Integrated Technical Services

Safety First, Service Always

Service S-124 – Guide d’utilisateur

Septembre 2025

Registre des changements

Version	Date	Description	Signature
1.0	2025-09-08	Coordonnées de la zone d'essai mis à jour, section sur les signatures ajoutée et limite des résultats changée à 2000.	M.R.C
1.1	2025-09-03	Section 2.5 déplacée en section 3, correction du nom des figures.	M.R.C

Table of Contents

Section 1 INTRODUCTION

1.1 CONTEXTE

Le service S-124 de la garde côtière canadienne a été conçu pour distribuer les avis à la navigation (AVNAVs) produits par la garde côtière canadienne suivant le standard S-124 et conforme au standard de sécurité “Secure Exchange of Maritime Information” (SECOM).

OBJECTIF

L’objectif de ce document est de décrire comment utiliser le service S-124 pour les utilisateurs comme les distributeurs de produits S-100 et les développeurs de systèmes de navigation maritime (ECDIS, ECS, etc.). Ce service ne consiste pas à télécharger des fichiers ou des cartes. Il s’agit d’un service web exposant un API REST qui accepte des requêtes HTTPS et qui retourne des réponses avec des informations qui doivent être décodées.

Ce document inclut des exemples que vous pouvez suivre pour savoir comment utiliser le service S-124 et décoder ses réponses.

IMPORTANT

Le service S-124 est encore sous développement et représente nos meilleurs efforts pour satisfaire les spécifications S-124 / S-100. À moins d’avis contraire, ls utilisateurs devraient utiliser ce service pour des buts de tests seulement. Des changements sur ce service pourraient survenir sans préavis.

Section 2 Comment utiliser le Service S-124 de la GCC

Le service S-124 de la GCC est un API REST accessible avec le URL:

https://s124.ccg-gcc.gc.ca/api/secom/v1/object

Pour les développeurs, le lien openAPI suivant décrit le service S-124 de la GCC ainsi que ses paramètres:

https://s124.ccg-gcc.gc.ca/swagger-ui/index.html

2.1 REQUÊTE DÉTAILLÉE GET

La requête détaillée GET: https://s124.ccg-gcc.gc.ca/api/secom/v1/object
retourne une liste d'objets 'DataResponse' contenant 3 champs:

- 'data',
- 'exchangeMetadata' and
- 'ackRequest'

Le champ 'data' contient l'AVNAV S-124 encodé en Base64 (voir figure 2.1).

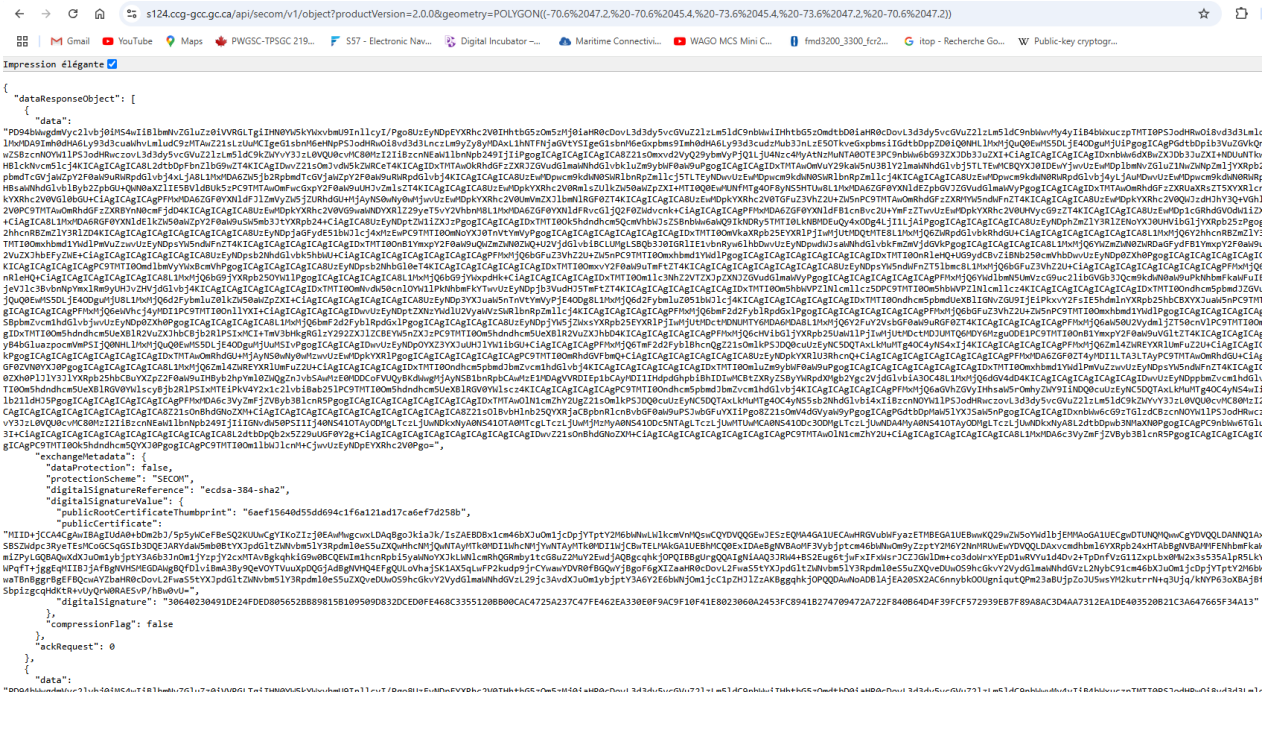

Figure 2.1. S-124 GET Detailed Request Response

Le champ ‘exchangeMetadata’ contient des informations comme la signature numérique du contenu du champ ‘data’, le certificat public qui sert à valider la signature et l’algorithme de la signature utilisée (voir figure 2.1).

La requête retourne les AVNAVs S-124 dans un format ‘S-100 Exchange Set’ (fichier .zip) lorsque le paramètre containerType=1 est utilisé (voir ci-dessous).

Exemple:

Cette requête retourne les AVNAVs actifs dans la zone canadienne des essais S-100, la réponse est retournée sous forme d’un exchange set S-100 contenant les AVNAVs S-124 dans la version 2.0.0:

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?geometry=POLYGON((-73.6%2045.5,%20-73.5%2045.5,%20-73.2%2045.8,%20-73.2%2046.0,%20-72.6%2046.0,%20-69.8%2047.2,%20-71.0%2047.2,%20-73.5%2046.1,%20-73.5%2045.7,%20-73.6%2045.7,%20-73.6%2045.5))&containerType=1&productVersion=2.0.0

Le champ ‘responseText’ de la réponse contient des informations validant le nombre de résultats retournés et les paramètres de la requête. Ce champ est à la fin de la réponse:

```
"responseText":"Get Request of NWs returned 1 results with the following
filters: dataReference=null, containerType=S100_ExchangeSet , validFrom=null,
validTo=null, dataProductType=null, productVersion=2.0.0, geometry=POLYGON((-
73.6 45.5, -73.5 45.5, -73.2 45.8, -73.2 46.0, -72.6 46.0, -69.8 47.2, -71.0
47.2, -73.5 46.1, -73.5 45.7, -73.6 45.7, -73.6 45.5)), page=null,
pageSize=null, max pageSize is 2000"
```

2.2 PARAMÈTRES DE LA REQUÊTE DÉTAILLÉE GET

Le URL pour la requête détaillée GET est https://s124.ccg-gcc.gc.ca/api/secom/v1/object et les paramètres suivants peuvent être utilisés. Les paramètres sont séparés avec le caractère '&'.

2.2.1 productVersion

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?productVersion=2.0.0

Le paramètre **productVersion** définit la version S-124 des AVNAVs retournés.
La version S-124 par défaut est 2.0.0. Les versions S-124 supportés sont 1.0.0, 1.5.0 et 2.0.0.

2.2.2 containerType

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?containerType=1

Le paramètre **containerType** peut être utillisé si vous voulez les résultats sous forme d'AVNAVs individuels ou regroupés dans un Exhange set S-100 (dans un fichier.zip).
Quand **containerType**=1, 1 seul champ 'data' est retourné et il contient le contenu (encodé en Base64) du fichier .zip file contenant le exchange set S-100.
Quand **containerType**=0 (valeur par défaut), plusieurs champs 'data' sont retournés, un champ 'data' pour chaque AVNAV S-124.

2.2.3 dataReference

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?dataReference=00000000-0000-0000-0000-000000171894

Le paramètre **dataReference** peut être utilisé quand le ID d'un AVNAV spécifique est connu. Le id dataReference de l'AVNAV est donné sous la forme XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX.

La requête résumée GET (voir section 2.4) retourne la valeur du champ 'dataReference' associé à chaque AVNAV dans la réponse.

2.2.4 geometry

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?geometry=POLYGON((-73.6%2045.5,%20-73.5%2045.5,%20-73.2%2045.8,%20-73.2%2046.0,%20-72.6%2046.0,%20-69.8%2047.2,%20-71.0%2047.2,%20-73.5%2046.1,%20-73.5%2045.7,%20-73.6%2045.7,%20-73.6%2045.5))

Le paramètre **geometry** peut être utilisé si souhaitez avoir les résultats appartenant à une zone spécifique définie par un point, polyline, ou un polygone. Vous devez utiliser une géométrie définie en langage WKT. (https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry). Nous recommandons d'utiliser une géométrie de type POLYGON, LINESTRING ou POINT.

N.B: Le polygone de l'exemple ci-dessus est celui de la zone canadienne des essais S-100.

2.2.5 validFrom

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?validFrom=20250101T000000

Le paramètre **validFrom** peut être utilisé si vous voulez obtenir les AVNAVs S-124 publiés après le moment défini par validFrom. Quand le paramètre **validFrom** est utilisé, les résultats incluent les AVNAVs ('auto-annulés') annulateurs S-124. Aussi, quand le paramètre **validFrom** est utilisé, les résultats sont retournés comme une page parmi plusieurs pages.

2.2.6 validTo

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?validTo=20250101T000000

Le paramètre **validTo** peut être utilisé si vous voulez obtenir les AVNAVs S-124 publiés avant le moment défini par le paramètre validTo. Quand le paramètre **validTo** est utilisé, , les résultats incluent les AVNAVs 'auto-annulés' annulateurs S-124. Aussi, quand le paramètre **validTo** est utilisé, les résultats sont retournés comme une page parmi plusieurs pages.

2.2.7 page

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?pageSize=10&page=1

Le paramètre **page** sert à avoir les résultats comme une page parmi plusieurs pages.
Pour voir combien de pages sont retournés, le nombre de pages est affiché dans le champ

pagination à la fin des résultats retournés (voir figure 2.2).
La **page** par défaut est 1 lorsque seulement le paramètre **pageSize** est donné.

