UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

INTEROPERABILITAT ENTRE ELS ESTÀNDARDS DE SEGURETAT DE LA INFORMACIÓ GENÒMICA

Grau en Enginyeria Informàtica

Especialitat Tecnologies de la informació

MARTÍ FERNÁNDEZ CABALLÉ

Director: Jaime M. Delgado Merce
Departament d'Arquitectura de Computadors

Juny, 2020

Dedicat a la meva germana, la Mariona
que segur que se'n en surt d'aquesta

Agraïments

Primer de tot vull agrair a en Jaime Delgado el suport i l'ajuda en la direcció d'aquest projecte i que m'hagi facilitat l'opció de treballar sobre MPEG-G.

També a la Sílvia Llorente per la seva ajuda realitzada juntament amb en Jaime durant tot el projecte.

I agrair a en Daniel Naro per tota l'ajuda en temes més específics de MPEG-G, en els apartats de la seguretat i privacitat.

De la mateixa forma donar les gràcies als meus pares (Marc i Fina) i germana (Mariona) per mantenir-me i aguantar-me i haver-me educat. I també al meu gat Eto. Sense vosaltres no seria el mateix.

Per últim, donar les gràcies a tots els meus amics, tant de la facultat com fora d'ella. I com no a la gent del 7vago per fer els viatges diaris d'AVE d'aquests 4 anys més entretinguts.

A tots: GRÀCIES

Martí Fernández Caballé

Girona, 22 de juny de 2020.

Interoperabilitat entre els estàndards de seguretat de la informació genòmica

Martí Fernández Caballé

Resum

Projecte centrat en l'estudi de la seguretat dels formats actuals de les dades genòmiques, en concret MPEG-G i el definit per GA4GH i en la seva interoperabilitat. En concret exposa una proposta d'utilització dels algoritmes de crypt4gh de GA4GH a MPEG-G així com una implementació minimalista d'aquesta. També analitza la interoperabilitat entre les APIs i la possibilitat d'aprofitat la privacitat de MPEG-G en el mapatge hts-get.

Resumen

Proyecto enfocado en el estudio de la seguridad de los formatos actuales de los datos genéticos, concretamente MPEG-G y el estándar definido por GA4GH y la interoperabilidad entre ellos. En concreto, se expone una propuesta de uso de algunos algoritmos de criptografía de GA4GH en MPEG-G, así como la realización de una implementación mínima. También analiza la interoperabilidad entre las API y la posibilidad de aprovechar la privacidad de MPEG-G en el mapaje de hts-get.

Abstract

Project focused on the study of the security of current formats of genomic data, specifically MPEG-G and the standard defined by GA4GH and the interoperability between them. Specifically, it exposes a proposal for the use of some GA4GH cryptography algorithms in MPEG-G, as well as a minimal implementation of it. Also the interoperability between the APIs and the possibility of taking advantage of the privacy of MPEG-G in the hts-get mapping.

Índex

Índex de figures

Índex de taules

Acrònims

ADN - Àcid desoxiribonucleic.
AES - Advanced Encryption Standard.
API - Interfície de programació d'aplicacions.
AU - Access Unit.
AUEP - AccessUnitEncryptionParameters.
AWS - Amazon Web Services.
BAM - Binari de SAM file.
BCF - Binari de VCF File.
CRAM - Compressed Genome Sequence Alignment File.
CTR - Counter.
DG - Dataset Group.
DH - Diffie Hellman.
DT - Dataset.
EC - Elliptic Curve / Corba El·líptica.
EP - EncryptionParameters.
FDIS - Final Draft International Standard.
FIB - Facultat d'Informàtica de Barcelona.
GA4GH - Global Alliance for Genomics and Health.
GCM - Galois/Counter Mode.
GDPR - Reglament General de Protecció de Dades.
GEP - Gestió de Projectes.
ID - Identificador.
IDE - Entorn integrat de desenvolupament.
IEC - International Electrotechnical Commission.
ISO - Organització Internacional per a l'Estandardització.
IV - Initialization Vector.
JTC1 - Joint Technical Committee 1.
KLV - Key Length Value.
MPEG - Moving Picture Experts Group.
MPEG-G - MPEG standard for Genomic Information Representation.
OAEP - Optimal Asymmetric Encryption Padding.
PBKDF2 - Password-Based Key Derivation Function 2.
PDI - Personal Docent i Investigador.
PhD - Doctor.
PRF - Pseudorandom Function.
RFC - Release For Comment.

RSA - Rivest, Shamir i Adleman.
SAM - Sequence Alignment/Map.
SC29 - Sub Committee 29.
TFG - Treball Final de Grau.
TIC - Tecnologies de la informació i la comunicació.
UML - Llenguatge de modelització unificat.
UPC - Universitat Politècnica de Catalunya.
VCF - Variant Call Format.
WG11 - Working Group 11.
XACML - eXtensible Access Control Markup Language.
XML - eXtensible Markup Language.

Capítol 1

Introducció i context

Projecte realitzat en el marc d'un Treball de Final de Grau [114] en modalitat A, del Grau en Enginyeria Informàtica a la **Facultat d'Informàtica de Barcelona** de la **Universitat Politècnica de Catalunya** [120] i titulat *Interoperabilitat entre els estàndards de seguretat de la informació genòmica.*

La finalitat principal del projecte era estudiar els formats actuals de dades genòmiques, principalment la seva seguretat i privacitat, un tema molt important amb dades mèdiques a causa de la llei de protecció de dades. I finalment també mirar, amb poc detall, les compatibilitats entre formats i les seves APIs.

Projecte realitzat per en Martí Fernández Caballé i dirigit per en Jaime M. Delgado Merce del Departament d'Arquitectura de Computadors (DAC).

En aquest document es troba la informació del projecte. Començant pels capítols 1, 2, 3 i 4 que introdueixen el projecte, el posen en context i defineixen objectius i temps. Posteriorment es troben els capítols 5 i 6 els quals informen i introdueixen els estàndard actuals però que han requerit un temps d'estudi i comprensió. Ja a partir d'aquests es troba la contribució realitzada en aquest projecte.

1.1 La tecnologia en l'àmbit de la medicina

La medicina és una branca de les ciències de la salut que s'ocupa de la ciència, prevenció i cura de les malalties humanes [132].

Al llarg de la història ha tingut una gran evolució i millora, des de l'Antiga Grècia amb Hipòcrates [131], per a molts anomenat "el pare de la medicina", fins a l'actualitat, passant per l'Antiga Roma, Renaixement, etc.

Aquesta millora, entre altres, ha permès la reducció de la mortalitat i millora de l'esperança de vida, com es pot veure a la figura 1.1, gràfica que va des de 1960, amb una esperança al voltant dels 52 anys, fins a l'any 2019, amb una esperança d'uns 73 anys. Aquesta evolució és deguda a l'augment del coneixement mèdic provinent de la investigació científica, sobretot a partir del segle XVII.

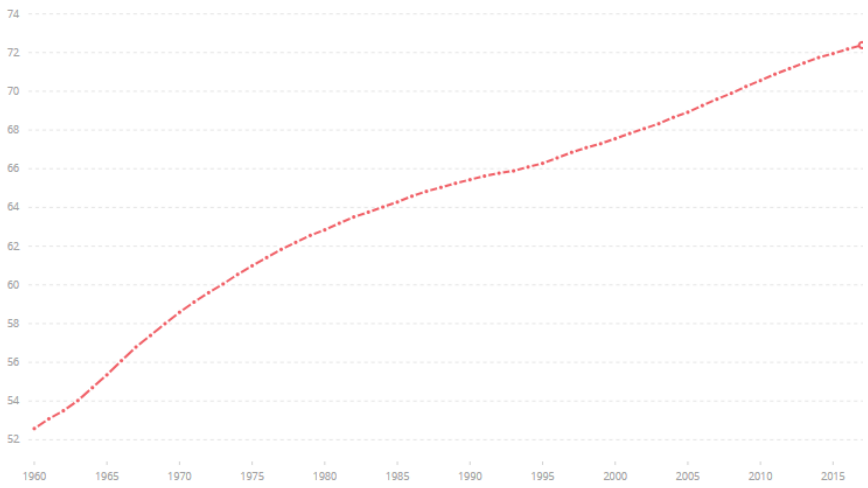


Figura 1.1: Evolució de l'esperança de vida. [27]

Aquest creixement ve marcat per la millora en la recerca i en tenir el coneixement més ordenat, així com nous plantejaments o el rigor científic. Aquest coneixement però, també es veu marcat amb les millores tecnològiques.

Les millores de la tecnologia en aquests àmbits són diverses. Una de les més importants i més antiga, en l'àmbit de la recerca, és el microscopi [133]. També trobem el descobriment de la radioteràpia o el modern robot Da Vinci [21] que millora considerablement les cirurgies. I ja més cap al segle XXI trobem un clar augment de tecnologies com el Big Data i Machine Learning en l'àmbit mèdic [83] degut a la major facilitat per recollir dades, emmagatzemar-les i transportar-les.

1.2 La importància de les dades genòmiques

Les millores tecnològiques permeten millorar molt en diferents camps, però un dels que més millora i que apareix a mitjans del segle XIX és el camp de la genètica [67] [129].

Aquestes millores han permès millorar l'estudi del genoma humà (apartat 1.2.1) i també una reducció de cost de la seqüenciació d'un genoma, com es pot veure a la figura 1.2 i una millora en la facilitat de l'extracció.

Aquesta millora permet un gran augment de recollida de dades, ja que és menys costos i més senzill i ràpid. Aquestes dades genètiques permeten estudiar un seguit de malalties [22], però també tenen utilitat en altres aspectes com en criminalística o ciència forense, història, antropologia i la bioinformàtica.

1.2.1 El genoma humà

El genoma humà [130] [86] és el material genètic que conté els cromosomes d'un ésser humà. Els cromosomes són la forma com l'ADN s'organitza dins les cèl·lules.

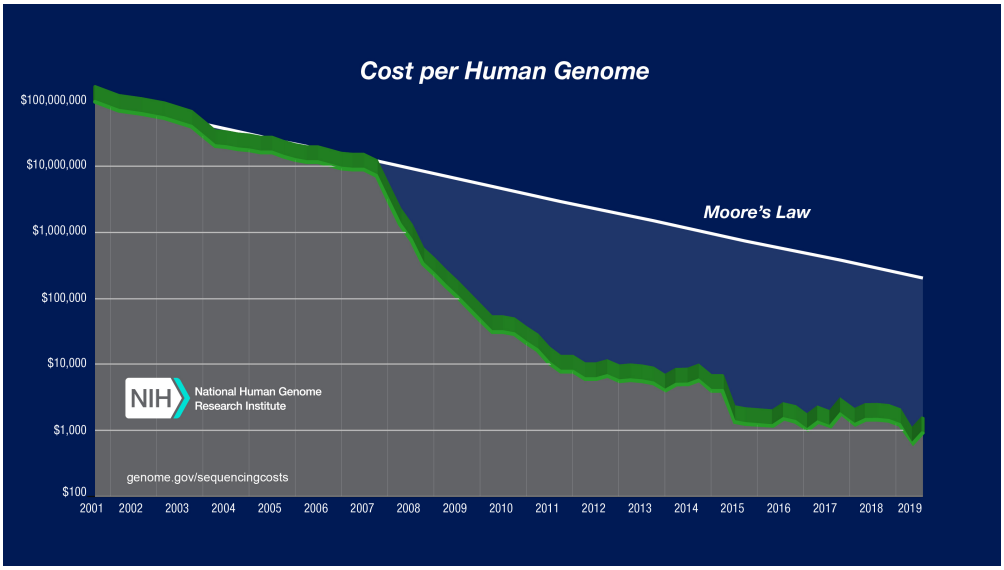


Figura 1.2: Evolució del preu d'extracció. [136]

Aquest genoma pot patir danys que són causes de malalties com el càncer, però al ser hereditari també pot venir amb danys i causar malalties. A més, un 97% del qual, no es sap perquè serveix.

ADN

L'ADN [36] (àcid desoxiribonucleic [124]) és un àcid nucleic [125] contingent de la informació genètica a llarg termini, i que s'encarregada de dirigir el desenvolupament i funcionament de tots els éssers vius coneguts i alguns virus. Es podrien considerar plantilles per produir altres parts de les cèl·lules, com les proteïnes, entre altres.

Els segments d'ADN que porten aquesta informació genètica reben el nom de gen, però altres seqüències d'ADN tenen una funció estructural o ajuden a regular l'ús d'aquesta informació genètica.

Bàsicament les molècules d'ADN tenen dues propietats importants:

- **Poden fer còpies de si mateix.** Si es separen les dues cadenes, es pot fer servir cadascuna per formar una de nova i per tant una nova molècula d'ADN.
- **Poden portar informació.** Una cadena és un codi per a la creació de proteïnes.

Gens

Un gen [128] [66] és un segment d'ADN que codifica una proteïna especifica. Per exemple un gen codifica la proteïna insulina.

Els gens son la unitat bàsica de la genètica i els humans en tenim al voltant de 20.000 i 25.000 per cada cèl·lula del cos. Aquests gens representen el 3% de tot l'ADN, la resta (97%) actualment no es sap que fan exactament.

Els gens estan localitzats dins els cromosomes, al nucli cel·lular [134]. Cada gen ocupa dins el cromosoma una posició determinada. El conjunt de gens d'una espècie s'anomena genoma.

Cromosomes

Dins de les cèl·lules, l'ADN s'organitza en estructures anomenades cromosomes [126] [65]. Que són les molècules d'ADN envasades al voltant de proteïnes anomenades histones. Aquests cromosomes són duplicats abans que les cèl·lules es divideixin, en un procés anomenat replicació de l'ADN. Els organismes eucariotes (animals, plantes, fongs i protists) emmagatzemen l'ADN dins del nucli cel·lular, en canvi els procariotes (eubacteris i arqueobacteris), es troba dins del citoplasma de la cèl·lula.

Els humans disposem d'un total de 23 parells de cromosomes. La meitat provinents del pare i l'altra meitat de la mare. Dels 23 parells, un parell és el que determina el sexe de l'individu. Les dones tenen 2 cromosomes X. Els homes tenen un cromosoma X i un Y. La mare li aporta un cromosoma X al fill, mentre que el pare pot contribuir sigui amb un cromosoma X o amb un cromosoma Y. És el cromosoma del pare el que determina si el nadó és un masculí o femení.

Els trets genètics poden ser dominants o recessius:
- Els trets dominants són controlats per un gen en el parell de cromosomes.
- Els trets recessius requereixen que ambdós gens en el parell de gens treballin junts.

Moltes característiques personals, com l'alçada, són determinades per més d'un gen. No obstant això, algunes malalties, com l'anèmia drepanocítica, poden ser ocasionades per un canvi en un sol gen.

El cromosoma més gran, el cromosoma 1, conté uns 8000 gens. El cromosoma més petit, el cromosoma 21, conté uns 300 gens. (El cromosoma 22 hauria de ser el més petit, però els científics es van equivocar quan els van numerar per primera vegada).

1.3 Les dades genòmiques a la tecnologia

Actualment, el fet de la reducció del cost i la major facilitat per aconseguir les seqüències genòmiques fa que es disposi cada vegada d'una quantitat superior de dades. Aquestes dades s'emmagatzemen i tenen una gran utilitat per la recerca [136] anomenada bioinformàtica però també en altres àmbits, i en un futur quan es vagi descobrint que significa cada un dels gens encara serà més important.

Tot aquest creixement fa que cada vegada hi hagi més dades que, al tractar-se de dades mèdiques, és molt important i indispensable que tinguin una correcta seguretat i privacitat.

1.3.1 Protecció de dades

La protecció de dades és un tema molt important a la nostra societat cada vegada més digitalitzada i amb més dades a la xarxa. Un clar exemple és la nova normativa europea GDPR [35]. Però també tenim la *Ley Organica de Protección de Datos* (LODP). En el cas d'informació mèdica és molt important i és un dels nivells que més seguretat i privacitat es necessita. [62] [40].

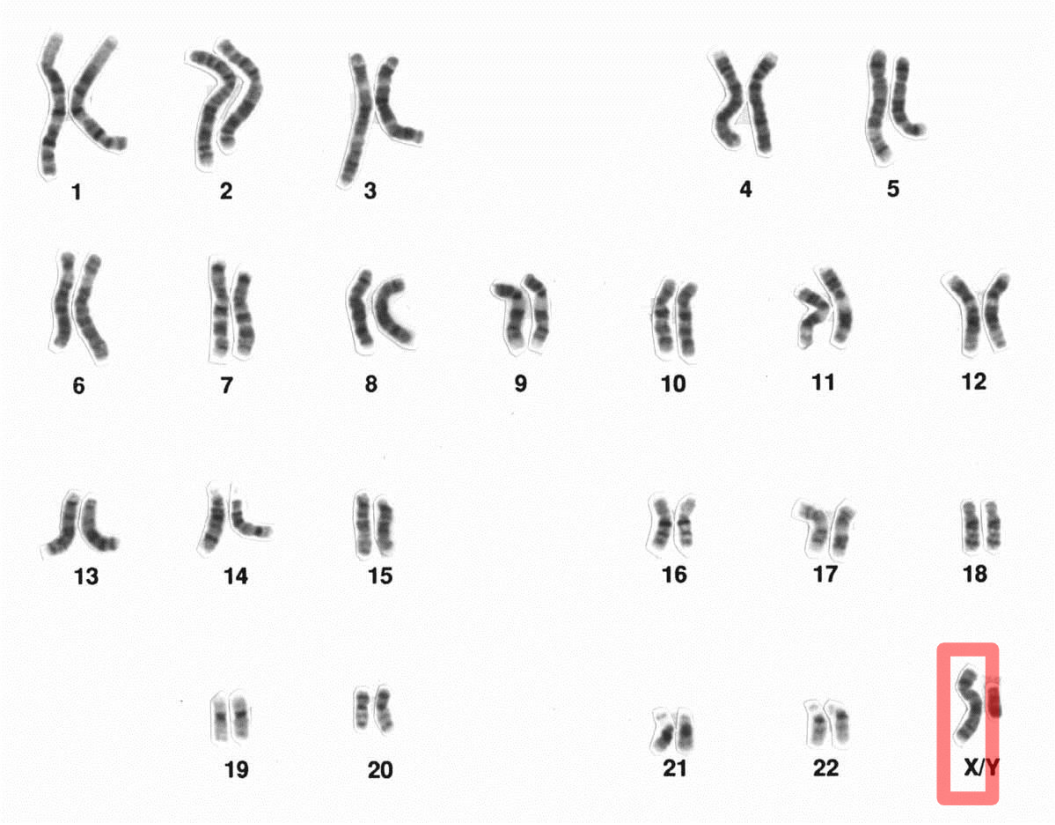


Figura 1.3: Cariograma de cromosomes. [127]

1.3.2 GDPR

La GDPR, *General Data Protection Regulation*, és la regulació Europea pel qual el Parlament Europeu, el Consell de la Unió Europea i la Comissió Europea tenen la intenció de reforçar i unificar la protecció de dades personals per tots els estats dins de la Unió Europea (UE). I vigent des de l'any 2018.

També s'ocupa de l'exportació de dades personals fora de la UE i les condicions necessàries per fer-ho legalment. L'objectiu principal del GDPR és donar control als ciutadans i residents sobre les seves dades personals i simplificar l'entorn regulador dels negocis internacionals unificant la regulació dins de la UE. Un resum referent al GDRP en general es pot veure a l'annex A.1, on trobem les possibles sancions, que fa el Data Protection Officer entre altres.

També trobem dins d'aquesta alguns apartats específics sobre dades sensibles, el que inclou dades mèdiques i dades genètiques.

L'article 9 [7] parla de dades més sensibles de les dades personals, en concret de les que diguin la raça o ètnia, opinions polítiques, creences religioses o filosòfiques, pertinença a un sindicat, el tractament de dades genètiques, dades biomètriques de forma exclusiva d'una persona física, dades sobre salut o dades sobre la vida sexual o l'orientació sexual d'una persona física. Aquestes no es poden recol·lectar i tractar excepte en el cas que es compleixi

alguns dels punts següents. [1]

- Es doni el consentiment explícit pel propòsit especificat.
- Quan sigui necessari per matèria de dret laboral, seguretat social o de protecció social.
- Quan sigui necessari per protegir a l'interessat o algú altre i l'interessat no estigui capacitat legalment o físicament per donar consentiment.
- Quan es realitza per part d'una fundació, associació o qualsevol altre organisme sense ànim de lucre amb un objectiu polític, filosòfic, religiós o sindical i amb la condició que el processament tingui relació exclusivament amb els membres o ex-membres o a persones que tinguin contacte regular amb ell en relació amb els seus propòsits i en cap cas es poden divulgar fora de l'organisme sense consentiment explícit.
- Son dades que s'han donat fet publiques prèviament per l'interessat.
- Per temes referents a procediments legals o per ordre judicial.
- Si el tractament és necessari per raons d'interès públic substancial i que sigui proporcional a l'objectiu perseguit.
- En referència a medicina preventiva, capacitats per realitzar una feina, diagnòstic mèdic, tractament social o de salut, sistemes i serveis sanitaris o atenció social. Processades per o sota la responsabilitat d'un professional sotmès a l'obligació del secret professional segons la legislació de l'estat concret o normes establertes per organismes nacionals competents.
- Si és necessari per raons de salut pública o assegurar alts estàndards de qualitat i seguretat a l'assistència sanitària i productes o dispositius mèdics. Els estats ha de definir mesures adequades i específiques per a salvaguardar els drets i les llibertats de l'interessat, en particular el secret professional.
- Si és necessari per arxius per interès públic, investigació científica o històrica o estadístiques. Segons la llei de l'estat membre i que ha de ser proporcional a l'objectiu perseguit, respectant l'essència del dret a protecció de dades i preveure mesures adequades i específiques per a salvaguardar els drets fonamentals i els interessos de l'interessat. Aquestes salvaguardes garanteixen que es posin en pràctica mesures tècniques i organitzatives. Quan els propòsits ho permeti les dades s'han de recol·lectar de forma anònima.

Els Estats membres poden mantenir o introduir més condicions, incloses limitacions pel que fa al tractament de dades genètiques, dades biomètriques o dades relatives a la salut.

A més, a part de l'article també hi trobem 'recitals' com són el 34 [91], 35 [92], 51 [93], 53 [94] i 54 [95], que augmenten la informació.

Com en la resta de dades i com es veu a l'annex A.1 les dades s'han de protegir d'una forma que assegurin la seva privacitat en funció del perill, importància, etc. El GDPR no especifica quines mesures concretes s'han d'utilitzar però si que es recomana encriptació de totes les dades.

[1] Els punts han estat reduïts i en algun cas es pot haver perdut informació, no necessària per tenir una visió general de la normativa.

1.4 Els formats actuals de dades genòmiques

Actualment per la gestió de les dades genòmiques emmagatzemades hi ha un gran nombre de formats diferents que permeten l'estudi i emmagatzematge de les dades genètiques d'una persona. Els principals en aquest projecte són GA4GH i MPEG-G, els quals s'han tractat en aquest projecte.

1.5 Motivació

Els actors implicats en aquests camps de dades genètiques són diversos, van des de les persones encarregades de realitzar els formats i processar les dades fins al pacient passant per metges, científics entre altres. Sobretot és important per qui treballa amb dades genètiques, i el seu interès està a tenir unes dades que puguin anar ràpidament per aconseguir conclusions tan aviat com sigui possible. A més com més dades i d'una importància com les genètiques, com s'ha vist anteriorment on les dades genètiques és el que defineix els cèl·lules, més conclusions es poden treure. Un clar exemple de l'ús de la genètica és la seqüenciació del genoma del SARS-CoV-2 [26].

Per tant és necessari i es fan estàndards que siguin ràpids, fàcils de transportar, etc. Però també és molt important la seguretat i integritat, perquè aquestes dades no acabin en males mans ni es modifiqui la informació provocant errors, com ja s'ha vist amb les lleis, és per això que es va voler realitzar aquest projecte.

Capítol 2

Abast

2.1 Objectius

Aquest projecte tenia un objectiu principal que era veure els diferents estàndards que hi ha actualment per tractar dades genòmiques i estudiar la seva seguretat i privacitat, així com la seva interoperabilitat entre ells i poder millorar el que hi ha actualment. Per fer-ho hi havia tres subobjectius.

2.1.1 Afegir crypt4gh a MPEG-G

Crypt4gh de GA4GH utilitza un sistema d'encriptació diferent de MPEG-G amb els avantatges i inconvenients que això suposa. El primer subobjectiu del projecte era afegir l'encriptació que utilitza crypt4gh a MPEG-G. Tan conceptualment i realitzant una proposta de col·laboració a l'estàndard MPEG-G, com realitzar una implementació bàsica per comprovar el funcionament.

Un altre punt tingut en compte en aquest apartat era comprovar si el que comenta el document de crypt4gh [37] sobre X25519, que diu que no té una distribució de bits completament uniforme, és cert o no.

2.1.2 Mapping entre APIs

El segon subobjectiu del projecte era mapar conceptualment l'API que utilitza GA4GH, en concret Hts-get [41], sobre l'API de MPEG-G i així aplicacions que actualment treballin amb l'API hts-get es puguin seguir utilitzant amb fitxers amb el format MPEG-G.

2.1.3 Seguretat i privacitat de les APIs

El tercer subobjectiu del projecte era afegir la seguretat i privacitat de MPEG-G al *mapping* de l'API, i poder utilitzar els camps que hi ha als fitxers MPEG-G sobre aquests temes, que en canvi GA4GH no té, sinó que suposa que l'aplicació ja ho gestiona.

2.2 Eines i tecnologies

Les eines TIC emprades per la realització del projecte han estat les següents.

Cocalc	IDE Online Python amb llibreries
Overleaf	Editor online de Latex
Mendeley	Gestor de referències
Github	Control de versions de codi
Dropbox	Emmagatzematge de fitxers al núvol
Google Drive	Emmagatzematge de fitxers al núvol
Gantt Project	Programa per diagrames de Gantt
Intellij	IDE per desenvolupar amb Java
Draw.io	App per elaborar diagrames
Amazon Web Services	Màquina virtual

Taula 2.1: Eines utilitzades. (Elaboració pròpia)

2.3 Coneixements

Per la realització del projecte s'ha utilitzat els següents coneixements, entre altres que no apareixen, adquirits durant el Grau en Enginyeria Informàtica o que s'han obtingut per poder realitzar el projecte.

- Programació orientada a objectes.
- Llenguatge de programació Java i llibreries criptogràfiques.
- Programació en Python.
- XML i XSD Schemas [135].
- JAXB.
- Maquina virtual Amazon Web Services EC2.
- Linux Bash.
- Seguretat Informàtica.
- Criptografia (AES, Chacha20, etc).
- Privacitat.
- Introducció a la genètica.
- Especificació MPEG-G i GA4GH.
- Lectura i comprensió de documentació tècnica.
- Estadística: distribució normal, binomial i llenguatge R.
- Maven.
- OpenID, OAuth2.0 i TLS.

2.4 Seguiment

Per realitzar el seguiment del procés s'han anat realitzant reunions periòdicament amb el director, quan el contingut s'havia avançat suficient o sorgien dubtes. També s'ha mantingut el contacte via e-mail amb el director per tenir un seguiment constant del projecte.

Finalment es van marcar dates límits per anar-se complint en els períodes previstos i així assolir els objectius del projecte en el termini establert.

2.5 Metodologia

Per tal de realitzar el projecte s'ha realitzat en una metodologia en cascada. La raó és que a causa de la necessitat de inicialment haver de fer una anàlisi, posteriorment l'esquema conceptualment i finalment la implementació feia que el més senzill fos seguir aquesta metodologia.

Inicialment es va definir el projecte i els seus objectius, especificats anteriorment. Un cop es tenien els objectius es va definir, quasi naturalment, les tasques i el seu ordre, definides al capítol 3.

Un cop definides es va procedir a realitzar-les. Mentre s'anaven realitzant les tasques s'ha dut a terme un control, per veure que els objectius es complien i els terminis també, tot i no haver marcat unes dates exactes per cada tasca però si una data aproximada per cada una degut a la irregularitat de les hores dedicades setmanalment per factors externs, així com la desconeixença del temps requerit en cada apartat.

Capítol 3

Planificació

El projecte al tractar-se d'un TFG realitzat a la FIB en un Grau d'enginyeria Informàtica [114] hauria de tenir una durada equivalent a 18 crèdits ECTS, on un crèdit equival a entre 25 i 30 hores de treball, tant autònom com dirigit [25]. Per tant entre les 540h i 450h. El temps total d'aquest projecte ha estat de 575h.

Pel que fa a la planificació del projecte les dates més importants han estat les següents:
- Data inici projecte: Octubre 2019
- Data final projecte: 22 de Juny de 2020
- Data lectura: 29 Juny - 3 Juliol de 2020

Altres dates destacades han estat:
- Entrega Fita Inicial: 16 de març de 2020.
- Elecció torn de lectura: fins al 2 de juny de 2020.
- Limit reunió de seguiment: 2 de juny de 2020.
- Data limit entrega memòria: 25 de juny de 2020.

Dedicació

Es van dedicar hores de forma puntual i segons disponibilitat des de l'octubre de 2019 fins al 10 de febrer de 2020. I s'han dedicat una mitjana de 4 hores diàries (no festius) entre el 10 de febrer i el 25 de juny de 2020.
Entre el dia 10 de febrer i el dia de lectura hi ha hagut unes 20 setmanes aproximadament. Es pot veure amb més detall a l'apartat 3.2.

Recursos

A les tasques, si no s'especifica el contrari, ha sigut necessari el personal, mínim un ordinador, connexió a internet, el software que s'especifica a cada tasca, un local i els recursos necessaris com electricitat, aigua, etc.

3.1 Tasques

A la Taula 3.1 es veu el resultat de les tasques amb la seva dedicació en hores i les dependències. S'agrupen en 8 grans grups.

Id	Tasca	Durada	Dependències
T1	**Gestió del projecte**	**85 h**	
T1.1	Contextualització i abast	20 h	-
T1.2	Planificació temporal	15 h	-
T1.3	Planificació econòmica	10 h	T1.2
T1.4	Informe sostenibilitat	5 h	-
T1.5	Informe seguiment	5 h	T1.1, T1.2, T1.3, T1.4
T1.6	Reunions	15 h	-
T1.7	Gestions	15 h	-
T2	**Treball previ**	**150 h**	
T2.1	Documentació MPEG-G	70 h	-
T2.2	Documentació crypt4GH	10 h	-
T2.3	Documentació hts-get	15 h	-
T2.4	Documentació tecnologies	10 h	-
T2.5	Documentació encriptació	35 h	-
T2.6	Familiarització amb el programari	10 h	-
T3	**Cryt4gh a MPEG-G**	**25 h**	
T3.1	Afegir seguretat	20 h	T2.1, T2.2, T2.4, T2.5
T3.2	Document col·laboració	5 h	T3.1
T4	**Implemenració MPEG-G**	**145 h**	
T4.1	Especificació	15 h	T3.1
T4.2	Documentació llibreries	10 h	T2.5
T4.3	Documentació JAXB	5 h	-
T4.4	XSD Schemas	5 h	T2.1, T2.4, T2.6
T4.5	Implementació boxes	30 h	T2.1, T2.6
T4.6	Implementació KLV	15 h	T2.1, T2.4
T4.7	Tests classes	30 h	T4.1, T4.2, T4.3, T4.4, T4.5, T4.6
T4.8	Implementació final	35 h	T4.7
T5	**Script refutar**	**35 h**	
T5.1	Maquina AWS	5 h	-
T5.2	Scripts	15 h	T2.2, T2.5
T5.3	Estadística	5 h	T2.2, T2.5
T5.4	Execució i resultats	10 h	T5.1, T5.2
T6	**Mapping entre APIs**	**30 h**	
T6.1	Mapping API	15 h	T2.3, T3.1
T6.2	Seguretat mapping API	15 h	T6.1
T7	**GDPR**	**5 h**	
T8	**Documentació**	**100 h**	
T8.1	Documentació i memòria	80 h	-
T8.2	Presentació final i defensa	20 h	T9.1
TOTAL		**575 h**	

Taula 3.1: Tasques a realitzar. (Elaboració pròpia)

3.1.1 Gestió del projecte

T1.1 - Contextualització i abast (20h)

Posar en context el treball dins l'àmbit de la informàtica i la medicina, sobretot en l'apartat de la genètica. Valorar l'abast del projecte i definir com es realitzarà el seguiment de les tasques. Utilitzant Overleaf [84] per redactar i Google Drive [38] per realitzar esquemes i resums.

T1.2 - Planificació temporal (15h)

Planificar temporalment les diferents tasques, el temps aproximat que portarà cada una i la seva descripció, així com les dependències i recursos.

Pel que fa a software utilitzant Overleaf, GanttProject [34] i Google Drive.

T1.3 - Planificació econòmica (10h)

Planificar econòmicament el projecte, valorant els recursos necessaris, personal i hores necessàries i el seu cost.

Utilitzant Overleaf i Google Drive.

T1.4 - Informe sostenibilitat (5h)

Realitzar un informe sobre la sostenibilitat del projecte. Tant en l'àmbit social, com mediambiental i econòmic. Pel que fa al software utilitzant Overleaf i Google Drive.

T1.5 - Informe de seguiment (5h)

Realitzar un informe de seguiment i de les variacions respecte a la planificació inicial en un punt intermedi.

T1.6 - Reunions (15h)

Reunions dutes a terme durant el projecte entre el director, l'estudiant i en casos altres PDI i PhDs. Aquestes reunions dutes a terme quan sigui necessàries i per resoldre dubtes o decidir la direcció del projecte.

T1.7 - Gestions (15h)

Realitzar les gestions necessàries pel TFG. Llegir normativa, inscriure projecte, etc.

3.1.2 Treball previ

T2.1 - Documentació MPEG-G (70h)

Llegir i entendre el funcionament de MPEG-G, l'apartat de metadata, protecció i API [48], la forma en què encripta, així com estan definides les seves boxes. Necessari el document on es troba aquesta especificació.

T2.2 - Documentació crypt4gh (10h)

Llegir i entendre el funcionament de crypt4gh [37] i els seus mètodes d'encriptació.

T2.3 - Documentació hts-get (15h)

Llegir i entendre el funcionament de hts-get [41], una de les APIs que utilitza GA4GH.

T2.4 - Documentació tecnologies (10h)

Documentar-se i comprendre diferents tecnologies desconegudes o amb poc coneixement com XML [44], KLV [138], Blake2b [102], TLS [29] entre altres. Necessari les documentacions d'aquestes tecnologies.

T2.5 - Documentació encriptació (35h)

Documentar-se i comprendre diferents mètodes d'encriptació i 'wrapping' com RSA [70], AES [3] [31], ChaCha20 [13], Poly1305 [14], AES Wrap [103], Corbes El·lítiques [61] [117], PKCS [58], Chacha20-Poly1305 [105] [80], etc.

T2.6 - Familiarització amb el programari (10h)

Instal·lar el programari necessari i entendre el seu funcionament. Alguns d'aquests són Intellij [20] per desenvolupar la implementació de MPEG-G amb Java. Cocalc per realitzar un script amb Python i llibreries criptogràfiques.

3.1.3 Cryt4gh a MPEG-G

T3.1 - Afegir seguretat (20h)

Pensar conceptualment com afegir la seguretat de crypt4gh a MPEG-G. Definint els camps XML necessaris, les variacions a l'especificació, entre altres.

T3.2 - Document col·laboració (5h)

Realitzar un document amb angles especificant com es podria afegir la seguretat per poder col·laborar amb l'ISO d'MPEG-G apartat 3. Utilitzant Overleaf.

3.1.4 Implementació MPEG-G

Implementar les boxes de protecció i l'encriptació de MPEG-G amb Java.

T4.1 - Especificació (15h)

Especificar les classes necessàries i diagrames per veure el funcionament. I realitzar l'UML de la implementació.

T4.2 - Documentació llibreries (10h)

Buscar i documentar-se sobre les llibreries necessàries.

T4.3 - Documentació JAXB (5h)

Entendre el funcionament i utilització de JAXB [51] per convertir els esquemes XML a Java fàcil i ràpidament.

