impôt sur le revenu
[SIGNATURE] AB2MS643NQNY7X5BMSHCW7FJ5DQ2LCOVZ60PVXU05NATQ0BBQKE753DIFQOV75WZNXBW2UEHL23Q3QOX60XJRG0SK35SLLCM3B6GJSly
```




2D-DOC

DC02-FR04

```
===== Start retrieving informations =====

[HEADER] DC02FR04BYT11A231A2401
[DATA] 26FR244900010M/TAIEB/AOUNI0221 RUE GUSTAVE MAREAU01811890260000418002FACTURE FNB0036P01D5,16019139571334007060840
[2D doc Information]
- Pays de service des prestations : FR
- Code postal ou code cedex du point de service des prestations : 49000
- Ligne 1 de la norme adresse postale du bénéficiaire de la prestation : M/TAIEB/AOUNI
- Ligne 4 de la norme adresse postale du point de service des prestations : 1 RUE GUSTAVE MAREAU
- Numéro de la facture : 11890260000418
- Catégorie de document : FACTURE FNB
- Sous-catégorie de document : GP
- Montant TTC de la facture : 5,16
- Numéro de client : 139571334
- Heure de l'association entre le document et le code 2D-Doc : 060840
- Type de document : Justificatif de domicile - Factures de fournisseur d'énergie | Factures de téléphonie | Factures
[SIGNATURE] LEZ3Y0JYCUDCG4MUFQLILRT3W2GMI52PSFDU5L4V664J6LF7Y4YEA VYEUQL6TVCPHJ6KXBF476IZSBJ7Q5YZSAM2FA32JH664DBGY
```



2D-DOC

DIFFICULTÉS RENCONTRÉES

Peu de connaissance du langage

Le code devait être réalisé de préférence en Java. Nous en avons très peu fait l'an dernier et nous avons donc dû nous remettre plus ou moins à niveau. En effet, nous avons eu quelques difficultés avec la gestion des librairies et l'organisation du code. Cependant, Java est un langage de programmation très bien documenté et nous avons pu avancer rapidement malgré notre manque d'expérience.

Sujet complexe

Il nous a fallu un certain moment pour bien assimiler la complexité du projet. Pour savoir quelle entité signe quoi. On a en premier lieu vérifié le certificat du participant avant la signature du 2D doc car nous avions mal interprété le schéma du projet. Ce n'est qu'après avoir eu plus de détail que nous avons rajouté la partie de vérification de la signature du 2D-Doc.

Déchiffrer les informations de la zone message du 2D doc

Nous avons rencontré des difficultés sur le déchiffrement de la zone de 2D-doc. Le principe n'est pas extrêmement compliqué une fois bien compris. Cependant le travail est fastidieux car il faut rentrer à la main tous les ID possibles puis, au niveau du code, il fallait repérer les balises `<RS>`, `<GS>` ou `<>` qui délimitent la fin d'une donnée ou la troncature de celle-ci. Cette difficulté a également été rencontrée lors du découpage entre la zone message et la signature. La technique est de passer par des tableaux de bytes car les balises sont invisibles dans le format String.

Utilisation des URL

Lorsque nous avons dû exploiter les URL pour récupérer les certificats des participants, nous avons compris qu'il fallait utiliser les requêtes RFC4387. Nous n'avons pas trop eu de difficultés pour les certificats signés par FR03. En revanche, nous avons mis plus de temps à comprendre et écrire la bonne URL pour les autres AC car nous cherchions trop compliqué. Finalement, il suffisait de prendre l'URL telle quelle et rajouter le « *?name=IDparticipant* ».

Vérification de la TSL

La vérification de la TSL a été une partie qui nous a posé beaucoup de problèmes. Nous ne savions pas comment nous y prendre puis nous sommes partis sur la vérification d'un document XML. A partir de là, nous avons suivi des méthodes qui ont toutes abouties sur des erreurs de code assez compliquées à résoudre. Nous n'avons d'ailleurs toujours pas réussi à résoudre le problème.

AMÉLIORATIONS POSSIBLES

Arrangement du code

Comme expliqué précédemment nous n'avons pas l'habitude de coder en Java. Une des améliorations possibles serait donc d'optimiser l'agencement du code qui n'est pour l'instant pas très bien organisé. En effet, nous avons essayé de découper le code en plusieurs classes pour clarifier l'exécution et soulager Decrypt2Ddoc qui constituait au départ notre seule classe dans laquelle tout était codé.

Vérification des certificats

Lors des vérification de certificat, on ne vérifie que la révocation, la période de validité et la signature. On pourrait ajouter la vérification du nom du sujet, du nom de l'émetteur et des extensions de clé.

Affichage

L'affichage des résultats dans la console est assez basique. On pourrait utiliser une librairie qui permettrait de faire des tableaux ou du moins, rendre l'affichage plus attrayant qu'une simple liste d'informations.

Vérification de la TSL

Nous avons rencontré beaucoup de problèmes lors de la vérification de la TSL. Par conséquent et à cause d'un manque de temps, nous n'avons pas pu implémenter cette vérification. Elle pourrait donc être une piste d'amélioration afin de compléter la vérification de la chaîne de confiance.

Prendre en charge la version DC01

Le code ne permet de vérifier que les 2D-doc avec une version supérieure à 01. Il serait intéressant d'intégrer la vérification de ces anciens 2D-doc qu'on retrouve encore sur quelques papiers.

CONCLUSION

Notre projet permet ainsi la vérification de l'intégrité d'un 2D-doc, le décodage de ses données et la vérification de la chaîne de confiance.

Cependant, il n'est pas impossible de réaliser des attaques sur ce dispositif. En voici un exemple :

Man in the middle



Il est en effet possible d'attaquer via un man in the middle lorsque le participant récupère la TLS sur le site de l'ANTS. L'attaquant pourrait intercepter le trafic réseau en établissant une connexion de type SSL (avec un outil comme ZAP par exemple) avec la victime et renvoyer au participant une fausse TLS.

Ça aurait pour but de briser la chaîne de confiance vue que l'on ne pourrait plus vérifier avec certitude les certificats.

L'attaquant pourrait ainsi faire croire à la victime que ses documents sont bien officiels, mais pourrait également rajouter sa propre autorité de certification racine avec lequel il pourrait fausser des 2D-doc.

Notre code possédant la TSL et la plupart des certificats en dur, nous évitons ainsi ce type d'attaque en réduisant le nombre de requêtes HTTP. Cependant il est nécessaire de vérifier régulièrement l'intégrité de la TSL et la mettre à jour car elle est la base de la chaîne de confiance.