pbWU+CiAgICAgICAgICAgIDxTMTI0OnRoZVJlZmVyZW5jZXMgeGxpbms6aHJlZj0iI0NDRy5TMTI0LkNBMDEuTi4wNDY0LjI1LjEiLz4KICAgICAgICA8L1MxMjQ6TkFWV0FSTlByZWFtYmxlPgogICAgICAgIDxTMTI0Ok5BVldBUk5
QYXJ0IGdtbDppZD0iQ0NHLlMxMjQuQ0EwMS5OLjA0NjQuMjUuMSI+CiAgICAgICAgICAgIDxTMTI0OmZpeGVkRGF0ZVJhbmdlPgogICAgICAgICAgICAgICAgPFMxMjQ6ZGF0ZVN0YXJ0PgogICAgICAgICAgICAgICAgIDxTMTA
wOmRhdGU+MjAyNS0wNy0wMzwvUzEwMDpkYXRlPgogICAgICAgICAgICAgICAgPC9TMTI0OmRhdGVTdGFydD4KICAgICAgICAgICAgPC9TMTI0OmZpeGVkRGF0ZVJhbmdlPgogICAgICAgICA8UzEyNDp3YXJuaW5nSW5mb3JtYXR
pb24+CiAgICAgICAgICAgICAgICA8UzEyNDppbmZvcm1hdGlvbj4KICAgICAgICAgICAgICAgICAgICA8UzEyNDpsYW5ndWFnZT5lbmc8L1MxMjQ6bGFuZ3VhZ2U+CiAgICAgICAgICAgICAgICAgPFMxMjQ6dGV4dD5BIHN1YnN
1cmZhY2UgbW9vcmluZyBoYXMgYmVlbiBkZXBsb3llZCBpbiB0aGUgdmljaW5pdHkgb2YgNDggNDUuNzAxTiAwNTMgMzEuOTU4VyB3aXRoIGEgdG9wIGZsb2F0YXRpb24gYXQgMTA1IG1ldGVycyBiZWxvdyB0aGUgc3VyZmFjZS7CoAp
DYXV0aW9uLjwvUzEyNDp0ZXh0PgogICAgICAgICAgICAgICAgPC9TMTI0OmluZm9ybWF0aW9uPgogICAgICAgICAgICAgICAgPFMxMjQ6bmF2d2FyblR5cGVEZXRhaWxzIGNvZGU9Im90aGVyIj5vdGhlcjogT2JzdHJ1Y3Rpb24gKE9
0aGVyKTwvUzEyNDpuYXZ3YXJuVHlwZURldGFpbHM+CiAgICAgICAgICAgIDwvUzEyNDp3YXJuaW5nSW5mb3JtYXRpb24+CiAgICAgICAgICAgIDxTMTI0OmhlYWRlciB4bGluazpocmVmPSIjQ0NHLlMxMjQuQ0EwMS5OLjA0NjQuMjU
uMCIvPgogICAgICAgICAgICA8UzEyNDpnZW9tZXRyeT4KICAgICAgICAgICAgICAgIDxTMTAwOnBvaW50UHJvcGVydHk+CiAgICAgICAgICAgICAgICAgPFMxMDA6UG9pbnQgZ21sOmlkPSJDQ0cuUzEyNC5DQTAxLk4uMDQ2NC4
yNS5sb2NhdGlvbi4xIiBzcnNOYW1lPSJodHRwczovL3d3dy5vcGVuZ2lzLm5ldC9kZWYvY3JzL0VQU0cvMC80MzI2IiBzcnNEaW1lbnNpb249IjIiPgogICAgICAgICAgICAgICAgICAgICA8Z21sOnBvcyBzcnNOYW1lPSJodHR
wczovL3d3dy5vcGVuZ2lzLm5ldC9kZWYvY3JzL0VQU0cvMC80MzI2IiBzcnNEaW1lbnNpb249IjIiPjQ4Ljc2MTY4MyAtNTMuNTMyNjMzPC9nbWw6cG9zPgogICAgICAgICAgICAgICAgIDwvUzEwMDpQb2ludD4KICAgICAgICA
gICAgICAgIDwvUzEwMDpwb2ludFByb3BlcnR5PgogICAgICAgICAgICA8L1MxMjQ6Z2VvbWV0cnk+CiAgICAgICAgPC9TMTI0Ok5BVldBUk5QYXJ0PgogICAgPC9TMTI0Om1lbWJlcnM+CjwvUzEyNDpEYXRhc2V0Pgo=","exchange
Metadata":{"dataProtection":false,"protectionScheme":"SECOM","digitalSignatureReference":"ecdsa-384-sha2","digitalSignatureValue":
{"publicRootCertificateThumbprint":"6aef15640d55dd694c1f6a121ad17ca6ef7d258b","publicCertificate":"MIID+jCCA4CgAwIBAgIUdA0+bDm2bJ/5p5yWCeFBeSQ2KUUwCgYIKoZIzj0EAwMwgcwxLDAqBgoJk
iaJk/IsZAEBDBx1cm46bXJuOm1jcDpjYTptY2M6bWNwLWlkcmVnMQswCQYDVQQGEwJESzEQMA4GA1UECAwHRGVubWFyazETMBEGA1UEBwwKQ29wZW5oYWdlbjEMMAoGA1UECgwDTUNQMQwwCgYDVQQLDANNQ1AxHjAcBgNVBAMMFU1DU
CBJZGVudGl0eSBSZWdpc3RyeTEsMCoGCSqGSIb3DQEJARYdaW5mb0BtYXJpdGltZWNvbm5lY3Rpdml0eS5uZXQwHhcNMjQwNTAyMTk0MDI1WhcNMjYwNTAyMTk0MDI1WjCBwTELMAkGA1UEBhMCQ0ExIDAeBgNVBAoMF3Vybjptcm46b
WNwOm9yZzptY2M6Y2NnMRUwEwYDVQQLDAxvcmdhbml6YXRpb24xHTAbBgNVBAMMFENhbmFkaWFuIENvYXN0IEd1YXJkMScwJQYKCZImiZPyLGQBAQwXdXJuOm1ybjptY3A6b3JnOm1jYzpjY2cxMTAvBgkqhkiG9w0BCQEWIm1hcnRpb
i5yaWNoYXJkLWNlcmRhQGRmby1tcG8uZ2MuY2EwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQ3JRW4+BS2Eug6tjwFxIFxWsrJCZJGWlDm+co3doWrxYEpD1wRVYu1d4Dv2+TpDnfVzG11ZxpLbx0MW2x3s535AlpR5LkY/CsTgYJI171df
zkfhjv5FerOsJZpzWPqfT+jggEqMIIBJjAfBgNVHSMEGDAWgBQfDlviBmA3By9QeVOYTVuuXpDQGjAdBgNVHQ4EFgQULoVhajSK1AX5qLwFP2kudp9jrCYwawYDVR0fBGQwYjBgoF6gXIZaaHR0cDovL2FwaS5tYXJpdGltZWNvbm5lY
3Rpdml0eS5uZXQveDUwOS9hcGkvY2VydGlmaWNhdGVzL2NybC91cm46bXJuOm1jcDpjYTptY2M6bWNwLWlkcmVnMHcGCCsGAQUFBwEBBGswaTBnBggrBgEFBQcwAYZbaHR0cDovL2FwaS5tYXJpdGltZWNvbm5lY3Rpdml0eS5uZXQve
DUwOS9hcGkvY2VydGlmaWNhdGVzL29jc3AvdXJuOm1ybjptY3A6Y2E6bWNjOm1jcC1pZHJlZzAKBggqhkjOPQQDAwNoADBlAjEA20SX2AC6nnybkOOUgniqutQPm23aBUjpZoJU5wsYM2kutrrN+q3Ujq/kNYP63oXBAjBfa7Rhnkg2i
9LLsVeJ7vB9oM+XdoLSGSbpizgcqHdKtR+vUyQrW0RAESvP/hBw0vU=","digitalSignature":"306402303557780C131B5D04E82E84B2662859309FCA48873061F7D6052361855964F47FCE6E0ABC4727CEF64FE12FD5769
3EB100230434F4192DB6A1EEA76289F4EEC1E57A1764638614496350968229AFFA5B96276E78E52AD576AD6CB6633EE8AAAC8424"},"compressionFlag":false},"ackRequest":0}],"pagination":
{"totalItems":1351,"maxItemsPerPage":10},"responseText":"Get Request of NWs returned 10 results with the following filters: dataReference=null, containerType=null",
validFrom=null, validTo=null, dataProductType=null, productVersion=null, geometry=null, page=1, pageSize=10, max pageSize is 1000"}

Figure 2.2

2.2.8 pageSize

https://s124.ccg-gcc.gc.ca/api/secom/v1/object?pageSize=10&page=1

Le paramètre **pageSize** peut être utilisé pour obtenir les résultats comme une page parmi plusieurs pages.

La valeur par défaut du paramètre **pageSize** est 2000 lorsque seulement le paramètre **page** est donné.

2.3 COMMENT CONVERTIR LA RÉPONSE DE LA REQUÊTE DÉTAILLÉ GET

Comme expliqué dans la section 2.1, ***<u>le contenu du champ 'data' est encodé en Base64</u>***.

Quand **containerType**=1 (S-100 Exchange set), vous devez suivre les prochaines étapes pour décoder le résultat en un texte humainement lisible:

1) Copier le contenu (encodé en Base64) du champ 'data'. Ne pas inclure les guillemets "".

Figure 2.3

2) Coller le texte encodé en Base64 dans la zone de texte du site web (ou tout autre outil pouvant convertir du contenu Base64 en fichier). Par exemple, aller sur https://base64.guru/converter/decode/file et cliquer sur le bouton "Decode Base64 to File" button. Cliquer sur le lien application.zip pour télécharger le fichier .zip contenant le S-100 Exchange set. Voir figure 2.4.

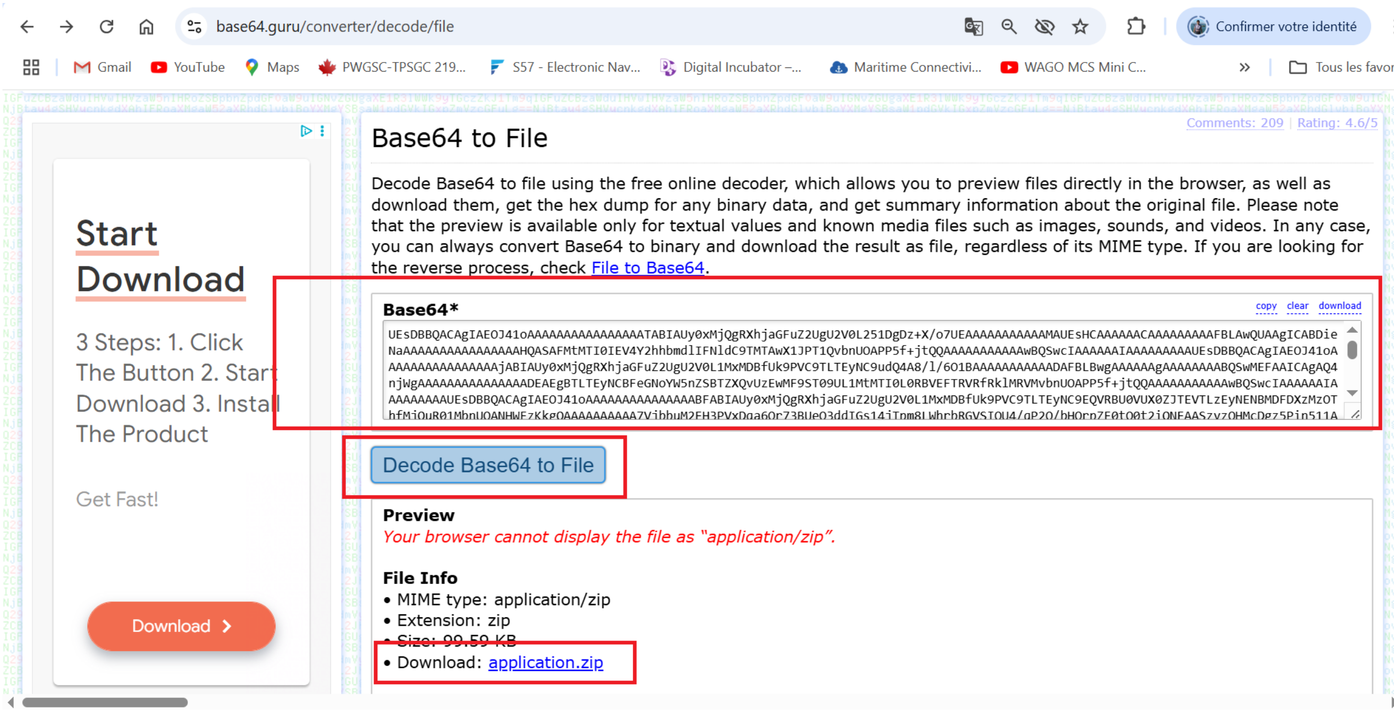

Figure 2.4

Si le paramètre **containerType** n’est pas utilisé, ou est mis à 0, la réponse sera une liste d’objets contenant chacun un AVNAV S-124.
Pour convertir les AVNAVs, suivre les étapes suivantes:

1) Pour chaque champ ‘data’, copier le contenu encodé en Base64. Voir figure 2.5.

Figure 2.5

2) Coller le contenu encodé en Base64 dans le zone de texte du site web (ou un outil pouvant convertir du Base64 en texte) https://www.base64decode.org/ et cliquer sur le bouton “Decode”.
Le résultat converti sera affiché dans la zone de texte du bas. Voir figure 2.6.

Figure 2.6

2.4 REQUÊTE RÉSUMÉE GET

La requête résumée GET retourne la liste des AVNAVs S-124 actifs. Le URL pour la requête résumée GET est https://s124.ccg-gcc.gc.ca/api/secom/v1/object/summary .

Les champs retournés par la requête résumée GET sont décrites ci-dessous :

Figure 2.7

2.4.1 dataReference

Le valeur du champ **dataReference** peut être utilisée avec le paramètre **dataReference** de la requête détaillée GET pour obtenir un AVNAV spécifique.

2.4.2 info_identifier

Le champ **info_identifier** est associé avec le id de l'AVNAV S-124.

2.4.3 Info_name

Le champ **info_name** est associé avec le titre de l'AVNAV S-124.

2.4.4 info_status

Le champ **info_status** est associé avec le statut de l'AVNAV S-124.

2.4.5 info_description

Le champ **info_description** est associé avec la description de l'AVNAV S-124.

2.4.6 info_productVersion

Le champ **info_productVersion** est associé avec la version du standard S-124.