T4.4 - XSD Schemas (5h)

Adaptar els esquemes XML (XSD Schemas) del document de MPEG-G [48] a la implementació a realitzar deixant només el necessari.

T4.5 - Implementació boxes (30h)

Implementar mitjançant Java les boxes dels fitxers MPEG-G amb tots els seus paràmetres. Utilitzant Intellij IDEA.

T4.6 - Implementació KLV (15h)

Implementar la lectura i escriptura del fitxer en format KLV. Utilitzant Intellij IDEA.

T4.7 - Tests classes (30h)

Testejar les classes implementades per comprovar el seu correcte funcionament.

T4.8 - Implementació final (35h)

Fusionar les implementacions i afegir el necessari per tenir una implementació total que permeti llegir i escriure fitxers MPEG-G de prova on bàsicament la seva part important serien els boxes de metadata i protecció. Utilitzant el software Intellij IDEA.

3.1.5 Script refutar

T5.1 - Maquina AWS (5h)

Posar en marxa una instància d'EC2 [6] d'AWS per executar els Scripts i tota la gestió necessària.

T5.2 - Scripts (15h)

Generar scripts amb Python i llibreries criptogràfiques per refutar o confirmar el que diu el document de cypt4gh [37] sobre que la corba el·líptica X25519 [61] no té una distribució totalment uniforme. Necessari Cocalc.

T5.3 - Estadística (5h)

Comprendre els diferents tipus de distribucions com la binomial, normal, uniforme discreta entre altres, per poder realitzar tests de normalitat sobre la distribució resultant dels scripts en Python.

T5.4 - Execució i resultats (10h)

Executar els scripts a la màquina virtual i analitzar els resultats utilitzant R per fer tests de normalitat.

3.1.6 Mapping entre APIs

T6.1 - Mapping API (15h)

Realitzar conceptualment i esquemàticament el mapatge de l'API HTS-Get [42] a l'API de MPEG-G. Mirant les crides que té hts-get i permetent que es cridin sobre fitxers MPEG-G.

T6.2 - Seguretat mapping API (15h)

Realitzar conceptualment la forma d'afegir al mapatge de l'API HTS-Get seguretat i privacitat dels fitxers MPEG-G. I analitzant AAIConnect [69] de GA4GH per realitzar l'autorització.

3.1.7 GDPR

A partir dels coneixements del treball realitzat a l'assignatura ASMI (Aspectes Socials i Mediambientals de la Informàtica) [8] acabar d'ampliar la informació per posar més en context en la GDPR les dades genòmiques i mediques.

3.1.8 Documentació

T8.1 - Documentació i memòria (80h)

Documentar tot el necessari durant el projecte i finalment realitzar la memòria del projecte. Utilitzant Overleaf, Google Drive, Mendeley [68] i Dropbox.

T8.2 - Presentació final i defensa (20h)

Preparar i realitzar la presentació i defensa del treball. Utilitzant Google Drive.

3.2 Dedicació setmanal

La distribució d'hores durant les setmanes de l'any 2020 aproximadament ha estat la que s'observa a la figura 3.1. S'hi han de sumar unes 65 hores invertides durant el 2019 no presents al gràfic i invertides entre octubre i desembre.

3.3 Canvis respecte a la planificació inicial

Al document de Gestió de Projectes s'especificava la planificació inicial del projecte i es definia la planificació temporal, i també una previsió de les despeses que suposaria i pel que fa a l'informe de seguiment es pot veure el punt intermedi del projecte les variacions i progressió respecte a la planificació inicial.

Pel que fa a la planificació econòmica inicial com que no és un projecte que hi hagi les despeses reals i per tant no podem tenir resultats econòmics, i en cas de realitzar-se una aproximació

no real les grans variacions serien en les hores invertides, per tant només modificar les taules de la previsió inicial amb les hores de dedicades, s'ha decidit finalment no s'ha realitzat al final.

Respecte al material utilitzat només hi ha la variació que en lloc d'utilitzar Cocalc per l'execució de l'script, s'ha utilitzat AWS, per tant el cost de Cocalc hagués estat 0, ja que per programar i prou és gratuït, i d'AWS s'hauria invertit uns 50 € calculats mitjançant l'aproximació que fa AWS Educate en dòlars.

A part de l'informe de seguiment respecte a la planificació inicial s'han modificat algunes coses, les quals totes estaven previstes a la gestió del risc inicial, i no han suposat cap gran problema.

1. S'ha invertit més temps del previst en el Prototip de la seguretat de MPEG-G degut a la dificultat de la implementació de les 'boxes' de MPEG-G.

2. S'ha invertit més temps del previst en valorar la frase sobre la corba X22519 de crypt4gh, ja que s'han realitzat més proves de les inicialment previstes.

3. La implementació de l'API, inicialment prevista, no s'ha dut a terme però si l'especificació, a causa del comentat al punt 1 i 2 i que es va valorar que era millor acabar els punts previs.

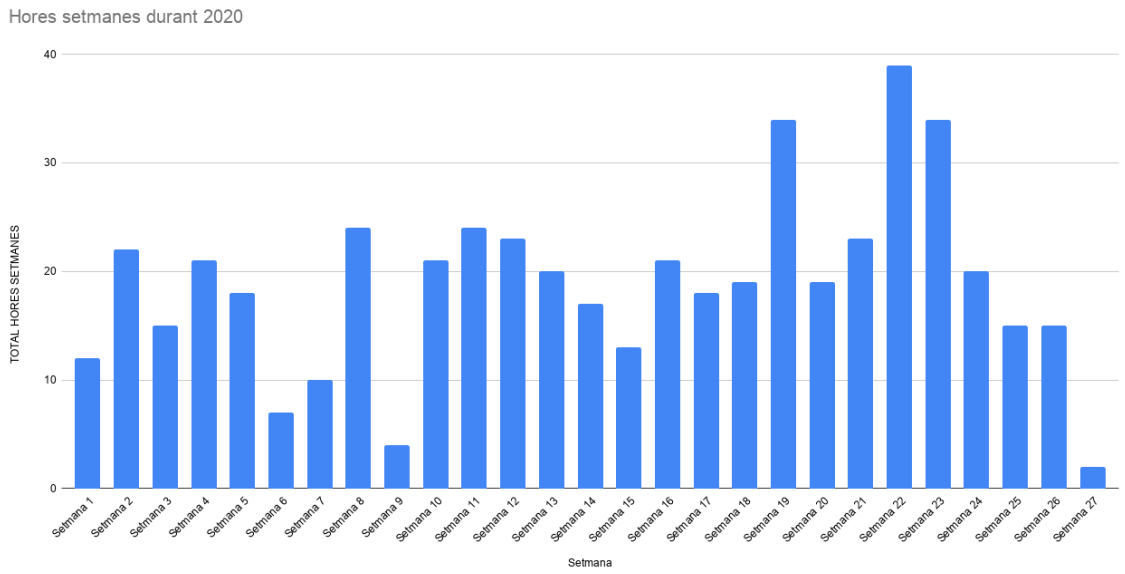


Figura 3.1: Hores invertides durant el 2020. (Elaboració pròpia)

Capítol 4

Sostenibilitat i compromís social

Al tractar-se d'un projecte d'Enginyeria Informàtica i sense haver de fer cap objecte material no té gaire sentit realitzar un gran estudi sobre la sostenibilitat amb gran profunditat del projecte, sobretot en l'aspecte ambiental i econòmic. Això és degut al fet que en la informàtica bàsicament és necessari un ordinador, amb tot el que comporta la seva fabricació i transport i la seva electricitat, i la de les empreses que proveeixen serveis, o el local. Però això no és molt diferent d'altres projectes, per exemple d'enginyeria industrial o arquitectura, entre altres, on majoritàriament també s'utilitzen ordinadors però en molts casos és una part molt petita tant del cost com de l'impacte ambiental.

A més en la informàtica la majoria de vegades s'utilitzen estructures que ja existeixen i amb poc consum, com el cas de la connexió a internet i no màquines específiques per cada projecte.

És per tot això que en la informàtica el cost econòmic és majoritàriament del personal i l'impacte ambiental del local on es troba.

4.1 Econòmica

Econòmicament aquest projecte és equivalent als actuals, ja que majoritàriament el cost és humà i dels ordinadors necessaris. Com podem veure a l'apartat de planificació econòmica del document de Gestió de Projectes, dels 23.985 € totals 15.210 € són de personal, per tant al voltant del 63%, a més tenint en compte que als 23.985 € ja hi ha sumada la contingència i imprevistos.

Possiblement en recursos humans és superior el tema hores perquè al tractar-se d'un estudiant la formació ha de ser superior que algú que ja porti temps treballant amb el tema.

4.2 Social

Aquest projecte millora el coneixement de l'autor, ja que assoleix nous coneixements de tecnologies no estudiades fins al moment. També suposa una millora per la comunitat que defineix els estàndards, ja que és una persona més treballant per la millora d'aquests i fent propostes. En cas de sortir bé podria millorar també la vida a la gent que treballa amb les dades per tenir més opcions d'encriptació i per poder utilitzar la mateixa API amb uns fitxers diferents.

Aquest projecte no és una necessitat real, sense es pot viure, però sí que pot permetre millorar tal com passa en moltes recerques científiques, que no és una cosa necessària però acaba permetent la millora de la societat.

4.3 Ambiental

Pel que fa a l'impacte ambiental del projecte al ser un projecte d'enginyeria informàtica és bastant petit respecte altres tipus de projectes. Per exemple un ordinador consumeix una potència màxima d'uns 125W, si pensem que durant tot el projecte són 575h, com s'especifica a 3, en total seria un consum de 71,9KWh. Per comparar-ho, una nevera té un consum de potència d'uns 250W, i està sempre encesa, que implica unes 8760 hores l'any i per tant un consum anual d'uns 2190KWh.

A part d'això s'ha de sumar el consum de l'espai de treball, de les companyies que subministren internet o serveis al núvol i molts altres factors. Però generalment el consum és mínim i per tant no s'ha plantejat reduir-lo.

L'únic que tindria un consum elevat seria els Data Centers dels serveis al núvol, aquests però gestionen dades per molta gent i per tant individualment acaba essent un consum petit, a més que funcionen amb energia elèctrica i que per tant pot provenir d'energies renovables o nuclears, tot i que això seria millorable.

Capítol 5

MPEG-G

5.1 Que és MPEG?

MPEG són les sigles de *Moving Picture Experts Group* [72], en català Grup d'experts en imatges en moviment. Es tracta d'un grup de treball format per ISO i IEC per tal de definir estàndards per compressió i transmissió d'àudio i vídeo. La seva primera reunió va ser el 1988 i ha anat creixent amb diferents indústries, universitats i institucions d'investigació fins a l'actualitat, que es realitzen reunions trimestrals de més de 500 persones i per tant és un dels grups de treball més grans de món. La designació oficial de l'MPEG és ISO/IEC JTC1/SC29 /WG11. [139]

Estàndards

MPEG ha dissenyat diferents estàndards, que permeten que es pugui descodificar independentment de com estigui implementada la codificació o descodificació. Uns dels més importants són:

- **MPEG-1** [73] - Grup d'estàndards de codificació d'àudio i vídeo de l'any 1996. El més conegut és el *MPEG-1 audio layer 3*, conegut com a MP3. [9]
- **MPEG-4** [74] - Grup d'estàndards de codificació d'àudio i vídeo de l'any 1999. El més conegut és la part 14, conegut com a MP4 i publicat el 2003. [71]

I ja més actuals trobaríem:

- **MPEG-DASH** [72] - Iniciat l'any 2011. És un estàndard que permet l'streaming en alta qualitat sobre HTTP adaptant-se a les capacitats de cada servidor-usuari.
- **MPEG-H** [72] - Grup d'estàndards de codificació d'alta eficiència i Media Delivery en entorns heterogenis de l'any 2013.

5.2 Que és MPEG-G?

MPEG-G [112] [75] es tracta d'un estàndard ISO [45] (ISO/IEC 23092) adreçat a la representació d'informació genòmica desenvolupat pel grup de treball MPEG. Aquest estàndard pretén permetre centralitzar tots els formats que trobem actualment en un estàndard. [72]

També ofereix privacitat i seguretat i el seu principal avantatge és la seva reducció d'espai respecte altres, ja que es basa a no repetir les dades que siguin iguals en totes les mostres i així obté la millora com es pot veure a la figura 5.1.

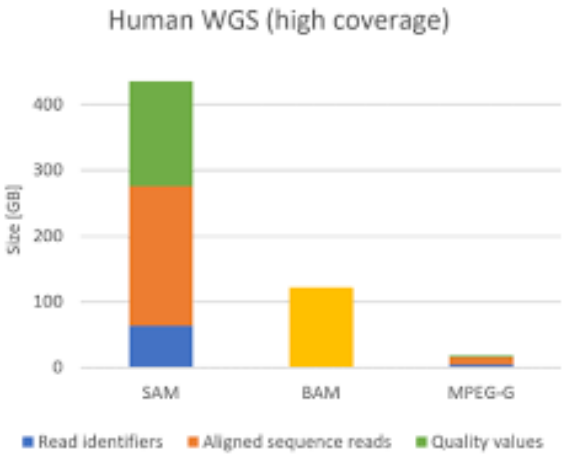


Figura 5.1: Espai utilitzat per fitxers segons [4].

Actualment disposa de 5 parts i s'està treballant en una sisena. La part tractada en aquest projecte és la *Part 3: Metadata and application programming interfaces (APIs)* [48] tot i que també fa referència a les dues primeres parts: *Part 1 - Transport and storage of genomic information* [46] i *Part 2 - Coding of genomic information* [47].

5.3 Estructura fitxer MPEG-G

L'especificació de l'estàndard dels fitxers MPEG-G fa que tinguin una organització que es podria dir jeràrquica, ja que s'organitza en diferents nivells i grups. Els grups que trobem són els següents:

- DatasetGroup
- Dataset
- AccessUnit
- DescriptorStream

El *DatasetGroup* (DG) és el contenidor de Datasets, com es pot veure a la figura 5.2. Aquest té 'box' de protecció i metadata. La 'box' de protecció del DG és la que conté les dades necessàries per, juntament amb la *Key* necessària, desencriptar les 'boxes' dels Datasets, com es veu a la figura B.1 i B.2.

El *Dataset* (DT), que és el contenidor d'*AccessUnits* i/o *Descriptor Streams*. També té les 'boxes' de protecció i metadata. La de protecció s'utilitza per desencriptar les boxes del nivell inferior com es veu a la figura B.2 i B.3. Finalment ja al nivell de les dades dins el DT, trobem *l'accessunit* (AU) i els *DescriptorStreams* (DS). Aquests són els que permeten l'accés a les dades. A la imatge 5.3 es veu com accedeixen els AU, on a diferència dels Descriptor Stream,

els AccessUnit tenen més informació i que per tant no s'ha de consultar externament, com si passa als Descriptor Streams (DSC mode) que no tenen headers i per tant han de consultar informació externa.

Pel que fa a les 'boxes' de protecció de cada nivell a l'apartat 5.4 s'especifica que contenen cada una.

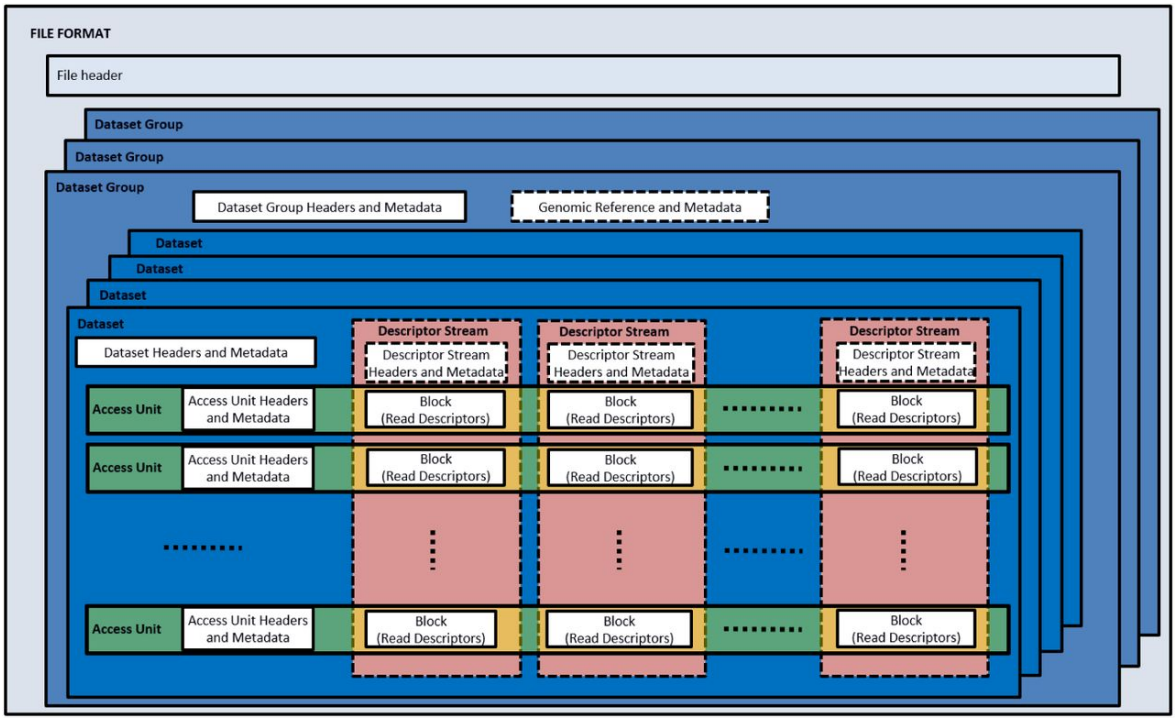


Figura 5.2: Estructura fitxer MPEG-G. [4]

.

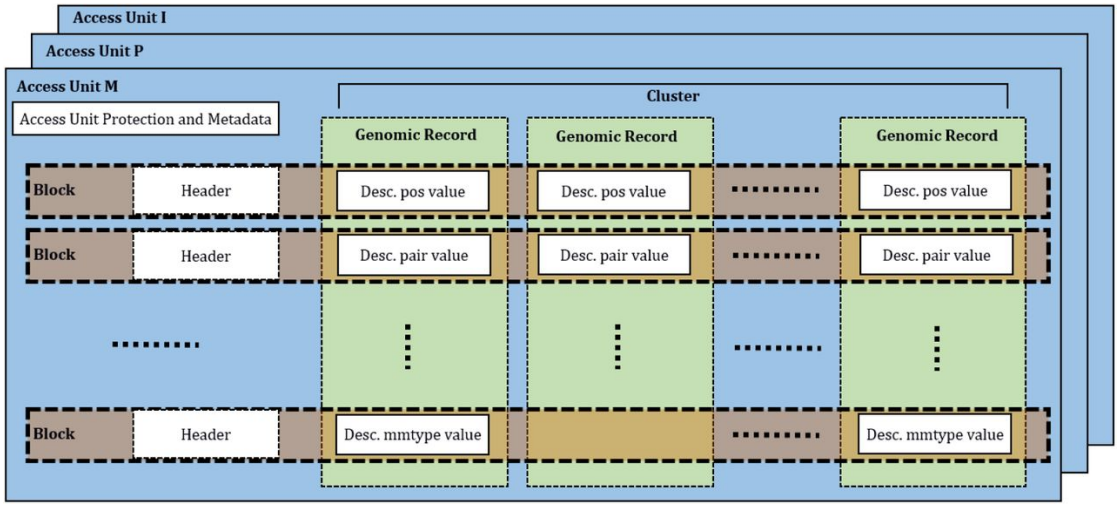


Figura 5.3: Estructura d'un AccessUnit de MPEG-G. [4]

.

5.3.1 KLV

Les 'boxes' de protecció, així com les dades, estat organitzades amb el que seria una estructura d'arbre utilitzant KLVs. Els KLV [138] són estructures d'*encoding* que indiquen la clau, longitud i el valor. En MPEG-G hi ha els següents que es veuen a la taula 5.1.

Key	Valor
rfmd	Reference Metadata
aupr	AccessUnit Protection
dtpr	Dataset Protection
dtmd	Dataset Metadata
dgpr	DatasetGroup Protection
dgmd	DatasetGroup Metadata
dspr	DescriptorStream Metadata
dshd	DescriptorStream Header
pars	encoding_parameters()
auin	AccessUnit Information Box
auhd	AccessUnit Header

Taula 5.1: Keys utilitzats en l'especificació de MPEG-G [48]. (Elaboració pròpia)

Per les dades no s'utilitza cap Key concreta.

5.4 Protecció

Per poder dur a terme l'encriptació dels gen-info (*boxes* i *blocks*), com s'ha dit a l'apartat anterior, s'utilitzen les 'protection boxes'. Aquestes estan definides al document [48] com un XSD Schema [2] [118]. Per poder dur a terme les operacions a aquestes 'boxes' hi ha altres esquemes definits com els de transport de claus o localitzacions a encriptar que es descriuen a continuació, igual que les 'boxes'. La utilització d'aquestes s'explica a l'apartat 5.5.

5.4.1 Keys

Pel que fa als KeyTransport són els objectes que contenen les claus (*Wrap*) o amb els que s'aconsegueix la clau (*Derivation*). Per aconseguir la clau és necessari que s'enviï una altra clau o KeyTransport que permet aconseguir la clau del KeyTransport actual. Els KeyTransport estan definits com a la figura 5.4 on conté el nom i després un dels tres tipus que s'especifiquen més avall.

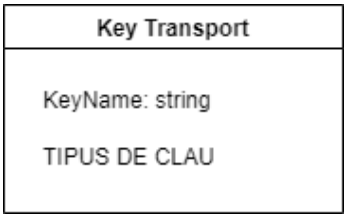


Figura 5.4: Estructures de KeyTransport. (Elaboració pròpia)

El *Wrapping* es pot veure a la figura 5.5, on l'objecte *Key Wrap* és el KeyTransport actual, s'agafa el nom de la clau per deswrapejar (kek a Symmetric i publicKeyName a Asymmetric) i s'aplica l'algoritme d'unwrapping al paràmetre wrappedKey per aconseguir la clau.

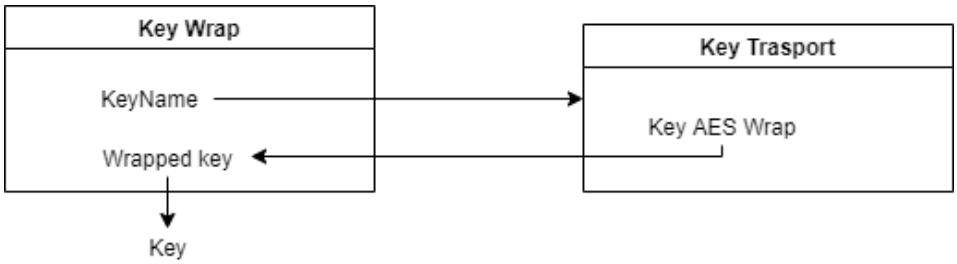


Figura 5.5: Key Wrapping. (Elaboració pròpia)

El derivation es veu a la figura 5.6 on l'objecte Key Transport actual és el Key Derivation de la figura, i només conte la informació necessària per aconseguir la clau a través d'una master key, l'anomenada *Derived key* a la figura.

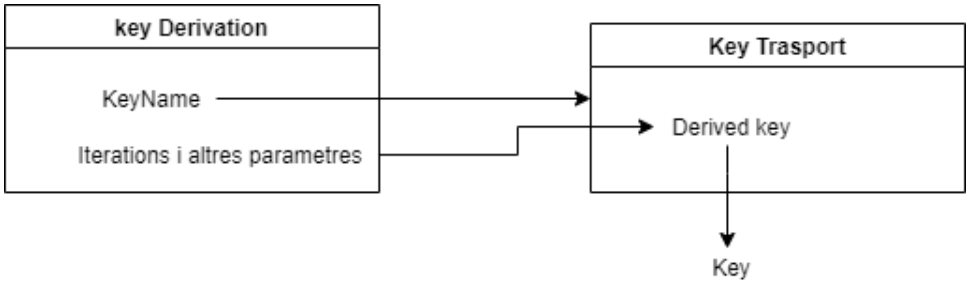


Figura 5.6: Key Derivation. (Elaboració pròpia)

A la definició de MPEG-G hi ha els següents tipus de KeyTransport definits.

- KeyAsymmetricWrap (figura 5.7): es tracta d'un *key wrap* que utilitza RSAES-OAEP [70] i per tant té els paràmetres necessaris, com la funció de hash (SHA-2), funció de mask (MGF que ha de ser una de SHA-2 [1]) i la wrappedKey i PublicKeyName.

- KeySymmetricWrap (figura 5.8): es tracta d'un *key wrap* que utilitza AESWrap [103] i té els paràmetres necessaris, com la wrappedKey i kek.

- KeyDerivation (figura 5.9): es tracta de la key Derivation com diu el nom i per tant te el Salt, Iteracions, lenght i PRF, a part del passwordName. Per fer derivation utilitza l'algortime PBKDF2 [58].

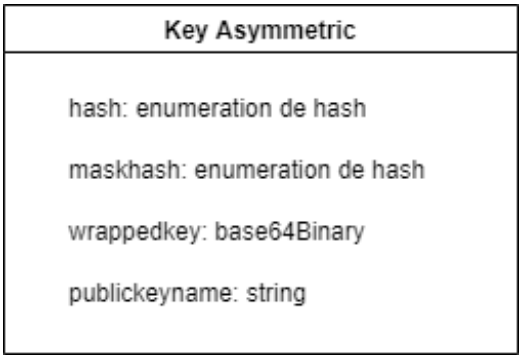


Figura 5.7: Tipus KeyAsymmetric. (Elaboració pròpia)

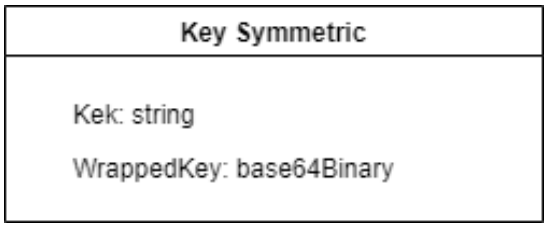


Figura 5.8: Tipus KeySimmetric. (Elaboració pròpia)

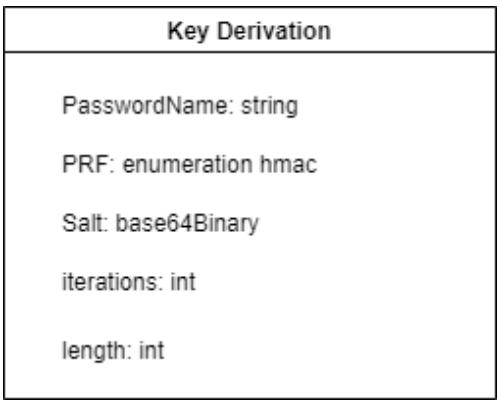


Figura 5.9: Tipus KeyDerivation. (Elaboració pròpia)

5.4.2 Localització

Per tal de saber la localització a desencriptar i les dades necessàries per fer-ho, hi ha les estructures EP (EncryptionParameters) per les 'boxes', i AUEP (AccessUnitEncryptionParameters) per AccessUnit.

Les dades que contenen es poden veure a la figura 5.10. I en el cas d'EP són el cipher, que és un valor del cipherURI i és el que s'utilitza per desencriptar la 'box'; el keyName, que és el nom de la clau que s'ha d'utilitzar amb el cipher; IV, que s'utilitza en tots els ciphers; TAG, que només s'utilitza en AES-GCM; encryptionLocation, que és l'URI de la 'box' que

es vol desencriptar o encriptar; i finalment el configurationID, que en el cas de DatasetGroup i Dataset no fa res, només és necessari en el cas que l'URI apunti a un AccessUnit, on aquest serà el necessari per unwrapejar la clau equivalent amb ConfigurationID de la llista de WrappedKeys d'AccessUnitEncryptionParameters.

Pel que fa a AUEP també té el cipher però conte IV i TAG de tipus *in* i *block*. El *block* hi és present sempre i l'*in* només quan hi ha un information block. Per tant si es fa servir AES-CTR per un lloc on no hi ha *information block* només s'utilitzaria aublockIV. També hi trobem el wrapped i configurationID que és la clau a deswrapejar per desencriptar el block i per deswrapejar la clau s'utilitza l'algoritme de RFC3394 [103] AES-Wrap i la clau del nivell superior que apunta a l'accessunit i que té el mateix configurationID que aquí.

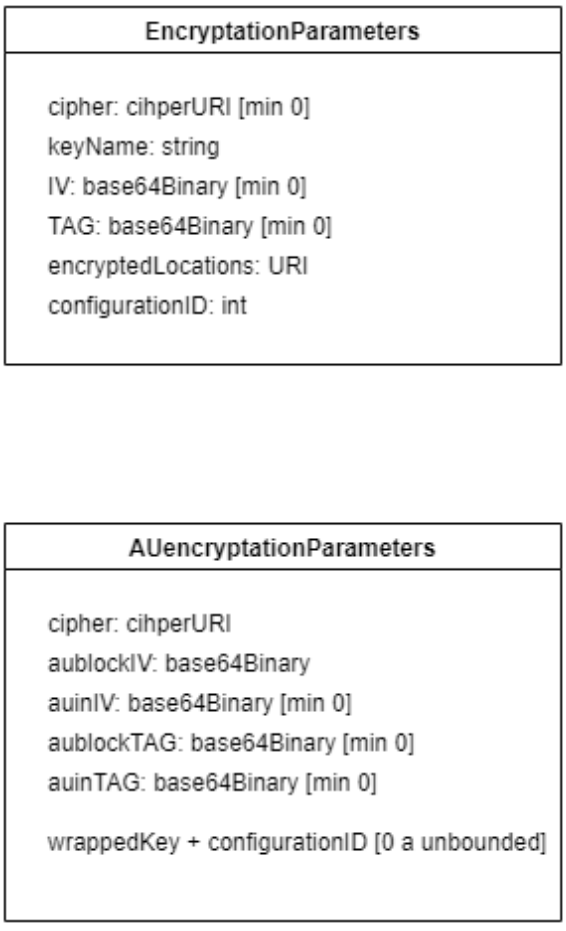


Figura 5.10: Estructura d'EncryptionParametrersType i AccessUnitEncryptionParametersType. (Elaboració pròpia)

Per tal de localitzar les posicions en què es vol accedir s'utilitzen URIs [11] [140], la definició de quina apunta a cada cosa es troben a la part 3 de MPEG-G [48] per les 'boxes' de protecció i metadata als seus respectius apartats.

5.4.3 Boxes

Les 'boxes', igual que les estructures anteriors, estan definides amb un XSD Schema i per tant es defineixen en XML [16]. Aquestes 'boxes' contenen les EP, Signatures per validar la integritat i la política d'accés a les dades.

En referència a la Signature està definit a la part 3 de MPEG-G [48] quines dades s'utilitzen per fer la firma i després validar.

Pel que fa a la Policy s'utilitza XACML per definir qui pot fer cada cosa.

La 'box' de protecció del DatasetGroup, que veiem a la figura 5.11, conté la política de privacitat (Policy) del DatasetGroup, els signatureParameters, que són les firmes dels Datasets com està especificat a [48] i pot anar de 0 fins a les necessàries en funció dels Dataset que hi hagui i que s'hagi realitzat la signatura. També trobem els EP que n'hi pot haver des de 0 fins al nombre de Datasets que hi hagui, on cada un la seva EncriptedLocation apuntaria a un Dataset del DatasetGroup. Finalment també hi ha els KeyTransportAES que serien els KeyTransport necessaris per realitzar el cipher d'EP, i amb diferents formes de *wrapping* segons usuaris que ho hagin de llegir.

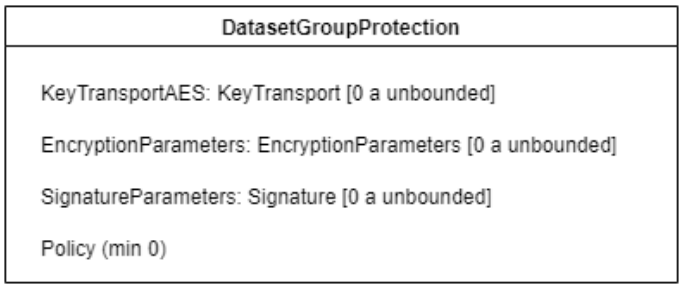


Figura 5.11: 'Box' de protecció del DatasetGroup. (Elaboració pròpia)

En el cas del DatasetGroup és molt igual, però en lloc de gestionar el nivell de Datasets gestiona el d'AccessUnits. Es pot veure a la figura 5.12.

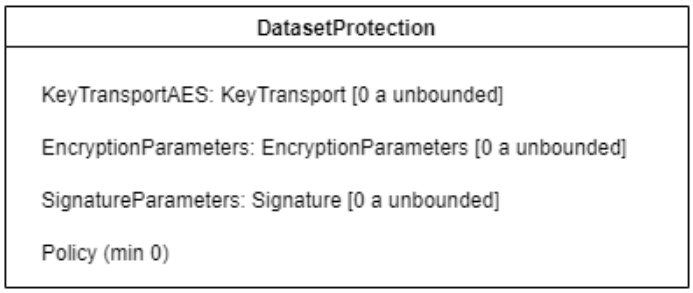


Figura 5.12: 'Box' de protecció del Dataset. (Elaboració pròpia)

Finalment trobem la 'box' de protecció de l'AccessUnit, que sí que és una mica diferent, ja que en lloc d'EP utilitza AUEP i només disposa d'això i de SignatureParameters, ja que la clau com s'ha dit s'obté del nivell superior i s'identifica amb el configurationID.

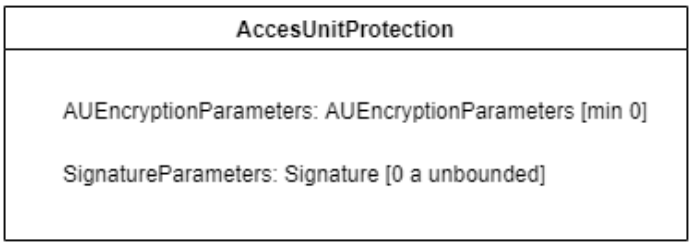


Figura 5.13: 'Box' de protecció de l'AccessUnit. (Elaboració pròpia)

5.4.4 XACML

Per definir la política de privacitat del fitxer s'utilitza XACML [28].

XACML és un estàndard que serveix per definir polítiques d'accés en format XML i per tant es defineixen unes regles que quan es processa una petició s'analitzen per saber si està autoritzada o no. Un dels objectius de XACML, com en tots els estàndards és trobar un punt comú perquè tothom utilitzi el mateix i així tenir interoperabilitat entre les implementacions de control d'accés de múltiples fabricants.

5.5 Criptografia

Amb les 'boxes', EP i keys ja podem utilitzar l'algoritme per desencriptar, que serà al que s'indica a CipherURI, el paràmetre dins l'EncryptionParameter, i que es veu a la figura 5.14.

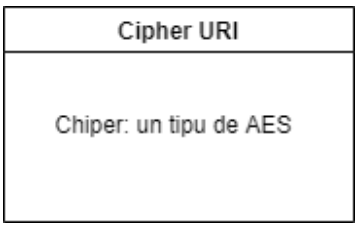


Figura 5.14: Estructures de CipherURI. (Elaboració pròpia)

Els valors disponibles de cipher són els que estan a la taula 5.2.

Algoritme	**Mides**
AES-CTR	128, 192, 256
AES-GCM	128, 192, 256

Taula 5.2: Ciphers disponibles a MPEG-G [48]. (Elaboració pròpia)

Amb aquest cipher que es troba a l'EP, el qual l'EncryptedLocation concorda amb l'URI. A aquests EP es troba també el nom de la clau a utilitzar amb el Cipher. Aquesta clau s'ha de trobar a KeyTransportAES de la 'box', i per deswrapejar la clau o aconseguir-la es fa amb una altra clau que pot venir externament. Pot haver-hi més d'una clau per una mateixa EncryptedLocation i que tu només tinguis accés a una. A part d'això també s'utilitzen els paràmetres necessaris segons si és GCM o CTR.

