La cryptologie dans TousAntiCovid

Tom Kubasik, Hugo Hersemeule
Info002 - Cryptologie

Introduction

TousAntiCovid est l’application utilisé par le gouvernement français dans la lutte contre l’épidémie du COVID-19

Originellement appelé StopCovid, l'application avait pour principale fonctionnalité le contact tracing par Bluetooth. Devenue TousAntiCovid, elle embarque maintenant les Certificats de Vaccinations, de rétablissement et de test dans le cadre du Pass sanitaire.

Étant une application du gouvernement Français gérant des données personnelles de santé, la sécurité et donc la cryptologie est un sujet important dans le développement de l’application.

Les certificats de vaccinations

Les certificats de vaccinations étaient, avant de passer sous une norme européenne, sous la norme 2D-DOC.

Créée par l’Agence nationale des titres sécurisés (ANTS) en partenariat avec des entités privées à la demande du Ministère de l'intérieur, le norme 2D-Doc permet de lutter contre la fraude documentaire tout en développant les échanges de documents administratifs entre les administrations et les citoyens.

Un 2D-Doc est un code-barres bi-dimen sionnel, il contient les données suivantes:

- 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

Pour garantir l'authenticité des informations, la date d'émission du document est verrouillée par une signature électronique du hash de ces données, qui garantit l'identification de l'organisme émetteur et l'intégrité du document.

Signature

Le 2D-Doc est signé électroniquement par une clé privée générée par l'administration française dont la clé publique est connue comme officielle. Cela permet de contrôler l'authenticité du document pour tout scanner disposant de la clé publique du signataire du document.
On peut voir dans le Json de configuration de TousAntiCovid disponible à cette adresse :
https://app-static.tousanticovid.gouv.fr/json/version-37/Config/config.json
Que l'application dispose des clés officielles d'émission de certificat de type 2D-Doc pour en vérifier l'authenticité d'un certificat importé dans l'application.


```
{
            "name":"app.walletPubKeys",
            "value":[
                {
                    "auth":"FR03",
                    "pubKeys":{
                        "AHP1":"-----BEGIN PUBLIC
KEY-----MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEPnxJntwNwme9uHSasmGFFwdC0FWNEpuc
gzhjr+/AZ6UuTm3kL3ogEUAwKU0tShEVmZNK4/lM05h+0ZvtboJM/A==-----END PUBLIC
KEY-----",
                        "AHP2":"-----BEGIN PUBLIC
KEY-----MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEOYUgmx8pKu0UbyqQ/kt4+PXSpUprkO2Y
LHmzzoN66XjDW0AnSzXorFPe556p73Vawqaoy3qQKDIDB62IBYWBuA==-----END PUBLIC
KEY-----",
                        "AV01":"-----BEGIN PUBLIC
KEY-----MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE1T9uG2bEP7uWND6RT/lJs2y787BkEJoR
MMLXvqPKFFC3ckqFAPnFjbiv/odlWH04a1P9CvaCRxG31FMEOFZyXA==-----END PUBLIC
KEY-----",
                        "AV02":"-----BEGIN PUBLIC
KEY-----MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE3jL6zQ0aQj9eJHUw4VDHB9sMoviLVIlA
DnoBwC43Md8p9w655z2bDhYEEajQ2amQzt+eU7HdWrvqY23Do91Izg==-----END PUBLIC
KEY-----",
                        "ING1":"-----BEGIN PUBLIC
KEY-----MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEaMOVmK4FAlYeU8tUGpQgtpuLgwSn/arX
/4ltKt/wYCole7rB4ECRLtSH4domHU/Z1cjUmVfyGH6NcboWrys9ww==-----END PUBLIC
KEY-----"
                    }
                }
            ]
        },
```


Si la clé publique nécessaire pour décrypter le certificat n'est pas dans le json de configuration, il sera impossible d'importer le certificat.