Section 3 Validation de la signature

3.1 VALIDATION DES SIGNATURES DU S-100 EXCHANGE SET

Pour valider la signature associée à un fichier S-124 .gml contenu dans un S-100 exchange set, il faut suivre les étapes suivantes (Les étapes 2 et 3 sont pour les utilisateurs avancés, il est recommandé d'aller directement à l'étape 4 si vous n'êtes pas un développeur ou un utilisateur avancé:

1) Extraire le certificat du fichier catalog.xml file. Copier la valeur du certificat publique contenu dans la balise "<certificate>" (voir figure 3.1).

Figure 3.1

2) (Optionnel) Décoder la valeur du certificat publique qui est encodé en Base64. Les 2 options suivantes peuvent être suivies:

   a) Site web https://www.base64decode.org/ : Mettre la valeur du certificat publique de l'étape 1) dans la zone de texte du haut et appuyer sur le bouton « Decode ». Voir figure 3.2.

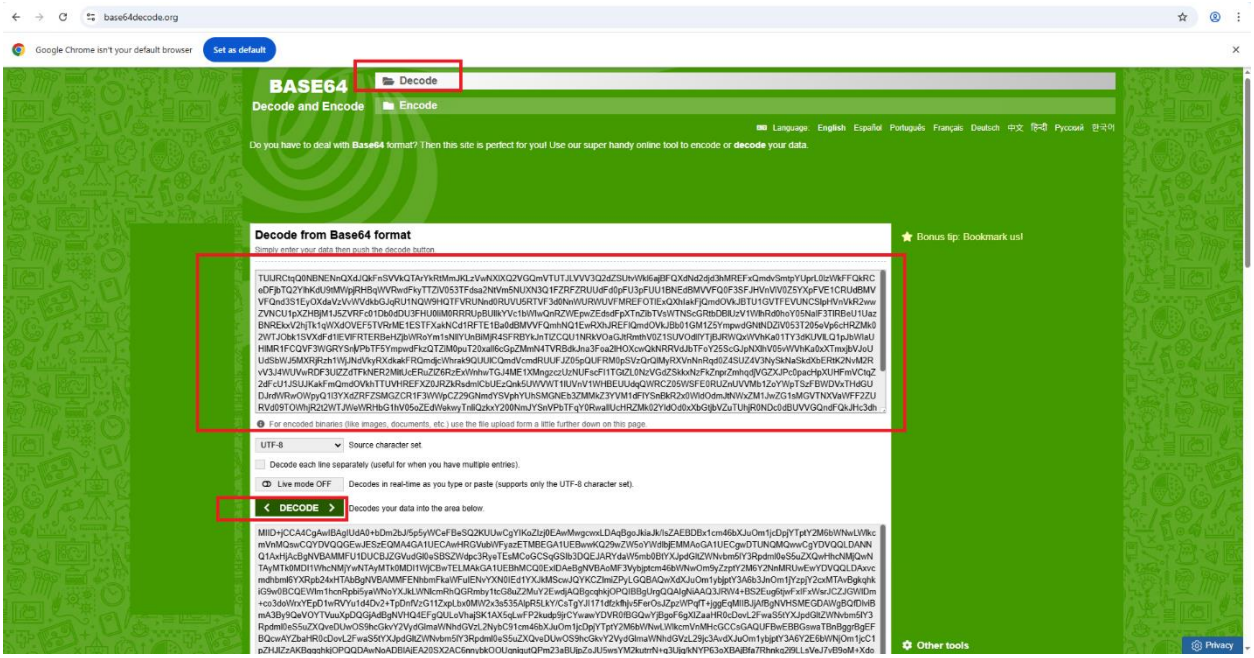

Figure 3.2

b) Utiliser un outil pour décoder du Base64 comme un langage de programmation (Base64.getDecoder() en Java par exemple):

```
String decodedCertificateValue = new String(Base64.getDecoder().decode(certificateValue));
```

Figure 3.3

3) (Optionnel) Avec la valeur décodée du certificate, extraire la clé publique du certificat. Les options suivantes peuvent être utilisées:

a) Mettre la valeur du certificat dans un fichier nommé ccg_certificate.pem et entouré le contenu par -----BEGIN CERTIFICATE----- et -----END CERTIFICATE----- (voir figure 3.4).

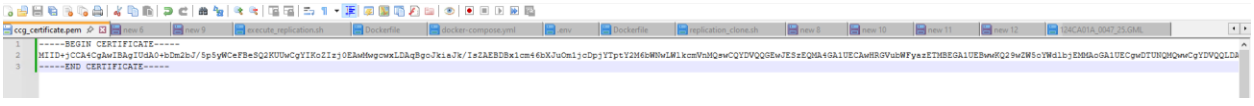

Figure 3.4

Exécuter la commande openssl dans un terminal WSL (Environnement de Linux dans Windows) :

openssl x509 -pubkey -noout -in ccg_certificate.pem > ec-secp384-r1-pub-key-extracted.pem (voir figure 3.5)

La commande crée le fichier ec-secp384-r1-pub-key-extracted.pem contenant la clé publique (voir figure 3.6).

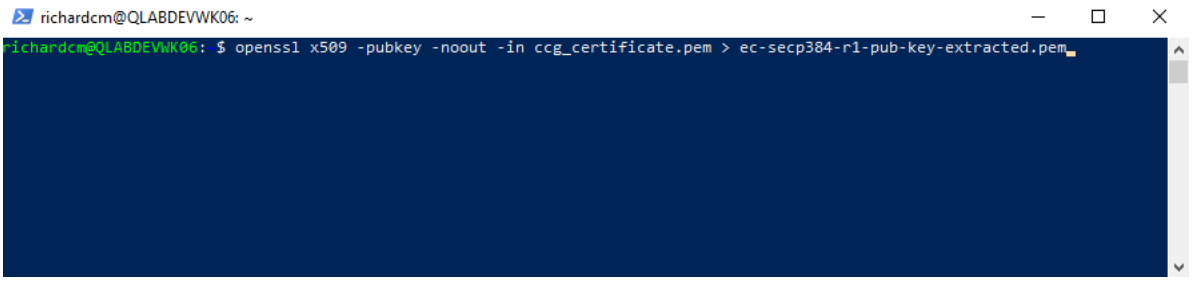

Figure 3.5

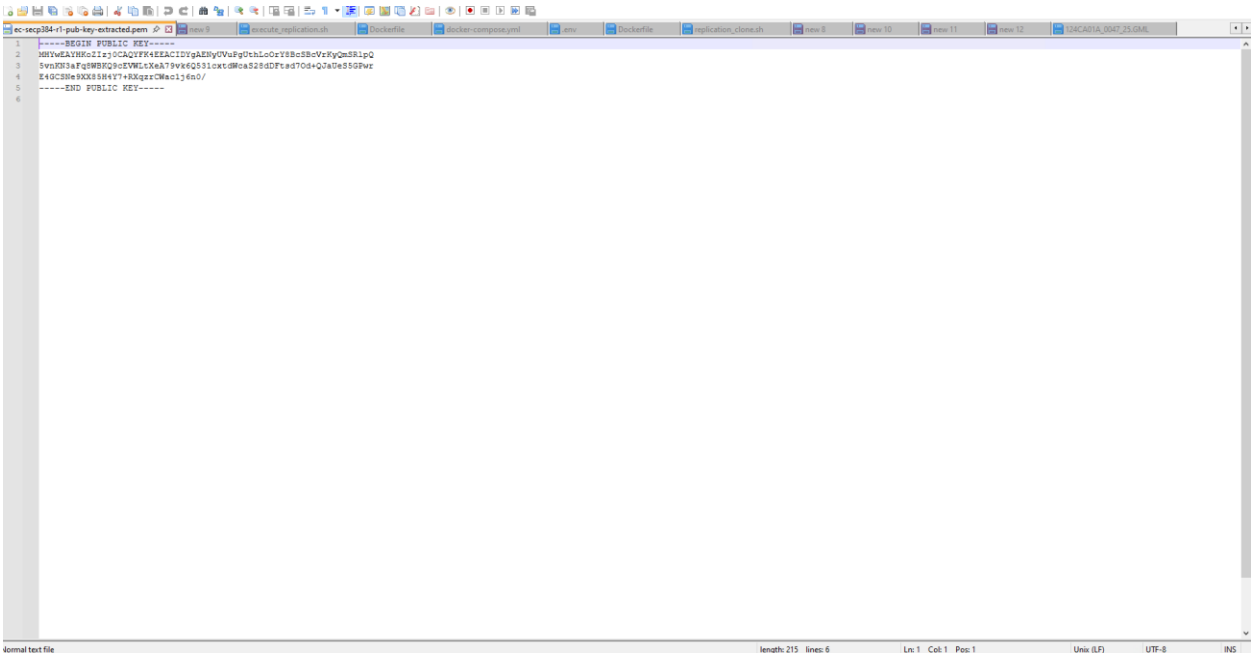

Figure 3.6

b) Utiliser un langage de programmation (comme Java avec la librairie java.security) pour charger le certificat décodé et extraire la clé publique.

```
X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(decodedCertificateValue.getBytes());
PublicKey publicKey = keyFactory.generatePublic(publicSpec);
```

Figure 3.7

4) Valider la signature avec la clé publique et la valeur de la signature. Pour valider la signature, utiliser l'outil en ligne (option a) ou un langage de programmation comme Java (option b).

   a) Si vous avez sauté l'étape 2 et 3, copier la clé publique extraite du certificat public de la GCC (le texte qui suit).

   -----BEGIN PUBLIC KEY-----

   MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAENyUVuPgUthLoOrY8BcSBcVrKyQmSRlpQ

   5vnKN3aFq8WBKQ9cEVWLtXeA79vk6Q531cxtdWcaS28dDFtsd7Od+QJaUeS5GPwr

   E4GCSNe9XX85H4Y7+RXqzrCWac1j6n0/

   -----END PUBLIC KEY-----

   Aller sur le site web https://emn178.github.io/online-tools/ecdsa/verify/ et sélectionner ECDSA->Verify Signature
   i) Sélectionner UTF-8 pour le 'Input Encoding' (voir figure 3.8);

   ii) Ajouter le contenu du fichier pour lequel vous voulez valider la signature dans la zone de texte 'Input' (voir figure 3.8). Dans cet exemple, le contenu du fichier 124CA01C_2446_25.GML est validé. (voir figure 3.9);

   iii) Sélectionner SECG secp384r1 / X9.63 ansip384r1 / NIST P-384 pour l'option 'Curve' (voir figure 3.8);

   iv) Sélectionner SHA384 pour le champ 'Signature Algorithm' (voir figure 3.8);

   v) Sélectionner 'Pem Text' pour l'option 'Public Key - Type' (voir figure 3.10);

   vi) Ajouter le contenu de la clé publique (surrounded by -----BEGIN PUBLIC KEY-----

   and -----END PUBLIC KEY-----) dans le champ 'Public Key – Data' (voir figure 3.10);

   vii) Sélectionner Base64 pour le champ 'Signature – Type' (voir figure 3.11);

   viii) Ajouter le contenu de la balise "<S100_SE_DigitalSignature>" associé avec la signature du fichier à valider (124CA01C_2446_25.GML dans cet exemple, voir figure 3.12) dans le champ 'Signature – Data' (voir figure 3.11);

   ix) Cliquer sur le bouton 'verify' (voir figure 3.8).