En el cas de GCM [64] [23] s'utilitza TAG i IV, mentre que en el cas de CTR [24] només és necessari l'IV.

D'aquesta manera amb la clau, el cipher i paràmetres i amb les dades encriptades a la que apunta aplicant l'algoritme i si tot és correcte es pot desencriptar les dades i aconseguir la 'box' tal com es veu a la figura 5.15.

Per encriptar seria el mateix però a l'invers.

El cas de les dades finals és diferents, ja que s'ha d'utilitzar l'AccessUnit per accedir-hi. En concret no s'utilitza EncryptedParameters com en els altres casos sinó AccessUnitEncryptionParameters, que igual que EncryptedParameters té el cipher i els paràmetres, que en aquest cas són més, ja que hi ha els paràmetres pel *Block* i per l'*Information Block* com ja s'ha dit a l'apartat Localització. A part d'això la resta varia una mica. La clau no es troba en un KeyTransport sinó que es troba directament la clau *wrapped* en una llista i es deswrapeja la clau que concorda amb el configurationID del nivell superior, i amb la clau també del nivell superior, com ja s'ha dit a l'apartat Localització. Amb aquestes dades ja es pot procedir a desencriptar com es pot veure a la figura 5.16.

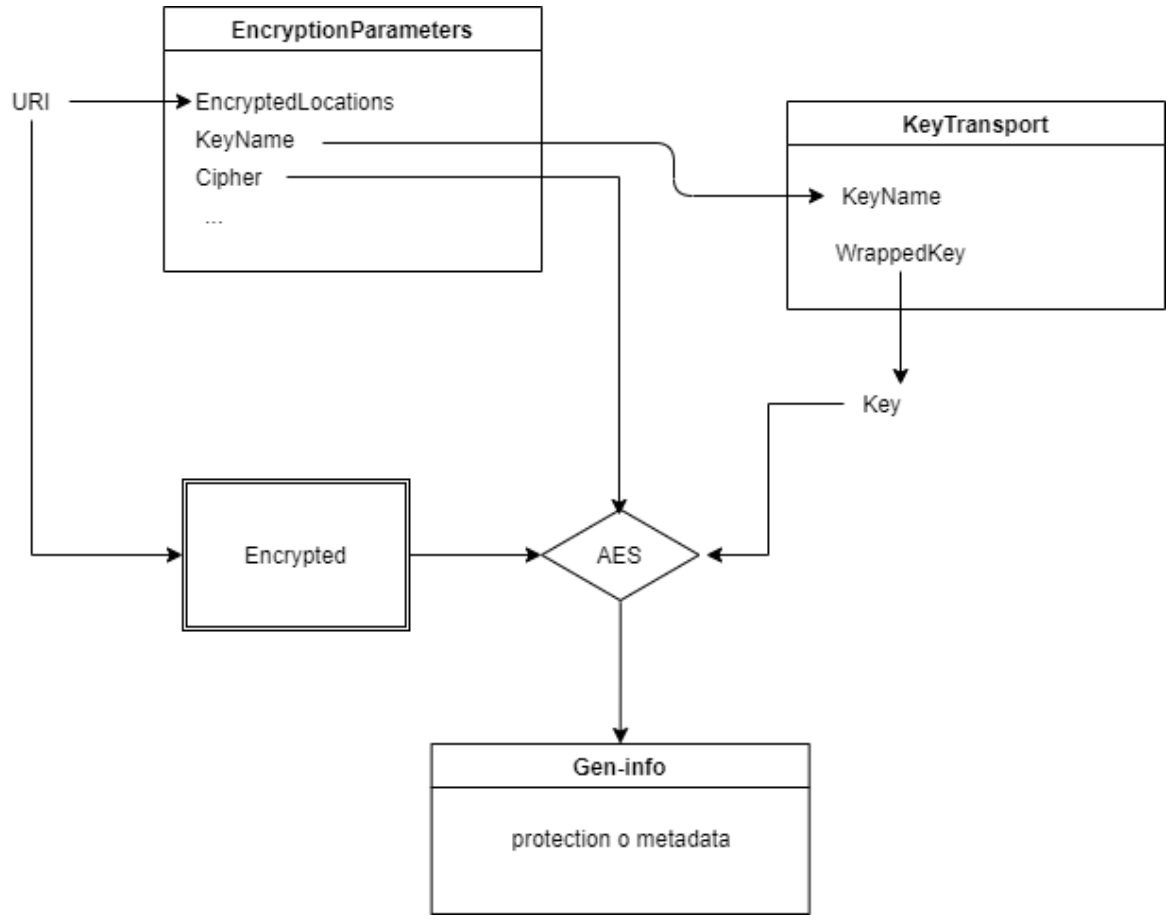


Figura 5.15: Esquema encriptar i desencriptar de les 'boxes'. (Elaboració pròpia)

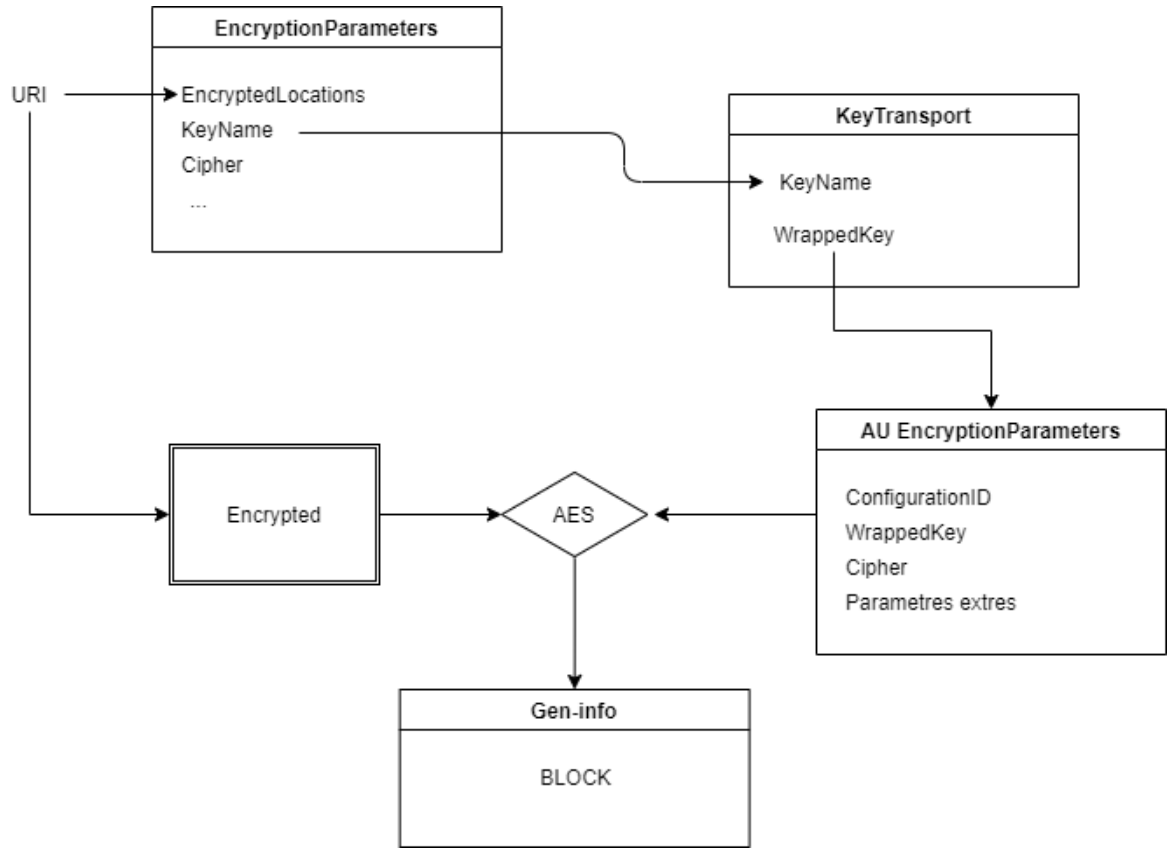


Figura 5.16: Esquema encriptar i desencriptar dels blocs. (Elaboració pròpia)

5.5.1 AES - Advanced Encryption Standard

AES és un sistema d'encriptació de clau simètrica, creat per Joan Daemen i Vincent Rijmen com a proposta pel concurs plantejat pel National Institute of Standards and Technology, a finals del segle XX, sent el guanyador del mateix.

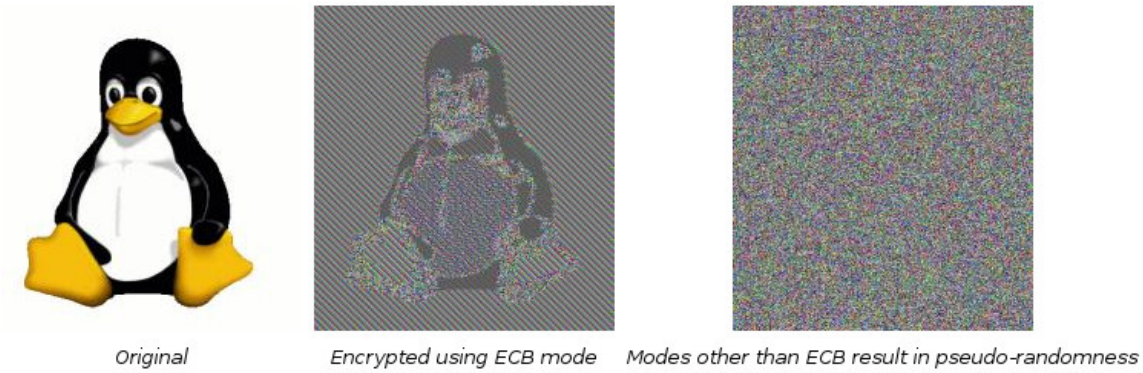


Figura 5.17: Comparació modes encriptació AES. [137]

Disposa de diferents modes d'encriptació, tots ells basats en l'encriptació per blocs i clau

simètrica. El més senzill és ECB i que es podria considerar equivalent a un algoritme de flux igual que en el cas de CTR.

En general els modes clàssics són els de la figura 5.18 i on es pot veure el nivell de seguretat de cada un i altres paràmetres. El cas de la seguretat es pot veure clarament a la figura 5.17 on un es veu la imatge encriptada amb ECB i després amb un dels modes altres modes amb seguretat 'high'.

Metrics-Modes	ECB	CTR	CBC	CFB	OFB
Security	Low	Medium	High	High	High
parallelism	Yes	Yes	No	No	No
Decryption	Yes	Now	Yes	No	No
Random access	Yes	Yes	No	No	No
Rapidity	Yes	Yes	No	No	No
Complexity	No	Now	Yes	Yes	Yes
Error propagation	No	Now	Yes	Yes	Yes
Implementation cost	Low	Low	High	High	High

Figura 5.18: Modes AES i la seves característiques. [60]

En el cas de GCM no apareix a la taula, ja que no és dels modes clàssics, ja que a més d'encriptació ofereix autentificació de missatge. A part de GCM, també hi ha altres modes com CCM, EAX o CWC, però com es veu a la figura 5.19 GCM és el més recomanable. A més el mode GCM permet aconseguir grans rendiments i canals de comunicació d'alta velocitat amb maquinari barat.

Comparison of 4th generation schemes

	CCM	EAX	CWC	GCM
Provably secure?	✓	✓	✓	✓
Unpatented?	✓	✓	✓	✓
Any length nonce?	✗	✓	✗	✓
One key?	✓	✓	✓	✓
On-line?	✗	✓	✓	✓
Can preprocess static headers/AD?	✗	✓	✓	✓
Fully parallelizable?	✗	✗	✓	✓
Preserves alignment?	✗	✓	✓	✓
Fully specified?	✓	✓	✓	✓

Figura 5.19: Modes AES amb autentificació. [19] de [109]

5.5.2 Consideracions

A part de tot el referent a l'encriptació perquè sigui segur s'han de seguir unes pràctiques per evitar problemes de seguretat. Alguns dels principals són els següents:

- No enviar les claus privades.
- No enviar claus secretes sense wrapejar o encriptar.
- No s'ha d'utilitzar un IV i Key iguals per encriptar coses diferents, sempre s'ha de variar l'IV.
- Mantenir actualitzat el software que s'utilitzi per possibles forats de seguretat.
- Enviar per canal segur les dades (TLS, SSH, etc).
- Comprovar la identitat de qui envia (Certificats).

Entre altres.

Capítol 6

GA4GH

6.1 Que és GA4GH

Global Alliance for Genomics and Health és una organització que realitza estàndards per tractar dades genòmiques. GA4GH [32] ha adoptat un estàndard que emmagatzema les dades en format BAM/SAM que es descriu breument a l'apartat 6.2, i que inicialment eren dos independents. I també té un estàndard d'encriptació anomenat crypt4gh, i desenvolupat per ells, i que es descriu a l'apartat 6.3. Però a diferència de MPEG-G els fitxers no porten la política de privacitat incorporada, sinó que suposen que l'aplicació serà l'encarregada de gestionar-ho.

6.2 BAM/SAM

SAM (*Sequence Alignment/Map format*) és un format de text *TAB-delimited* amb una capçalera, opcional i prèvia als alignments, i els alignments. Les capçaleres comencen amb '@'. Els alignment no tenen res a l'inici però tenen 11 camps obligatoris per a informació d'alineació essencial, com ara la posició de mapatge, i el nombre variable de camps opcionals per a informació flexible o d'alineació específica. [113]

Tenint l'alignment[1] de la figura 6.1. El seu SAM seria el de la figura 6.2.

```
Coor     12345678901234  5678901234567890123456789012345
ref      AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

+r001/1        TTAGATAAAGGATA*CTG
+r002         aaaAGATAA*GGATA
+r003       gcctaAGCTAA
+r004                     ATAGCT..............TCAGC
-r003                           ttagctTAGGC
-r001/2                                      CAGCGGCAT
```

Figura 6.1: Valors alignment. [113]

[1] Alineaments de Seqüències d'ADN. https://es.wikipedia.org/wiki/Secuencia_de_ADN

```
@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

Figura 6.2: SAM equivalent a la figura 6.1. [113]

6.3 Crypt4gh

GA4GH també ha estipulat estàndard de seguretat per aquestes dades, l'anomenen Crypt4gh i es pot trobar a [37]. Al document bàsicament especifiquen com s'ha d'encriptar i les capçaleres on s'especifica la versió, el magic number,etc. També les capçaleres per cada paquet entre altres. Però a la secció 3.3 del document s'especifica l'encriptació, en concret es diu els algoritmes que utilitza i la forma d'obtenció de les claus. També trobem l'apartat de desencriptar i els edit list, aquest permeten accedir a només una part de les dades desencriptades, ja que no són necessàries o per alguna altra raó.

Centrant-nos en els algoritmes que utilitzen tenim X25519 per intercanvi de claus i ChaCha20-Poly1305 com a algoritme de clau simètrica. Pel que fa a ChaCha20-Poly1305 és un algoritme de clau secreta, equivalent a AES-GCM, ja que AES seria equivalent a ChaCha20 [13] i Poly1305 és per verificació d'integritat, per tant equivalent al mode GCM.

Capítol 7

Incongruència X25519

En el document de crypt4gh [37] de GA4GH trobem que a l'apartat 3.3.1 diu el següent *"As the X25519 algorithm does not produce a completely uniform bit distribution, and many possible (Ksw, Kpr) pairs can produce the same output, the Diffie-Hellman key is hashed along with the two public keys to produce the final shared key. The hash function used to do this is Blake2b, as described in [RFC7693]."*.

Perquè un algoritme sigui segur ha de complir que tingui una distribució binomial de 0,5 entre tots els seus bits i que no hi hagui dependències entre ells, i a causa del fet que sembla estrany que una corba tan utilitzada com la X25519 [12] es digui això, ja que se suposa que ha de ser criptogràficament segur, va sorprendre, i per això es va decidir realitzar un seguit de scripts per comprovar-ho.

7.1 Scripts

Per poder comprovar la distribució de bits es van realitzar quatre scripts (anomenats scripts tot i no ser en bash ni un script 100%, ja que l'únic que realitzen és una crida a una llibreria i analitzen els resultats). Aquests es poden trobar a l'annex A.3 amb el Readme.md corresponent per poder localitzar els fitxers dins el directori.

Els Scripts es volien utilitzar per refutar o acceptar el que es diu sobre X22519 a crypt4gh de GA4GH i es va decidir realitzar amb Python, per variar del llenguatge utilitzat en altres parts d'aquest projecte.

Pel que fa a les llibreries criptogràfiques es van utilitzar les que es troben a [87].

Script 1

L'script 1 agafa una clau A i realitza un keyAgreement amb una clau B per aconseguir la sharedKey i el mateix amb la mateixa A i una C, realitzant dues comprovacions dels bits a cada iteració per contar el nombre de zeros que apareixen a cada bit. El nombre d'iteracions no està definit sinó que es para l'script en rebre un signal.

Per fer les comprovacions es van realitzar 4 execucions de l'script amb diferents iteracions.

Script 2

En aquest script el que es pretenia era fer l'anàlisi del nombre de zeros a cada bit, es fa com a l'script 1 però amb la diferència que les iteracions són un nombre fix i només es realitza amb

2 claus i no 3. A més es va executar 9 vegades per obtenir diferents resultats. Cada valor és el nombre de zeros que apareixen a cada bit durant l'execució.

Script 3

En aquest script el que es pretenia era fer l'anàlisi del nombre de zeros a cada bit, com als altres però amb la diferència que també es calcula la clau fent el blake2b amb l'OR i concatenat la sharedkey i les dues públiques com diu el document de crypt4gh. Al document no queda molt clar com s'ha de fer si amb OR o concatenat, per això es fa de les dues formes, tot i que posteriorment he pogut aclarir que es tracta de concatenar.

L'script es va executar 9 vegades, on cada valor és el nombre de zeros que apareixen a cada bit durant l'execució. Tot i això, aquesta segona part de blake2b [39] [88] [15] no va quedar correctament, ja que resultats erronis possiblement per un error a la llibreria i per tant no s'ha tingut en compte.

Script 4

En aquest script a més de fer un anàlisi com a l'script2 s'analitza les dependències entre bits, per tant el nombre de bits que és igual amb la resta a cada iteració i que el resultat es pot veure en una matriu el notebooke de R de l'annex A.3.

7.2 Màquina virtual

Per tal de poder executar l'script durant un bon temps era necessari tenir una màquina engegada. La forma més fàcil era una màquina virtual. I aprofitant que Amazon ofereix AWS Educate [10] es va triar aquesta opció. Per tal de posar en marxa la màquina virtual va ser necessari aprendre com funciona Amazon Web Services. Per començar i un cop dins AWS es va entrar a EC2, que és el servei de màquines virtuals d'Amazon, un cop dins com es veu a la figura 7.1 es va definir una nova instància com es veu a 7.2. Un cop iniciada la instància des d'un terminal s'accedia a través de ssh i també es copiava l'script mitjançant scp. Per tal de poder executar l'script era necessari nacl [77], ja que s'utilitza una llibreria, en concret la de *publicKey*. Ja finalment un cop acabada la instal·lació es podia procedir a executar els scripts, i mitjançant la bash command *top* es podia veure com s'estava executant l'script Python.

7.3 Estadística

Per tal de valorar els resultats obtinguts i per tant contrastar si el que es comenta sobre la corba X25519 és cert o no, el que necessitàvem era saber si la distribució de bits de la clau segueix una distribució binomial amb p=0.5 [123], que seria la segura. El document de crypt4gh diu que no segueix una distribució uniforme [119] de bits, per tant, que no segueix una binomial amb p=0.5 amb una desviació estàndard petita.

Per poder realitzar això es van agafar les dades resultants de cada bit dels scripts i amb R es va fer l'anàlisi utilitzant t.test i qqnorm [121]. L'anàlisi es va fer amb una normal, ja que sabem que si una distribució binomial té suficients punts la podem tractar com una normal [81].

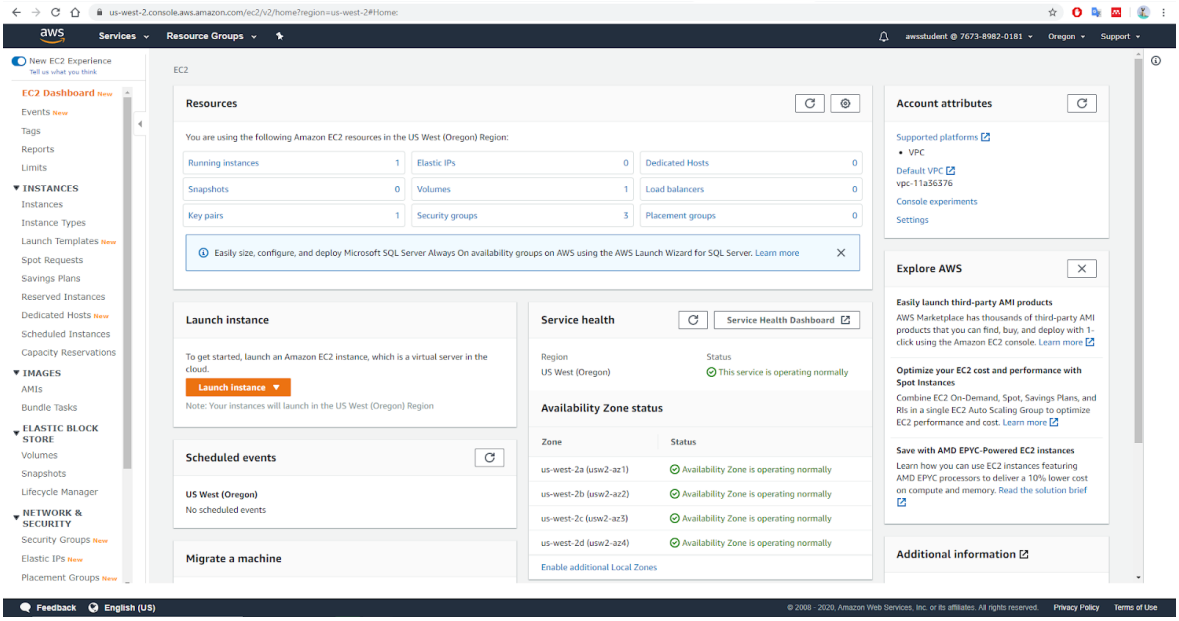


Figura 7.1: AWS Panel. Captura de https://aws.amazon.com/

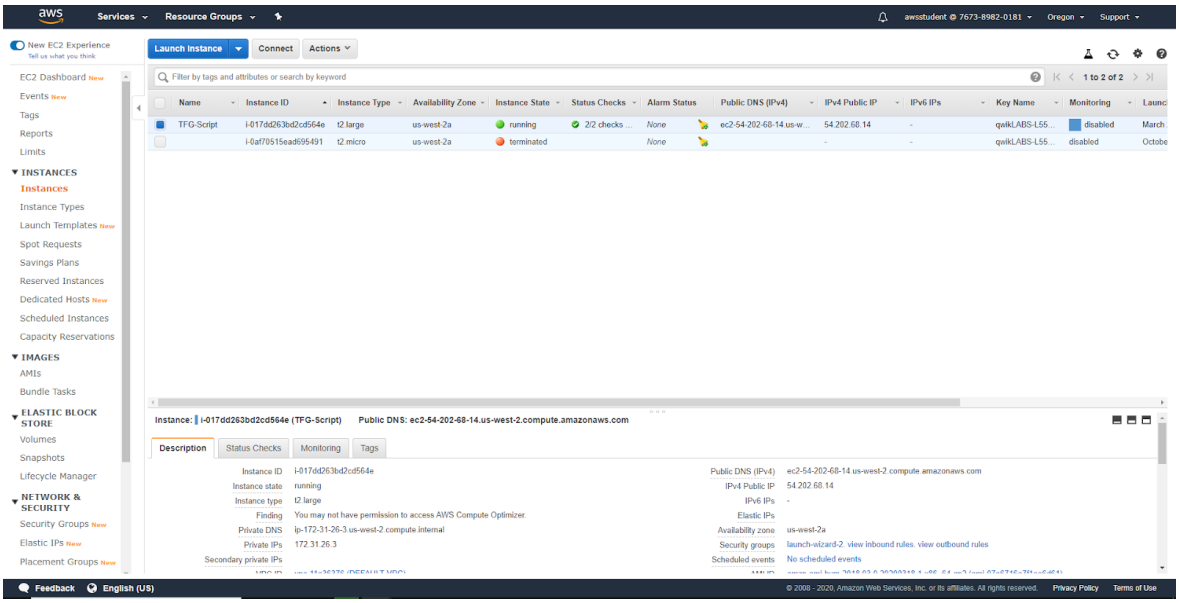


Figura 7.2: Menú instàncies AWS. Captura de https://aws.amazon.com/

Al t.test és on hi trobem el p-value, el qual si té un valor molt baix permet refutar la hipòtesi nul·la, que en aquest cas era que la distribució seguia una normal. Per aquest test es van haver d'estandaritzar les dades (mitjana a 0), ja que si no el p-value no és correcte. També es va haver de mirar les gràfiques de normalitat, on es pot observar si s'aproximen a una normal o no, ja que el p-value amb depèn de quines distribucions no és del tot correcte [122].

7.4 Resultats

Amb els resultats de diverses execucions, i que es poden veure o obtenir des de l'annex A.3, i després d'executar els notebooks de R s'obtenia que els p-value en tots els casos, tant en anàlisi de zeros com de dependències, eren 1. Per tant no podem refutar la hipòtesi nul·la però tampoc acceptar-la, ja que el p-value només ens permet refutar.

A continuació, i per tal de tenir més evidencies es van realitzar comprovacions gràfiques, que inclouen QQ Plot, histogrames o blox plots. A la figura C.1 trobem el QQPlot de les dades obtingudes amb l'script 1 a la 4 execució, que és de les que més iteracions va fer. A la gràfica es veu que els punts segueixen la normal excepte molt pocs outliders que es desvien molt poc.

També veiem l'histograma a la figura C.2 on clarament la gran part de dades està al centre, tenint en compte que el nombre d'execucions de l'script va ser 94785920, i per tant el nombre de zeros està al 50 per cent, a més que també es veu que segueix una normal.

I ja a la figura C.3 podem veure el QQplot de la shared key amb la segona clau, on veiem que també s'aproxima molt una normal. Per tant la clau A, tant amb B com C genera una distribució de bits normal i amb desviació estàndard molt petita, com es pot veure a l'annex A.3 on hi ha tots els resultats.

Als resultats de l'script 2 de la figura C.4, en el seu histograma també veiem unes dades molt semblants a una normal. El mateix passa amb la figura C.5 d'un qqplot de les dades de l'script 3 o a l'histograma de les mateixes dades a la figura C.6.

Pel que fa a les dependències entre bits, a l'annex A.3 a l'script 4 es veu la matriu amb els bits iguals on tots són aproximadament la meitat de vegades igual entre ells.

7.5 Conclusions

Un cop observat els resultats, on tots s'obté p-value = 1, que s'aproximen a una distribució normal i l'interval de confiança és molt petit i amb molta probabilitat i la desviació estàndard és molt petita, que per tant que totes les dades estan centrades i que també veiem que no hi ha dependències entre bits, podem afirmar que el que es diu al document de crypt4gh molt possiblement és fals, ja que no ho podem afirmar al cent per cent.

Capítol 8

Crypt4gh a MPEG-G

L'objectiu principal del treball era mirar la interoperabilitat entre estàndards i per tant mirar si es podien ajuntar. Per fer-ho es va escollir afegir encriptació de crypt4gh a MPEG-G.

La raó és que actualment MPEG-G [72] no inclou els algoritmes d'encriptació que utilitza crypt4gh i es podrien afegir per permetre un ventall més ampli d'opcions. En concret crypt4gh utilitza ChaCha20-Poly1305 [80] [100] com a algoritme de clau simètrica, X25519 [61] per encriptació de clau publica i Blake2b [102] com a hash; mentre que MPEG-G utilitza AES [103], RSA [70] i SHA-2 [1] respectivament.

8.1 Motivació

L'objectiu d'afegir és disposar de més algoritmes a MPEG-G i a la vegada poder tenir fitxers encriptats de la mateixa forma en cas de treballar amb els 2 tipus, cosa que també es busca amb el *mapping* de l'API. Pel que sabem també l'algoritme de crypt4gh (ChaCha20-Poly1305) és més ràpid en dispositius mòbils i IoT que no pas AES [106] [78] i per exemple és el que utilitza Google en TLS 1.2 [29] o TLS 1.3 [101] (segons el que suporti el navegador) per Android [18].

8.2 Esquema

Per tal d'afegir aquesta opció només cal modificar l'especificació, ja que es tracta d'un estàndard i no d'una implementació.

Bàsicament, per tal d'aconseguir que es pugui realitzar els nous mètodes i tenint en compte l'especificació de MPEG-G s'ha de modificar el següent:

- Afegir un KeyTransport per corbes el·líptiques. [115]
- Afegir a cipher les opcions de CipherURI.
- Afegir *Nounce* a EncryptionParameters i AUEncryptionParametes.
- Afegir l'explicació de les dades necessàries i com recuperar les claus.

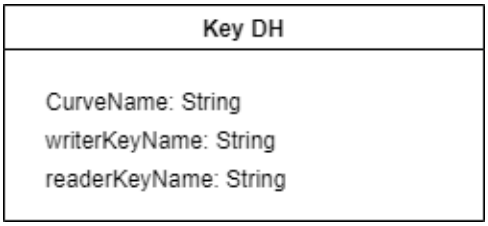


Figura 8.1: Esquema Box KeyTransport DH. (Elaboració pròpia)

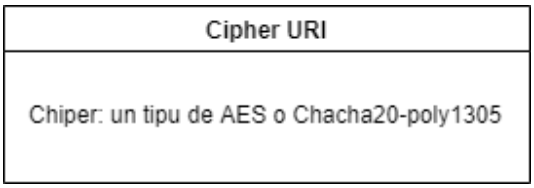


Figura 8.2: Afegit cipher ChaCha20-Poly1305. (Elaboració pròpia)

8.2.1 KeyTransport

Per tal de poder realitzar la recuperació de la clau com es mostra a l'apartat 8.3 és necessari definir un nou KeyTransport, en concret el que es veu a la figura 8.1, on trobem el nom de la corba i les claus del reader i writer. Una la publica i l'altre la privada.

8.2.2 Cipher

Per tal de poder utilitzar el ChaCha20-Poly1305 com a cipher hem d'afegir el cipher a CipherURI com es veu a la figura 8.2 i afegir el *Nounce* a EP i AUEP com es veu a la figura 8.3. Aleshores si s'utilitza Cipher ChaCha20-Poly1305 s'haurà d'utilitzar l'IV i el *Nounce*.

8.3 Criptografia

En el moment de definir el fitxer i per tant el keyTransport per writer ha de posar el nom de la seva clau publica i generar la keyShared utilitzant la seva privada i posar el nom de la clau publica del lector. Un cop el lector ho rebi haurà d'utilitzar la clau publica del writer i la seva clau privada relacionada amb la readerKeyName per obtenir la sharedKey i per tant, la clau del KeyTransport.

Aquest sistema per transportar claus és més un estil KeyDerivation que no pas KeyWrap, ja que no hi ha la clau emmagatzemada sinó la forma d'aconseguir-ho igual que el derivation, tot i que en concret és KeyExchange. També es podria afegir blake2b que serveix per hash i també com hmac [116]

8.4 Contribució MPEG-G

Amb tot l'exposat prèviament es va realitzar un document de contribució a MPEG, en l'estàndard MPEG-G. Aquest document es troba a l'annex A.2 i ha estat la base perquè el director del projecte, com a membre de MPEG, prepares el document per dur a terme una contribució oficial a MPEG-G.

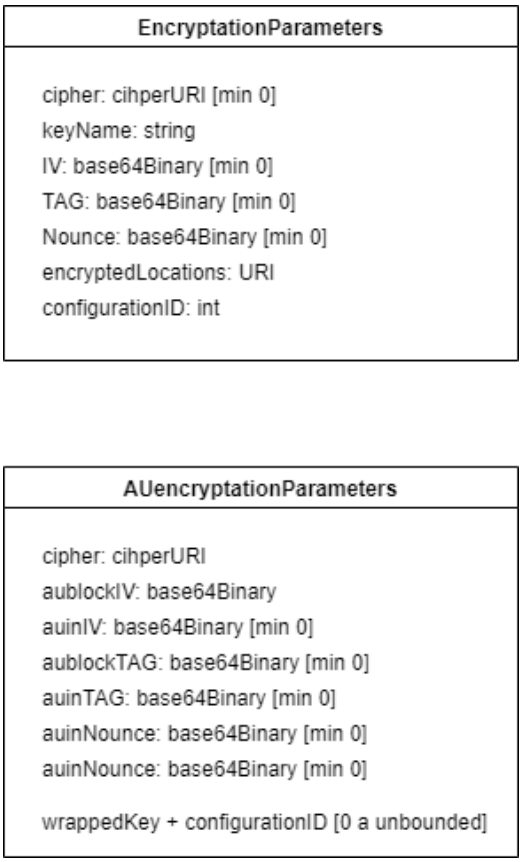


Figura 8.3: EP i AUEP afegint *Nounce* per ChaCha20-Poly1305. (Elaboració pròpia)

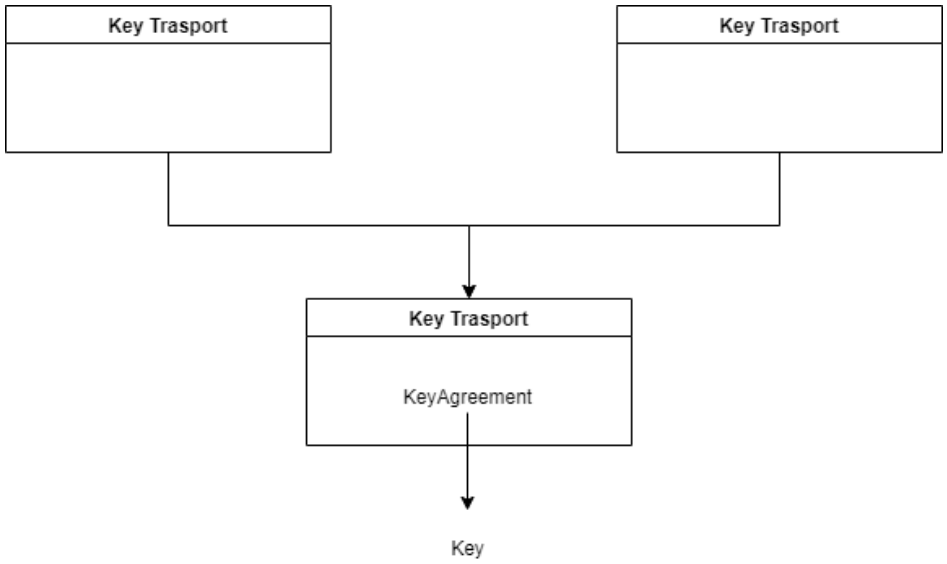


Figura 8.4: Esquema obtenir KeyTransport DH. (Elaboració pròpia)

Capítol 9

Prototip MPEG-G

Per tal de dur a terme l'especificació de MPEG-G amb les noves opcions d'encriptació es va optar per realitzar un prototip de la Part 3 de MPEG-G [48], el referent a seguretat i metadata, per tal de poder entendre més el funcionament i comprovar els algoritmes afegits.

La implementació va estar realitzada en Java. Aquesta és una simplificació de l'especificació original per adaptar-se a les necessitats del projecte i utilitzant només coses rellevants.

Per tant s'ha fet el següent diferent respecte l'especificació de MPEG-G part 3:

- L'AccessUnit només té dades, no l'information block.
- No s'utilitza *Policy*.
- Només hi ha un block per cada AU. I per tant només un signature paràmetre a AU.
- WrappedKey d'AccessUnit només té un valor i sense estar wrapejat. S'ha canviat respecte a l'XML [118] i és en format Key de java i només un valor en lloc de List.
- ConfigurationID no s'utilitza, al només haver-hi una wrappedKey. Es segueix una mica el mètode 'Revised' de la tesi d'en Daniel Naro [79] apartat 7.2.
- KeyTransportAES només té una clau per EncryptedLocation, quan en realitat n'hi poden haver més segons els diferents de wraps que es vulgui.
- No s'ha implementat el mode *DSC*.
- S'envien claus privades tot i que no s'hauria de fer.

Altres comentaris a tenir en compte:

- Finalment Blake2b no s'ha implementat com a hash ni hmac per la dificultat de trobar llibreries en condicions.
- S'utilitza la versió de Java 11 però amb dependències de java 6/8 per JAXB [50] perquè l'11 no ho suporta.

- El Javadoc no està detallat completament pel fet que la implementació software no era el principal objectiu del projecte.

Pel que fa als paràmetres per encriptar s'ha de tenir en compte el següent: *Note that GCM mode has a uniqueness requirement on IVs used in encryption with a given key. When IVs are repeated for GCM encryption, such usages are subject to forgery attacks. Thus, after each encryption operation using GCM mode, callers should re-initialize the cipher objects with GCM parameters which have a different IV value.*

En el cas de ChaCha20 és igual però amb el Nounce.

A la figura 9.1 es veu la forma en què es va decidir implementar. Utilitzant keys auxiliars pels valors de tot el DatasetGroup, Dataset i AccessUnit, aquestes són 'dgva', 'dtva' i 'auva' respectivament. Amb això saps la longitud de les dades de tota aquella zona i per tant facilita la implementació.

I finalment al només haver-hi un block a cada AccessUnit no s'ha definit cap *key* auxiliar.

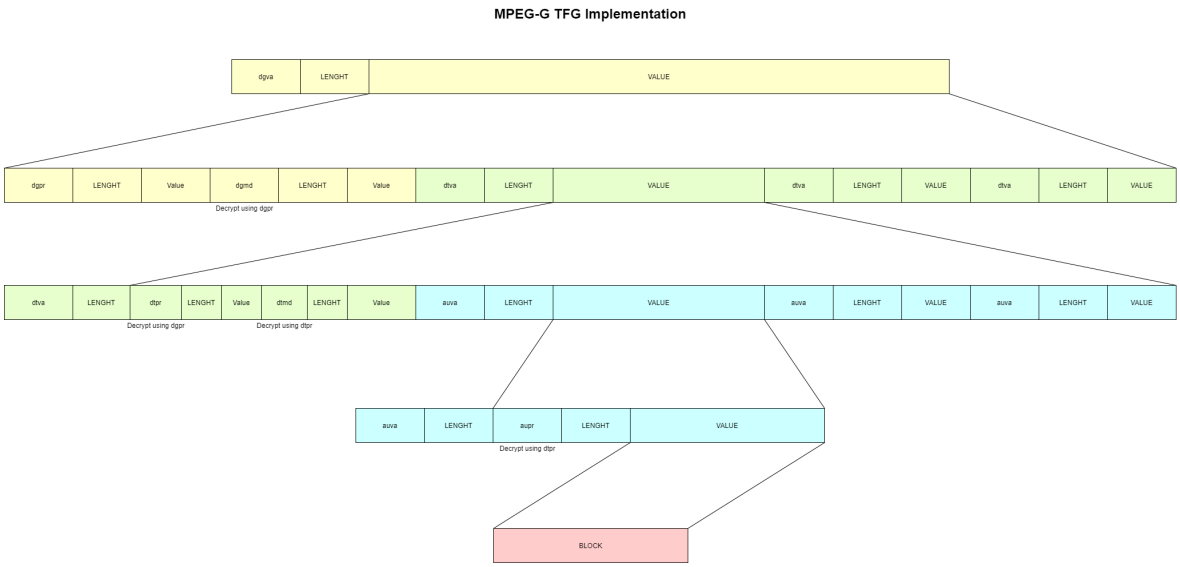


Figura 9.1: Diagrama implementació fitxer MPEG-G. (Elaboració pròpia)

9.1 Especificació

Per tal de dur a terme la implementació es va realitzar un esquema UML inicial, el qual no s'ha modificat a posteriori ni realitzat un final al no ser un dels objectius del projecte i només utilitzar-se per la implementació d'un prototip.

A la figura D.1 i D.2 es veuen les classes inicials, que posteriorment hi ha hagut algunes diferències. El més destacable és que no s'utilitza AccessUnit ID, sinó que es fa a través de ReferenceType que diu la posició dins el cromosoma. I també s'han afegit les claus 'Simple'.

9.2 Implementació

Finalment seguint el que s'havia definit a l'especificació personal es va realitzar la implementació, aquesta es pot trobar a l'annex A.4 i també es troba el javadoc [53] a l'annex A.5.

En general les funcions que s'ha utilitzat es troben a [85] i en concret tot el referent a criptografia [54]. I tenim la informació necessària referent a criptografia a la documentació d'Oracle [108] [52] i les recomanacions de mides de clau i paràmetres al document d'ENISA [5].

9.2.1 Encriptació

Per tal d'encriptar i desencriptar una box d'un nivell es va decidir fer des d'una box d'un nivell superior i per tant com es veu a la figura 9.2 on per desencriptar s'agafa els bits de la 'box' encriptada i que apunta la localització a desencriptar, i utilitzant els paràmetres es desencripta i es retorna la 'box' corresponent. Per encriptar és a la inversa i es veu a la figura 9.3, s'agafa la box, s'aplica el que es troba a EP i es retornen els bytes de la box encriptada.

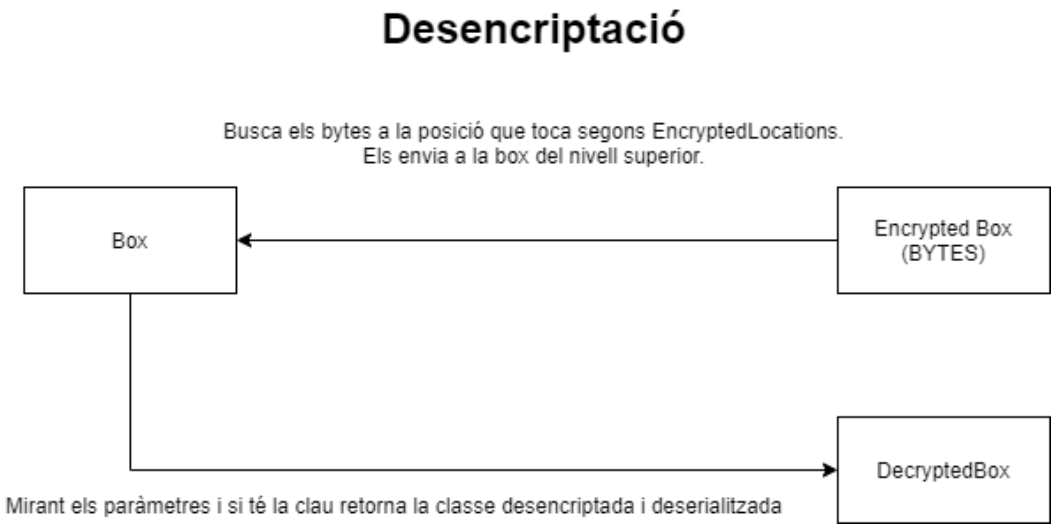


Figura 9.2: Procediment implementació desencriptació. (Elaboració pròpia.)

Al moment d'encriptar les boxes i dades del block teníem dues opcions, utilitzar SealedObject [104] i serialitzar [107] o serialitzar i utilitzar cipher. Una per encriptar agafa l'objecte, el converteix en un SealedObject amb els paràmetres necessaris i per tant és l'objecte encriptat i després el serialitza, i l'altre respectivament agafa l'objecte, el serialitza i aplica el cipher als bytes serialitzats. Per desencriptar el que utilitza SealedObject agafa els bytes, desserialitza i després desencripta el SealedObject, mentre que amb cipher es desencripta els bytes i després es desencripta l'objecte. Es poden veure les opcions a la figura 9.4. Finalment es va optar per l'opció de no utilitzar SealedObject sinó encriptar directament, ja que per cipher no utilitza un altre objecte sinó els bytes i això per guardar-ho quan està encriptat és més fàcil, a més que encripta tots els bytes i no es veu si és un objecte serialitzat.

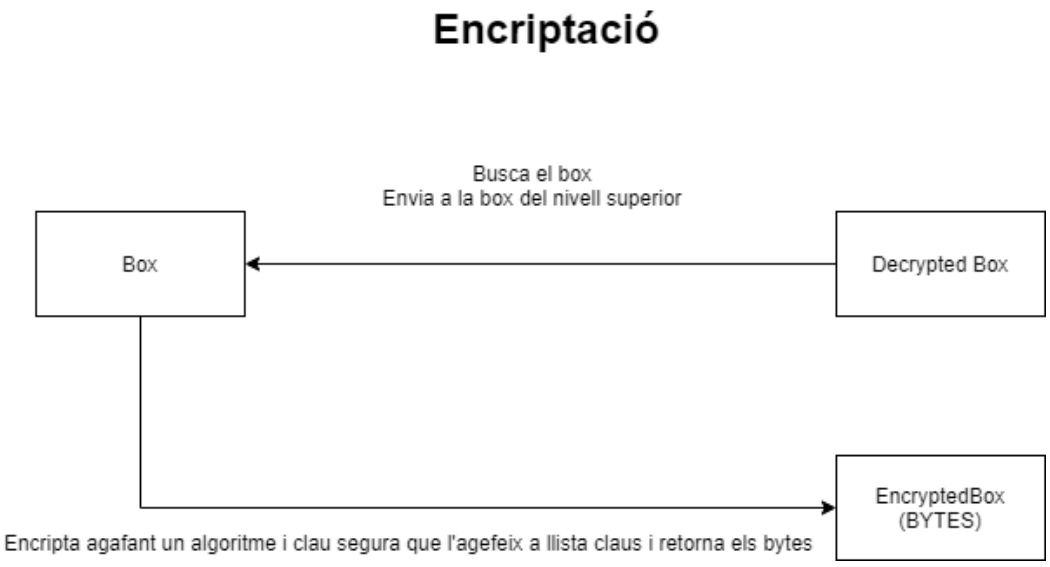


Figura 9.3: Procediment implementació encriptació . (Elaboració pròpia.)

9.2.2 Classes

Per tal de fer la implementació i com ja s'ha vist a l'UML es va fer mitjançant classes. Les classes implementades finalment són les següents i a les que s'adjunta una petita descripció.

Boxes

Pel que fa a les boxes de protecció i metadata s'han implementat les classes de la figura E.1 que són les següents.

- DatasetGroupProtectionType és la classe que defineix la 'box' de protecció d'un DatasetGroup. Té els 4 paràmetres definits al XSD Schema i les funcions necessàries per gestionar els EP i per encriptar i desencriptar les 'boxes' dels Dataset, entre altres funcions.
- DatasetProtectionType és la classe que defineix la 'box' de protecció d'un Dataset. Té els 4 paràmetres definits al XSD Schema i les funcions necessàries per gestionar els EP i per encriptar i desencriptar les 'boxes' dels AccessUnit, entre altres funcions.
- AccessUnitProtectionType és la classe que defineix la 'box' de protecció d'un AccessUnit. Té les funcions necessàries per gestionar l'encriptació i desencriptació dels blocks.
- ReferenceType és la classe encarregada de gestionar les posicions i dades que contenen els AccessUnit, amb la posició i nom.
- DatasetGroupMetadataType és la classe que permet definir la metadata dels DatasetGroup. Bàsicament conte el seu ID.
- DatasetMetadataType és la classe que permet definir la metadata dels Dataset. Bàsicament conte el seu ID.

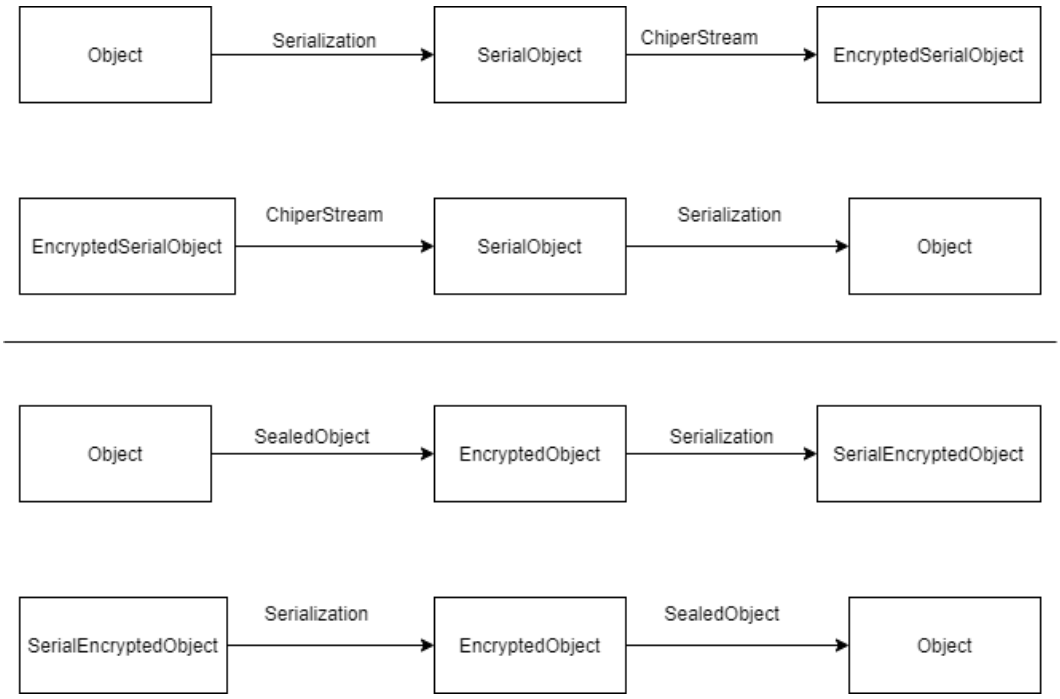


Figura 9.4: Diferencia entre *sealedObject* i sense. (Elaboració pròpia)

Data

Pel que fa als diferents nivells són les classes de la figura E.2 que són les següents.

- Dataset és la classe que representa un Dataset. Té les seves 'boxes' amb les dues opcions, si estan encriptades o no. Un array dels AccessUnit que conté i encryptedLocations que són els References dels AccessUnit encriptats. A més és la classe que gestiona els AccessUnit i evita que n'hi hagi més d'un amb Reference que s'intercalin o altres gestions.
- DatasetGroup és la classe que representa un DatasetGroup. Té les seves 'boxes'. Un array dels Dataset que conté. A més és la classe que gestiona els Dataset i altres gestions.
- AccessUnit és la classe que permet l'accés als blocs. En aquesta implementació només conte un bloc, però a la realitat més d'un. S'encarrega de desencriptar i encriptar les dades i permetre llegir. A més conte la informació del Reference.
- Block és la classe encarregada de guardar els bytes de dades.
- MPEG-G és la classe per gestionar un fitxer MPEG-G. D'aquesta manera amb un main pots executar diferents classes MPEG-G i per tant treballar amb diferents fitxers alhora.

Enumeracions

Pel que fa a enumeracions trobem de la figura E.4 que són les següents.

- CipherURIType és l'enumeració que defineix els ciphers disponibles, en aquest cas són els definits per MPEG-G i l'afegit, el ChaCha20-Poly1305.
- DadesType és l'enumeració que defineix els tipus de dades que pot apuntar un URI.
- WrapType és l'enumeració que defineix els tipus de wrap, els que estan definits a MPEG-G i l'afegit, l'EC.

Claus

Pel que fa als KeyTransport trobem les classes de la figura E.6. Bàsicament és un KeyTransport amb diferents tipus de claus amb els seus paràmetres. S'han definit els Simple i SimplePublic per poder simular el fet de quan s'enviarien les claus per un tercer canal. També s'ha definit un keysList per gestionar les claus que s'enviarien per aquest segon canal.

Tipus

Pel que fa als tipus paràmetres que tenen les boxes de protecció s'han implementat les classes de la figura E.5 que són les següents.

- AccessUnitEncryptionParametersType defineix els paràmetres i la seva gestió.
- EncryptionParametersType defineix tots els paràmetres necessaris a l'especificació dels EP i que s'específica com es defineix més endavant.
- SignatureType permet gestionar les signatures i comprovacions d'integritat necessàries. Per aquesta classe s'ha hagut de sobreescriure el hashCode a altres per comprovar si són iguals els valors i no pas posició de memòria, etc.
- PolicyType seria la classe a on es definiria la política de privacitat, al no utilitzar-se en aquesta implementació està buida.

Utilitats

Pel que fa a utilitats s'han implementat les classes de la figura E.3 que són les següents.

- URI, que és la classe per apuntar al gen-info, amb els ids dels DatasetGroup i Dataset i Reference per poder saber quin AccessUnit utilitzar. Aquest té les funcions per definir on apunta.
- BytesUtilities és una classe on s'han definit funcions útils a diferents classes per poder gestionar els bytes, en concret disposa de les funcions per sumar byteArrays, buscar KLV en una byteArray i serialitzar i desserialitzar, entre altres.
- KLV és la classe que permet la simulació de l'estructura d'un KLV.
- KeyGens es generen les claus o parells de claus de forma segura utilitzant paràmetres segurs, s'utilitza KeyGenerator de java [59].
- Pair és la classe equivalent al Pair de C++.
- CryptoForms és la classe que conté les funcions necessàries per encriptar. S'explica més avall amb més detall.

9.2.3 Main

El codi del main bàsicament el que es fa és crear el fitxer (classe MPEG-G), a ell afegeix alguns blocs de prova amb valors Random, i al mateix moment es guarden els URIs.

A continuació es creen URIs per les 'boxes' que s'han creat al fer blocs, Datasets i DatasetGroups i la generació de claus necessàries. Un cop es té això es posen els EP i finalment s'encripta tot i s'escriu el fitxer.

Després també es llegeix el fitxer per comprovar que va correctament.

9.2.4 Codi

El codi es pot trobar a l'annex A.4. Però algunes de les funcions o parts de codi més importants són els següents.

KLV

A continuació veiem com està fet el KLV, la seva definició és molt fàcil, ja que només té la K (key), L (longitud) i V (valor) i per tant el constructor només necessita la Key i el Value per generar el KLV calculant la lenght.