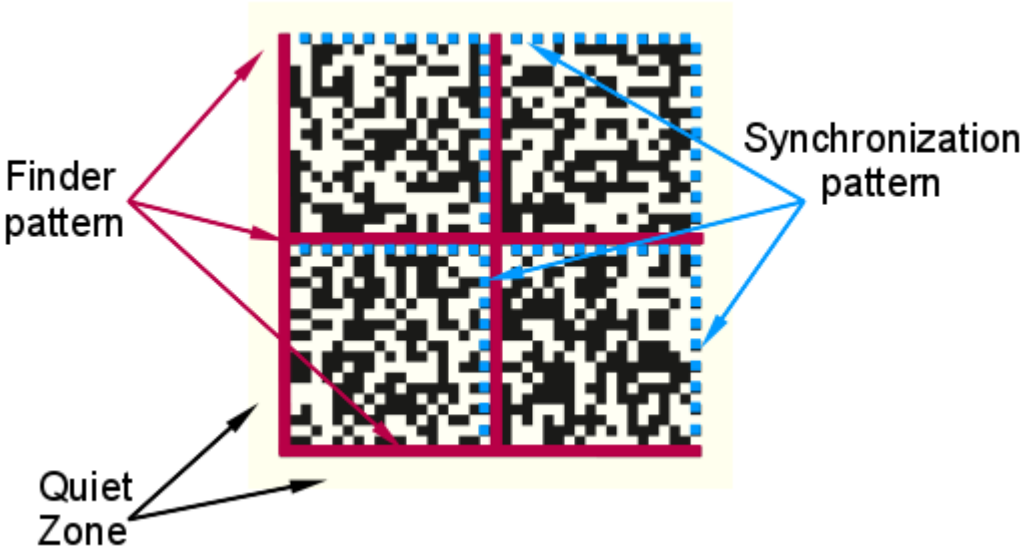
Si on enlève la clé privée utilisée, n'importe qui peut générer un pass vaccinal, cela est possible sur ce site : https://github.pathcheck.org/eu.dgc.html

Seul le fait que la clé publique pour déchiffrer les données n'est pas connue comme clé officielle assure que ce certificat n'est pas un certificat officiel.

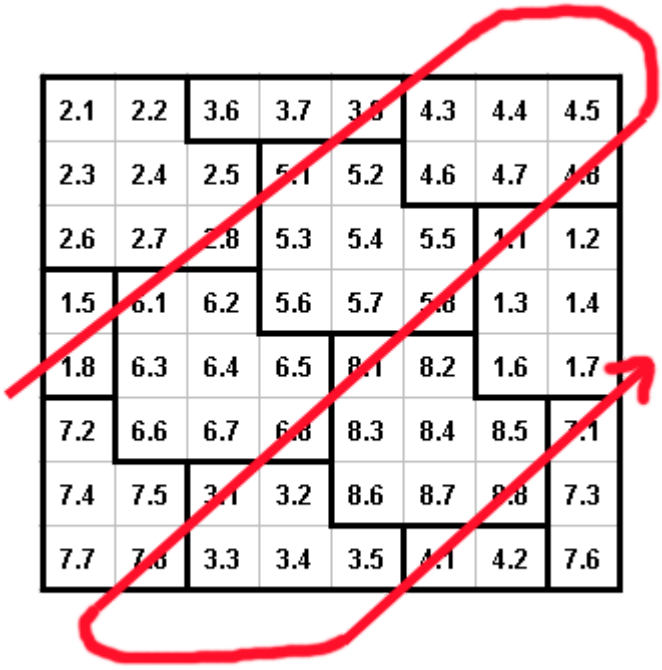
DataMatrix

La norme 2D-Doc est en fait une sorte de Data Matrix.
une datamatrix est une forme de barcode à deux dimensions inventé par l'entreprise ID-Matrix en 2005 et est principalement utilisée dans le marquage de produit ou de pièces mécaniques.
La NASA l'utilise notamment pour le marquage des composants de ses navettes spatiales.
Afin de faciliter le calibrage du lecteur de code-barres, un DataMatrix est entouré d'une barre noire sur les côtés gauches et bas, et des pointillés sur les deux autres côtés.

Un Datamatrix peut contenir jusqu'à 2335 caractères alphanumériques, c'est une quantité important d'information par rapport à sa surface réduite, mais cela n'est pas suffisant pour stocker un Certificat de Covid. Un seul certificat utilise donc 4 Datamatrix.