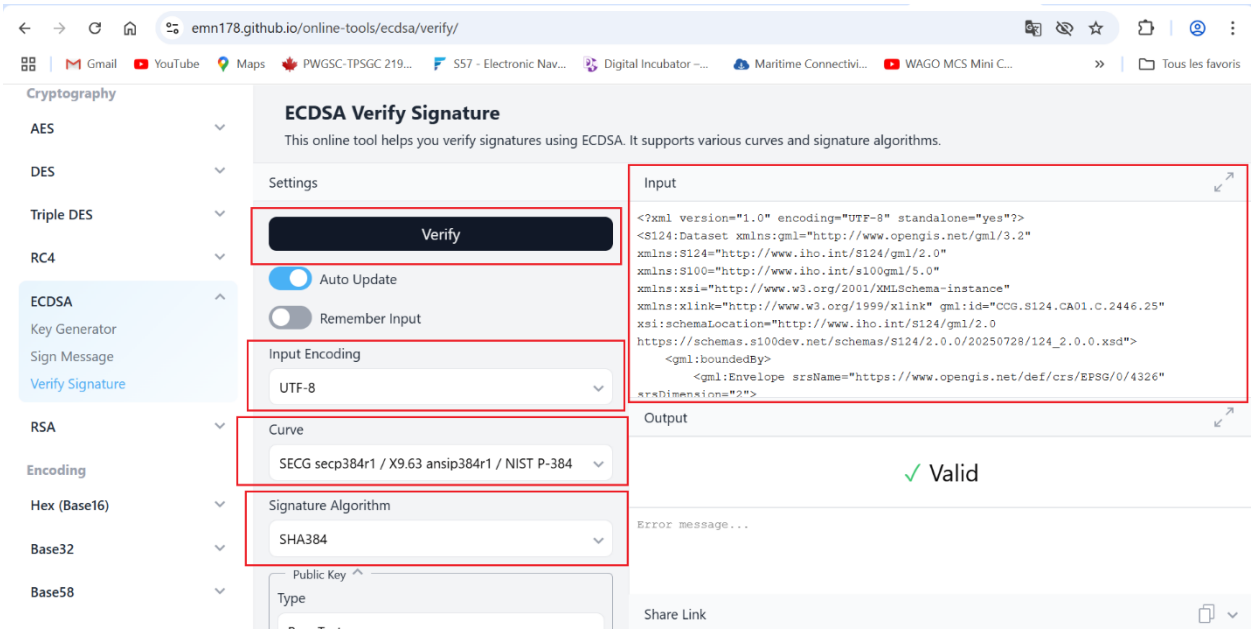

Figure 3.8

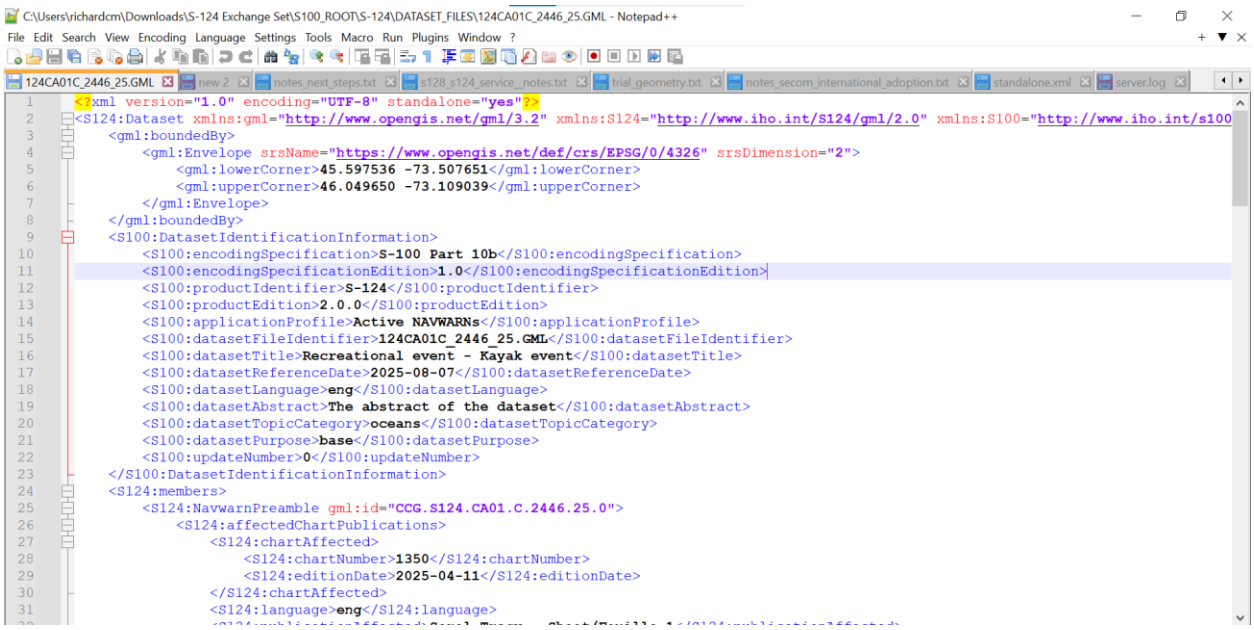

Figure 3.9

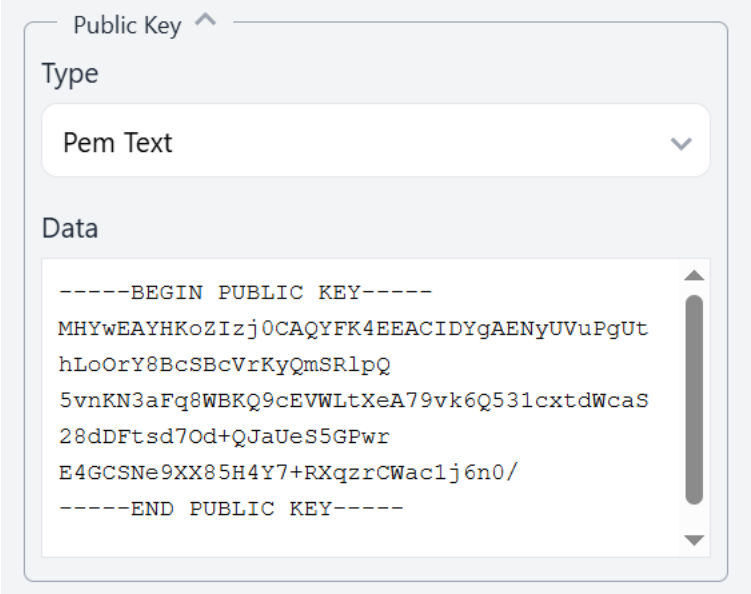

Figure 3.10

Figure 3.11

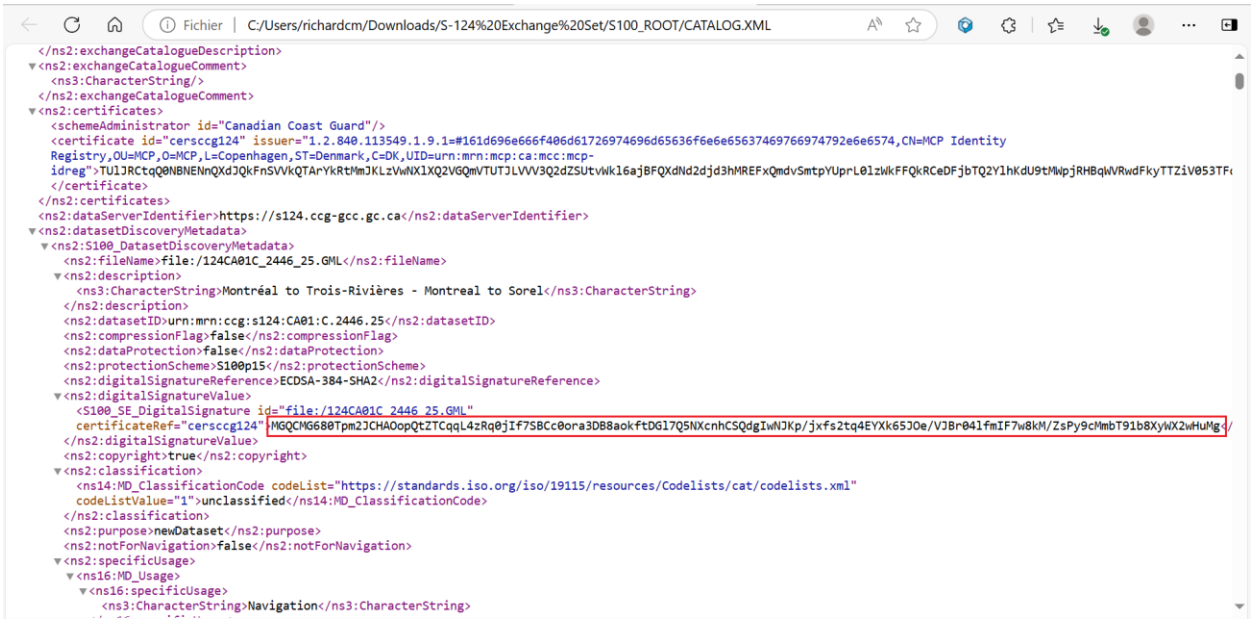

Figure 3.12

b) En Java ,

i) Assigner la valeur de la signature (figure 3.12) à la variable (String) signature;

ii) Assigner la valeur du certificat encodé (figure 3.1) à la variable (String) encodedPublicCertificate;

iii) Assigner le contenu texte du fichier (figure 3.9) associé avec la signature à la variable (String) content;

iv) Exécuter les commandes suivantes (utiliser l'algorithme 'SHA384WITHECDSA'):

```
byte[] decodedPublicCertificate = Base64.getDecoder().decode(encodedPublicCertificate);
byte[] publicCertificateValue = Base64.getDecoder().decode(decodedPublicCertificate);
X509Certificate certificate = (X509Certificate) CertificateFactory.getInstance("X.509").generateCertificate(new
ByteArrayInputStream(publicCertificateValue));
PublicKey publicKey = certificate.getPublicKey();

Signature ecdsaSignature = Signature.getInstance("SHA384withECDSA");
ecdsaSignature.initVerify(publicKey);
ecdsaSignature.update(content.getBytes());
boolean isValid = ecdsaSignature.verify(Base64.getDecoder().decode(signature.getBytes()));
```

3.2 VALIDATION DES SIGNATURES DE LA RÉPONSE DES REQUÊTES DÉTAILLÉES GET (LE CHAMP EXCHANGEMETADATA)

Valider la signature (voir figure 3.13) du champ exchangeMetadata associé avec le contenu du champ ‘data’ (voir figure 3.14), suivre les étapes suivantes (l’étape 2 est pour les utilisateurs intermediate/advanced users, nous recommandons d’aller directement à l’étape 3) si vous n’êtes pas un développeur ou un utilisateur avancé):