```
// Key del value.
protected String K;
// Lenght del value.
protected Integer L;
// Value del KLV.
protected byte[] V;


/**
 * Crea un nou objecte de KLV.
 * @param k Key del KLV.
 * @param v Value del KLV.
 */
public KLV(String k, byte[] v) {
    this.K = k;
    this.L = v.length;
    this.V = v;
}
```

URI

Pel que fa a l'URI bàsicament s'ha definit un id pel DG, un pel DT i un per l'AU i finalment el tipus de dades al que apunta, si és una 'box' de protecció o metadata o si són dades.

En cas d'estar a Dataset i voler apuntar a un AU al moment de crear el KLV l'id de DG és indiferent. El mateix passa amb la localització dins EncryptedParamaters, només es miren els valors que apunten a nivells inferiors, mai els d'un nivell superior.

```
// Id del DatasetGroup.
protected Integer dg_id;
// Id del Dataset.
protected Integer dt_id;
```

```
5 // Tipu de dades. O Metadata, Protection o Dades.
6 protected DadesType tipus;
7 // En cas d'apuntar a dades (accesunit i block) tambe ha de tenir ref per
      indicar l'acces Unit
8 protected ReferenceType ref;
```

Afegir EP

Per tal d'afegir els valors d'EncryptedParameters i fer-ho fàcil per l'usuari al main s'ha definit aquesta funció. En concret les mides com és Chacha a 256, AES a 256; també el tipus de hash el sha256 i en el cas de Derivation paràmetres estan definits automàticament. També es fa perquè es generen valors aleatoris segurs i així s'evita l'error de l'usuari en definir-ho. Com és el cas de IV i TAG i Nounce en cas de ChaCha20-Poly1305 que també es defineixen amb secureRandom com es veu a les línies 5, 10 i 32. També es genera aleatòriament el nom del keyTransport que es retorna, com es veu a la línia 12.

La funció a més torna el keyTransport que en el cas de la 'box' de protecció ha d'afegir a KeyTransportAES. Per això a la 'box' de protecció hi ha una funció que afegeix la clau després de fer la crida.

Aquesta funció només s'ha d'utilitzar a localització que estigui desencriptada, ja que si està encriptada es sobreescriu i no es podria desencriptar.

```
1  public KeyTransportType setNewForEPValues(CipherURIType cipher,
       KeyTransportType wrapper, URI id, WrapType wrapmode, KeyTransportType
       Eliptic2) throws Exception {
2      this.cipher = cipher;
3      byte[] IV = new byte[128/8];
4      SecureRandom random = new SecureRandom();
5      random.nextBytes(IV);
6      this.iv = IV;
7      if(cipher == CipherURIType.AES_256_GCM || cipher == CipherURIType.
       AES_128_GCM ||cipher == CipherURIType.AES_192_GCM) {
8          byte[] ta = new byte[128/8];
9          random.nextBytes(ta);
10         this.tag = ta;
11     }
12     this.keyName = random.toString();
13     this.encryptedLocations = id;
14     KeyTransportType kt = null;
15     if(cipher.equals(CipherURIType.CHACHA_20_POLY_1305)) {
16         switch (wrapmode) {
17             case EC:
18                 kt = new KeyEllipticCurveType(this.keyName,wrapper.getKeyName()
       , Eliptic2.getKeyName());
19                 break;
20             case Simetric:
21                 kt = new KeySymmetricWrapType(this.keyName,wrapper,nouChacha
       (256));
22                 break;
23             case Asimetric:
24                 kt = new KeyAsymmetricWrapType(this.keyName, "SHA-256", "SHA
       -256", wrapper, nouChacha(256));
25                 break;
```

```
26              case Derivation:
27                  kt = new KeyDerivationType(this.keyName, wrapper.getKeyName());
28                  break;
29          }
30          byte[] no = new byte[12];
31          random.nextBytes(no);
32          this.nounce = no;
33      } else {
34          switch (wrapmode) {
35              case EC:
36                  kt = new KeyEllipticCurveType(this.keyName, wrapper.getKeyName
    (), Eliptic2.getKeyName());
37                  break;
38              case Simetric:
39                  kt = new KeySymmetricWrapType(this.keyName,wrapper,nouAES(256))
    ;
40                  break;
41              case Asimetric:
42                  kt = new KeyAsymmetricWrapType(this.keyName, "SHA-256", "SHA
    -256", wrapper,nouAES(256));
43                  break;
44              case Derivation:
45                  kt = new KeyDerivationType(this.keyName, wrapper.getKeyName());
46                  break;
47          }
48      }
49      return kt;
50  }
```

Funcions criptografia

Per tal de gestionar tot el tema d'encriptació i desencriptació de les 'boxes' s'han creat unes funcions perquè no s'hagi de repetir el codi. En concret una que inicialitza el cipher [17] amb els valors de cipher, la clau i si és encriptació o desencriptació, que va definit pel boolean mode, el qual si és true és per encriptar i si és fals és per desencriptar. També s'han de cridar les funcions per inicialitzar els paràmetres del cipher, com GCMParamerersSpec o IVParamatersSpec. En el cas de ChaCha20 si anés en solitari necessitaria ChaCha20ParameterSpec params però a l'anar amb Poly1305 és IVParamatersSpec.

Després també tenim dues funcions que a partir d'un objecte passen a bytes o de bytes a un objecte per quan està encriptat o quan no utilitzant el cipher i dofinal.

```
1  public static Object getObject(byte[] dades, Cipher cipher) throws IOException,
       ClassNotFoundException, BadPaddingException, IllegalBlockSizeException {
2          InputStream istream = new ByteArrayInputStream(cipher.doFinal(dades));
3          ObjectInputStream inputStream = new ObjectInputStream(istream);
4          return inputStream.readObject();
5  }
6
7  public static byte[] setObject(Object a, Cipher cipher) throws IOException,
       BadPaddingException, IllegalBlockSizeException {
8          ByteArrayOutputStream baos = new ByteArrayOutputStream();
9          ObjectOutputStream oos = new ObjectOutputStream(baos);
```

```
10          oos.writeObject(a);
11          return cipher.doFinal(baos.toByteArray());
12  }
13
14  public static Cipher initCipher(boolean mode, EncryptionParametersType EP, Key
        sks) throws Exception {
15          Cipher cipher = Cipher.getInstance(EP.getCipher().value());
16          if (EP.getCipher().equals(CipherURIType.CHACHA_20_POLY_1305)) {
17                  if (mode) cipher.init(Cipher.ENCRYPT_MODE, sks, new
        IvParameterSpec(EP.getNounce()));
18                  else cipher.init(Cipher.DECRYPT_MODE, sks, new IvParameterSpec(
        EP.getNounce()));
19          } else {
20                  if (EP.getCipher() == CipherURIType.AES_256_GCM || EP.getCipher
        () == CipherURIType.AES_128_GCM || EP.getCipher() == CipherURIType.
        AES_192_GCM) {
21                          if (mode) cipher.init(Cipher.ENCRYPT_MODE, sks, new
        GCMParameterSpec(128, EP.getIV()));
22                          else cipher.init(Cipher.DECRYPT_MODE, sks, new
        GCMParameterSpec(128, EP.getIV()));
23                          cipher.updateAAD(EP.getTAG());
24                  } else {
25                          if (mode) cipher.init(Cipher.ENCRYPT_MODE, sks, new
        IvParameterSpec(EP.getIV()));
26                          else cipher.init(Cipher.DECRYPT_MODE, sks, new
        IvParameterSpec(EP.getIV()));
27                  }
28          }
29          return cipher;
30  }
```

Box de protecció

A l'apartat de la box de protecció les funcions més destacables són les següents. En aquest cas és el de protecció del Dataset.

Tenim la funció per desencriptar el nivell inferior (AccessUnit) on agafa els EP de l'URI que es passa i amb aquests aplica els algoritmes necessaris als bytes[] enviats i comprova la firma.

Fa semblant en el cas d'encriptar per la inversa, per exemple en lloc de guardar la firma la comprova. També té una funció per buscar la key dins KeyTransportAES.

```
1  protected List<KeyTransportType> keyTransportAES;
2  protected List<EncryptionParametersType> encryptionParameters;
3  protected List<SignatureType> signatureParameters;
4  protected PolicyType policy;
5
6  protected Key getKey(String KeyName, String KeyAlgortim, int KeyType) throws
       Exception {
7      for (KeyTransportType kp : keyTransportAES) {
8          if (kp.getKeyName().equals(KeyName)) {
9              return kp.getKey(KeyAlgortim,KeyType);
10         }
11     }
```

```
12      throw new Exception("No hi ha la clau");
13  }
14
15  public Object decryptAU(URI id, byte[] aubox) throws Exception {
16      boolean signature = false;
17      Object ob;
18      if (getEP(id).getCipher() == CipherURIType.CHACHA_20_POLY_1305) {
19          ob = getObject(aubox, initCipher(false, getEP(id), getKey(getEP(id).
        getKeyName(), "ChaCha20", Cipher.SECRET_KEY)));
20      } else
21          ob = getObject(aubox, initCipher(false, getEP(id), getKey(getEP(id).
        getKeyName(), "AES", Cipher.SECRET_KEY)));
22      for (SignatureType sig : signatureParameters) {
23          if (sig.getLocation().getRef().equals(id.getRef()) && sig.getLocation()
        .getTipus().equals(id.getTipus())) {
24              signature = true;
25              if (!sig.comprovarSign(ob)) throw new Exception("Signatura
        incorrecte");
26          }
27      }
28      if (!signature) System.out.print("Atenci , box sense comprovar signatura")
        ;
29      return ob;
30  }
31
32  public byte[] encryptAU(URI id, Object aubox) throws Exception {
33      if (signatureParameters == null) {
34          signatureParameters = new ArrayList<>();
35      }
36      novaSig(new SignatureType(aubox, id));
37      if (getEP(id).getCipher() == CipherURIType.CHACHA_20_POLY_1305) {
38          return setObject(aubox, initCipher(true, getEP(id), getKey(getEP(id).
        getKeyName(), "ChaCha20", Cipher.SECRET_KEY)));
39      } else
40          return setObject(aubox, initCipher(true, getEP(id), getKey(getEP(id).
        getKeyName(), "AES", Cipher.SECRET_KEY)));
41  }
```

Dataset

Pel que fa a una classe dels diferents nivells un dels casos és el Dataset que té bastants funcions però algunes de les més importants són parsejar on utilitza KLV i segons la Key parseja a un lloc o un altre.

També té per afegir un block, on retorna una còpia de l'URI del block si no dóna algun error abans. A aquest nivell també és l'encarregat de quan s'ha d'encriptar un nivell inferior cridar a la seva box de protecció i en el cas del Dataset afegir el Reference de l'AccessUnit encriptat a la llista, mentre que quan es desencripta es retira. O les funcions per encriptar-ho tot dels seus nivells inferiors.

Pel que fa a saber si té alguna box encriptada el que fa és mirar si hi ha la box i bytes null o a la inversa. I finalment la de getBytes que retorna els bytes amb forma de KLV del Dataset.

```
1  // Cont  els bytes de la box de protecci  si est  encriptada. Sino == null.
```

```
protected byte[] encryptedBoxP;
// Cont  els bytes de la box de metadata si est  encriptada. Sino == null.
protected byte[] encryptedBoxMd;
// Box de protecci  de la classe Dataset si esta desencriptada. Sino == null.
protected DatasetProtectionType DTProtection;
// Box de metadades de la classe Dataset si esta desencriptada. Sino == null.
protected DatasetMetadataType DTMetadata;
// ArrayList amb els accesunits pertanyents al Dataset.
protected ArrayList<AccesUnit> accesunits;

protected void parse(byte[] aux) throws Exception {
   while (aux != null) {
       Pair<KLV, byte[]> a = BytesUtilities.getFirstKLV(aux);
       aux = a.getValue();
       switch (a.getKey().getK()) {
           case "dtpr":
               try {
                   this.DTProtection = (DatasetProtectionType) BytesUtilities.deserialitzar(a.getKey().getV());
                   this.encryptedBoxP = null;
               } catch (Exception e) {
                   // You are converting an invalid stream
                   this.encryptedBoxP = a.getKey().getV();
                   this.DTProtection = null;
               }
               break;
           case "dtmd":
               try {
                   this.DTMetadata = (DatasetMetadataType) BytesUtilities.deserialitzar(a.getKey().getV());
                   this.encryptedBoxMd = null;
               } catch (Exception e) {
                   // You are converting an invalid stream
                   this.encryptedBoxMd = a.getKey().getV();
                   this.DTMetadata = null;
               }
               break;
           case "auva":
               addAU(a.getKey().getV());
               break;
           case "encr":
               encryptedLocations = (ArrayList<ReferenceType>) BytesUtilities.deserialitzar(a.getKey().getV());
               break;
           default:
               throw new Exception("Fitxer amb format incorrecte al parsejar Dataset");
       }
   }
}

public URI newBlock(URI uri, byte[] dades) throws Exception {
   if (uri.getRef() == null) throw new Exception("S'ha d'haver definit un Reference");
   if (antioverlapping(uri.getRef()))
       throw new Exception("Estas intentant afegir un block amb intersecci amb un existent");
```

```
53      AccesUnit aux;
54      aux = this.addAU();
55      aux.newBlock(dades, uri.getRef());
56      URI ret = new URI(uri);
57      ret.setTipus(DadesType.Dades);
58      return ret;
59  }
60
61  protected void EncryptAU(URI uri) throws Exception {
62      if (DTProtection == null) throw new Exception("Error de permisos");
63      AccesUnit a = getAU(uri.getRef());
64      if (uri.getTipus().equals(DadesType.AUProtection)) {
65          AccessUnitProtectionType aupr = a.getAUProtection();
66          byte[] res = this.DTProtection.encryptAU(uri, aupr);
67          a.setAUProtectionEncrypted(res);
68      } else if (uri.getTipus().equals(DadesType.AUMetadata)) {
69          throw new Exception("El reference no es pot encriptar, es on es troba la
        posicio");
70      } else throw new Exception("URI incorrecte per aquest nivell");
71  }
72
73  protected void DecryptAU(URI uri) throws Exception {
74      AccesUnit a = getAU(uri.getRef());
75      if (uri.getTipus().equals(DadesType.AUProtection)) {
76          byte[] aupr = a.getAUProtectionEncry();
77          AccessUnitProtectionType res = (AccessUnitProtectionType) this.
       DTProtection.decryptAU(uri, aupr);
78          a.setAUProtection(res);
79      } else if (uri.getTipus().equals(DadesType.AUMetadata)) {
80          throw new Exception("Reference mai esta encriptat");
81      } else throw new Exception("URI incorrecte");
82  }
83
84  protected void encryptAll() throws Exception {
85      for (AccesUnit auu : this.accesunits) {
86          auu.EncryptBlock();
87          URI b = new URI();
88          b.apuntarAccesUnitPr(0, 0, auu.getReference());
89          EncryptAU(b);
90      }
91  }
92
93  protected boolean isEncryptedBoxP() throws Exception {
94    if(DTProtection == null && encryptedBoxP == null) throw new Exception("Error
       box");
95    return (DTProtection == null);
96  }
97
98  protected byte[] actualitzarBytes() throws Exception {
99    byte[] pr;
100   byte[] md;
101   if (encryptedBoxP == null && DTProtection != null) {
102       pr = this.DTProtection.getKLV().toBytes();
103   } else if (encryptedBoxP != null && DTProtection == null) {
104       pr = new KLV("dtpr", encryptedBoxP).toBytes();
105   } else throw new Exception("Error amb boxes");
106   if (encryptedBoxMd == null && DTMetadata != null) {
```

```
107         md = this.DTMetadata.getKLV().toBytes();
108     } else if (encryptedBoxMd != null && DTMetadata == null) {
109         md = new KLV("dtmd", encryptedBoxMd).toBytes();
110     } else throw new Exception("Error amb boxes");
111     byte[] aux = BytesUtilities.addArrays(pr, md);
112     BytesUtilities.addArrays(aux, new KLV("encr", BytesUtilities.serialitzar(
        encryptedLocations)).toBytes());
113     return BytesUtilities.addArrays(aux, getAUBytes());
114 }
```

Keys

Pel que fa a les Key hi ha el wrap i unwrap en el cas del wrapping i constructora i getKey en el cas de Derivation i d'EC. En el cas de fer l'asymmetric és necessari OAEPParameters [82]. En concret el que fan és wrapejar la clau o deswrapejar-la obtenint la clau per deswrapejar de la llista de claus que s'enviarien per un tercer canal.

A continuació es veu la KeyAsymmetric amb OAEP.

```
1   public void WrapKey(KeyTransportType clau, Key unwrapped, String hashfu,
      String maskha) throws Exception {
2       PublicKey enc = (PublicKey) clau.getKey("RSA",Cipher.PUBLIC_KEY);
3       if(enc == null) throw new Exception("No es pot obtenir la clau per
      encriptar");
4       this.hashFunction = hashfu;
5       this.maskGenerationHashFunction = maskha;
6       this.publicKeyName = clau.getName();
7       String transformation = String.format("RSA/ECB/OAEPWith%sAndMGF1Padding",
      hashfu);
8       Cipher c = Cipher.getInstance(transformation);
9       OAEPParameterSpec par = new OAEPParameterSpec(hashfu, "MGF1", new
      MGF1ParameterSpec(maskha), PSource.PSpecified.DEFAULT);
10      c.init(Cipher.ENCRYPT_MODE, enc, par);
11      this.wrappedKey = c.doFinal(serialitzar(unwrapped));
12  }
13
14  public Key getKey(String wrappedKeyAlgorithm, int wrappedKeyType) throws
      Exception {
15      Key clau = llistaClaus.getKey(this.publicKeyName).getKey("RSA",Cipher.
      PRIVATE_KEY);
16      if (clau != null) {
17          String transformation = String.format("RSA/ECB/OAEPWith%
      sAndMGF1Padding",this.hashFunction);
18          Cipher c = Cipher.getInstance(transformation);
19
20          OAEPParameterSpec par = new OAEPParameterSpec(this.hashFunction, "
      MGF1", new MGF1ParameterSpec(this.maskGenerationHashFunction), PSource.
      PSpecified.DEFAULT);
21          c.init(Cipher.DECRYPT_MODE, clau, par);
22          byte[] aux = c.doFinal(this.wrappedKey);
23          return (Key) deserialitzar(aux);
24      }
25      return null;
26  }
27
```

```
28   @Override
29   public int hashCode() {
30       return Objects.hash(this.hashFunction, this.maskGenerationHashFunction,
       this.publicKeyName, Arrays.hashCode(this.wrappedKey),this.keyName);
31   }
```

A continuació es veu la KeyDerivation on no es guarda cap wrappedKey.