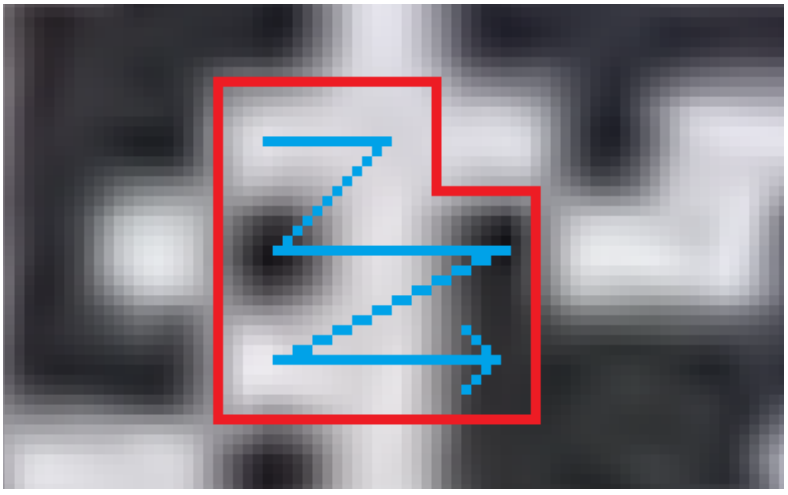


Les données sont stockées en bloc de 8 bits en forme de L et le barcode se lit en diagonale.



Pour un bloc X, on lit les pixels X.1, X.2, X.3, X.4, X.5, X.6, X.7, X.8

Si on prend en exemple ce bloc :



On peut en extraire la valeur binaire 00101001 soit en décimal 41.

La structure du certificat

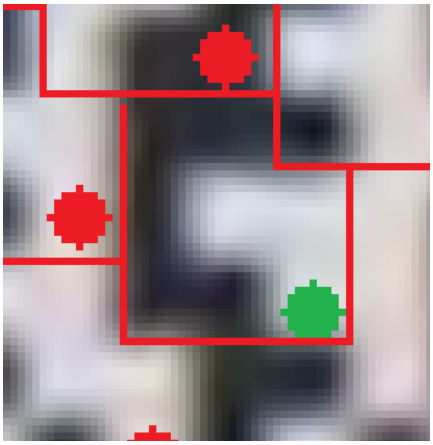
On retrouve sur le certificat plusieurs blocs information délimités par des “.” et commençant par la Lettre L

On a ainsi ces blocks :
L0 -> Nom de famille
L1 -> Prénom
L2 -> Date de naissance

L3 -> Maladie couverte
L4 -> Agent prophylactique
L5 -> Vaccin
L6 -> Fabricant du vaccin
L7 -> Nombre de doses faites
L8 -> Nom de doses attendue
L9 -> Date
LA -> État du cycle de vaccination
Dernière partie -> Signature de 64 octets en base 32

La première partie du certificat est en Encodé en format TEXT,
Les caractères sont décalés de 1 dans la Table ASCII pour les valeurs de 0 a 129,
Pour les valeurs entre 130 et 229, cela correspond à des paires de Chiffres entre 0 et 99,
Ainsi 134 se lit 134-130= 04. ce qui permet d'encoder deux octets en un.

Le premier bloc avec une valeur de 230 annonce la fin de l'encodage TEXT et le début de l'encodage C40.



L'encodage TEXT est très pratique pour stocker des nombres mais l'encodage C40 est plus efficace pour stocker les lettres majuscules qui dont est constitué en grande majorité le Certificat 2D-Doc.

Le C40 utilise une table de caractères réduite à 40 caractères, encodés en base 40.
Par exemple, la paire d'octets **22 EB** se lit comme un nombre de 16 bits 0x22EB (= 8939). On soustrait 1, puis on décode en base 40 : $5 * 40^2 + 23 * 40 + 18$

Ces trois nombres en base 40 nous donnent les trois index des caractères dans le charset.

5 = "J"
23 = "J"
18 = "E"

La compression est efficace car on encode 3 octets en seulement 16 bits au lieu de 24 normalement.

Contact Tracing

L'une des fonctionnalités fortes mises en avant par TousAntiCovid est le **Contact Tracing** *(ou traçage numérique)*. Il s'agit de pouvoir notifier à un utilisateur qu'il est cas contact, c'est-à-dire qu'il a été en proximité physique avec une personne positive au Covid19, bien que la personne positive était encore asymptomatique au moment où les deux personnes se sont croisées.