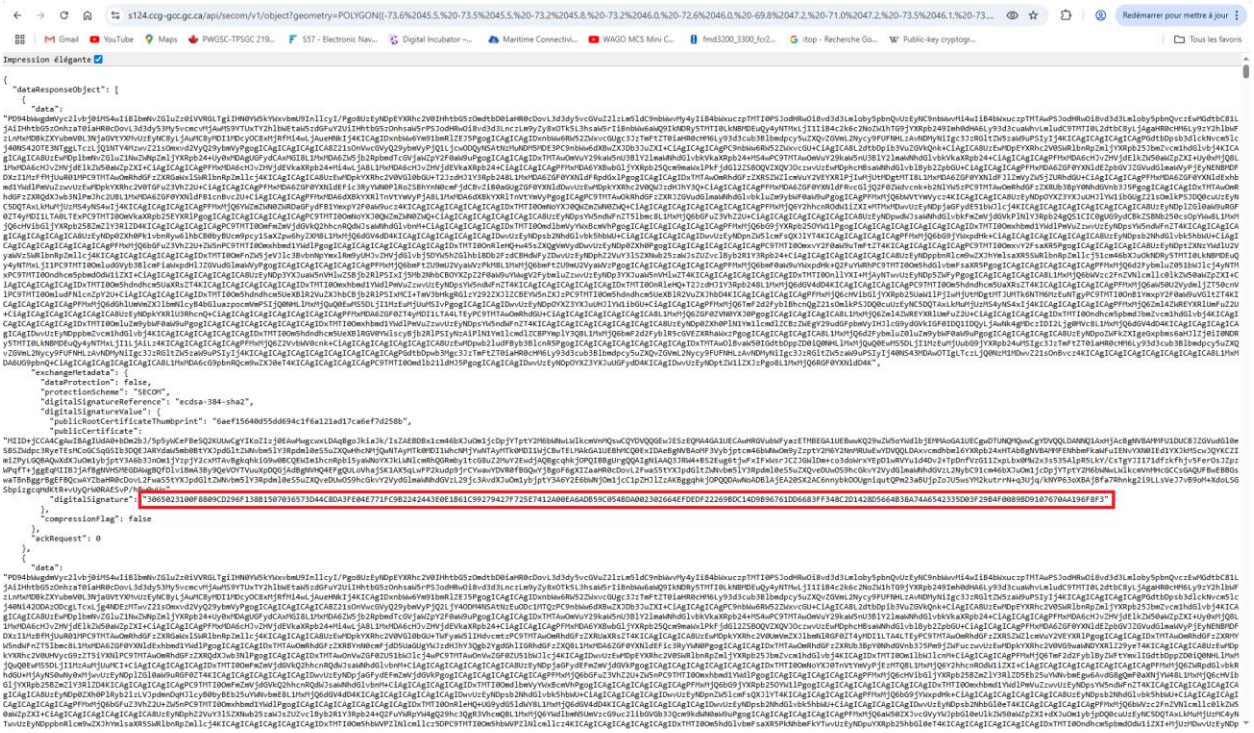

Figure 3.13

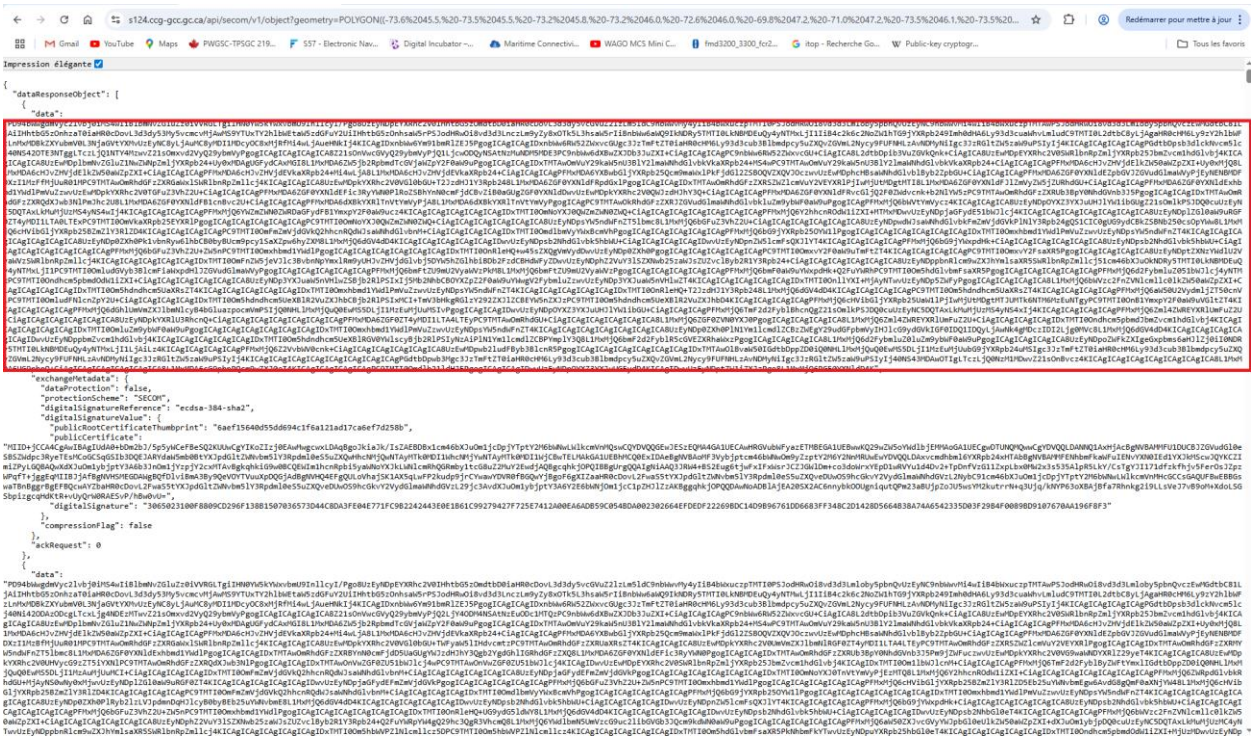

Figure 3.14

1) Extraire la valeur du champ exchangeMetaData->digitalSignatureValue->publicCertificate de la réponse de la requête GET détaillée (voir figure 3.15).

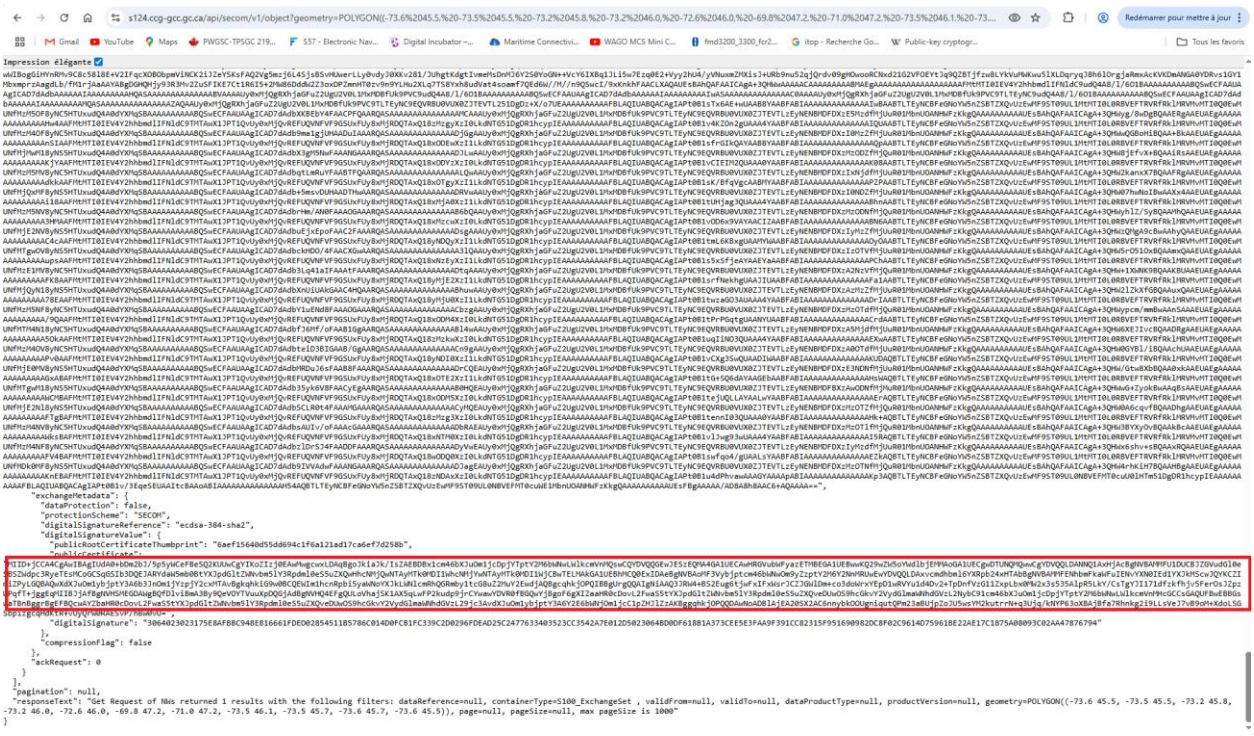

Figure 3.15

La valeur du certificat est déjà décodée, alors l'étape 2 de la section précédente n'est pas requise.

2) (Optionnel) Avec la valeur du certificat de la réponse de la requête, extraire la clé publique du certificat. Les 2 options suivantes peuvent être utilisées:

   a) Mettre le certificat décodé dans un fichier nommé ccg_certificate.pem et l'entourrer par -----BEGIN CERTIFICATE----- et -----END CERTIFICATE----- (voir figure 3.16)

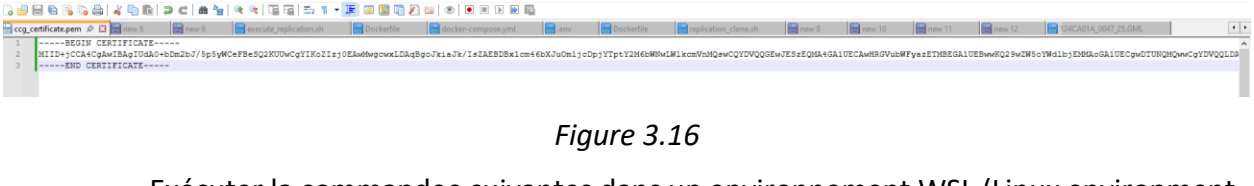

Figure 3.16

Exécuter la commandee suivantes dans un environnement WSL (Linux environment in Windows) ou dans un terminal Linux :

openssl x509 -pubkey -noout -in ccg_certificate.pem > ec-secp384-r1-pub-key-extracted.pem (see figure 3.17).

Cette commande créera le fichier ec-secp384-r1-pub-key-extracted.pem contenant la clé publique (voir figure 3.18).

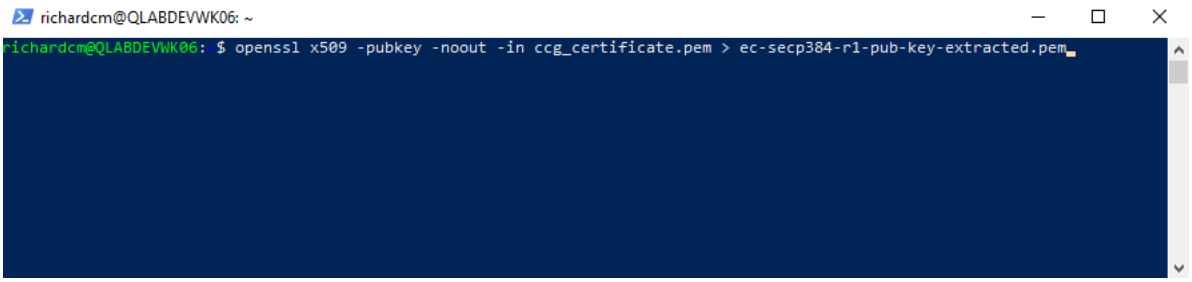

Figure 3.17

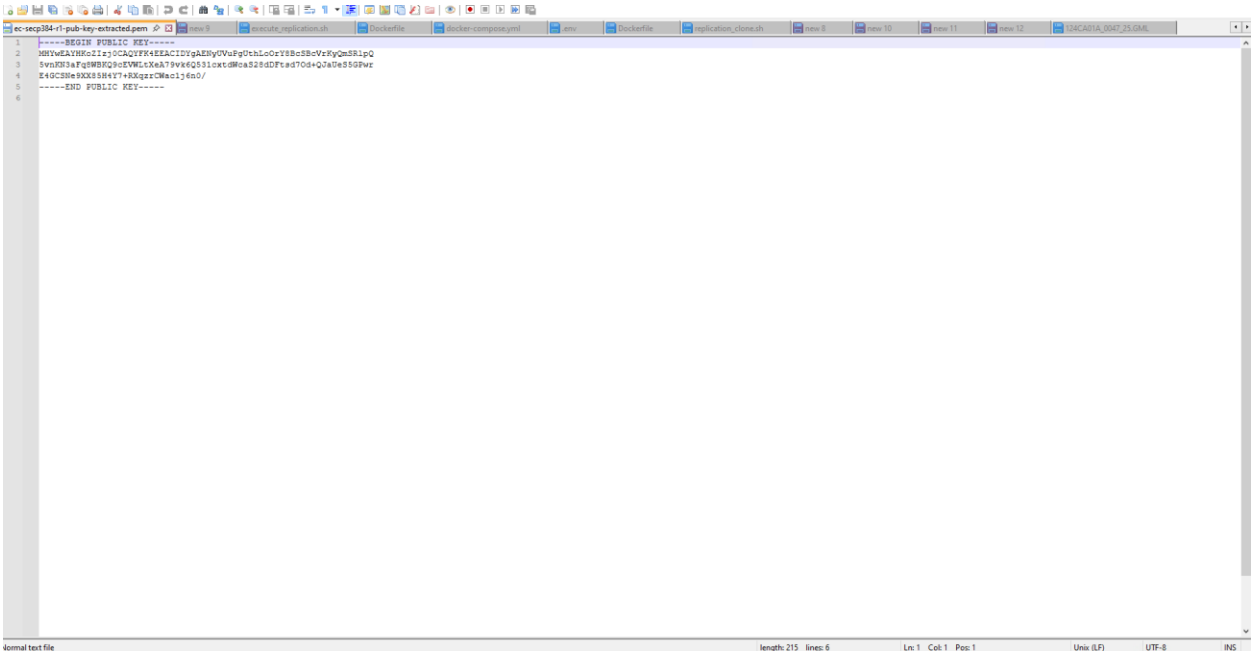

Figure 3.18

b) Utiliser un langage de programmation comme Java avec la librairie java.security pour charger le certificat décodé et extraire la clé publique (voir figure 3.19).

```
X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(decodedCertificateValue.getBytes());
PublicKey publicKey = keyFactory.generatePublic(publicSpec);
```

Figure 3.19

3) Valider la signature avec la clé publique et la valeur de la signature. Pour valider la signature, utiliser un outil en ligne (option a) or un langage de programmation comme Java (option b).

a) Si vous avez sauté l'étape 2, copier la clé publique extraite du certificat publique de la GCC (Le contenu textuel suivant) :

-----BEGIN PUBLIC KEY-----

MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAENyUVuPgUthLoOrY8BcSBcVrKyQmSRlpQ

5vnKN3aFq8WBKQ9cEVWLtXeA79vk6Q531cxtdWcaS28dDFtsd7Od+QJaUeS5GPwr

E4GCSNe9XX85H4Y7+RXqzrCWac1j6n0/

-----END PUBLIC KEY-----

Aller sur le site web https://emn178.github.io/online-tools/ecdsa/verify/ et sélectionner ECDSA->Verify Signature
i) Sélectionner Base64 pour le champ 'Input Encoding' (voir figure 3.20);

ii) Copier le contenu du champ 'data' (voir figure 3.21) associé à la signature à valider et l'ajouter dans la zone de texte 'Input' (voir figure 3.20);

iii) Sélectionner SECG secp384r1 / X9.63 ansip384r1 / NIST P-384 pour le champ 'Curve' (voir figure 3.20);

iv) Sélectionner SHA384 pour le champ 'Signature Algorithm' (voir figure 3.20);

v) Sélectionner Pem Text pour le champ 'Public Key – Type' (voir figure 3.22);

vi) Ajouter la clé publique (entouré par -----BEGIN PUBLIC KEY-----

et -----END PUBLIC KEY-----) dans la zone de texte 'Public Key - Data' (voir figure 3.22);

vii) Sélectionner HEX pour le champ 'Signature – Type' (voir figure 3.23). Le format de la signature dans la réponse de la requête est encodé en HEX au lieu de Base64;

viii) Copier le contenu du champ exchangeMetadata->digitalSignatureValue->digitalSignature associé au champ 'data' qui est validé (dans cet exemple, voir figure 3.24) et l'ajouter dans la zone de texte 'Signature - Data' (voir figure 3.23);

ix) Cliquer sur le bouton verify (voir figure 3.20).