```
1  public KeyDerivationType(String nom, String password) throws
       InvalidKeySpecException, NoSuchAlgorithmException {
2      super(nom);
3      this.passwordName = password;
4      this.salt = new byte[200];
5      Random random = new Random();
6      random.nextBytes(this.salt);
7      this.iterations = 10;
8      this.length = 256;
9      this.prf = "SHA256";
10 }
11
12
13 public Key getKey(String wrappedKeyAlgorithm, int wrappedKeyType) throws
       Exception {
14     String transformation = String.format("PBKDF2WithHmac%s",this.prf);
15     SecretKeyFactory kf = SecretKeyFactory.getInstance(transformation);
16     char[] pass = Base64.getEncoder().encodeToString(llistaClaus.getKey(this.
       passwordName).getKey("AES", Cipher.SECRET_KEY).getEncoded()).toCharArray();
17     KeySpec specs = new PBEKeySpec(pass, this.salt, this.iterations, this.
       length);
18     return kf.generateSecret(specs);
19 }
```

A continuació es veu la KeyEllipticCurve on es calcula la clau a partir de dues altres.

```
1  public KeyEllipticCurveType(String nom, String Key1, String Key2) {
2      super(nom);
3      this.writerKeyName = Key1;
4      this.readerKeyName = Key2;
5  }
6
7  public Key getKey(String wrappedKeyAlgorithm, int wrappedKeyType) throws
       Exception {
8      PublicKey clau2 = (PublicKey) llistaClaus.getKey(this.writerKeyName).getKey
       ("X25519", Cipher.PUBLIC_KEY);
9      PrivateKey clau1 = (PrivateKey) llistaClaus.getKey(this.readerKeyName).
       getKey("X25519", Cipher.PRIVATE_KEY);
10     if (clau1 != null && clau2 != null) {
11         KeyAgreement ka = KeyAgreement.getInstance(this.curveName);
12         ka.init(clau1);
13         ka.doPhase(clau2, true);
14         byte[] encoded = ka.generateSecret();
15         if(wrappedKeyType != Cipher.SECRET_KEY) throw new Exception("EC nom s
       pot obtenir SecretKeys");
16         return new SecretKeySpec(encoded,wrappedKeyAlgorithm);
17     }
```

```
18      return null;
19  }
```

A continuació es veu la KeySymmetricWrap, on també es veu una cosa que també s'ha fet a algun altre punt de refer el hashcode [49] per poder comprovar la signature només dels valors.