Cette fonctionnalité a notifié plus de 2 millions de personnes qu'elles étaient cas contact.

Pour qu'une telle technologie soit acceptée par la majorité de la population, il faut des garanties en termes de protection des données et de sécurité.
D'ailleurs, il est intéressant de comprendre que le Contact Tracing n'est pas une application de tracking, et n'utilise que le bluetooth, et en aucun cas les données de bornage GSM ni de géolocalisation.

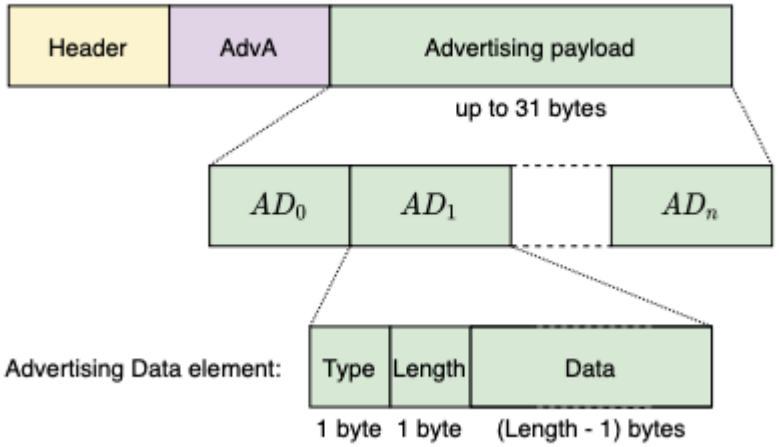
C'est là que le protocole **ROBERT** *(ROBust and privacy-presERving proximity Tracing)* entre en jeu.

Le Bluetooth Low Energy

L'une des technologies employées par le protocole ROBERT est la technologie **Bluetooth Low Energy** ou BLE. C'est une technologie sans fil à moyenne portée permettant d'échanger des données entre des smartphones qui sont à proximité afin de conserver une trace du contact entre leurs propriétaires.

Le BLE n'a pas été conçu pour mesurer les distances, mais le fait que la portée soit limitée, ainsi que la possibilité de mesurer l'intensité du signal reçu permettent d'avoir une estimation de la proximité des deux appareils assez précise.
De plus, le BLE consomme peu d'énergie, ce qui est très important puisque sa fonction première est de tourner en arrière-plan sur des appareils fonctionnant sur batterie.

À l'origine, cette technologie a été pensée pour l'échange entre un appareil central et des appareils périphériques, les appareils périphériques peuvent communiquer avec l'appareil central et ils peuvent se connecter sur des canaux et ainsi échanger des données.
Pour ce faire, le BLE possède un système de découverte d'appareils, appelé le **BLE advertising**, et il est intéressant de voir que ce système de découverte est suffisamment libre dans sa conception pour que son usage soit détourné.
En effet, dans le cas du contact tracing, on va utiliser la procédure de découverte pour réaliser l'échange de données, ce qui permet d'échanger des données sans véritable connexion entre appareils.

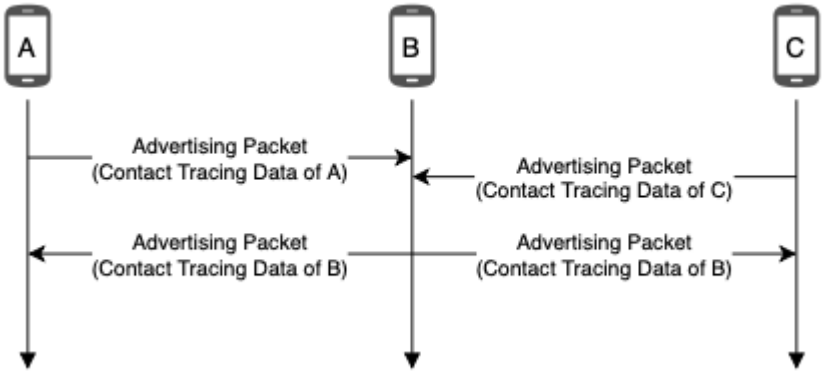


Voici un schéma représentant un paquet de BLE advertising. Le bloc AdvA représente l'adresse de l'émetteur.