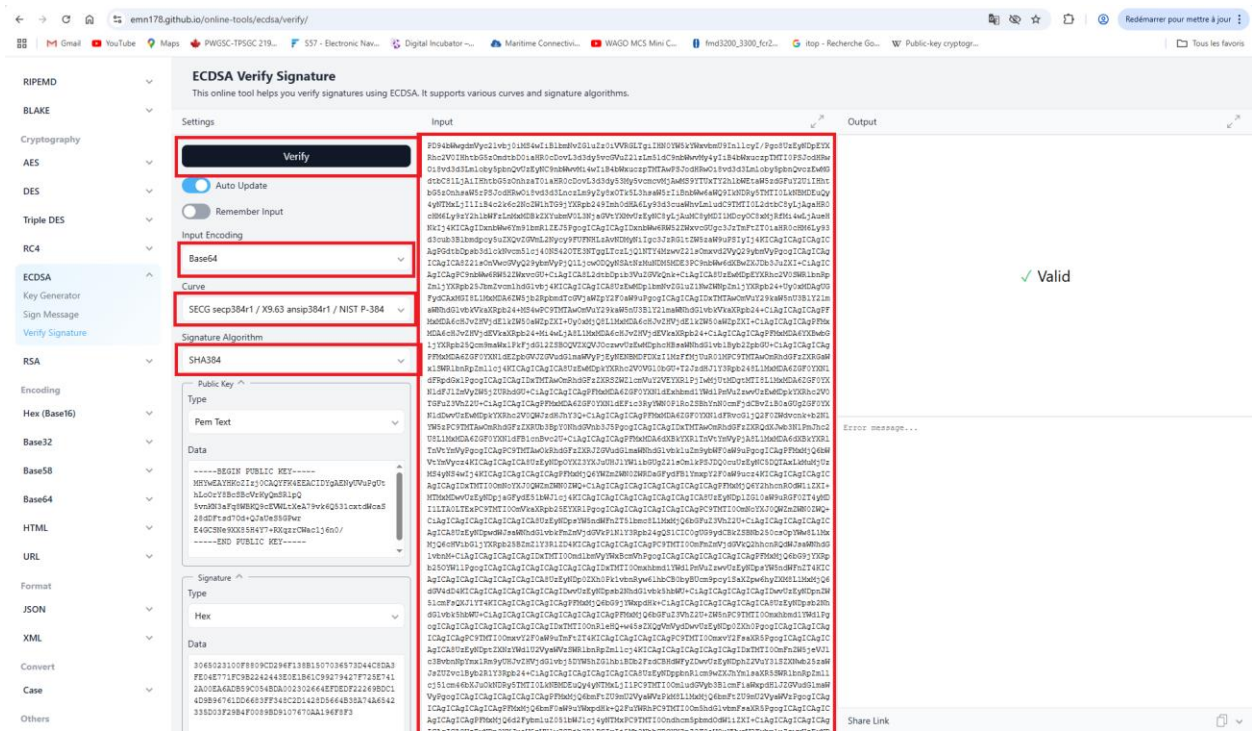

Figure 3.20

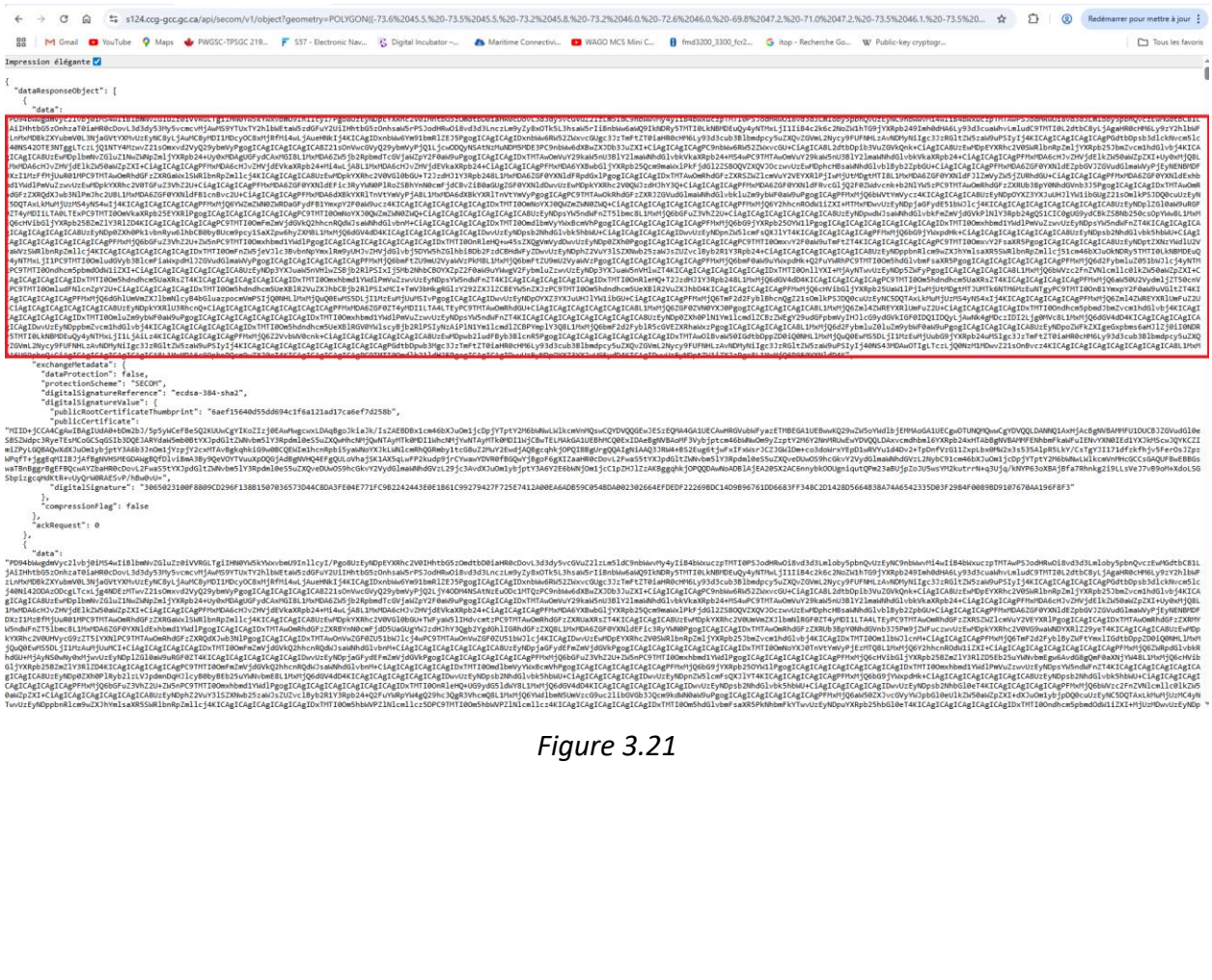

Figure 3.21

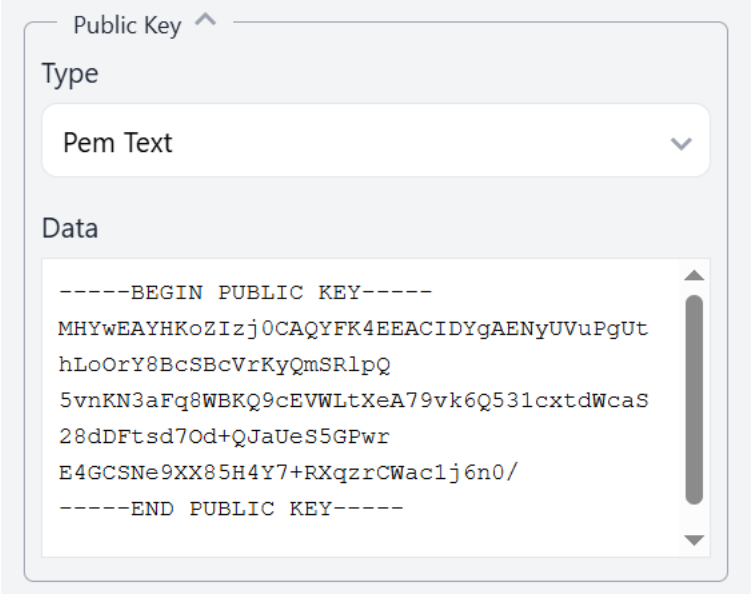

Figure 3.22

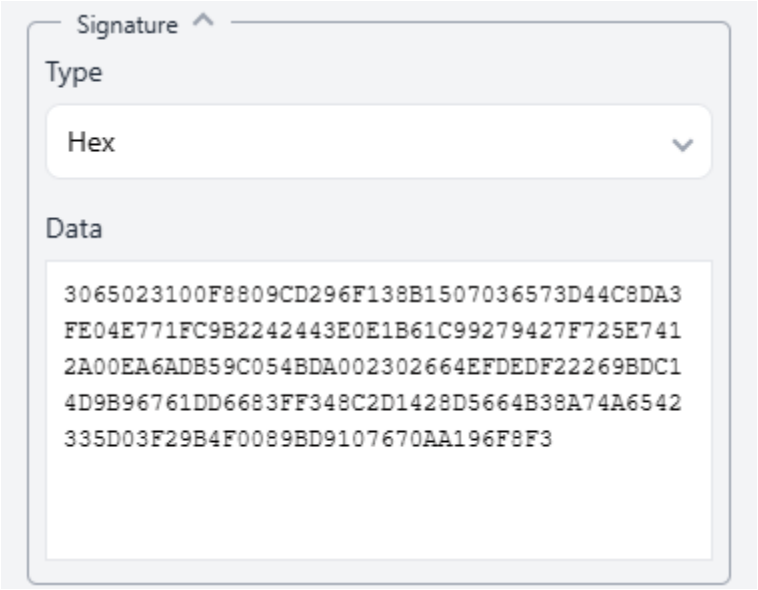

Figure 3.23

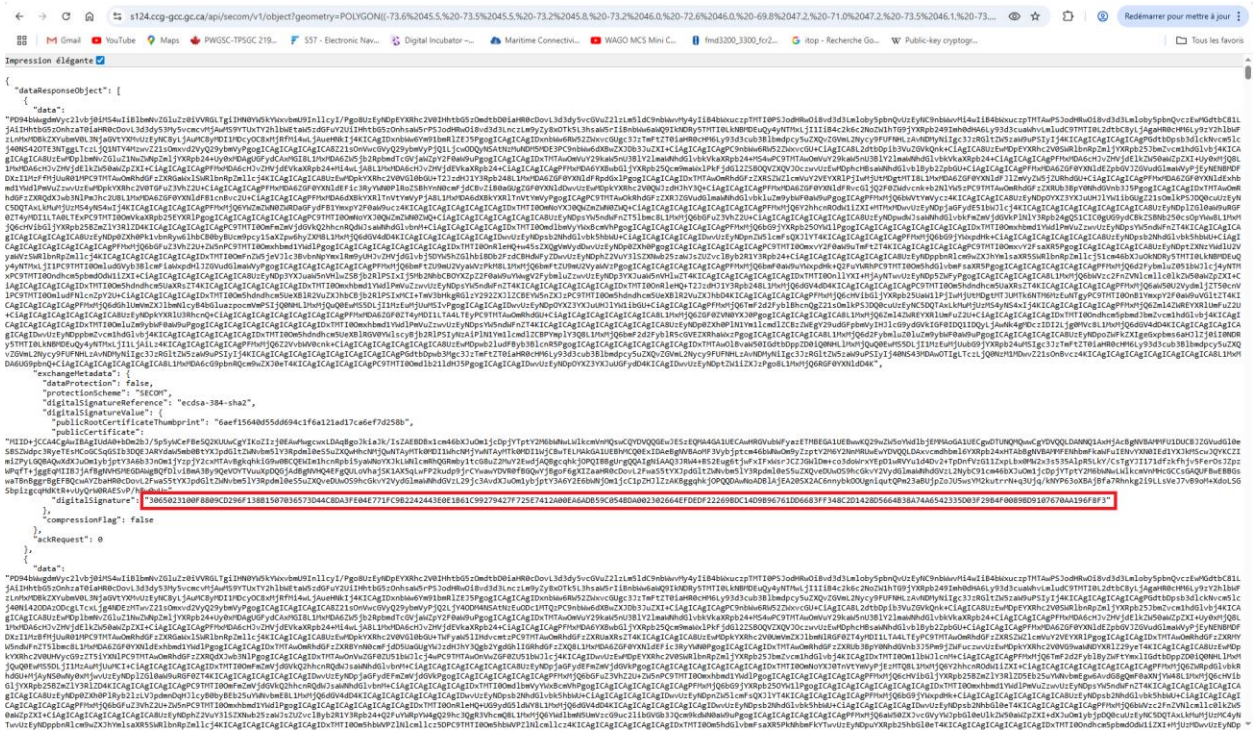

Figure 3.24

b) En Java,

i) Assigner la valeur de la signature (voir figure 3.24) à la variable (String) signature. Noter que la signature de la réponse de la requête est encodé dans le format HEX, ce sera géré en Java;

ii) Assigner le contenu du certificate public décodé de la réponse (voir figure 3.15) à la variable (String) decodedPublicCertificate;

iii) Assigner le contenu (voir figure 3.21) du champ 'data' associé à la signature à la variable (String) content. Noter que le champ 'data' de la réponse est encodé en Base64 et sera géré en Java;

iv) Exécuter les commandes suivantes (utiliser l'algorithme 'SHA384WITHECDSA'):

```
byte[] publicCertificateValue = Base64.getDecoder().decode(decodedPublicCertificate);
X509Certificate certificate = (X509Certificate) CertificateFactory.getInstance("X.509").generateCertificate(new
ByteArrayInputStream(publicCertificateValue));
PublicKey publicKey = certificate.getPublicKey();

Signature ecdsaSignature = Signature.getInstance("SHA384withECDSA");
ecdsaSignature.initVerify(publicKey);
ecdsaSignature.update(Base64.getDecoder().decode(content));
boolean isValid = ecdsaSignature.verify(HexFormat.of().parseHex(signature));
```