```
1  public Key getKey(String wrappedKeyAlgorithm, int wrappedKeyType) throws
       Exception {
2      Key clau = llistaClaus.getKey(this.kek).getKey("AES",Cipher.SECRET_KEY);
3      if (clau != null) {
4          Cipher c = Cipher.getInstance("AESWrap");
5          c.init(Cipher.UNWRAP_MODE, clau);
6          return c.unwrap(this.wrappedKey,wrappedKeyAlgorithm, wrappedKeyType);
7      }
8      return null;
9  }
10
11 public void WrapKey(KeyTransportType clau, Key unwrapped) throws Exception {
12     Key enc = clau.getKey("AES",Cipher.SECRET_KEY);
13     if(enc == null) throw new Exception("No es pot obtenir la clau per
       encriptar");
14     this.kek = clau.getName();
15     Cipher c = Cipher.getInstance("AESWrap");
16     c.init(Cipher.WRAP_MODE, enc);
17     this.wrappedKey = c.wrap(unwrapped);
18 }
19
20 @Override
21 public int hashCode() {
22     return Objects.hash(this.kek, Arrays.hashCode(this.wrappedKey), this.
       keyName);
23 }
```

9.3 Test

Per tal d'assegurar el seu funcionament s'han implementat un seguir de tests que es poden veure a l'annex A.4 on es proven les funcions més importants. Aquests tests són minimalistes per tal de valorar les limitacions definides per aquest projecte. Per fer els tests s'ha utilitzat JUnit [57].

També s'ha utilitzat el Random [89], igual que al main, per posar dades de mostra i poder realitzar els tests.

9.4 Conclusions

Tenint en compte les consideracions indicades inicialment s'ha aconseguit acabar la implementació de la seguretat dels fitxers MPEG-G. Aquesta implementació té funcions que es poden utilitzar a un main per gestionar fitxers 'MPEG-G model TFG' o si s'utilitza el main existent genera un fitxer encriptat ("TFG.mpg") i un sense encriptar ("TFG2.mpg"). Si s'obre el fitxer encriptat amb un bloc de notes es veu el que hi ha a la figura 9.5.

Finalment també s'ha passat la implementació a Maven [63] per fer més fàcil la seva execució a diferents entorns.

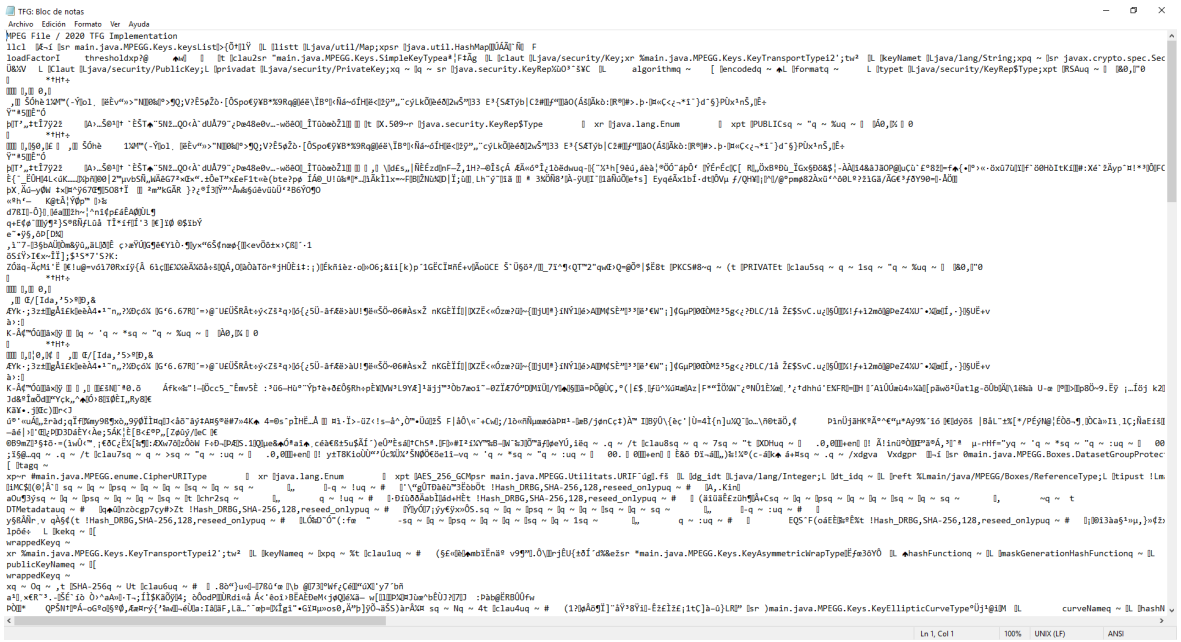


Figura 9.5: Captura de fitxer obert amb bloc de notes. (Elaboració pròpia)

Capítol 10

Interoperabilitat APIs

Pel que fa a les APIs, a l'especificació de MPEG-G part-3 hi trobem definida l'API. Per tant podríem considerar que un servidor pot tenir fitxers MPEG-G amb la seva API, equivalent a l'especificació de MPEG-G, implementada.

Per altra banda trobem que GA4GH te hts-get [41], que és una API de recuperació de dades genòmiques massives amb un model de client / servidor i que pot treballar sobre diferents proveïdors d'API de fitxers SAM, BAM, VCF, CRAM o BCF, pero no sobre MPEG-G. [43].

Tenint en compte tot això un dels objectius del projecte era fer un *mapping* que permeti accedir a un servidor amb una API MPEG-G a través de crides hts-get. A més s'ha de tenir en compte que abans de la definició de l'API de MPEG-G ja hi havia la de hts-get, i per tant moltes aplicacions possiblement funcionen amb hts-get, a més que MPEG-G té més crides i opcions que hts-get. Per totes aquestes raons el mapatge només s'ha realitzat de hts-get a MPEG-G. Per tant es tracta de passar de la figura 10.2 a la figura 10.1.

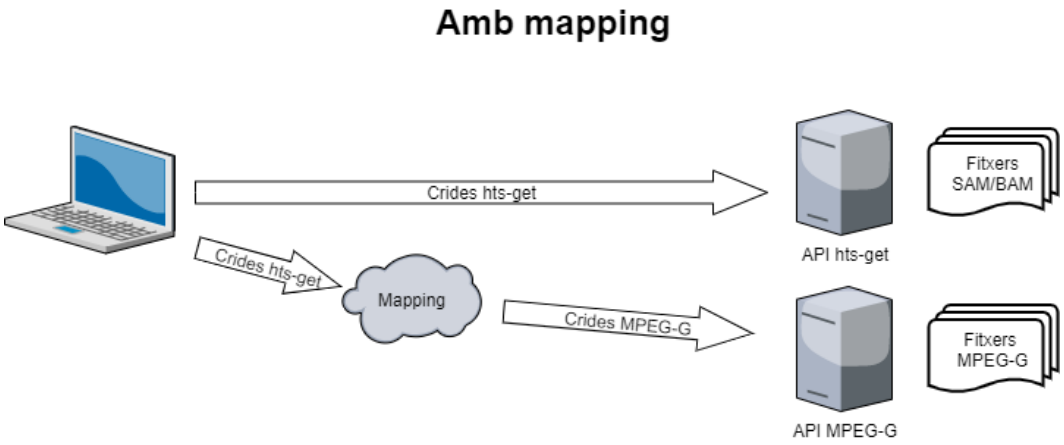


Figura 10.1: Crides hts-get a APIS. (Elaboració pròpia)

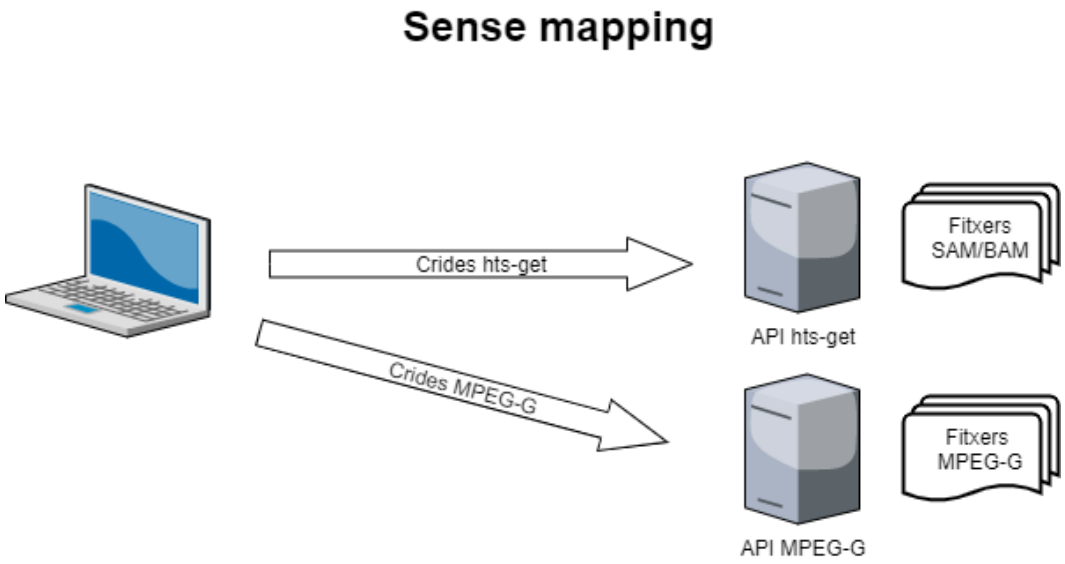


Figura 10.2: Crides a APIS sense *mapping*. (Elaboració pròpia)

Amb aquest *mapping* el client ha de saber a quin tipus de proveïdor fa la crida per definir els paràmetres, i per veure com es passa l'ID de la crida. També hi hauria l'opció de fer un *Mapping* transparent com es veu a la figura 10.3, en concret al mapatge caldria definir per defecte alguns paràmetres de l'entrada del SimpleFilter de MPEG-G, tot i que això no s'especifica amb detall a aquest projecte. I en cas de voler disposar de dos tipus de servidor el mapatge hauria de mantenir una taula per saber on es troba cada *id* i fer la crida al servidor pertinent.

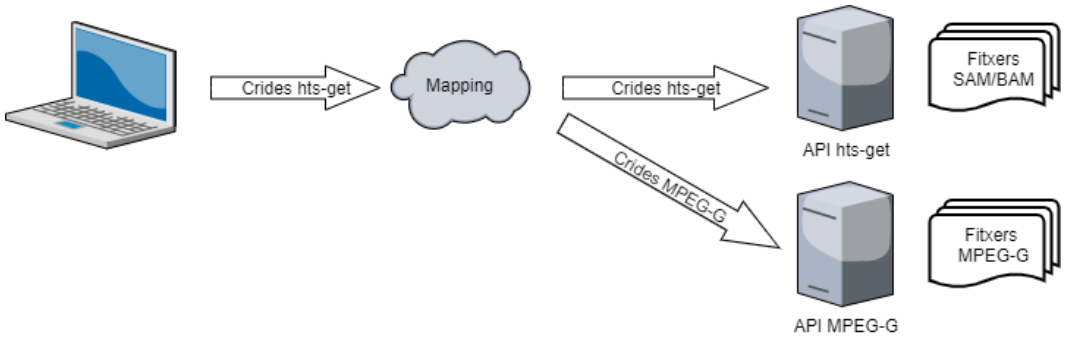


Figura 10.3: Crides a APIS amb *mapping* transparent. (Elaboració pròpia)

10.1 hts-get

Hts-get [41], és una API pensada per actuar sobre fitxers CRAM/SAM/VCF. A hts-get has de fer la crida GET que es veu a la figura F.2 amb els paràmetres segons especifica la mateixa figura F.2 i que es troben amb més detalls a [41]. Els ids hts-get no te'ls diu, sinó que els has d'obtenir d'alguna forma externa. Hts-get respon amb un *ticket*, que és el que es veu a la figura F.1, amb aquests pots agafar els urls i demana les dades, aquest procés es veu

a la figura 10.4 i on per acabar d'obtenir les dades has de concatenar les dades retornades per cada url, amb el qual tens uns headers que et permeten protegir les dades de l'url destí i que siguin accessibles només utilitzant les *key* dels paràmetres header retornats, però això és opcional. També un cop tens les dades resultants pots comprovar la seva integritat amb el *md5 digest*. Finalment els errors que pot retornar són els de la figura F.3.

També s'he de tenir en compte que hts-get admet múltiples implementacions de servidors i que els servidors poden respondre amb més informació de la sol·licitada però no menys.

A més aquesta API no proporciona una manera de saber els ids de ReadGroupSets vàlids i els clients els obtenen mitjançant algun altre mecanisme. El mateix passa amb la privacitat i seguretat dels fitxers, hts-get no l'incorpora, però sí que diu que és necessari utilitzar TLS 1.2 o superior i si el servidor demana autorització s'ha d'enviar el token pertinent, però aquesta especificació d'API no gestiona l'autorització d'usuaris sinó que també s'ha de fer de forma externa.

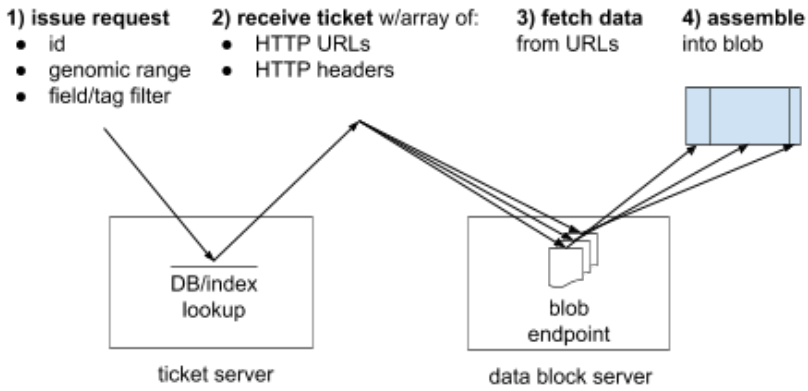


Figura 10.4: hts-get ticket. [41]

10.2 API MPEG-G

MPEG-G té definida una especificació d'API a la seva Part 3. [1] A diferència de hts-get disposa de més d'una crida i amb diverses opcions per cada una, tant per obtenir estadístiques directament, com les dades o metadades. Resumidament les crides de les quals disposa són les de la taula 10.1, on es veu el nom de la crida, els paràmetres d'entrada i els de sortida. Els paràmetres de sortida són els de la taula G.1, en aquesta taula no apareixen els d'estadístiques, els quals tenen una gran quantitat de paràmetres i es poden trobar a la part 3 de MPEG-G, a la secció de l'API. A més de les sortides trobem els codis d'error que es poden veure a la taula G.1. I pel que fa als paràmetres d'entrada a la taula 10.2 trobem l'estructura SimpleFilter, una de les diverses disponibles que es troben al document MPEG-G Part 3.

[1] En concret en aquest projecte es treballa amb l'especificació preliminar FDIS del març de 2019 disponible a l'inici d'aquest projecte.

Genomic information		
Crida	**In**	**Out**
GetHierarchy	-	OutHierarchyT
GetDataBySimpleFilter	datasetGroupID, datasetID, simpleFilter	outRecordsT
GetDataByAdvancedFilter	datasetGroupID, datasetID, advancedFilter	outRecordsT
GetDataBySignature	datasetGroupID, datasetID, signature	outRecordsT
GetDataByLabel	datasetGroupID, labelID	outRecordsT[]
Metadata		
Crida	**In**	**Out**
GetMetadataFields	datasetGroupID, datasetID	outputMetadata[]
GetMetadataContent	datasetGroupID, datasetID, fieldName	outputMetadata[]
Protection		
Crida	**In**	**Out**
GetDatasetGroupProtection	datasetGroupID	outputProtection
GetDatasetProtection	datasetGroupID, datasetID	outputProtection
GetDatasetRegionProtection	datasetGroupID, datasetID, sequenceID, startPos, endPos	outputProtection[]
Reference		
Crida	**In**	**Out**
GetDatasetReference	datasetGroupID, datasetID, includeSequences(bool)	outputReference
Statistics		
Crida	**In**	**Out**
GetSimpleStatistics	datasetGroupID, datasetID, sequenceID, startPos, endPos	MaxSegments, outputStatistics[]
GetAdvancedStatistics	datasetGroupID, datasetID, sequenceID, startPos, endPos	MaxSegments, outputStatistics[]

Taula 10.1: Crides API MPEG-G (Elaboració pròpia)

SimpleFilterT
uint numberOfGroups;
st(v) groupNames[numberOfGroups];
bool classID[6];
uint sequenceID;
uint startPos;
uint endPos;
bool singleEndsStrand[3];
bool pairedEndsStrand[9];
bool includeClippedReads[2];
bool includeMultipleAlignments;
bool includeOpticalDuplicates;
bool includeQualityCheckFailed;
bool includeAuxRecords;
bool includeReadNames;
bool includeQualityValues;
bool mismatchesIncludeNs;

Taula 10.2: SimpleFilterT de la API MPEG-G. (Elaboració pròpia)

Pel que fa als genAuxRecords és una estructura amb un array de genAux per cada record de les dades que s'obtenen, aquest genAux és un array de genTags, estructura que es veu a la figura 10.5.

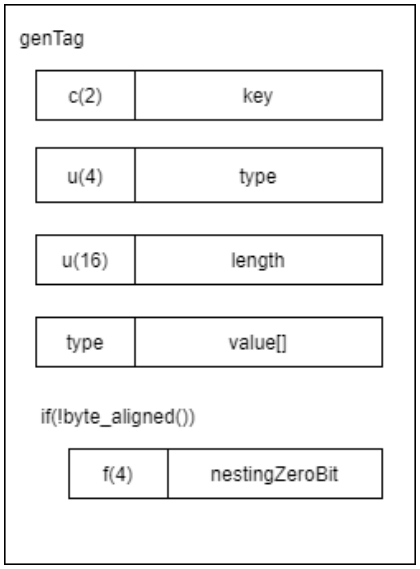


Figura 10.5: Estructura genTag de MPEG-G [48]. (Elaboració pròpia)

10.3 Mapping APIs

Tenint en compte, com ja s'ha dit, que l'API de MPEG-G té més opcions que la hts-get no hi ha problema per fer el mapatge de hts-get sobre MPEG-G, ja que les dades de hts-get es poden trobar a MPEG-G. El que sí que s'ha de tenir en compte és que hi ha dues opcions per realitzar el mapatge, o amb els mateixos paràmetres JSON d'entrada del GET http i a l'API de MPEG-G algunes entrades que els valors quedessin preestablerts o aprofitant que modificar el JSON de la crida GET és relativament fàcil, afegir un seguit d'opcions només disponibles si el proveïdor de fitxers és una API de MPEG-G i que permeti definir tots els paràmetres. El que estaria clar és que seria únicament sobre la crida GetDataBySimpleFilter de MPEG-G, ja que és el més semblant a hts-get. O GetMetadataFields en cas de demanar només els headers. Inicialment s'havia plantejat realitzar la primera opció però tenint en compte que afegir més paràmetres podria obrir el ventall a més opcions es va optar per la segona opció.

Per poder realitzar aquest *mapping*, bàsicament s'hauria de realitzar el que veu a la figura H.2 on si es fes la crida sobre una API MPEG-G seria necessària una traducció d'entrada i sortida per adaptar els paràmetres JSON a l'API de MPEG-G. A més, com es pot veure s'afegirien els paràmetres per SimpleFilter de MPEG-G que no es poden obtenir del JSON definit actualment, això també faria variar l'especificació en cas de cridar sobre API GA4GH, ja que tots aquests paràmetres haurien d'estar buits o inexistents.

A més d'això, per tal de poder saber la informació necessària de MPEG-G dels ids dels Datasets i DatasetGroup hauria de fer la crida a l'API MPEG-G directament o d'alguna altra forma que no impliqui hts-get.

Finalment podem veure a [33] que els reads són per fitxers SAM i per tant com que els fitxers SAM són el més semblant a MPEG-G, que no pas VCF, és per això les crides a hts-get cap

a MPEG-G haurien de ser reads. [90]

10.4 Traducció MPEG-G

Per acabar el mapatge com s'ha dit fa falta la traducció d'entrada i sortida. A l'apèndix C de MPEG-G explica com posar arxius SAM a MPEG-G, això no és exactament igual sinó que de fitxers MPEG-G vols aconseguir llegir les dades que llegiries d'un SAM amb els paràmetres que es demanen.

10.4.1 Entrada

Pel que fa a la traducció d'entrada es pot veure a la figura H.3. On inicialment s'ha de fer unes comprovacions dels errors per tal de respectar els errors dels paràmetres de MPEG-G i en cas d'error no seria necessari cridar a l'API MPEG-G. També a hts-get en modificar l'entrada, com ja s'ha dit, hts-get hauria de fer la comprovació que no hi haguessin aquests paràmetres en cas de fer la crida sobre API GA4GH. Un cop fetes les comprovacions ja es pot fer la crida a l'API de MPEG-G, on els paràmetres es mapegen de la forma que es veu a la figura H.4 en el cas de GetDataBySimpleFilter i a la figura H.1 per GetMetadataFields, i que gràcies a afegir els nous paràmetres al JSON la traducció és directa.

10.4.2 Sortida

Pel que fa a la sortida La traducció bàsicament consisteix en obtenir els url dels records que retornaria l'API de MPEG-G, per tant el servidor ho hauria de gestionar i tenir-los ja o realitzar un url per cada crida. Aquests completarien l'estructura dels *tickets* que retorna hts-get, on també hi hauria l'opció d'afegir els headers amb els que s'ha de cridar a les url de forma segura o el MD5 digest per verificar la integritat, tot i que no serien necessàries i per tant seria a gust del proveïdor de l'API de MPEG-G, tot i que recomanable. Com ja s'ha vist hts-get pot tornar més dades de les demanades però no menys, per tant els mpeg-grecords no s'haurien de modificar. En el cas dels errors també hi hauria un mapatge que no s'ha especificat, ja que bàsicament seria convertir els possibles errors de l'API de MPEG-G als codis de hts-get.

En aquest apartat no s'especifica cap esquema, ja que bàsicament s'ha de seguir el de la figura F.1 i el mapatge d'errors.

Capítol 11

Seguretat mapping APIs

Per poder tractar el tema de la seguretat en el mapatge de les APIs, i per tant poder utilitzar la privacitat que ja es troba en MPEG-G, en cas d'utilitzar hts-get sobre l'API de MPEG-G, tenint en compte que l'API de MPEG-G ja defineix la privacitat, l'únic que s'hauria de permetre és fer la crida hts-get amb l'usuari autentificat. Per seguir una mica la filosofia que utilitza GA4GH amb hts-get s'ha seguit en part a AAI Connect Profile [69], que a més es basa en OAuth 2.0 [96] [97] i OpenID [30] [76] que utilitzen JWS [98] i JWT [99].

Els diagrames d'OAuth 2.0 i OpenID són els que defineixen els passos a seguir perquè l'usuari s'identifiqui i pugui obtenir tokens d'accés sense que el servei necessiti implementar un sistema d'accés o de registre, sinó que s'utilitza un extern i mitjançant aquests protocols s'identifica a l'aplicació objectiu.

Aquests tokens estan amb formats JWS i JWT i que són formats JSON que permeten autentificar. JWS és el que permet firmar i per tant verificar un token, i els token són els que especifiquen que té accés i l'id de l'usuari entre altres.

A més hts-get recomana mínim TLS 1.2, tot i que si és possible és millor 1.3, ja que com es pot veure a [111] per 1.2 i [110] per 1.3 el cas de TLS 1.3 el primer que fa és definir un canal segur abans de començar a enviar certificats, a diferència de 1.2 que primer comprova la identitat mitjançant certificats però sense treballar a un canal segur i per tant és possible veure que s'envia fins que no es fa el KeyExchange.

11.1 AAI Connect Profile

Per tal de poder afegir autentificació al mapatge especificat a l'apartat 10, i per tant seguretat i privacitat, ens basem en AAI Connect Profile de GA4GH, com ja hem dit, és per això que hem d'entendre els paràmetres una mica diferents dels que es troben a AAI Connect, per poder fer l'adaptació d'aquest, tot i que la utilització és molt semblant i també es basa en OAuth2.0 i OpenID.

Prèviament hem de tenir en compte la terminologia que utilitza OpenID i que adopta AAI Connect.

- Claims Provider - Servidor que retorna *Claims*.
- ID Token – JSON Web Token (JWT) [99] que conté *Claim* sobre Authentication event.

- Issuer – Entitat que emet un seguit de *Claims*.
- OIDC – OpenID Connect, que és una capa d'autenticació a l'OAuth 2.0.
- Asserted/Assertion – Que està afirmat/validat per alguna organització.

Tenint en compte tot això i veient les definicions d'AAI Connect Profile [69] hauríem de tenir en compte les següents parts definides a hts-get i que poden variar una mica per adaptar-ho a MPEG-G.

GA4GH Claim – Un Claim JWT, que en el cas d'AAI Connect diu que està definit per la documentació tècnica de GA4GH. En el cas de treballar amb MPEG-G això seria un Claim JWT [55] [56] estil OAuth amb les dades necessàries.

Claim Management System – És servei que permet a usuaris *Claim Source* gestionar els seus *Claim Repositories*.

Claim Repository – És servei per gestionar l'emmagatzematge i obtenció de *Claims*, com si fos una BD, juntament amb metadata i/o logs relacionats amb la creació, modificació o eliminació de Claims.

Claim Source – Segons AAI Connect, és l'organització font de l'afirmació del *Claim* que com a mínim inclou l'organització associada a afirmar el *Claim*.

Aquesta no és necessàriament l'organització que guarda el claim o el broker que signa els tokens, és l'organització que té autoritat per afirmar el *claim* en lloc de l'usuari i és responsable de fer i mantenir l'assertion.

Identity Provider (IdP) – És un servei que ofereix als usuaris una identitat, els autentifica i dona *claims* al broker seguin protocols estàndard com OpenID Connect, SAML, etc. Alguns exemples són Google Identity i Facebook. Els IdPs poden ser els claims sources.

Broker –És un proveïdor de servei OIDC [30] que autentifiqui a un usuari (potencialment per un Identity Provider), recull els seus *claim* interns o els claims sources i emet els JWT claims que utilitzen Claim Clearinghouses. Els brokers poden ser Claim Clearinghouses d'altres Brokers.

Claim Clearinghouse – És consumidor de GA4GH/MPEG-G Claims, que venen donants per un Broker, i despres gestionar les decisions d'autorització, per tant mirar el claim i decidir a quina part pot accedir. Aquesta abstracció amb GA4GH està feta perquè permet diferents models de com permetre l'accés a les dades, tan donant un nou token per descarregar (actuant de Broker) com donant l'accés directament, com és el cas de l'API de MPEG-G.

Data Holder – Segons AII Connect és qui té les dades o una part d'ella i per tant qui gestiona l'accés, per tant en aquest cas seria el servidor de MPEG-G que tindria les dades i que gestionaria l'accés amb l'API de MPEG-G que treballa sobre els fitxers MPEG amb la privacitat definida pel data owner. El data owner i holder podrien ser el mateix. Aquests com a mínim té un Claim Clearinghouse.