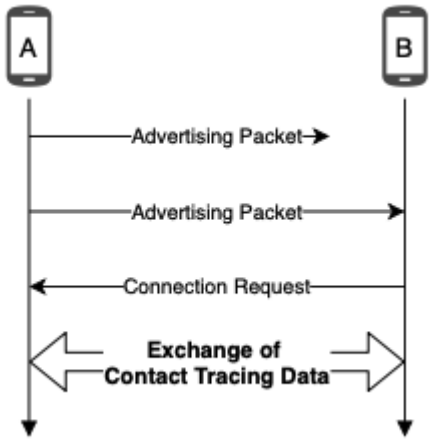
En voyant le schéma de ces paquets permettant l'échange de données en BLE lors de la première connexion entre deux appareils, on constate que l'adresse de l'émetteur est présente. Le BLE comprend donc la possibilité de randomiser les adresses en utilisant la spécification Bluetooth qui est déjà comprise dans tous les appareils Android et iOs.
Les adresses sont ainsi modifiées périodiquement, ce qui empêche les utilisateurs d'être traqués et peut protéger d'autres attaques telles que le *social graph reconstruction* (technique qui permettrait de retrouver toutes les relations entre les utilisateurs).

Quant au payload, il va surtout contenir le temps de transmission qu'il a fallu pour que le message passe d'un téléphone à un autre (ce qui permet d'avoir une estimation de la distance entre les téléphones) ainsi qu'un code de vérification d'intégrité.
En ajoutant le fait que l'adresse du transmetteur est cryptée et régulièrement modifiée aléatoirement, et bien que les communications Bluetooth en elle même ne sont pas confidentielles, c'est-à-dire que n'importe quelle personne bien ou mal intentionnée peut les observer, les attaques de *re-jeu* ne sont pas possibles. (Les attaques de *re-jeu* sont des attaques où l'attaquant capte un message puis en envoie un autre en se faisant passer pour l'utilisateur qui a envoyé le premier message).

À l'aide des paquets BLE advertising, une application telle que Tous Anti Covid peut utiliser un modèle de broadcast, sans jamais créer un canal d'échange entre les téléphones, car les paquets de découverte contiennent eux-mêmes les informations de *contact tracing*. (voir figure ci-dessous)



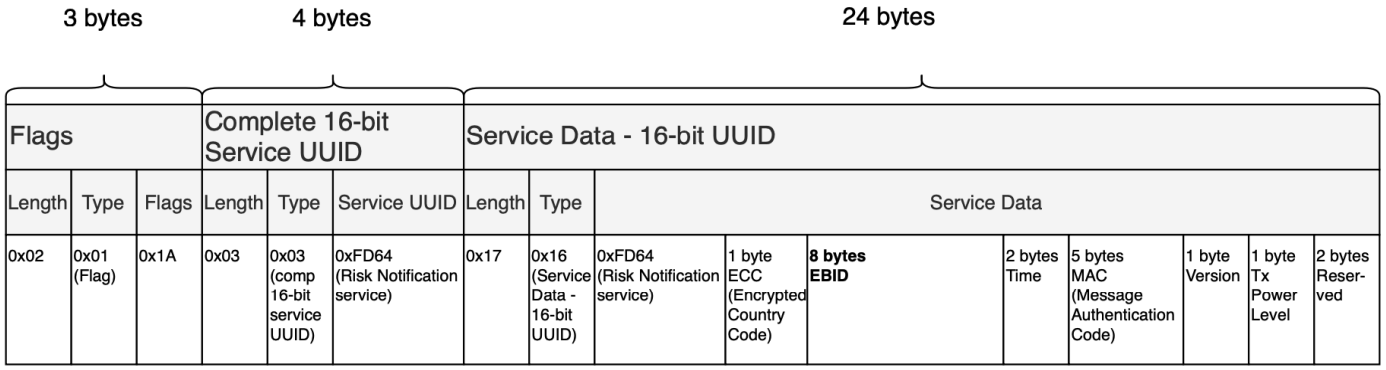
Néanmoins, l'équipe de Tous Anti Covid a décidé de rajouter la possibilité d'utiliser les BLE Advertising pour leur utilisation par défaut, c'est-à-dire la connexion entre 2 appareils dans l'unique objectif de pouvoir supporter des téléphones ayant des versions plus anciennes des systèmes d'exploitation.