3.3 VALIDATION DE LA SIGNATURE DU FICHIER CATALOG.SIGN (DANS LE S-100 EXCHANGE SET)

Pour valider la signature contenue dans le fichier catalog.sign associée avec le fichier catalog.xml du S-100 exchange set (voir figure 3.25), suivre les étapes suivantes (L'étape 1, 2 et 3 sont pour les utilisateurs intermédiaires/avancés, nous recommandons directement d'aller à l'étape 4 si vous n'êtes pas un développeur ou un utilisateur avancé):

1) (Optionnel) Extraire le certificat du fichier catalog.xml file. Copier la valeur du certificate contenu dans la balise `"<certificate>"` (voir figure 3.26).

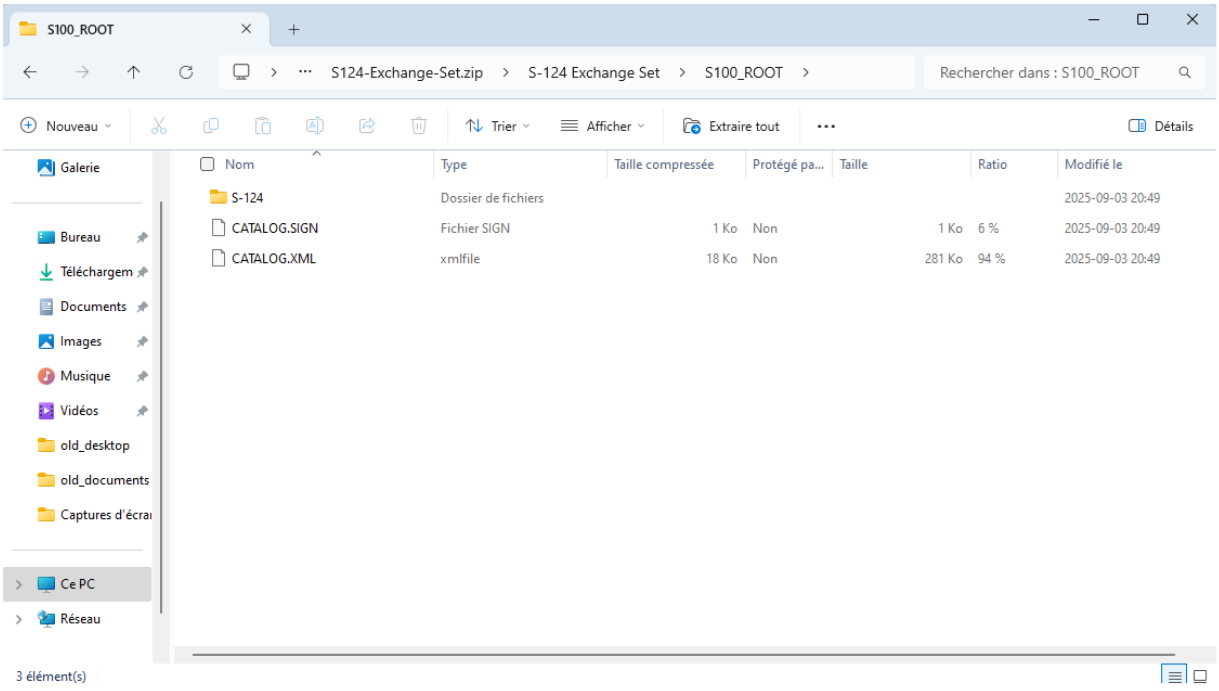

Figure 3.25

Figure 3.26

2) (Optionnel) Décoder la valeur du certificat encodé en Base64. Utiliser l'une 2 des deux options suivantes:

   a) Site web https://www.base64decode.org/ : Mettre la valeur du certificat publique de l'étape 1) dans la zone de texte du haut et appuyer sur le bouton « Decode ». Voir figure 3.27.

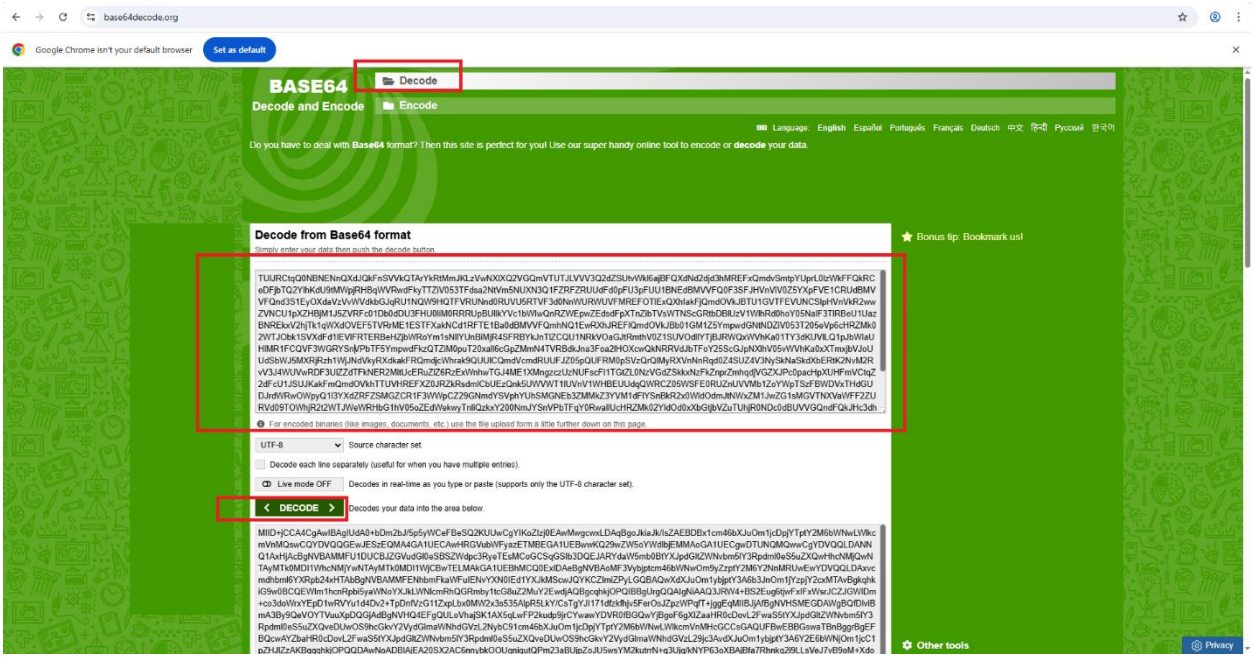

Figure 3.27

b) Utiliser un langage de programmation comme Java (Base64.getDecoder() en Java par exemple):

```
String decodedCertificateValue = new String(Base64.getDecoder().decode(certificateValue));
```

Figure 3.28

3) (Optionnel) Avec la valeur du certificat décodé, extraire la clé publique du certificat public de la GCC. Les 2 options suivantes peut être utilisées:

a) Mettre la valeur du certificate décodé de l'étape 2 dans un fichier nommé ccg_certificate.pem et l'entourrer -----BEGIN CERTIFICATE----- et -----END CERTIFICATE----- (voir figure 3.29);

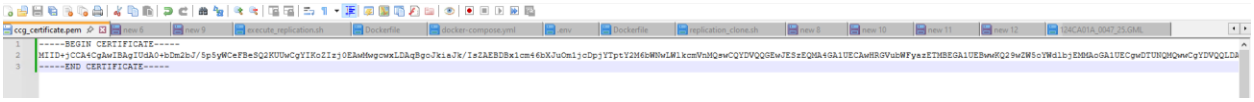

Figure 3.29

Exécuter la commande openssl suivante dans un terminal de l'environnement WSL (Linux environment in Windows) ou dans un terminal Linux:

openssl x509 -pubkey -noout -in ccg_certificate.pem > ec-secp384-r1-pub-key-extracted.pem (see figure 3.30)

Cette commande créera le fichier ec-secp384-r1-pub-key-extracted.pem contenant la clé publique (voir figure 3.31).

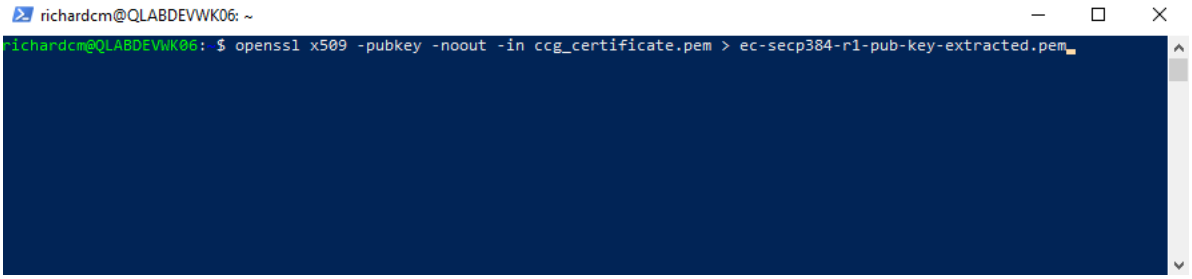

Figure 3.30

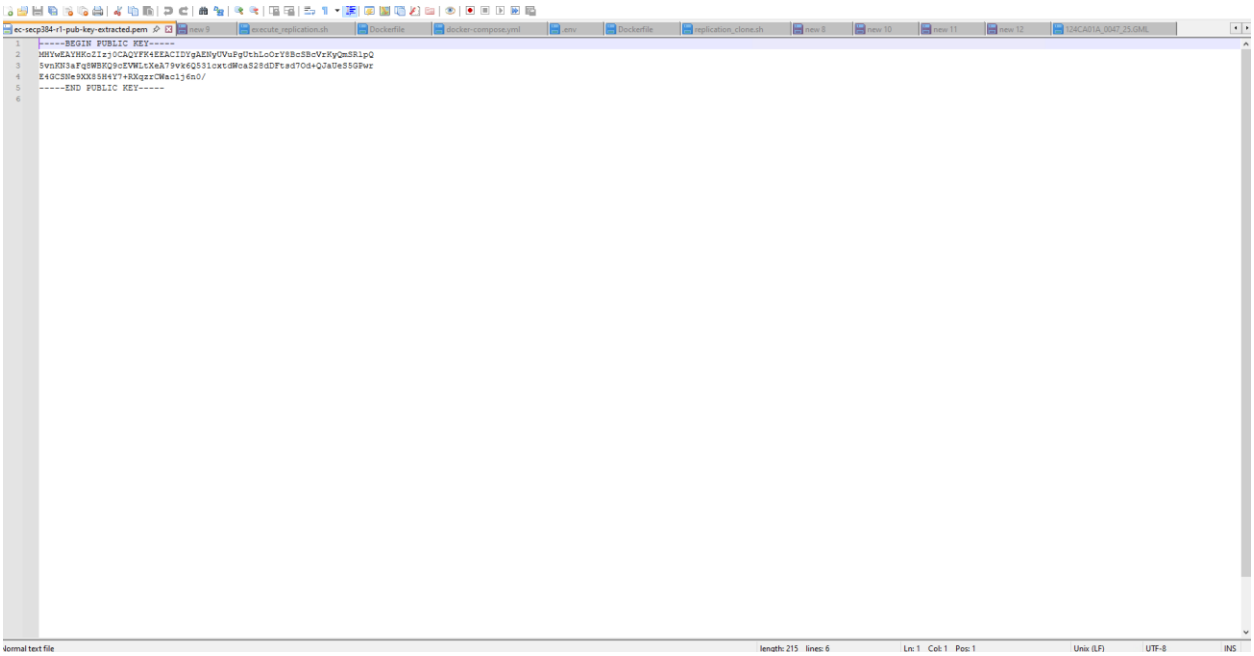

Figure 3.31

b) Utiliser un langage de programmation (comme Java avec la librairie java.security dans l'exemple qui suit) pour charger le certificat décodé et extraire la clé publique.

```
X509EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(decodedCertificateValue.getBytes());
PublicKey publicKey = keyFactory.generatePublic(publicSpec);
```