Data Owner – És l'organització de les quals són les dades i que defineixen qui pot accedir

i a què. Per tant són qui defineix les polítiques que ha de tenir la Policy de MPEG-G.

Embedded Token Issuer – És qui signa els Embedded Tokens. Pot ser el mateix broker.

Embedded Token – Un valor o entrada dins una llista o un valor de GA4GH/MPEG-G Claim que contingui un string JWS i firmat per un Embedded Token Issuer.

A la figura 11.1 d'AAI Connect s'observa el camí que es fa des d'un Claim Source fins a un Claim Clearinghouse que després l'utilitza el token del broker. A la figura no es mostren totes les relacions de confiança, on cada destinatari és típicament la *relying party* de l'autentificació.

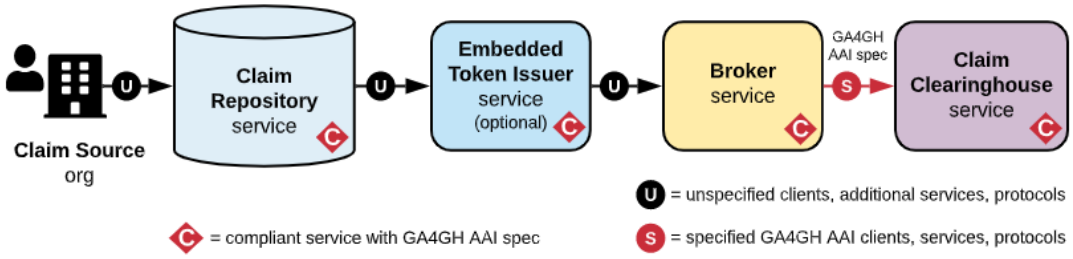


Figura 11.1: Auth flow de AAI Connect Profile [69].

Segons AAI Connect les implementacions poden introduir clients, serveis addicionals i protocols, que no estan detallats a la figura, per proporcionar els mecanismes per traslladar les dades entre Claim Respository i el Broker. Aquests han de complir una seguretat i privacitat equivalent a la d'AAI Connect Profile.

11.2 Especificació seguretat

Sense tenir en compte el que diuen AAI Connect Profile, Oauth i Open ID i per tant seguint una opció clàssica seria fer el que es veu a la figura 11.2. On suposaríem que el servidor MPEG-G s'encarrega de l'autentificació d'usuaris i per tant quan s'autentifiquen ofereix el token que després és el que s'utilitza per validar la identitat i saber qui és quan es fa la crida hts-get que es tradueix en una MPEG-G.

Per tant en concret l'aplicació que treballi sobre hts-get mapat hauria d'obtenir el token des del mateix servidor MPEG-G i d'aquesta forma indicar-ho amb la crida hts-get com a *Authorization: Bearer [access_token]*. D'aquesta forma el servidor MPEG-G rep una crida MPEG-G, ja que es fa el mapatge, i amb un token d'aquesta forma obté l'usuari i si està autoritzat o no, tal com fa l'especificació de l'API de MPEG-G. En cas de ser correcte retorna les dades amb els headers per assegurar que la crida la faci el que ho ha demanat.

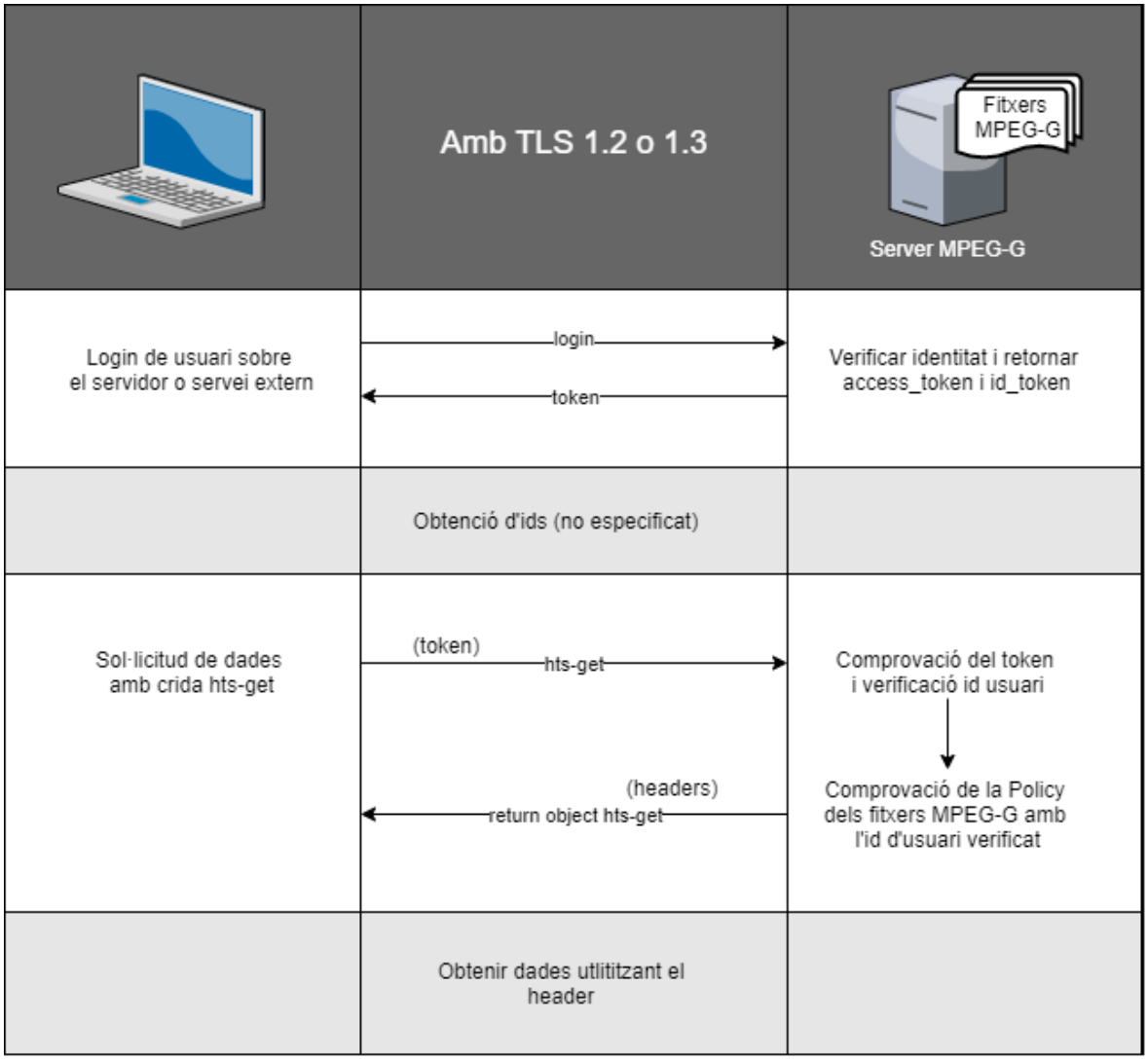


Figura 11.2: Diagrama autorització clàssica. (Elaboració pròpia)

Però si en lloc de fer això volguéssim utilitzar un sistema d'autentificació extern, i per tant basar-nos en AAI Connect Profile el que hauríem de fer és el següent.

Primer de tot consideraríem el servidor MPEG-G com el Data Holder, amb l'opció de també ser l'owner en funció de quina organització ofereixi l'API MPEG-G. També hauríem de considerar al servidor com el Claim Cleringhouse, ja que seria l'API MPEG-G que a través del token definiria si l'usuari pot accedir o no. Pel que fa al broker podria ser tant el mateix servidor com extern i referent al IdP sí que seria extern, per tant en el cas de ser broker intern hauria de saber qui és l'usuari a través d'un IdP i després generar el token.

Finalment el Claim Source, Repository, etc. seria tot extern, els quals serien les gestions dels que haurien de fer el IdP i serveis externs, tal com es veu a profile requieriments d'AAI Connect, ja que no especifica les necessitats d'aquests, només dels quals hem comentat que pot ser el servidor MPEG-G.

Pel que fa als **Profile Requieriments** serien iguals als d'AAI Connect.

Pel que fa als GA4GH Claims, aquí anomenats MPEG-G Claims contenen com a mínim les necessàries per l'API de MPEG-G que pugui identificar l'usuari.

Al no considerar-se important, no s'ha fet una especificació detallada com en el cas de **GA4GH JWT Format** d'AAI Connect adaptada a MPEG-G.

Capítol 12

Conclusions

Com ja s'ha definit a l'apartat 2 d'aquest document l'objectiu principal era *Observar els diferents estàndards que hi ha actualment per tractar dades genòmiques i estudiar la seva seguretat i privacitat, així com la seva interoperabilitat entre ells i poder millorar el que hi ha actualment*, també s'especificaven 3 subobjectius: *afegir la seguretat de crypt4gh a MPEG-G*, *mapping entre APIs* i *seguretat i privacitat de la API*.

Pel que fa a l'objectiu principal s'ha assolit amb èxit, ja que s'ha analitzat detalladament la seva seguretat com es veu en el cas de MPEG-G a l'apartat 5 on s'especifica tota la seva forma d'obtenir seguretat. I en el cas de GA4GH, en concret crypt4gh, es veu a l'apartat 6 on s'analitzen els algoritmes que utilitza, ja que a part d'això no disposa d'una estructura molt detallada.

Pel que fa a millorar el que hi ha actualment i que està relacionat amb el subobjectiu de *afegir la seguretat de crypt4gh a MPEG-G* ho podem veure a l'apartat 8 i a l'annex A.2. On després de l'estudi detallat dels estàndards de seguretat i algoritmes s'ha pogut afegir conceptualment els algoritmes de crypt4gh i sobretot permetre el keyAgreement amb el transport de claus. Una clau que no pot estar descoberta per dos usuaris diferents però sí que pot servir de clau per wrapejar la KeyTransport a utilitzar, i encriptar-la directament amb una clau especifica del KeyTransport per cada usuari. A més el ChaCha20-Poly1305 pot permetre als fitxers MPEG-G obtenir un major rendiment en cas que es vulguin obtenir en dispositius mòbils, ja que com hem vist obté un millor rendiment en aquests.

Relacionat amb tot això també trobem el prototip de l'apartat 9, on s'implementa l'encriptació i l'estructura relacionada a la seguretat de MPEG-G amb algunes variacions. Això ha permès acabar d'assolir l'objectiu principal de comprendre els estàndards actuals, i en afegir chacha20-poly1305 i EC, el tema de millora.

Ja referent al subobjectiu del mapatge d'APIs, on s'ha realitzat conceptualment. Aquest permet que a partir d'una aplicació que treballi amb hts-get poder treballar sobre un servidor amb una API de MPEG-G només variant uns paràmetres i afegint una petita traducció intermèdia. D'aquesta forma no és necessari un canvi total sinó més mínim. Tot i que a llarg termini, si les dades són utilitzades per realitzar grans estudis, es recomana treballar sobre l'API de MPEG-G, ja que a diferència de hts-get ofereix moltes més opcions i estadístiques.

A més de tot això, també s'ha fet un anàlisi detallat de la distribució de bits de la corba X25519, que es veu a l'apartat 7, per poder dir amb un gran percentatge de seguretat, ja que s'obtenen uns valors molt propers a una normal, però en ser estadística no és totalment segur, que el que es diu a crypt4gh és fals i com es poden veure les evidencies a l'apèndix A.3.

Finalment també s'ha analitzat la seguretat i privacitat del mapatge de les APIs, com es pot veure a l'apartat 11 on s'especifica com amb una crida hts-get es pot utilitzar la privacitat de MPEG-G i per tant l'aplicació no hauria de comprovar si té permisos o no, únicament accedir com a usuari al servidor i la política XACML dels fitxers MPEG-G serien els que definirien aquesta privacitat i que comprovaria l'API de MPEG-G cridada amb el mapatge de hts-get.

A part d'haver-se complert els objectius s'han après nous conceptes, com XSD Schemas, AWS, ChaCha20-Poly1305, Blake2b, transport de claus i derivation i wraping. A més s'han millorat apartats com la programació per temes criptogràfics en python i java i la programació en java en general, així com a la lectura de documents especialitzats i especificació de llibreries.

12.1 Adaptació a les competències

A continuació pots veure la relació de competències tècniques associades a aquest projecte i la seva justificació.

CTI2.2: Administrar i mantenir aplicacions, sistemes informàtics i xarxes de computadors (els nivells de coneixement i de comprensió són a les competències tècniques comunes). [Una mica]

Justificació: S'ha analitzat les APIs de MPEG-G i hts-get que treballa sobre la xarxa utilitzant GET de http i els paràmetres JSON.

CTI2.3: Demostrar comprensió, aplicar i gestionar la garantia i la seguretat dels sistemes informàtics (CEIC6). [En profunditat]

Justificació: S'ha analitzat la seguretat amb profunditat dels fitxers MPEG-G i els algoritmes d'encriptació. També el funcionament de TLS i OAuth 2.0.

CTI3.1: Concebre sistemes, aplicacions i serveis basats en tecnologies de xarxa, tenint en compte Internet, web, comerç electrònic, multimèdia, serveis interactius i computació ubiqua. [Bastant]

Justificació: S'ha analitzat les APIs de MPEG-G i hts-get que treballa sobre la xarxa utilitzant GET de http i els paràmetres JSON.

Capítol 13

Treball futur

De cara a un futur aquest projecte estaria obert a un seguit de millores o ampliacions. Tant en l'àmbit informàtic, com matemàtic/estadístic o legal.

Implementar la traducció entre APIs: Implementar la traducció d'entrada i sortida de hts-get a l'API MPEG-G.

Implementar seguretat API: A l'implementar el mapatge de les API també afegir privacitat i seguretat tal com s'especifica al capítol 11.

Millorar prototip: millorar el prototip de la seguretat de MPEG-G retirant totes les modificacions que s'han fet i igualar-ho a l'especificació real.

Anàlisi claus X25519: completar l'anàlisi de la distribució de bits de la corba X25519. Tant realitzant més iteracions com valorant amb diverses llibreries i llenguatges i mirant probabilitat que dos claus diferents generin una mateixa sharedKey.

Anàlisis normatives genètica: analitzar detalladament les normatives dels estats europeus respecte a la seguretat i privacitat de les dades genètiques, pel fet que la GDPR defineix uns mínims, però deixa obert a normatives dels estats membre.

Annex A

Fitxers Adjunts

Adjunt al treball es troba un ZIP amb els següents continguts.

A.1 GDPR

Un fitxer anomenat *gdpr-treball.pdf*. És un treball sobre GDPR realitzat per Nil Tosar, Aleix Lluch i l'autor d'aquest projecte Martí Fernández. Treball realitzat a l'assignatura de la Facultat d'Informàtica de Barcelona anomenada Aspectes Socials i Mediambientals de la Informàtica.

A.2 Contribució MPEG

Un fitxer anomenat *Add GA5GH file encryption to MPEG G.pdf* proposat com contribució MPEG-G, previ a les modificacions del director del projecte per adaptar-ho a una contribució oficial.

A.3 Scripts

Una carpeta anomenada */Script* en la qual es troben els 4 scrips, els notebooks de R i les dades. Està organitzat amb una carpeta per script. I per executar els scripts són necessàries les llibreries nacl, numpy i bitarray. I per executar el *.ipynb*, R. En ambdós casos és necessari Python 3.X.

A.4 Codi Prototip

Una carpeta anomenada */Implementacio* amb el codi del prototip. Dins d'aquesta hi ha la carpeta *./src* on es troben les classes (*./src/main/java/MPEGG/*) i els tests (*./src/test/java/*). També trobem *./xsd-schemas/* on hi ha els fitxers xsd de les *boxes*. I finalment també trobem exemples dels fitxers *.mpg* que genera la implementació.

A.5 Javadoc Prototip

Una carpeta anomenada */Javadoc* amb el javadoc del prototip.

Annex B

Esquemes fitxers

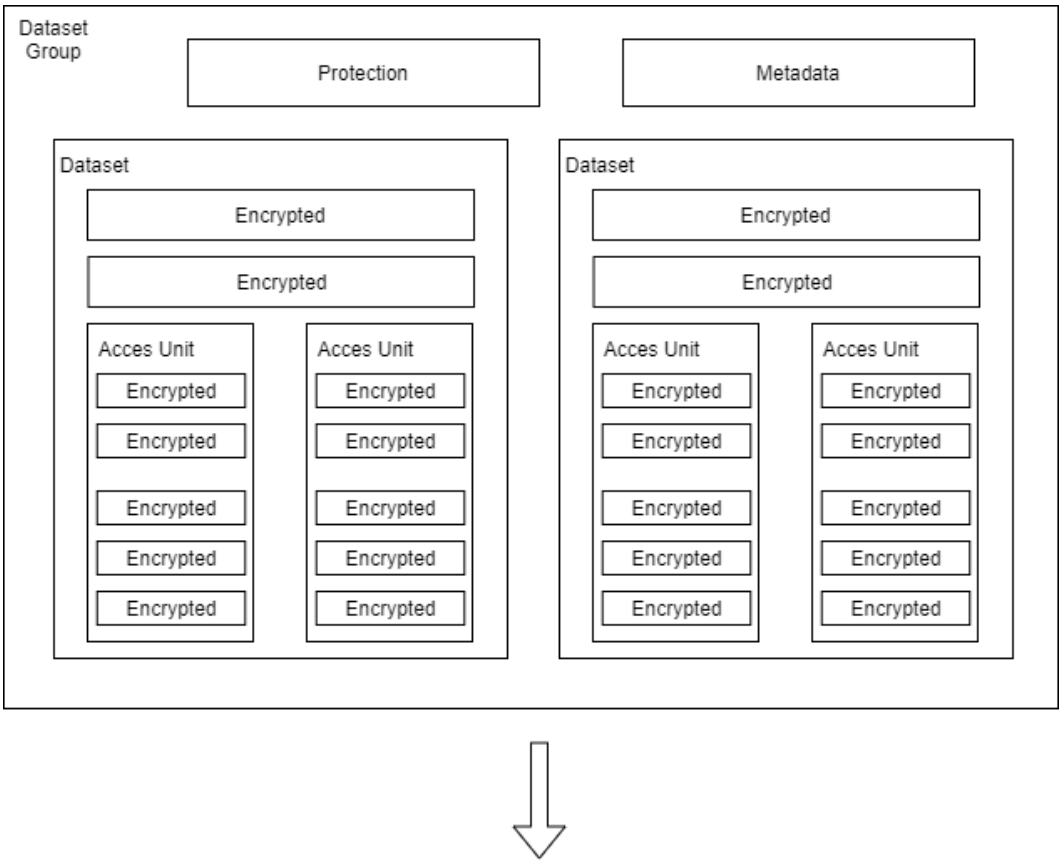


Figura B.1: Estructura d'un fitxer MPEG-G sense descriptor streams. Part 1. (Elaboració pròpia)

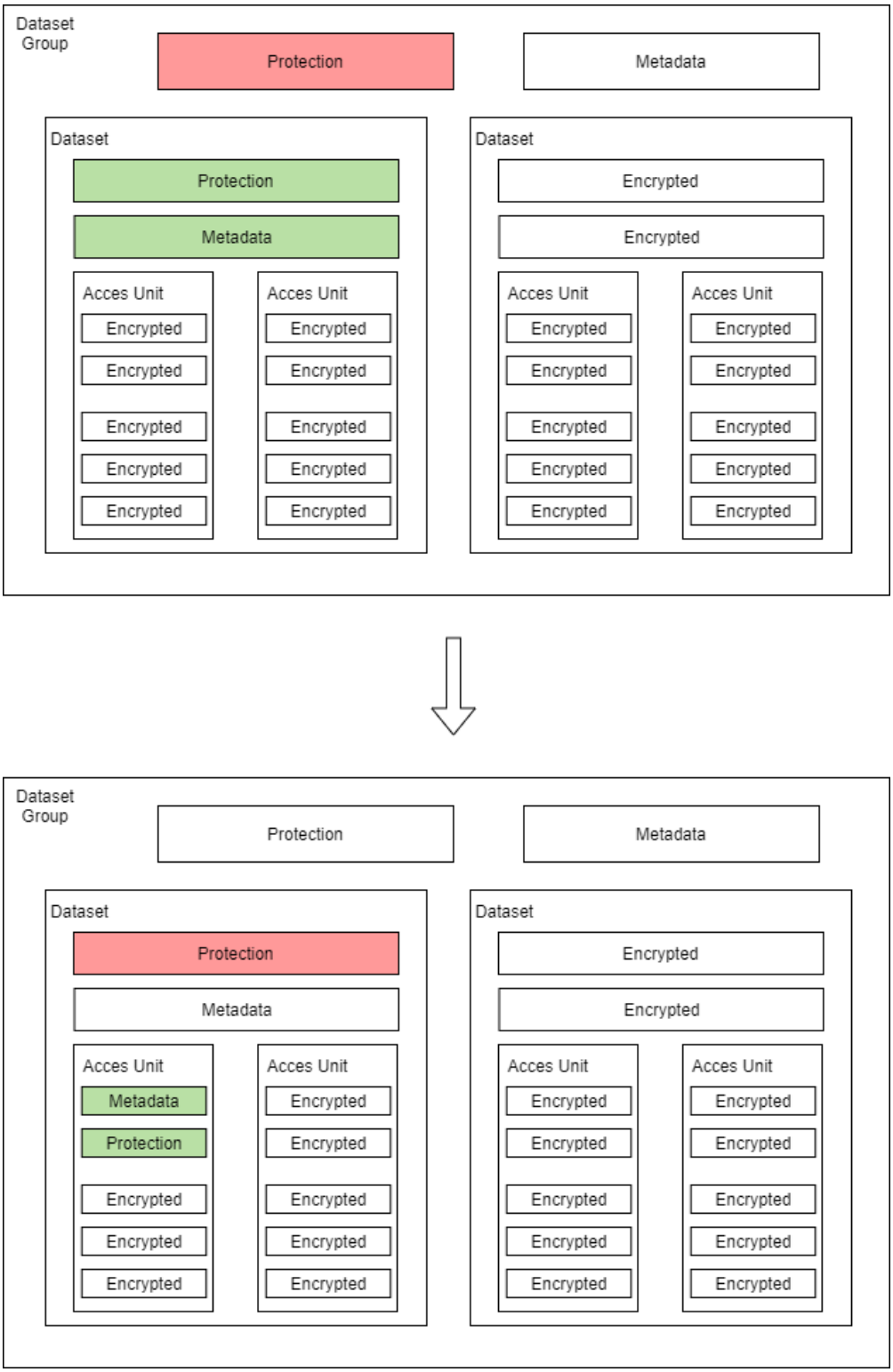


Figura B.2: Estructura d'un fitxer MPEG-G sense descriptor streams. Part 2. (Elaboració pròpia)

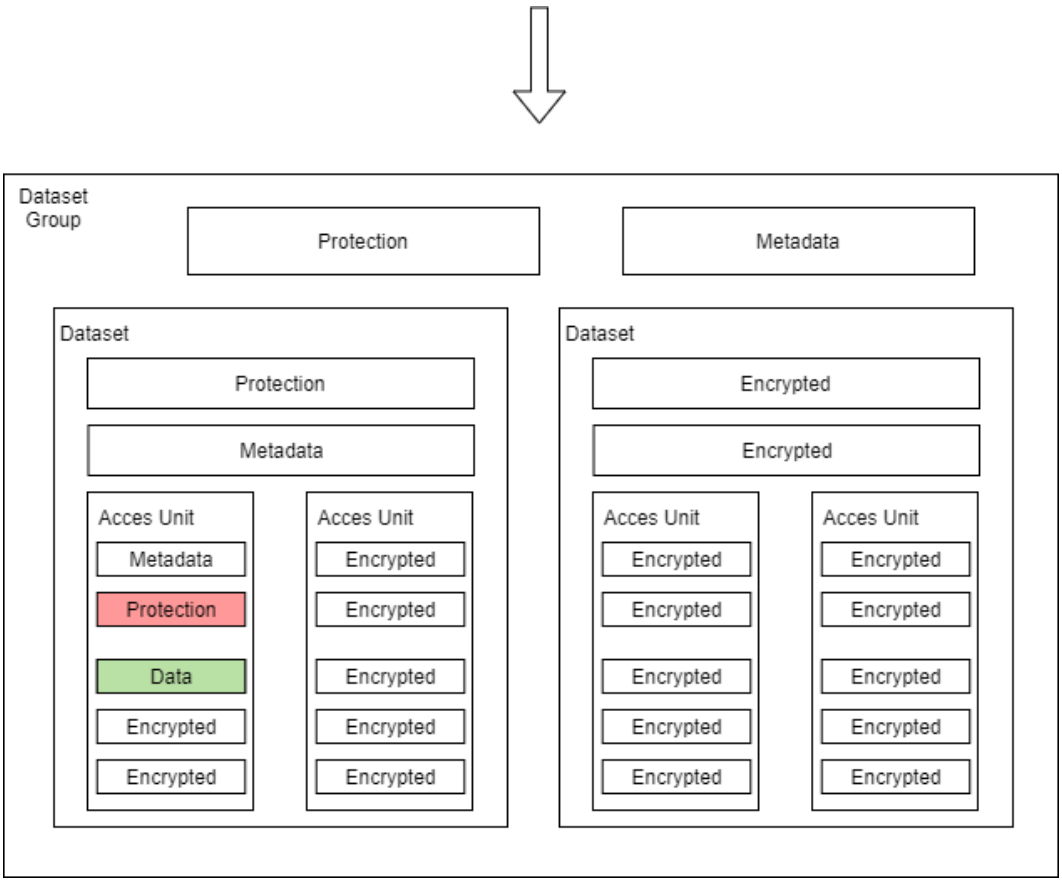


Figura B.3: Estructura d'un fitxer MPEG-G sense descriptor streams. Part 3. (Elaboració pròpia)

Annex C

Gràfics R

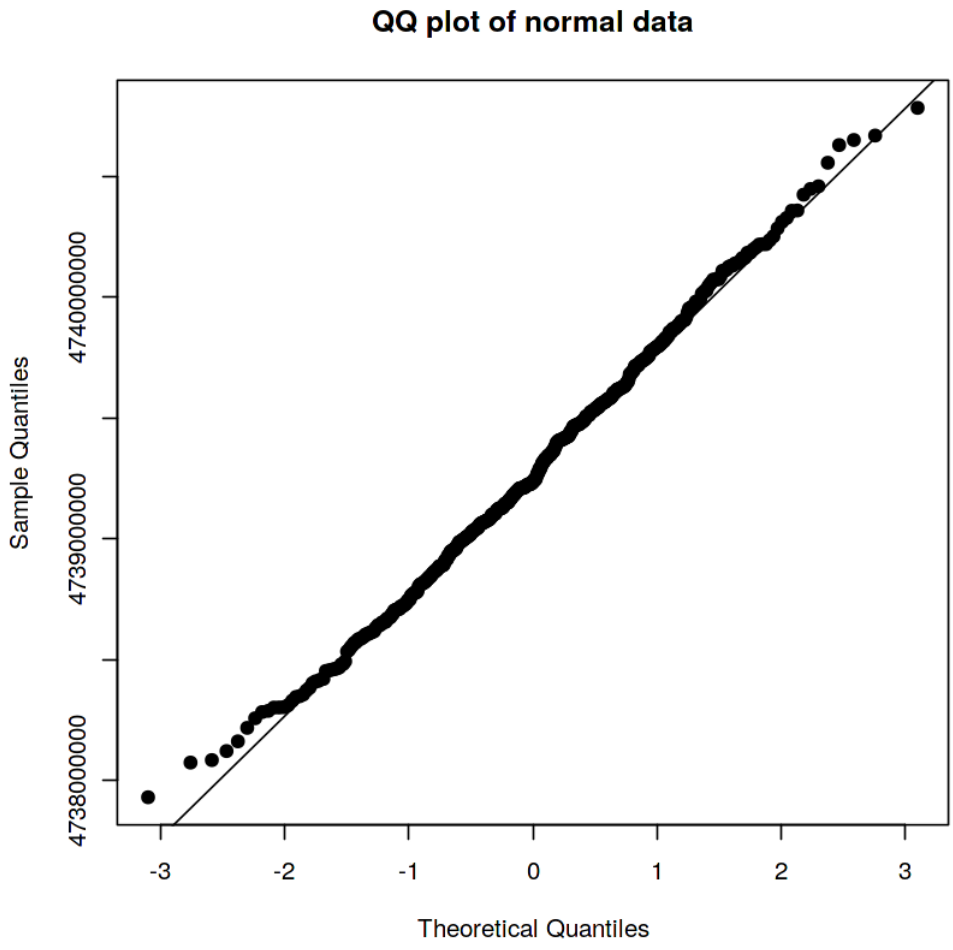


Figura C.1: QQ plot Script 1 i dades 4. (Elaboració pròpia)

.

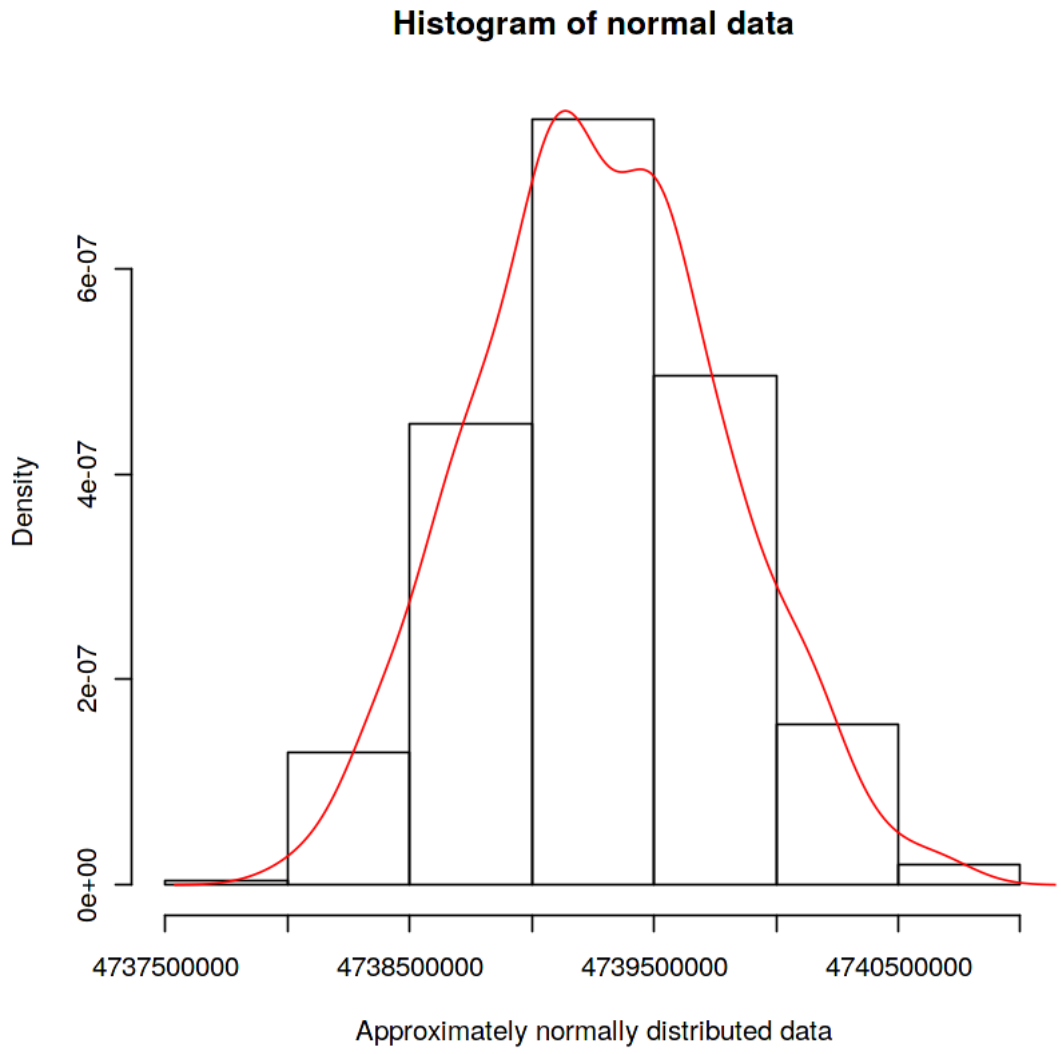


Figura C.2: Histograma Script 1 i dades 4. (Elaboració pròpia)

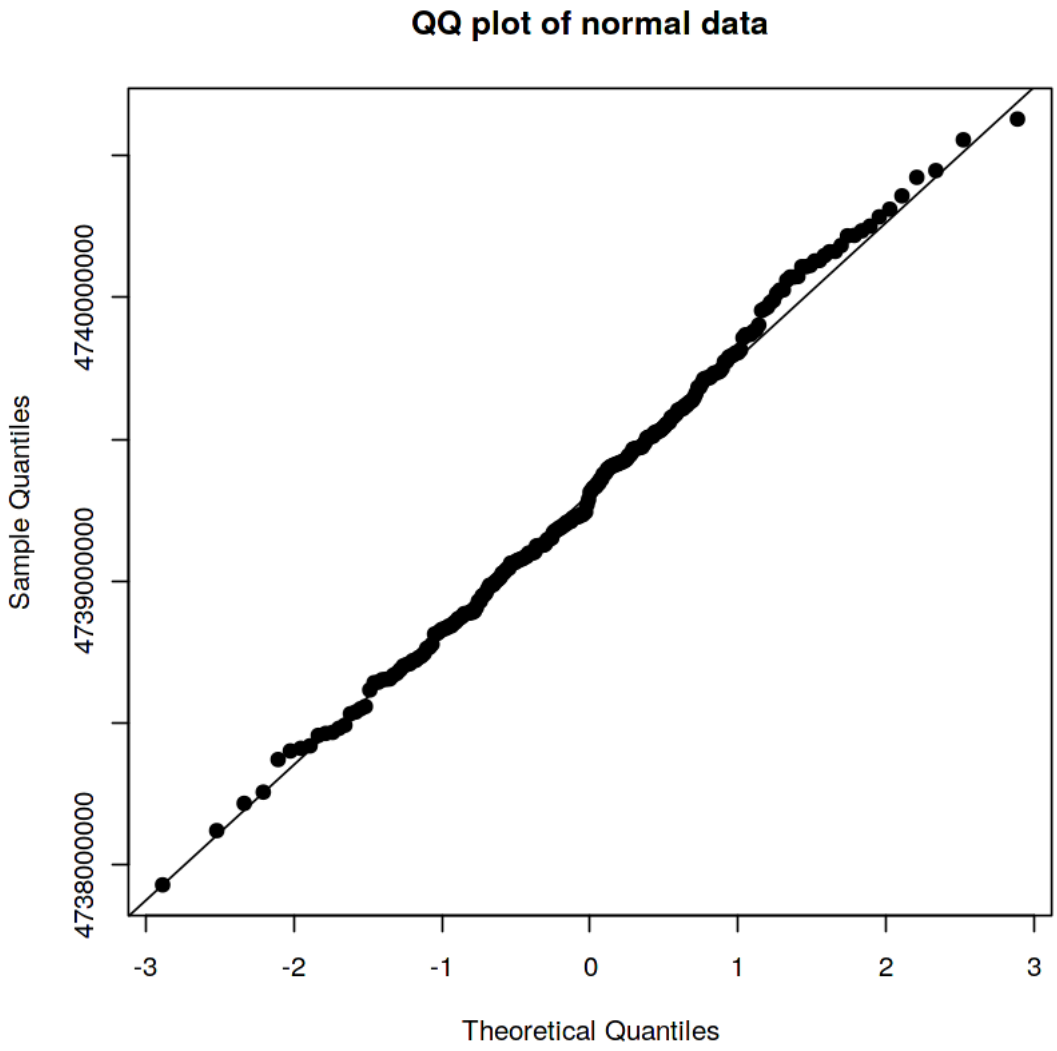


Figura C.3: QQ plot Script 1 i dades 4 amb segona clau. (Elaboració pròpia)

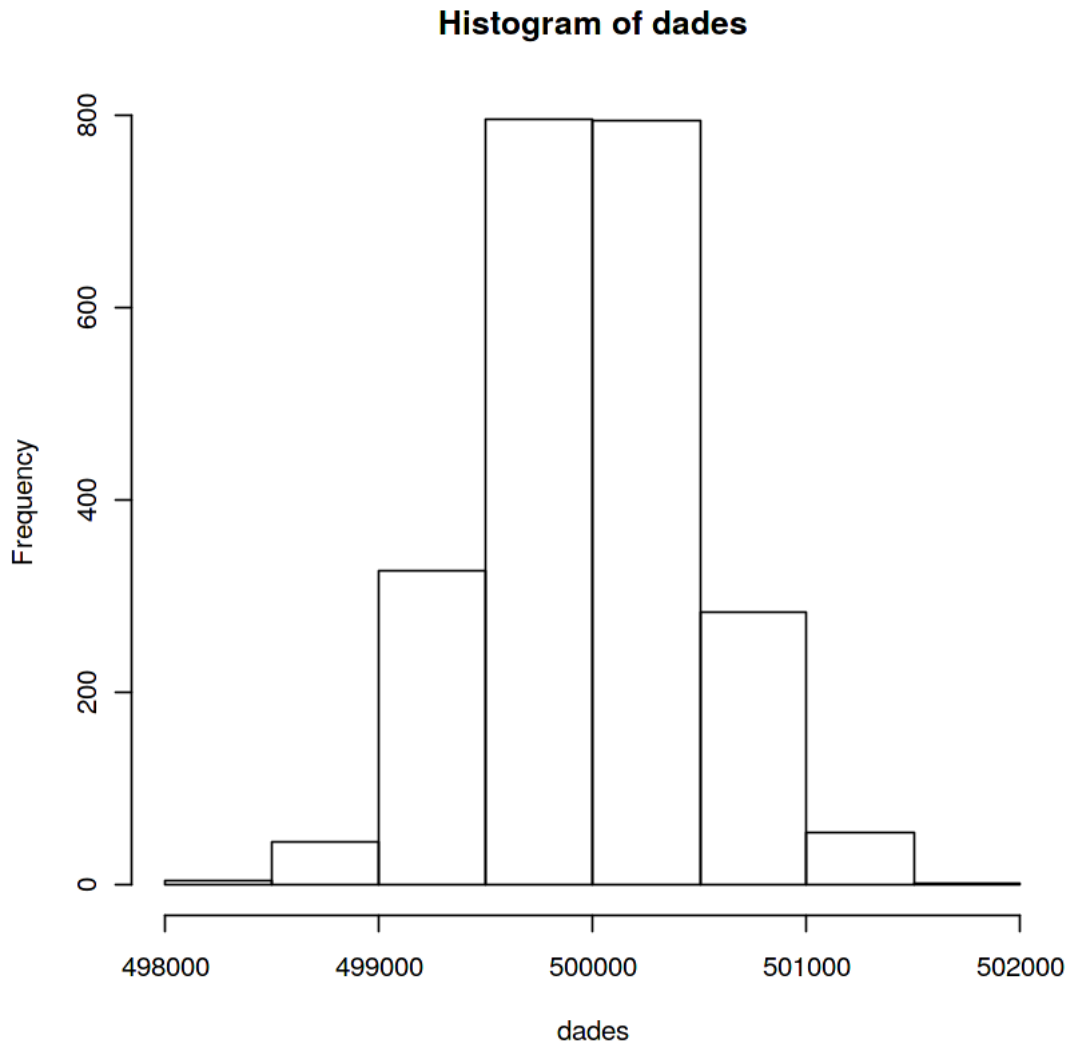


Figura C.4: Histograma Script 2. (Elaboració pròpia)

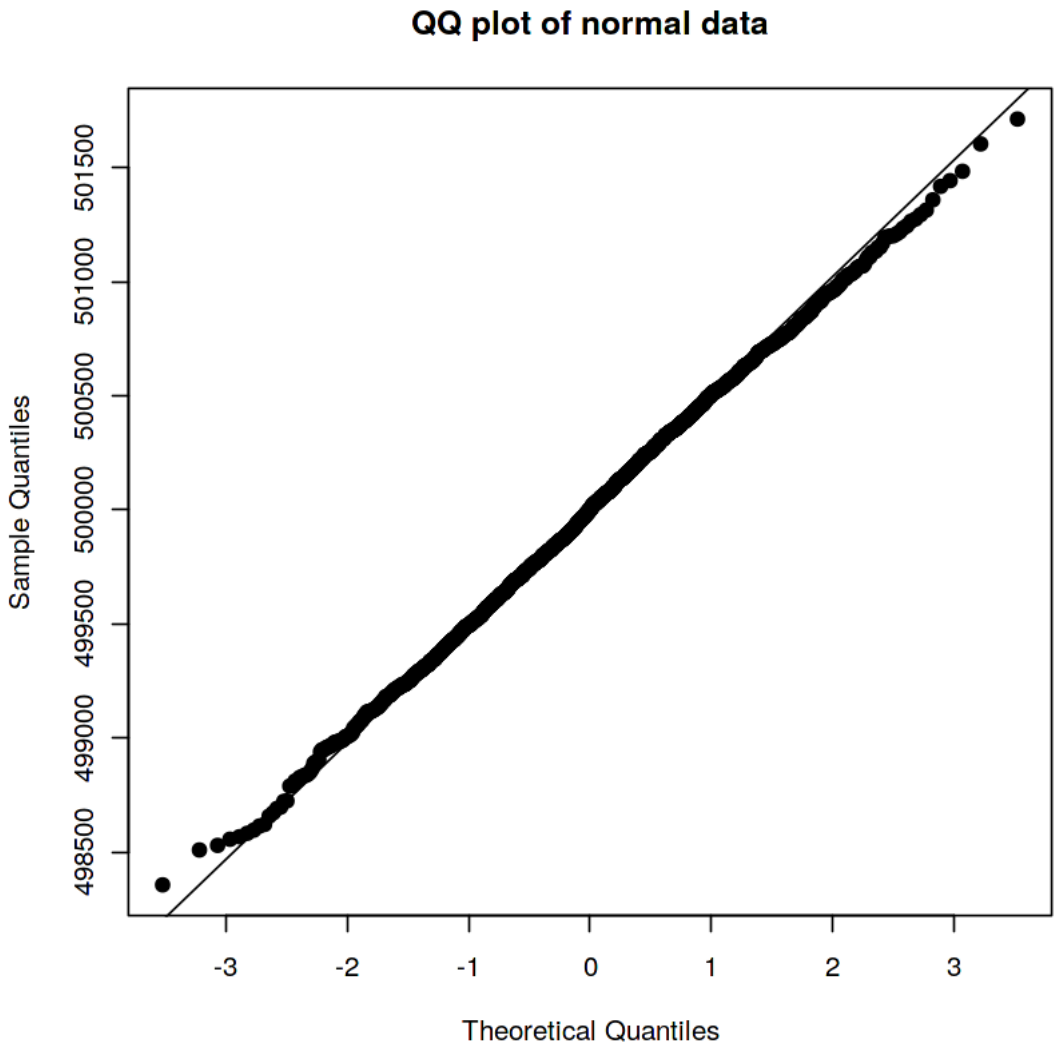


Figura C.5: QQPlot Script 3. (Elaboració pròpia)

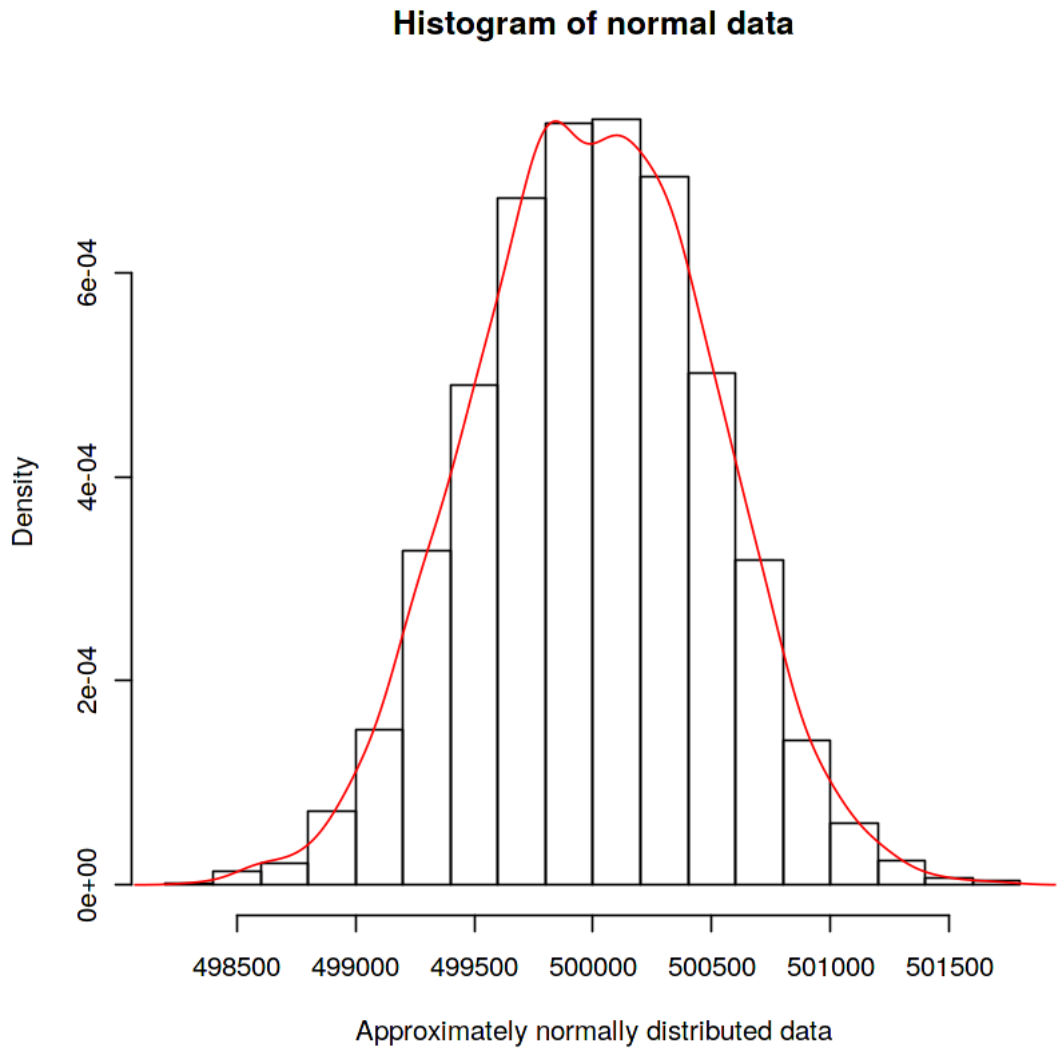


Figura C.6: Histograma Script 3. (Elaboració pròpia)

Annex D

UML

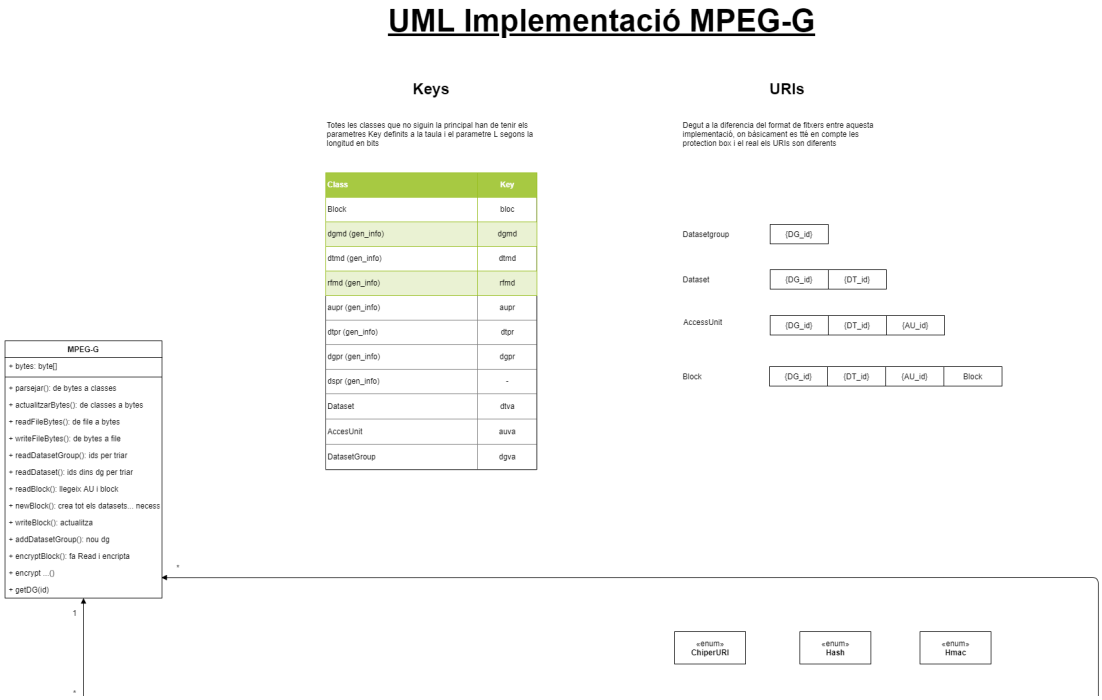


Figura D.1: UML previ de la implementació. Part 1. (Elaboració pròpia)

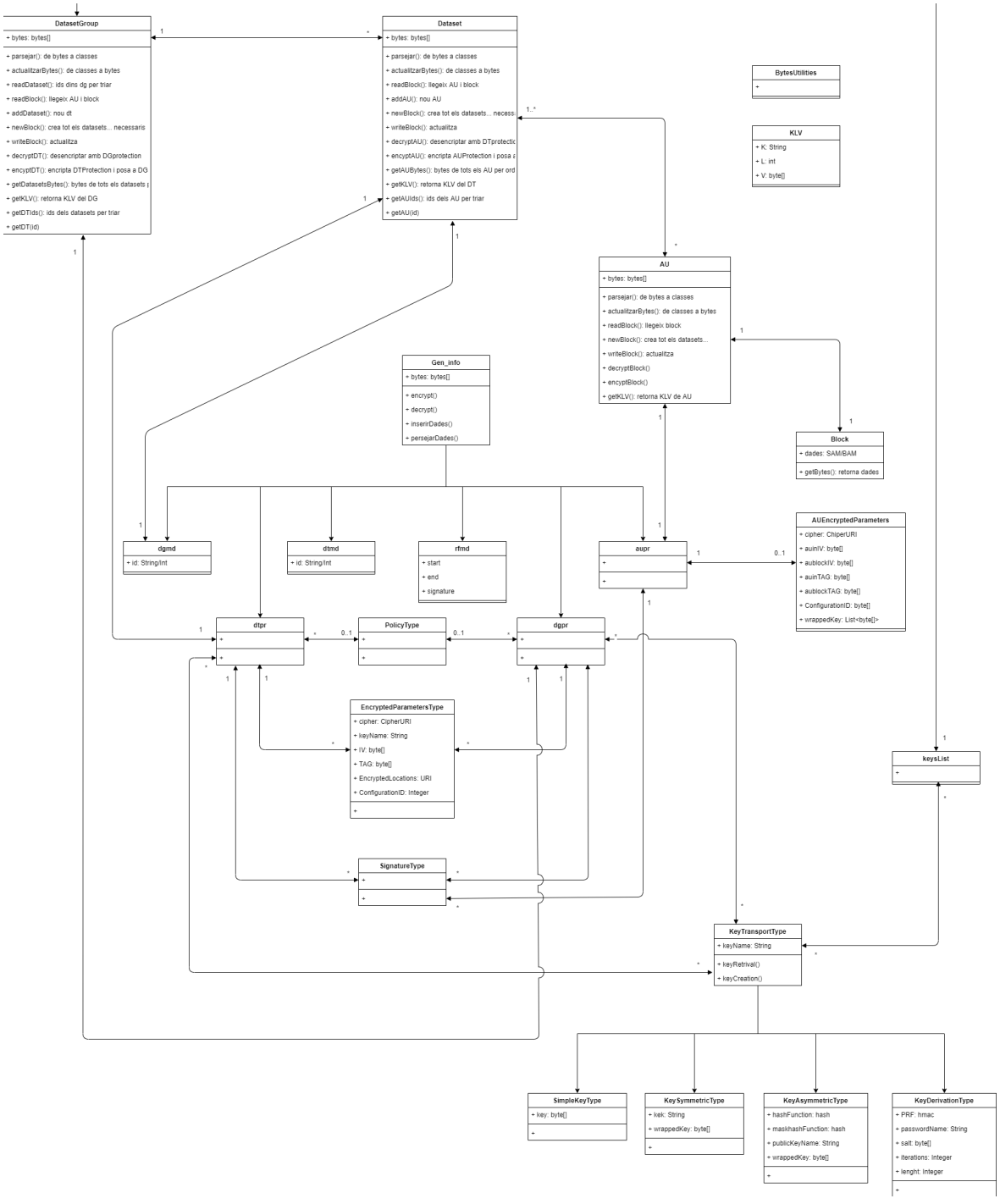


Figura D.2: UML previ de la implementació. Part 2. (Elaboració pròpia)

Annex E

Classes Java

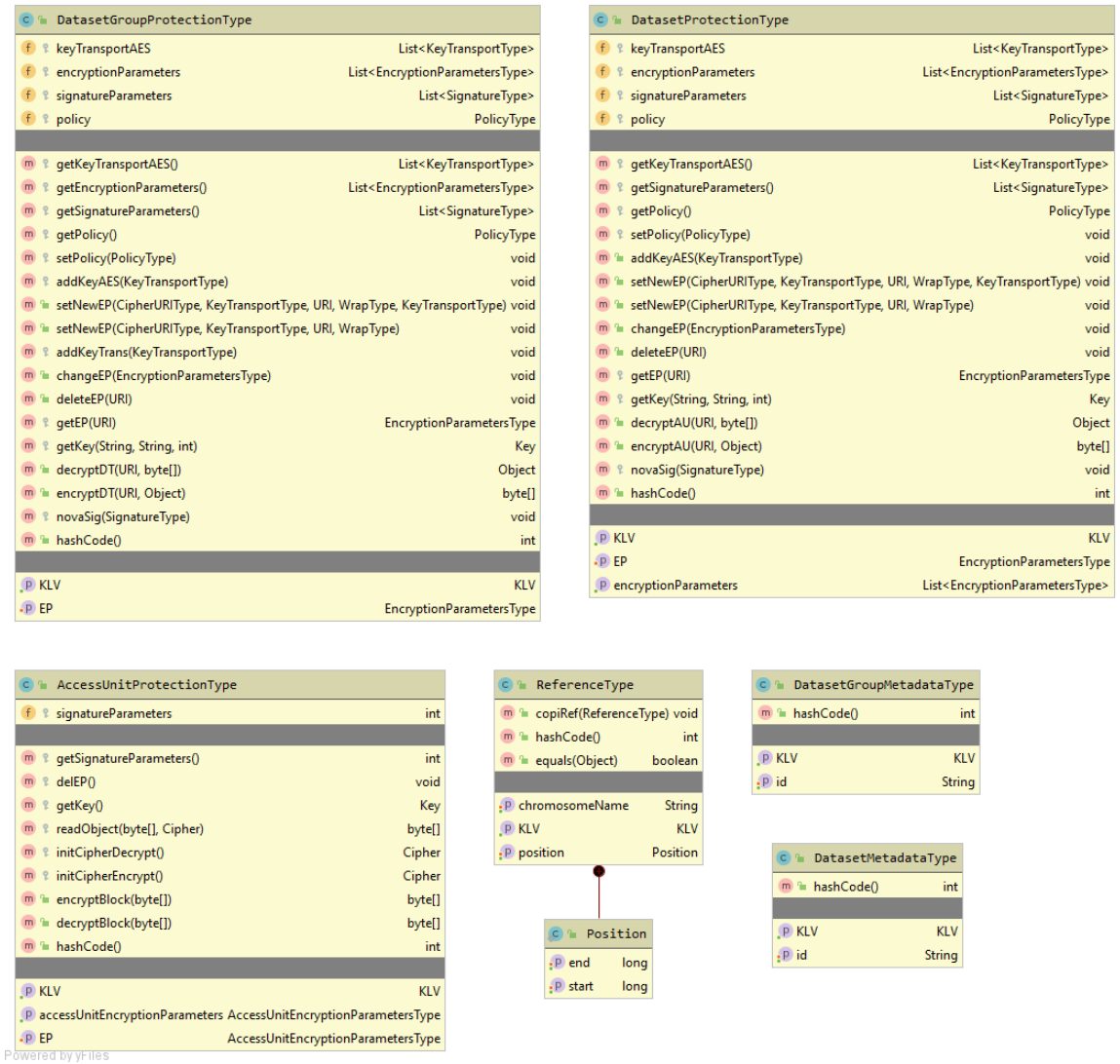


Figura E.1: Classes package Boxes. (Elaboració pròpia)

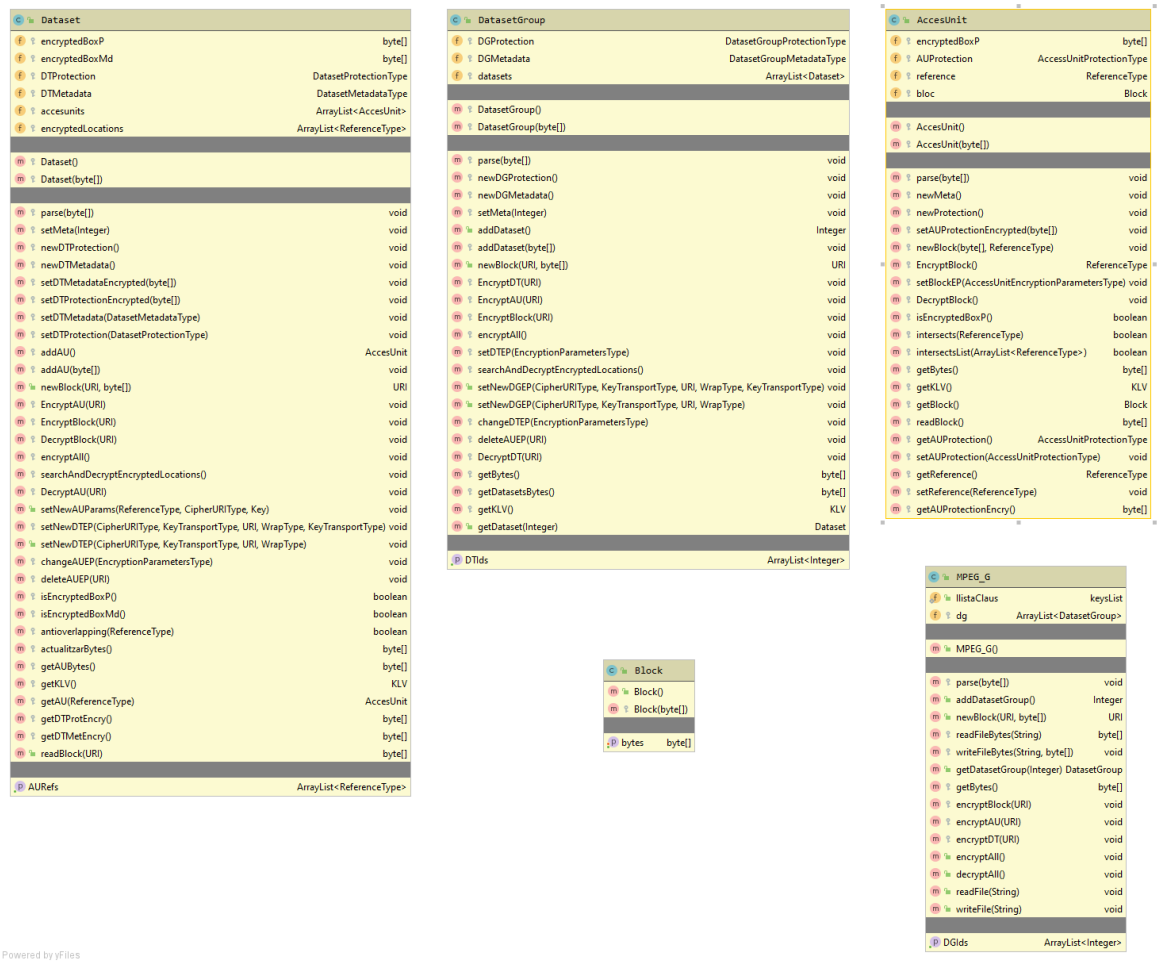


Figura E.2: Classes package Data. (Elaboració pròpia)

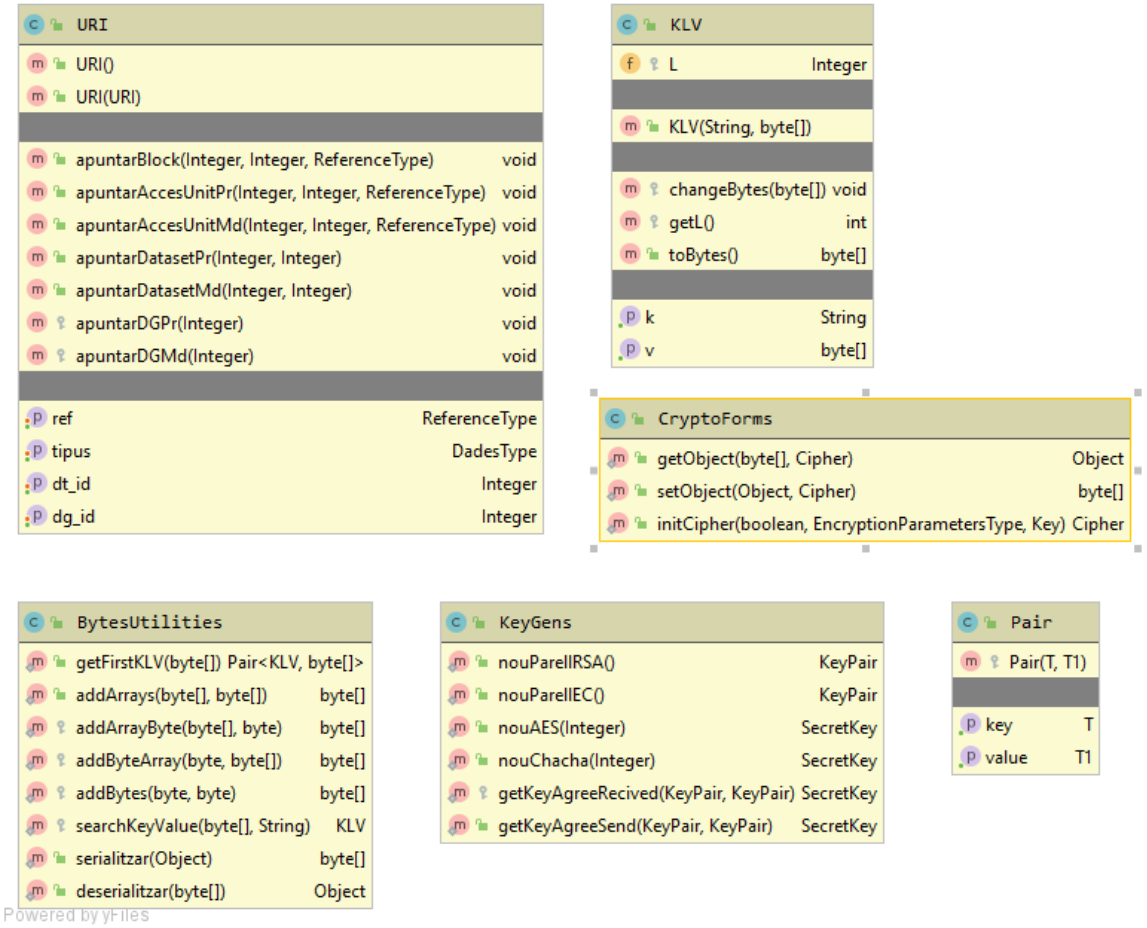


Figura E.3: Classes package Utilitats. (Elaboració pròpia)

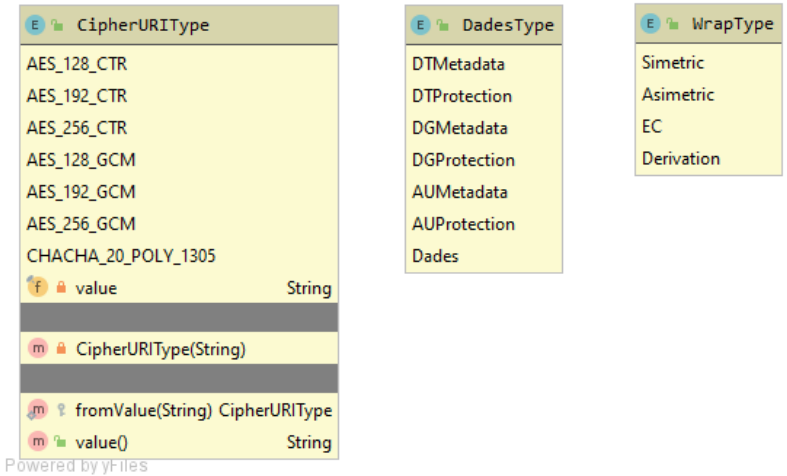


Figura E.4: Enumeracions. (Elaboració pròpia)

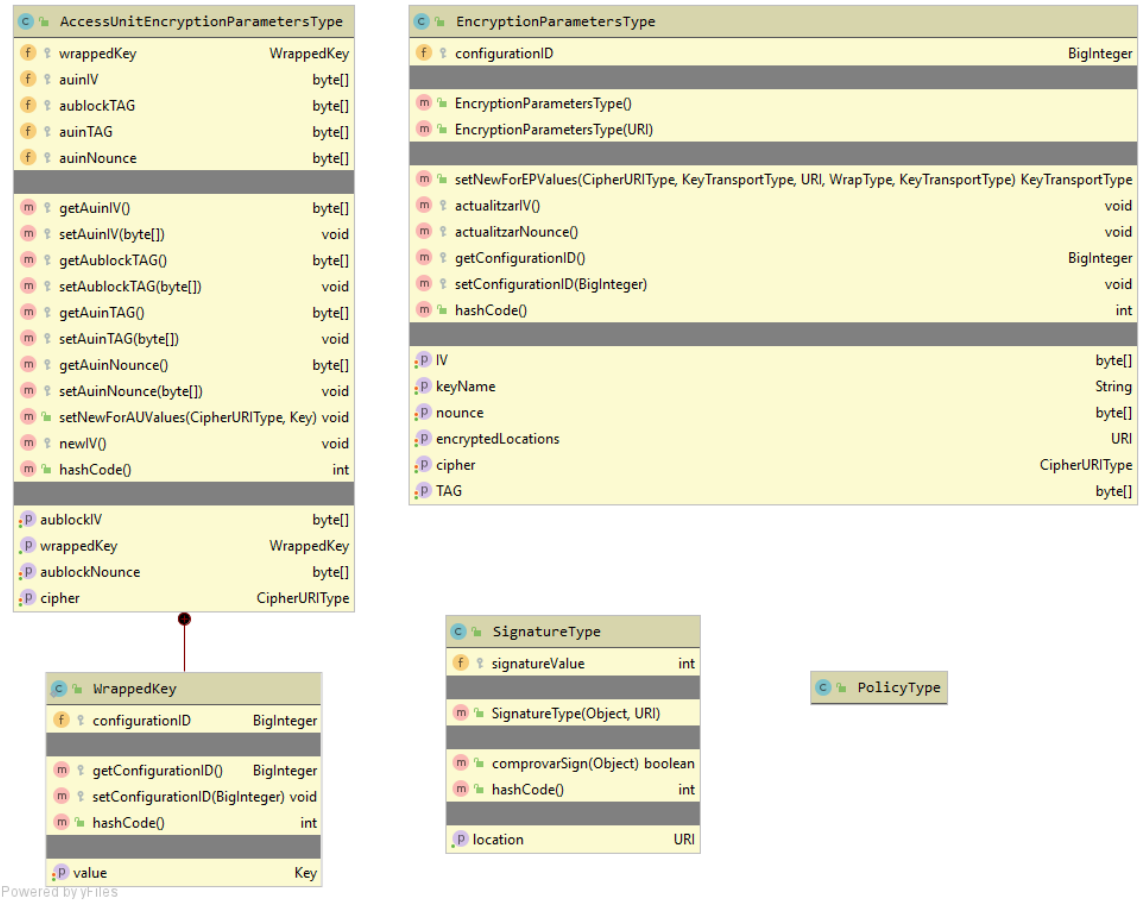


Figura E.5: Classes package Types. (Elaboració pròpia)

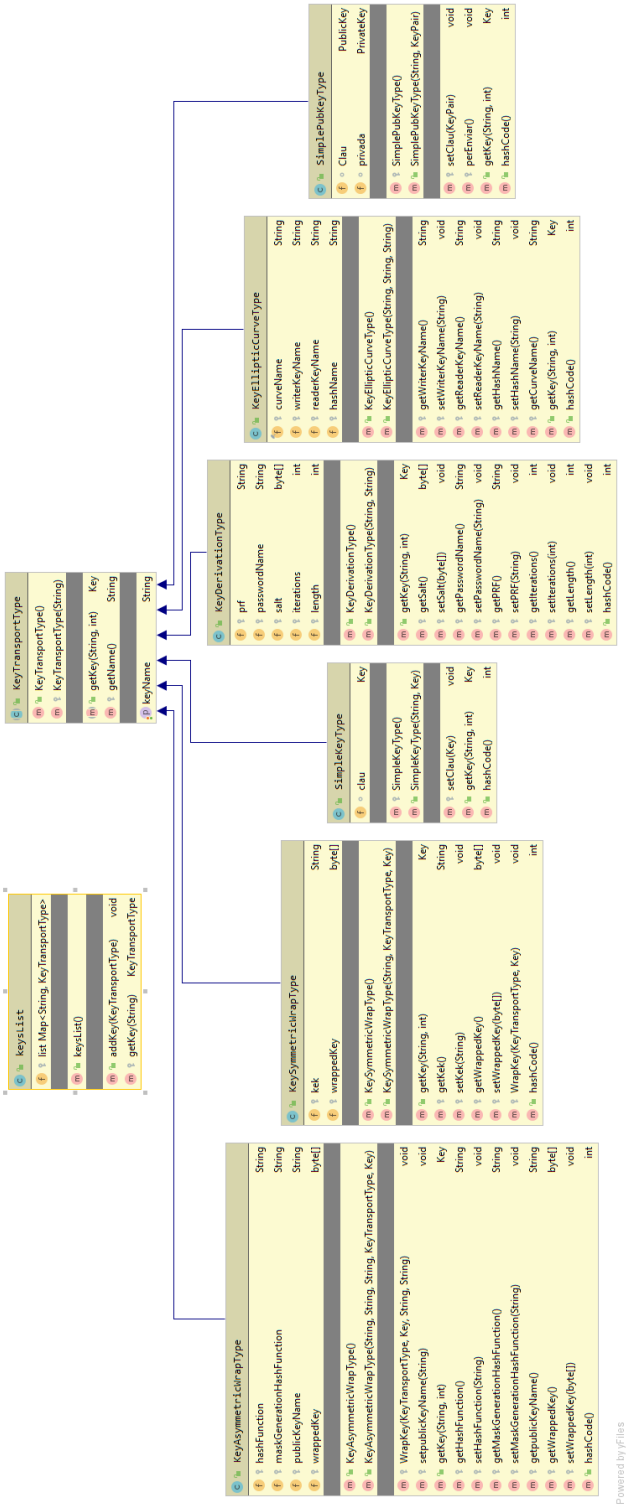


Figura E.6: Classes package Keys. (Elaboració pròpia)

Annex F

Esquemes hts-get

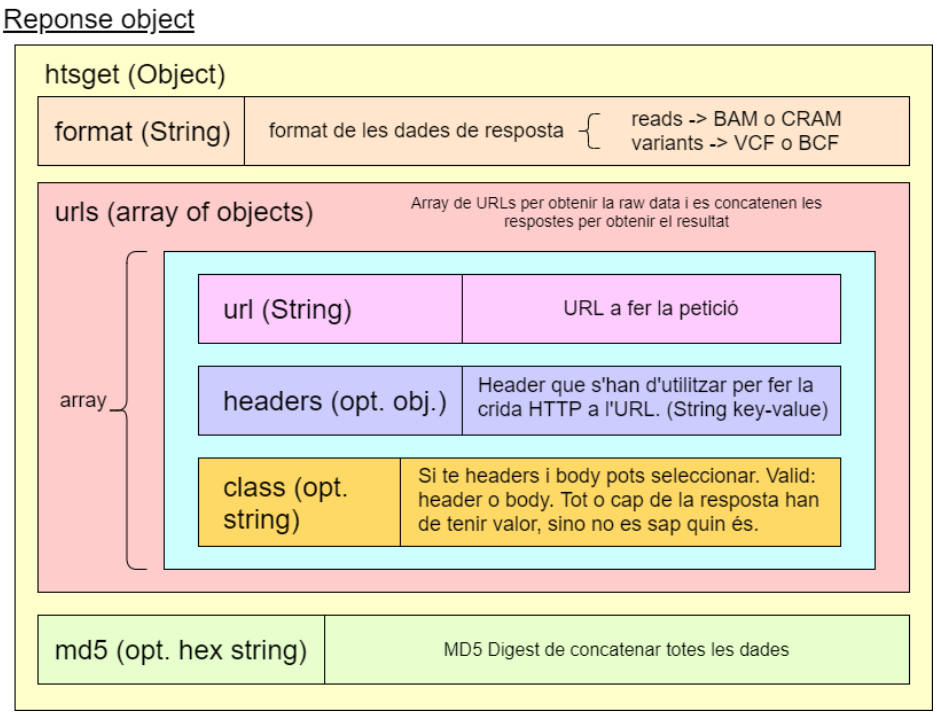


Figura F.1: Resposta hts-get [41]. (Elaboració pròpia)

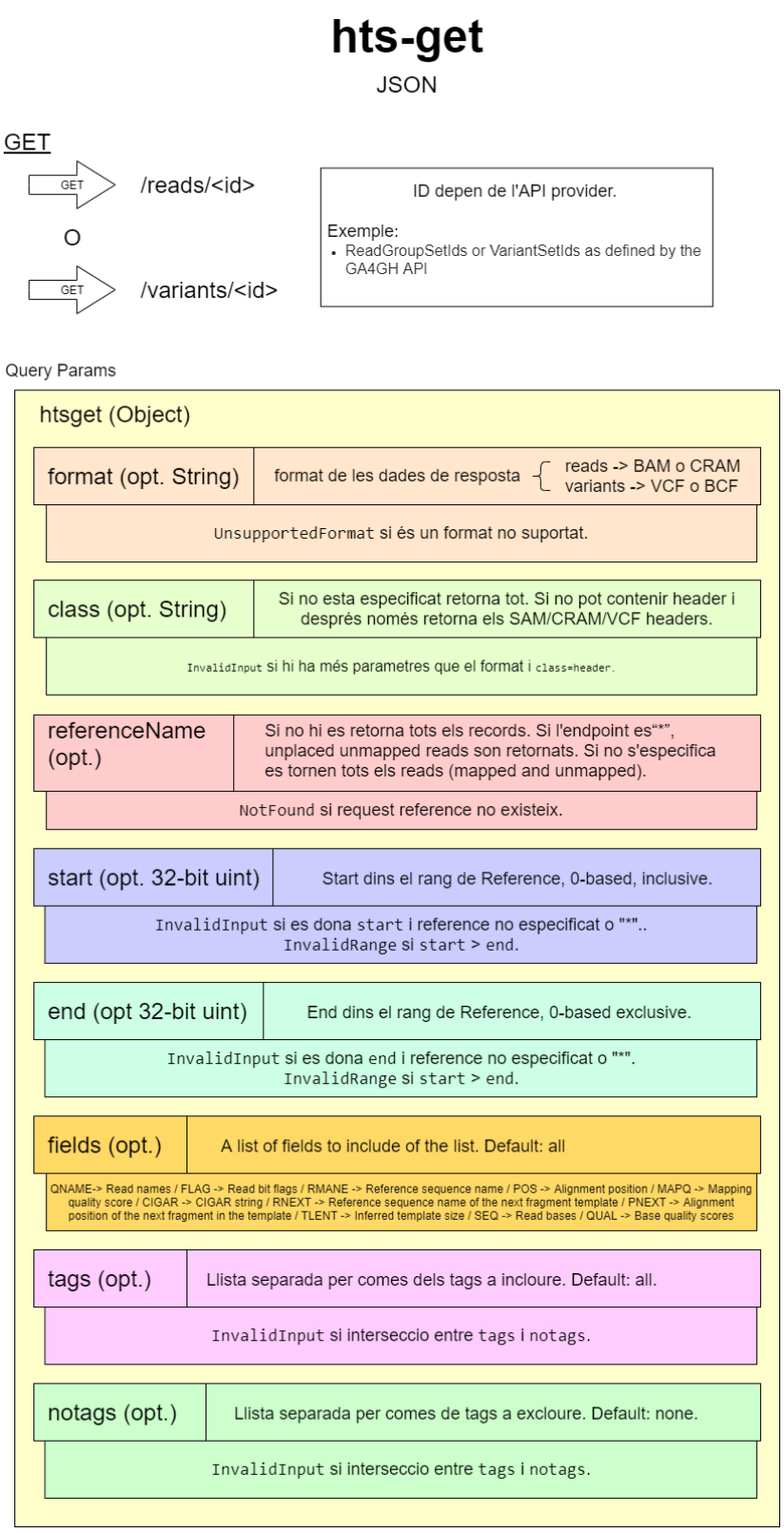
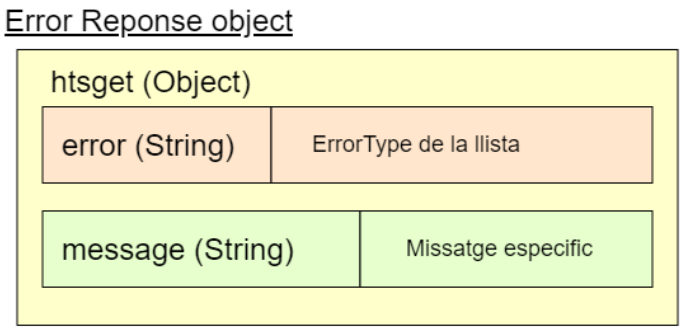


Figura F.2: Paràmetres crides hts-get [41]. (Elaboració pròpia)



Error type	HTTP status code	Description
InvalidAuthentication	401	Authorization provided is invalid
PermissionDenied	403	Authorization is required to access the resource
NotFound	404	The resource requested was not found
UnsupportedFormat	400	The requested file format is not supported by the server
InvalidInput	400	The request parameters do not adhere to the specification
InvalidRange	400	The requested range cannot be satisfied

Figura F.3: Errors hts-get [41]. (Elaboració pròpia)

Annex G

API MPEG-G

Esquemes en concret de l'especificació preliminar ISO/IEC FDIS 23092-3 del març de 2019.

Error codes are of type returnCodeT		
Name	**Value**	**Description**
G_SUCCESS	0	The operation completed successfully.
G_PARTIALLY_AUTHORIZED	1	The operation completes successfully, but only part of the queried content is returned due to partial lack of authorization.
G_NOT_AUTHORIZED	2	The operation is not authorized according the privacy rules.
G_VERIFICATION_FAILED	3	The digital signature, as specified in subclause 7.4, is not verified.
G_DECRYPTION_FAILED	4	It is not possible to complete the decryption process.
G_DATASETGROUP_NOTFOUND	5	The requested dataset group does not exist.
G_DATASET_NOTFOUND	6	The requested dataset does not exist.
G_ACCESSUNITS_NOTFOUND	7	The requested access unit does not exist.
G_REFERENCE_NOTFOUND	8	The requested reference does not exist.
G_SEQUENCE_NOTFOUND	9	The requested sequence does not exist.
G_METADATA_FIELD_NOTFOUND	10	The metadata element does not exist. If the element exists but is empty, this error is not returned.
G_INVALID_METADATA	11	The value of the metadata element, or the element itself, does not comply with the schema.
G_INVALID_REFERENCE	12	This error is returned when the genomic reference to be used in the decoding process is not compatible with the genomic information.
G_INVALID_PARAMETER	13	At least one parameter to the call is invalid.
G_INVALID_BITSTREAM	14	The provided bitstream is invalid.
G_UNLISTED_ERROR	15	An unlisted error.
	16 .. 255	Reserved

Taula G.1: Error codes de API MPEG-G. (Elaboració pròpia)

outHierarchyT	
uint datasetGroupsCount;	the number of dataset groups present in the bitstream
uint datasetGroupID[];	list of identifiers of dataset groups present in the bitstream
uint datasetsCount[];	list containing the number of datasets associated with each dataset group
uint datasetID[][];	list of identifiers of datasets associated with each dataset group
outRecordsT	
uint datasetGroupID;	identifier of a dataset group
uint datasetID;	identifier of a dataset associated with the dataset group identified by datasetGroupID
uint recordsCount;	number of records returned by the call
mpeggRecord records[]	list of records returned by the call
genAuxRecord auxInfo[];	list of returned genAuxRecord structures associated to the returned by the call records
outReferenceT	
uint seqCount;	Number of reference sequences returned by the call
st(v) seqName[];	List containing the names of the reference sequences returned by the call
uint seqStart[];	List containing the start position of the reference sequences returned by the call
uint seqEnd[];	List containing the end position of the reference sequences returned by the call
char refSequence[][];	Lists of ASCII characters representing the reference sequences returned by the call
outRegionProtectionT	
string sequenceName;	Name of the sequence where the protected region is defined
uint startPos;	Start position of the protected region.
uint endPos;	End position of the protected region.
uint classID;	the class of access units being encrypted as here specified
uint numKeys	the number of keys
st(v) keyName[numKeys]	Any key name present in this array allows to decrypt the content of an access unit which range intersects with the region defined by sequenceName, startPos, endPos, classID

Taula G.2: Estructures de sortida de API MPEG-G sense estadistiques. (Elaboració pròpia)

Annex H

Mapatge APIs

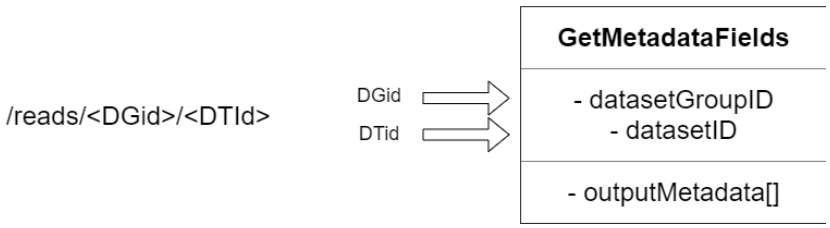


Figura H.1: Traducció entrada MPEG-G. (Elaboració pròpia)

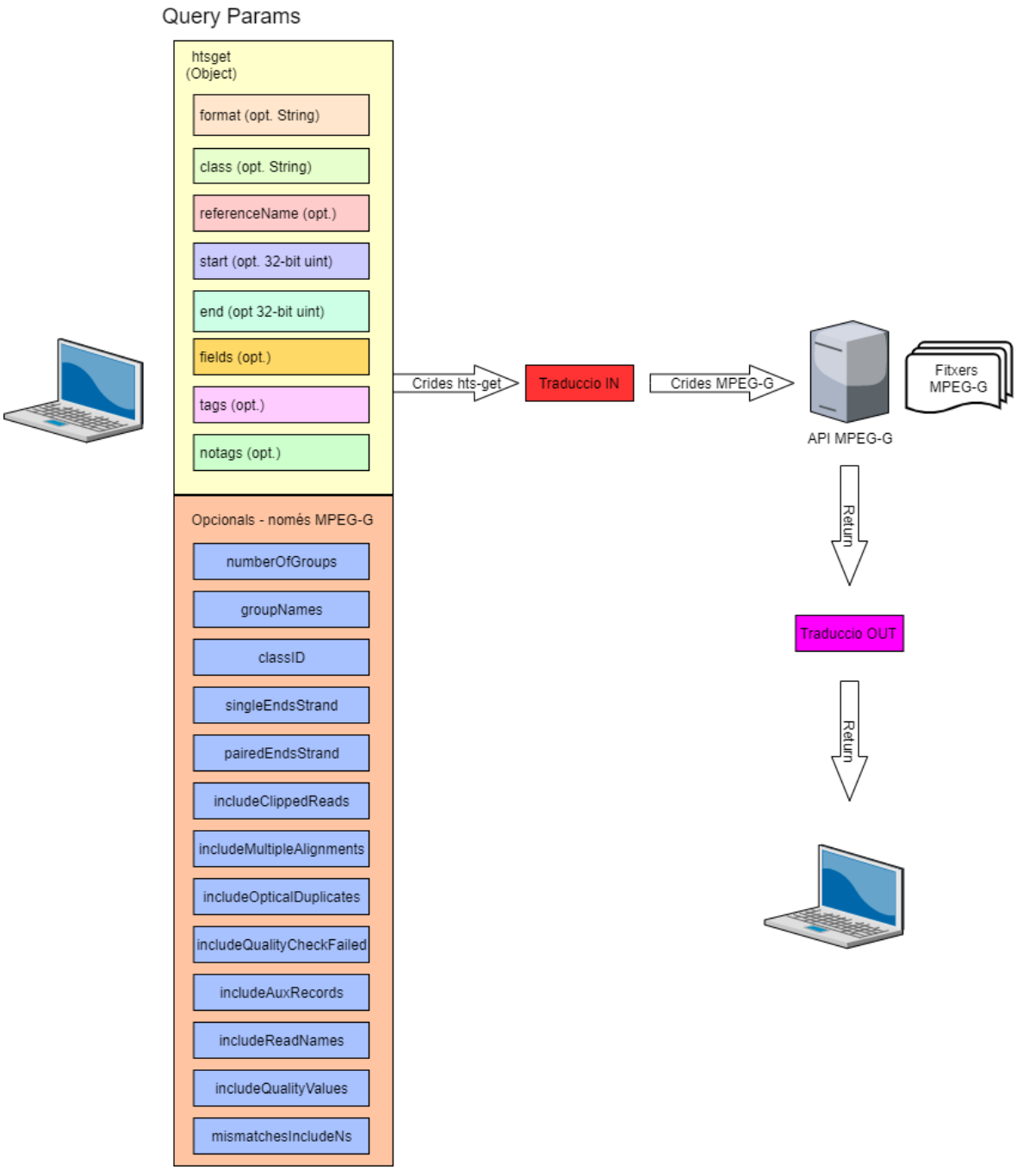


Figura H.2: Mapping hts-get a MPEG-G. (Elaboració pròpia)

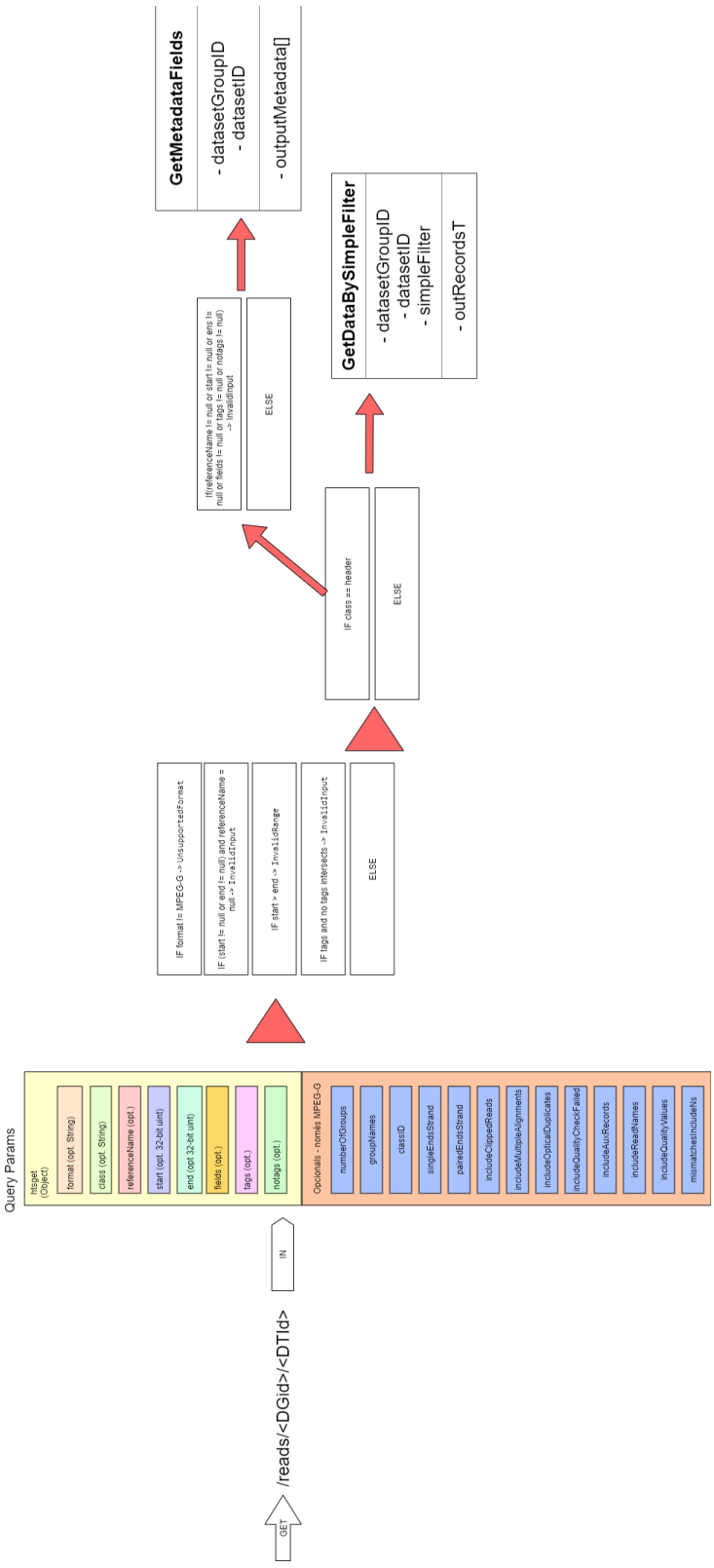


Figura H.3: Traducció entrada MPEG-G. (Elaboració pròpia)

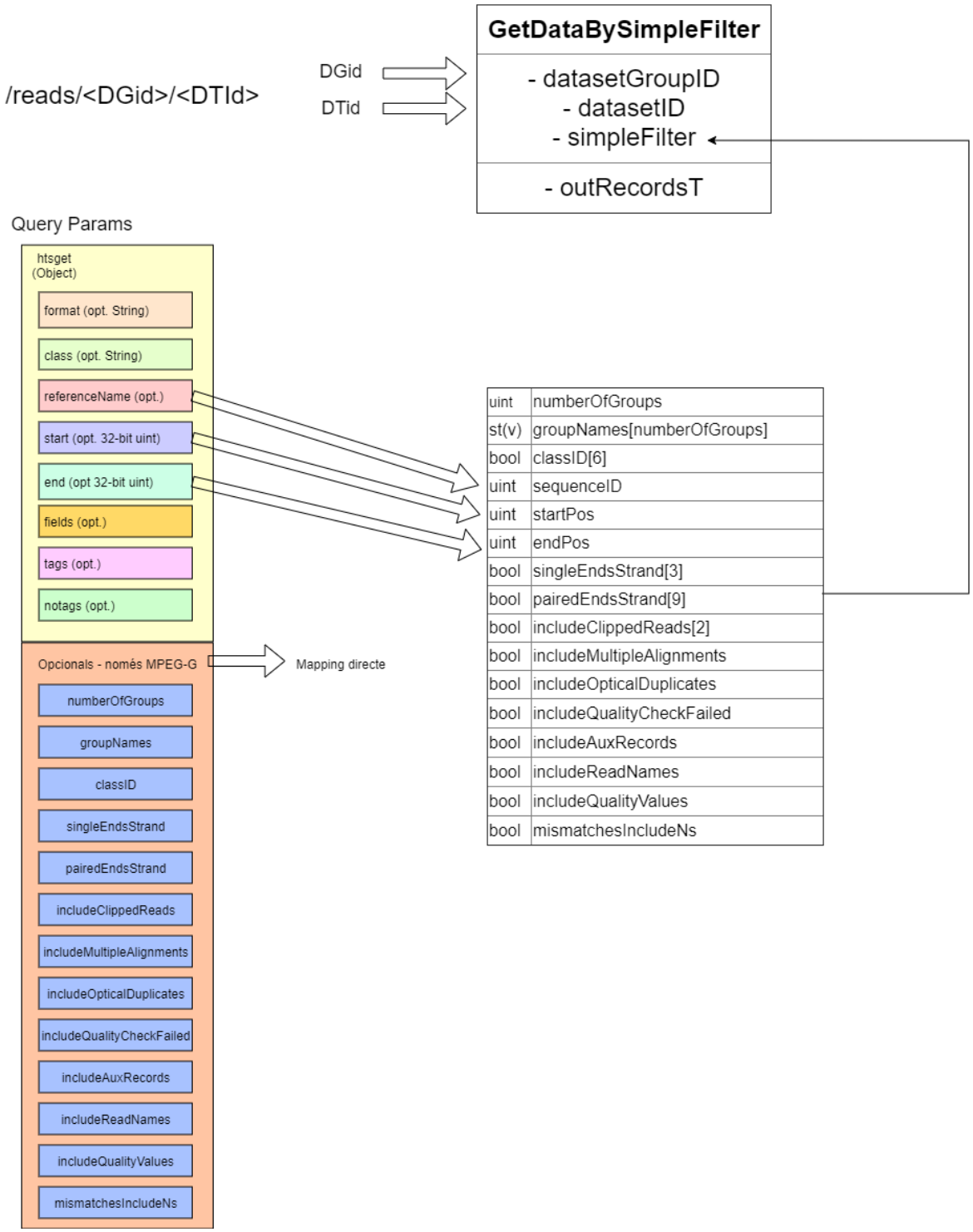


Figura H.4: Traducció entrada MPEG-G. (Elaboració pròpia)

Referències

[1] Internet Engineering Task Force (IETF). *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF) RFC 6234*. 2011. URL: https://tools.ietf.org/html/rfc6234.