Exemple de connexion entre deux appareils en BLE pour Tous Anti Covid lorsque les appareils sont trop anciens pour utiliser le modèle Broadcast.

Maintenant que nous nous sommes intéressés aux méthodes d'échanges de messages utilisés dans Tous Anti Covid, intéressons-nous au payload des messages eux-mêmes.

Payload des messages dans Tous Anti Covid



Ces payloads sont séparés en 3 parties :

- Les **Flags** : Ceux-ci doivent être inclus dans tous les **BLE advertising packet**.
- Le **Service UUID** : C'est l'uuid du service de notification de risque, dans le cas de Tous Anti Covid, il s'agit de *0xFD64*.
- Les **Service Data** :
    - On retrouve l'uuid du service de notification de risque, c'est-à-dire *0xFD64*.
    - L'**Encrypted Country Code : ECC**
    - **EBID :** Il s'agit de l'identifiant temporaire donné régulièrement randomisé dont nous parlions dans le chapitre sur le BLE.
    - **Time** : Enregistre le temps actuel au moment de l'échange, car cela va servir à l'encryption de l'adresse MAC.
    - **MAC** : Le message d'authentification encrypté avec l'EBID et Time en utilisant HMAC-SHA256.
    - Le reste étant utilisé pour des meta données.

Échange client-serveur

Afin de pouvoir notifier les utilisateurs de contact avec des personnes testées positives, il faut que des données soient échangées avec un serveur. C'est la seule chose ou TousAntiCovid est dépendant d'un serveur.

Pour s'assurer que les données ne soient pas interceptées, les échanges sont cryptés via AES avec un échange de clé ECDH (Elliptic curve Diffie Hellman).

Le principe est le suivant :

- le client génère une paire de clé ECC: **clientPrivKey**, **clientPubKey**
- le serveur génère une paire de clé ECC: **serveurPrivKey**, **serveurPubKey**
- le serveur et le client s'échange en clair leur clé publique
- le client calcule la clé partagée avec sa clé privée et la public du serveur: **clientPrivKey * serveurPubKey**

- le serveur calcule la clé partagée avec sa clé privée et la public du client: **serveurPrivKey * clientPubKey**
- Le client et le serveur se retrouve avec la même clé de déchiffrage.

Le code est implémenté comme cela dans l'application TousAntiCovid:

```
override fun getEncryptionKeys(rawServerPublicKey: ByteArray,
   rawLocalPrivateKey: ByteArray,
   kADerivation: ByteArray,
   kEADerivation: ByteArray): Pair<ByteArray, ByteArray> {
   val bouncyCastleProvider = BouncyCastleProvider()
   val keyFactory =
KeyFactory.getInstance(ALGORITHM_ECDH,bouncyCastleProvider)
   val serverPublicKey =
keyFactory.generatePublic(X509EncodedKeySpec(rawServerPublicKey))
   val localPrivateKey =
keyFactory.generatePrivate(PKCS8EncodedKeySpec(rawLocalPrivateKey))

   val keyAgreement = KeyAgreement.getInstance(ALGORITHM_ECDH)

   keyAgreement.init(localPrivateKey)
   keyAgreement.doPhase(serverPublicKey, true)

   return keyAgreement.generateSecret().use { sharedSecret ->
       SecretKeySpec(sharedSecret, HASH_HMACSHA256).safeUse<Pair<ByteArray,
ByteArray>> { secretKeySpec ->
           val hmac = Mac.getInstance(HASH_HMACSHA256)
           hmac.init(secretKeySpec)

           val kA = hmac.doFinal(kADerivation)
           val kEA = hmac.doFinal(kEADerivation)

           Pair(kA, kEA)
       }
   }
}
```

Sources

Open sourcing TousAntiCovid
https://gitlab.inria.fr/stopcovid19

Protocole Robert
https://github.com/ROBERT-proximity-tracing/documents

Documentation de la norme 2D-DOC
https://ants.gouv.fr/nos-missions/les-solutions-numeriques/2d-doc