Figure 3.32

4) Valider la signature avec la clé publique et la valeur de la signature. Pour valider la signature, utiliser un outil en ligne (option a) ou un langage de programmation comme Java (option b).

a) Si vous avez sauté les étapes 1) 2) et 3), prendre la clé publique suivante extraite du certificat publique de la GCC:

-----BEGIN PUBLIC KEY-----

MHYwEAYHKoZIzj0CAQYFK4EEACIDYgAENyUVuPgUthLoOrY8BcSBcVrKyQmSRlpQ

5vnKN3aFq8WBKQ9cEVWLtXeA79vk6Q531cxtdWcaS28dDFtsd7Od+QJaUeS5GPwr

E4GCSNe9XX85H4Y7+RXqzrCWac1j6n0/

-----END PUBLIC KEY-----

Aller sur le site web https://emn178.github.io/online-tools/ecdsa/verify/ et sélectionner ECDSA->Verify Signature.

i) Sélectionner ‘Base64’ pour le champ ‘Input Encoding’ (see figure 3.33);

ii) Copier le contenu du fichier catalog.xml (figure 3.34) associé à la signature à valider et l’ajouter à la zone de texte ‘Input’ (voir figure 3.33);

iii) Sélectionner SECG secp384r1 / X9.63 ansip384r1 / NIST P-384 pour le champ ‘Curve’ (voir figure 3.33);

iv) Sélectionner SHA384 pour le champ ‘Signature Algorithm’ (see figure 3.33);

v) Sélectionner Pem Text pour le champ ‘Public Key – Type’ (see figure 3.33);

vi) Ajouter le contenu de la clé publique (entouré par -----BEGIN PUBLIC KEY-----

et -----END PUBLIC KEY-----) dans la zone de texte ’Public Key – Data’ (voir figure 3.33);

vii) Sélectionner Base64 pour le champ ‘Signature – Type’ (see figure 3.33).

viii) Copier le cotenu du fichier catalog.sign (dans cet exemple, voir figure 3.35) et l’ajouter dans la zone de texte ‘Signature – Data’ (see figure 3.33);

ix) Cliquer sur le bouton verify.

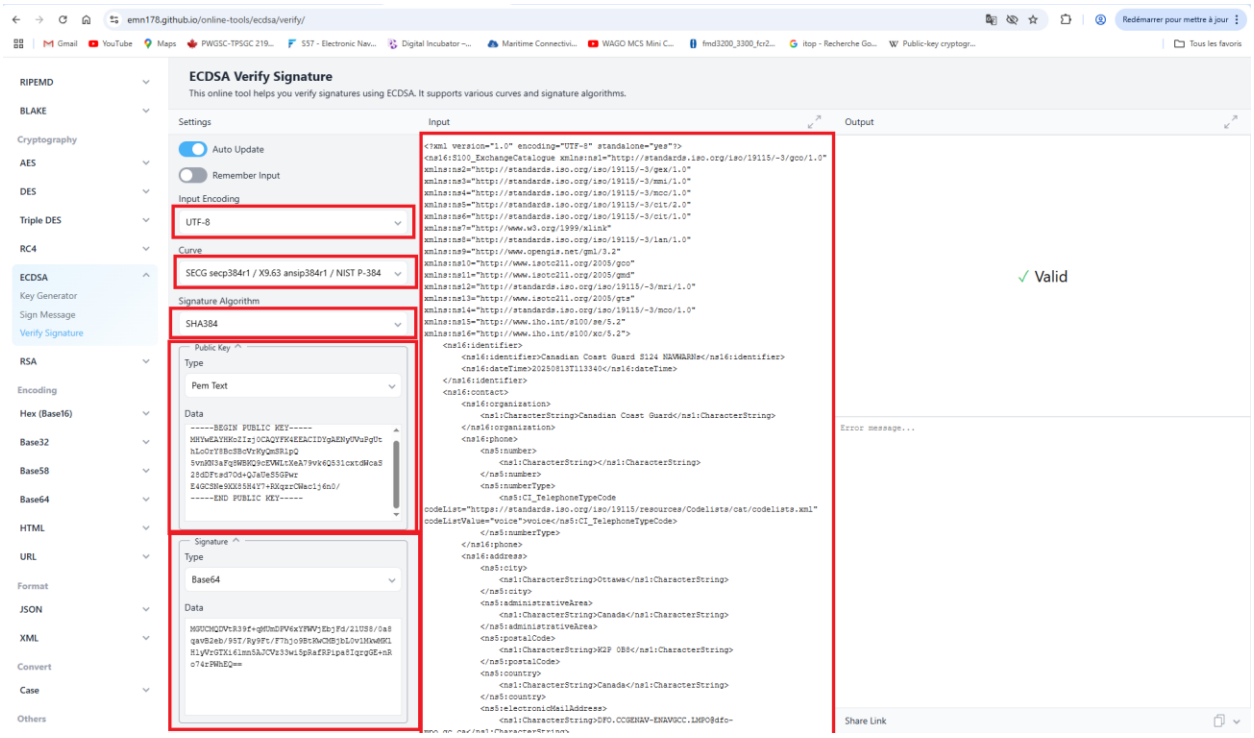

Figure 3.33

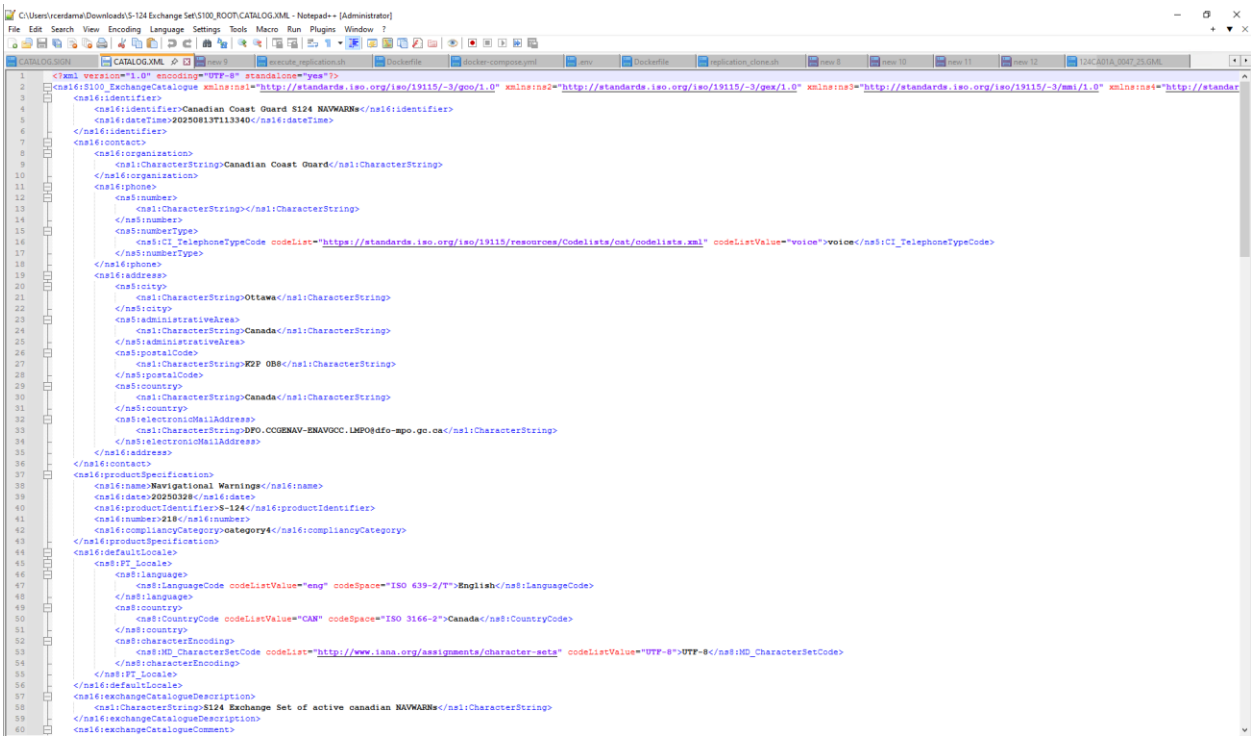

Figure 3.34

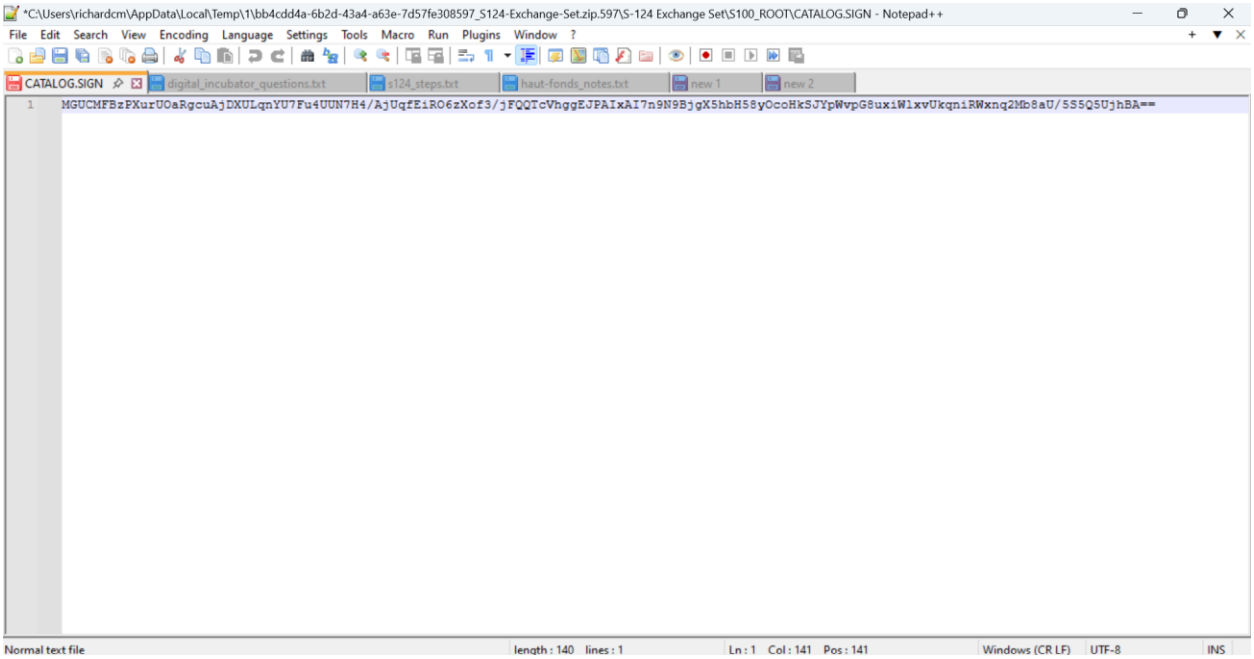

Figure 3.35

b) En Java,

i) Assigner la valeur de la signature (figure 3.35) à la variable (String) signature;

ii) Assigner la valeur du certificat public encodé du fichier catalog.xml (voir figure 3.26) à la variable (String) encodedPublicCertificate;

iii) Assigner le contenu du fichier catalog.xml (voir figure 3.34) à la variable (String) content;

iv) Exécuter les commandes suivantes (utiliser l'algorithme 'SHA384WITHECDSA'):

```
byte[] decodedPublicCertificate = Base64.getDecoder().decode(encodedPublicCertificate);
byte[] publicCertificateValue = Base64.getDecoder().decode(decodedPublicCertificate);
X509Certificate certificate = (X509Certificate) CertificateFactory.getInstance("X.509").generateCertificate(new
ByteArrayInputStream(publicCertificateValue));
PublicKey publicKey = certificate.getPublicKey();

Signature ecdsaSignature = Signature.getInstance("SHA384withECDSA");
ecdsaSignature.initVerify(publicKey);
ecdsaSignature.update(content.getBytes());
boolean isValid = ecdsaSignature.verify(Base64.getDecoder().decode(signature.getBytes()));
```