[2] Abrirllave.com. *Indicadores en XSD (XML Schema)*. URL: https://www.abrirllave.com/xsd/indicadores.php.

[3] "AES - Advanced Encryption Standard". A: (2005). URL: http://www.networksorcery.com/enp/data/aes.htm.

[4] Claudio Albert et al. "An introduction to MPEG-G, the new ISO standard for genomic information representation". A: *bioRxiv* October (2018). DOI: 10.1101/426353. URL: http://biorxiv.org/content/early/2018/10/08/426353.abstract.

[5] *Algorithms- Key Sizes and Parameters Report. 2013 recommendations*. Inf. tèc. URL: https://www.enisa.europa.eu/publications/algorithms-key-sizes-and-parameters-report/at_download/fullReport.

[6] *Amazon EC2*. URL: https://aws.amazon.com/ec2/.

[7] *Art. 9 GDPR – Processing of special categories of personal data — General Data Protection Regulation (GDPR)*. URL: https://gdpr-info.eu/art-9-gdpr/.

[8] *Aspectes Socials i Mediambientals de la Informàtica*. URL: https://www.fib.upc.edu/ca/estudis/graus/grau-en-enginyeria-informatica/pla-destudis/assignatures/ASMI.

[9] *Audio — MPEG*. URL: https://mpeg.chiariglione.org/standards/mpeg-1/audio.

[10] *Awseducate starter account services*. URL: https://s3.amazonaws.com/awseducate-starter-account-services/AWS_Educate_Starter_Accounts_and_AWS_Services.pdf.

[11] T Berners i R Fielding. "Uniform Resource Identifier (URI): Generic Syntax Status". A: *Journal of Chemical Information and Modeling* (2013). URL: https://www.ietf.org/rfc/rfc3986.txt.

[12] Daniel J Bernstein. *Curve25519: new Diffie-Hellman speed records*. Inf. tèc.

[13] Daniel J Bernstein. "ChaCha, a variant of Salsa20". A: *Workshop Record of SASC* (2008). URL: http://cr.yp.to/chacha/chacha-20080120.pdf.

[14] Daniel J Bernstein. *The Poly1305-AES message-authentication code*. Inf. tèc. URL: http://cr.yp.to/mac/poly1305-20050329.pdf.

[15] *BLAKE2b — PyCryptodome 3.9.7 documentation*. URL: https://pycryptodome.readthedocs.io/en/latest/src/hash/blake2b.html.

[16] Tim Bray et al. *Namespaces in XML 1.0*. Des. de 2009. URL: https://www.w3.org/TR/REC-xml-names/.

[17] *Cipher (Java SE 11 & JDK 11)*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/Cipher.html.

[18] Lucian Constantin. "Google speeds up encrypted Web communications in Chrome on Android." A: *Good Gear Guide* (2014), pàg. 6. URL: https://www.cio.com/article/2376777/google-speeds-up-encrypted-web-communications-in-chrome-on-android.html.

[19] *Choice of authenticated encryption mode for whole messages - Cryptography Stack Exchange*. URL: https://crypto.stackexchange.com/questions/18860/choice-of-authenticated-encryption-mode-for-whole-messages.

[20] Hamlet D'Arcy. "IntelliJ IDEA". A: *RefCardz* (2009). URL: https://www.jetbrains.com/idea/.

[21] *Da Vinci Surgery — Robotic Assisted Surgery for Patients*. 2020. URL: https://www.davincisurgery.com/.

[22] Luca Del Giacco i Cristina Cattaneo. "Introduction to genomics". A: *Methods in Molecular Biology* (2012). URL: https://www.genome.gov/About-Genomics/Introduction-to-Genomics.

[23] Morris Dworkin. "Recommendation for Block Cipher Modes of Operation Methods and Techniques". A: *National Institute of Standards and Technology Special Publication 800-38A 2001 ED* December (2001). URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38a.pdf.

[24] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Inf. tèc. 2007. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf.

[25] EEES. *Sistema de Trasferencia de Créditos Europeos*. 2016. URL: http://www.eees.es/es/ects.

[26] *El Vall d'Hebron seqüencia el genoma del coronavirus, un pas bàsic per obtenir vacunes*. URL: https://www.ccma.cat/324/el-vall-dhebron-sequencia-el-genoma-del-coronavirus-un-pas-basic-per-obtenir-vacunes/noticia/3002103/.

[27] *Esperanza de vida al nacer*. 2013. URL: https://datos.bancomundial.org/indicador/SP.DYN.LE00.IN.

[28] *eXtensible Access Control Markup Language (XACML) Version 3.0*. URL: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[29] A.G Fallis. *The Transport Layer Security (TLS) Protocol Version 1.2 Status*. Ag. de 2013. URL: https://www.rfc-editor.org/info/rfc5246.

[30] *Final: OpenID Connect Core 1.0 incorporating errata set 1*. URL: https://openid.net/specs/openid-connect-core-1_0.html.

[31] S Frankel et al. "The AES-CBC Cipher Algorithm and Its Use with IPsec". A: *Ietf.Org* (2003), pàg. 1. URL: https://tools.ietf.org/html/rfc3602.

[32] *GA4GH*. URL: https://www.ga4gh.org/.

[33] *GA4GH Genomics API — GA4GH Schemas 0.0.1 documentation*. URL: https://ga4gh-schemas.readthedocs.io/en/latest/.

[34] *GanttProject*. URL: https://www.ganttproject.biz/.

[35] General Data Protection Regulation. *General Data Protection Regulation (GDPR) – Official Legal Text*. 2016. URL: https://gdpr-info.eu/.

[36] GENIE. *DNA, genes and chromosomes — University of Leicester*. URL: https://www2.le.ac.uk/projects/vgec/schoolsandcolleges/topics/dnageneschromosomes.

[37] Global Alliance for Genomics and Health (GA4GH). "GA4GH File Encryption Standard". 2019. URL: http://samtools.github.io/hts-specs/crypt4gh.pdf.

[38] Google. *Nuestros productos*. 2018. URL: https://about.google/products/.

[39] *Hashing — PyNaCl 1.3.0 documentation*. URL: https://pynacl.readthedocs.io/en/stable/hashing/.

[40] *Health data in the workplace — European Data Protection Supervisor*. URL: https://edps.europa.eu/data-protection/data-protection/reference-library/health-data-workplace_en.

[41] *htsget protocol*. URL: http://samtools.github.io/hts-specs/htsget.html.

[42] *Htsget retrieval API spec v1.2.0*. URL: https://github.com/samtools/hts-specs/blob/master/htsget.md.

[43] *Htsjdk by Broad Institute*. URL: http://samtools.github.io/htsjdk/.

[44] Takeshi Imamura et al. *XML Encryption Syntax and Processing Version 1.1*. 2013. URL: https://www.w3.org/TR/xmlenc-core1/.

[45] "ISO Standards". A: URL: https://www.iso.org/standards.html.

[46] "ISO/IEC 23092-1, Information technology — Genomic information representation — Part 1: Transport and storage of genomic information". A: (). URL: https://www.iso.org/standard/57795.html.

[47] "ISO/IEC 23092-2, Information technology — Genomic information representation — Part 2: Coding of genomic information". A: (). URL: https://www.iso.org/standard/73536.html.

[48] *ISO/IEC 23092-3 - Information technology — Genomic information representation — Part 3: Metadata and application programming interfaces (APIs)*. URL: https://www.iso.org/standard/75625.html.

[49] *Java – How to override equals and hashCode – Mkyong.com*. URL: https://mkyong.com/java/java-how-to-overrides-equals-and-hashcode/.

[50] *Java 8 vs Java 11 - What are the Key Changes?* URL: https://blog.idrsolutions.com/2019/05/java-8-vs-java-11-what-are-the-key-changes/.

[51] *Java Architecture for XML Binding (JAXB)*. URL: https://www.oracle.com/technical-resources/articles/javase/jaxb.html.

[52] *Java Security Standard Algorithm Names*. URL: https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-names.html.

[53] *javadoc*. URL: https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html.

[54] *javax.crypto (Java SE 11 & JDK 11)*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/package-summary.html.

[55] *JSON Web Token Claims*. URL: https://auth0.com/docs/tokens/concepts/jwt-claims.

[56] *JSON Web Tokens*. URL: https://auth0.com/docs/tokens/concepts/jwts.

[57] *JUnit 5*. URL: https://junit.org/junit5/.

[58] Ed. K. Moriarty et al. *RFC 8018 - PKCS #5: Password-Based Cryptography Specification Version 2.1*. 2017. URL: https://tools.ietf.org/html/rfc8018.

[59] *KeyGenerator (Java SE 11 & JDK 11)*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/KeyGenerator.html.

[60] Sonia Kotel et al. *FPGA-Based Real-Time Implementation of AES Algorithm for Video Encryption*. ISBN: 9781618042620.

[61] Langley, Hamburg i Turner. "Elliptic Curves for Security". 2016. URL: https://tools.ietf.org/html/rfc7748.

[62] *Ley de Protección de Datos en sanidad: guía recomendada para médicos*. 2017. URL: https://protecciondatos-lopd.com/empresas/guia-centros-sanitarios/.

[63] *Maven – POM Reference*. URL: https://maven.apache.org/pom.html.

[64] David A Mcgrew i John Viega. *The Galois/Counter Mode of Operation (GCM)*. Inf. tèc.

[65] MedlinePlus. *Cromosomas*. URL: https://medlineplus.gov/spanish/ency/article/002327.htm.

[66] MedlinePlus. *Genes*. URL: https://medlineplus.gov/spanish/ency/article/002371.htm.

[67] MedlinePlus. *Genética*. URL: https://medlineplus.gov/spanish/ency/article/002048.htm.

[68] *Mendeley*. URL: https://www.mendeley.com/.

[69] Mikael Linden, Craig Voisin i David Bernick. *GA4GH Authentication and Authorization Infrastructure (AAI) OpenID Connect Profile*. URL: https://github.com/ga4gh/data-security/blob/master/AAI/AAIConnectProfile.md.

[70] K Moriarty et al. *PKCS #1: RSA Cryptography Specifications Version 2.2*. Inf. tèc. 2016. URL: https://tools.ietf.org/html/rfc8017.

[71] *MP4 File Format — MPEG*. URL: https://mpeg.chiariglione.org/standards/mpeg-4/mp4-file-format.

[72] MPEG. *The Moving Picture Experts Group Website*. 2013. URL: https://mpeg.chiariglione.org/.

[73] *MPEG-1 — MPEG*. URL: https://mpeg.chiariglione.org/standards/mpeg-1.

[74] *MPEG-4 — MPEG*. URL: https://mpeg.chiariglione.org/standards/mpeg-4.

[75] *MPEG-G Genomic Information Representation and Transport*. URL: https://mpeg-g.org/.

[76] N. Sakimura et al. *Draft: OpenID Connect Implicit Client Implementer's Guide 1.0 - draft 21*. 2017. URL: https://openid.net/specs/openid-connect-implicit-1_0.html.

[77] *nacl.hash — PyNaCl 1.3.0 documentation*. URL: https://pynacl.readthedocs.io/en/stable/api/hash/#nacl.hash.blake2b.

[78] Zakaria Najm et al. "On Comparing Side-channel Properties of AES and ChaCha20 on Microcontrollers". A: *2018 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2018*. Institute of Electrical i Electronics Engineers Inc., gen. de 2019, pàg. 552 - 555. ISBN: 9781538682401. DOI: 10.1109/APCCAS.2018.8605653.

[79] Daniel Naro. "Security strategies in genomic files". A: (2020). URL: https://futur.upc.edu/28491842.

[80] Yoav Nir i Adam Langley. *RFC 8439 - ChaCha20 and Poly1305*. 2018. URL: https://tools.ietf.org/html/rfc8439.

[81] *Normal Approximations - Mathematics A-Level Revision*. URL: https://revisionmaths.com/advanced-level-maths-revision/statistics/normal-approximations.

[82] *OAEPParameterSpec (Java SE 11 & JDK 11)*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/spec/OAEPParameterSpec.html.

[83] Ziad Obermeyer i Ezekiel J. Emanuel. *Predicting the future-big data, machine learning, and clinical medicine*. Set. de 2016. URL: www.ncbi.nlm.nih.gov/pmc/articles/PMC5070532/.

[84] Overleaf team. *Overleaf, Online LaTeX editor*. 2019. URL: https://www.overleaf.com.

[85] *Overview (Java SE 11 & JDK 11)*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/index.html.

[86] Mireia Pagès Guitart. *Apunts assignatura GENÈTICA MOLECULAR del Grau de Ciències Biomèdiques*. 2019.

[87] *Public Key Encryption — PyNaCl 1.3.0 documentation*. URL: https://pynacl.readthedocs.io/en/stable/public/.

[88] Python Software Foundation. *hashlib — Secure hashes and message digests — Python 3.7.3 documentation*. 2019. URL: https://docs.python.org/3/library/hashlib.html.

[89] *Random (Java SE 11 & JDK 11)*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Random.html.

[90] *reads — GA4GH Schemas 0.0.1 documentation*. URL: https://ga4gh-schemas.readthedocs.io/en/latest/schemas/reads.proto.html.

[91] *Recital 34 - Genetic Data — General Data Protection Regulation (GDPR)*. URL: https://gdpr-info.eu/recitals/no-34/.

[92] *Recital 35 - Health Data — General Data Protection Regulation (GDPR)*. URL: https://gdpr-info.eu/recitals/no-35/.

[93] *Recital 51 - Protecting Sensitive Personal Data — General Data Protection Regulation (GDPR)*. URL: https://gdpr-info.eu/recitals/no-51/.

[94] *Recital 53 - Processing of Sensitive Data in Health and Social Sector — General Data Protection Regulation (GDPR)*. URL: https://gdpr-info.eu/recitals/no-53/.

[95] *Recital 54 - Processing of Sensitive Data in Public Health Sector — General Data Protection Regulation (GDPR)*. URL: https://gdpr-info.eu/recitals/no-54/.

[96] *RFC 6749 - The OAuth 2.0 Authorization Framework*. URL: https://tools.ietf.org/html/rfc6749.

[97] *RFC 6819 - OAuth 2.0 Threat Model and Security Considerations*. URL: https://tools.ietf.org/html/rfc6819.

[98] *RFC 7515 - JSON Web Signature (JWS)*. URL: https://tools.ietf.org/html/rfc7515.

[99] *RFC 7519 - JSON Web Token (JWT)*. URL: https://tools.ietf.org/html/rfc7519.

[100] "RFC 7539 - ChaCha20 and Poly1305 for IETF Protocols". A: (2015). ISSN: 2070-1721. URL: https://tools.ietf.org/html/rfc7539%20http://www.rfc-editor.org/info/rfc7539..

[101] *RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3*. URL: https://tools.ietf.org/html/rfc8446.

[102] Ed. Saarinen M-J. i Jean Philippe Aumasson. *The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)*. 2015. URL: https://tools.ietf.org/html/rfc7693.

[103] J Schaad i R Housley. *[RFC3394] Advanced Encryption Standard (AES) Key Wrap Algorithm*. Inf. tèc. 2002. URL: https://tools.ietf.org/html/rfc3394.

[104] *SealedObject (Java SE 11 & JDK 11)*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/SealedObject.html.

[105] Segu-Info. *Protocolos ChaCha20 y Poly1305 para reforzar conexiones HTTPS*. URL: https://blog.segu-info.com.ar/2014/04/protocolos-chacha20-y-poly1305-para.html.

[106] *Sensors — Free Full-Text — PRISEC: Comparison of Symmetric Key Algorithms for IoT Devices — HTML*. URL: https://www.mdpi.com/1424-8220/19/19/4312/htm.

[107] *Serialization in Java*. URL: https://www.javahelps.com/2015/07/serialization-in-java.html.

[108] *Standard Algorithm Name Documentation*. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html.

[109] Petr Svenda. "Basic comparison of Modes for Authenticated-Encryption". A: (2004). URL: https://www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf.

[110] *The Illustrated TLS 1.3 Connection: Every Byte Explained*. URL: https://tls13.ulfheim.net/.

[111] *The Illustrated TLS Connection: Every Byte Explained*. URL: https://tls.ulfheim.net/.

[112] The Moving Picture Experts Group. *MPEG-G — MPEG*. 2017. URL: https://mpeg.chiariglione.org/standards/mpeg-g.

[113] The SAM/BAM Format Specification Working Group. "Sequence Alignment/Map Format Specification". A: (2020). URL: https://samtools.github.io/hts-specs/SAMv1.pdf.

[114] *Treball de Fi de Grau — Facultat d'Informàtica de Barcelona*. 2019. URL: https://www.fib.upc.edu/ca/estudis/graus/grau-en-enginyeria-informatica/treball-de-fi-de-grau.

[115] S Turner i D Brown. "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)". A: (2010). URL: http://tools.ietf.org/html/rfc5753.

[116] S Turner i L Chen. "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms". A: *RFC* (2011). URL: https://tools.ietf.org/html/rfc6151.

[117] S Turner, A. Langley i M. Hamburg. "RFC: 7748 - Elliptic Curves for Security This". A: (2016), pàg. 1 - 22. URL: https://tools.ietf.org/html/rfc7748.

[118] Tutorials Point (I) Pvt. Ltd. "XML Tutorial". A: (2017), pàg. 1 - 13. URL: https://www.w3schools.com/xml/default.asp.

[119] UNIVERSITAT de VALÈNCIA. *Modelo Uniforme Continuo*. URL: https://www.uv.es/ceaces/base/modelos%20de%20probabilidad/uniforme.htm.

[120] *UPC - UPC Universitat Politècnica de Catalunya*. URL: https://www.upc.edu/ca.

[121] Facultat d'Informàtica de Barcelona - UPC. *Apunts assignatura Mineria de Dades*. 2019. URL: https://www.fib.upc.edu/ca/estudis/graus/grau-en-enginyeria-informatica/pla-destudis/assignatures/MD.

[122] Facultat d'Informàtica de Barcelona - UPC. *Apunts assignatura Probabilitat i Estadística*. 2017.

[123] UNIVERSITAT de VALÈNCIA. *Distribución Binomial*. URL: https://www.uv.es/ceaces/base/modelos%20de%20probabilidad/binomial.htm.

[124] Viquipèdia. *Àcid desoxiribonucleic*. URL: https://ca.wikipedia.org/wiki/%C3%80cid_desoxiribonucleic.

[125] Viquipèdia. *Àcid nucleic*. URL: https://ca.wikipedia.org/wiki/%C3%80cid_nucleic.

[126] Viquipèdia. *Cromosoma*. URL: https://ca.wikipedia.org/wiki/Cromosoma.

[127] Viquipèdia. *Cromosoma X*. URL: https://ca.wikipedia.org/wiki/Cromosoma_X.

[128] Viquipèdia. *Gen*. URL: https://ca.wikipedia.org/wiki/Gen.

[129] Viquipèdia. *Genètica*. URL: https://ca.wikipedia.org/wiki/Gen%C3%A8tica.

[130] Viquipèdia. *Genoma humà*. URL: https://ca.wikipedia.org/wiki/Genoma_hum%C3%A0.

[131] Viquipèdia. *Hipòcrates*. URL: https://ca.wikipedia.org/wiki/Hip%C3%B2crates.

[132] Viquipèdia. *Medicina*. URL: https://ca.wikipedia.org/wiki/Medicina.

[133] Viquipèdia. *Microscopi*. URL: https://ca.wikipedia.org/wiki/Microscopi.

[134] Viquipèdia. *Nucli cel·lular*. URL: https://ca.wikipedia.org/wiki/Nucli_cel%C2%B7lular.

[135] W3Schools. *XML schema Element*. URL: https://www.w3schools.com/xml/schema_schema.asp.

[136] Kris A. Wetterstrand. *The Cost of Sequencing a Human Genome — NHGRI*. 2020. URL: https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost.

[137] Wikipedia. *Block cipher mode of operation*. URL: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation.

[138] Wikipedia. *KLV*. URL: https://en.wikipedia.org/wiki/KLV.

[139] Wikipedia. *Moving Picture Experts Group*. URL: https://es.wikipedia.org/wiki/Moving_Picture_Experts_Group.

[140] Wikipedia. "Uniform Resource Identifier (URI)". A: URL: https://en.wikipedia.org/wiki/Uniform_Resource_Identifier.