Security strategies in genomic files

Daniel Naro
Advisors: Jaime Delgado and Silvia Llorente

DOCTORAT EN ARQUITECTURA DE COMPUTADORS
UPC Departament d'Arquitectura de Computadors
February 11, 2020

Security strategies in genomic files

Daniel Naro
Advisors: Jaime Delgado and Silvia Llorente

DOCTORAT EN ARQUITECTURA DE COMPUTADORS

UPC Departament d'Arquitectura de Computadors

February 11, 2020

A la L9 del metro,
L9, tu havies d'estar acabada quan fes primer de batxillerat, ara entrego la tesis i encara no estàs acabada... Mira que m'haguessis anat bé aquests anys

Quiero dar las gracias a mis directores de tesis, Dr. Jaime Delgado y Dr. Silvia Llorente, por la oportunidad de hacer esta tesis, y por el soporte e ideas para llevarla a cabo.

También quiero dar las gracias a los miembros de la mitad BSC del proyecto: Josep Lluís Gelpí, Dmitry Repchevski, Romina Royo y Laia Codó. No solo por la oportunidad de completar mi trabajo implementando la otra parte de MPEG-G, pero también por resolver mis dudas y darme material para mejorar mi tesis.

I would like to thank the entirety of the MPEG group working on MPEG-G, and specially the "Illini group": Idoia Ochoa, Mikel Hernaez, Tom Paridaens and Jan Voges for the opportunity to cooperate in a larger endeavor.

I want to thank Dr. Manel Medina, Dr. Rubèn Tous, Dr. Jordi Domingo, Dr. Idoia Ochoa and Dr. Ramon Martí for the valuable corrections and feedbacks.

Vull donar-li les gràcies a la Cristina per haver-me donat l'empenta que em feia falta per llençar-me a fer la tesis.

Et je veux remercier ma famille pour le support pendant tout ce temps. Und insbesondere für die Korrekturen.

Contents

Chapter 1

Introduction

There are some aspects of genomic information that are almost universally known:

- We, humans, inherit our genetic code from our parents.
- It encodes how our bodies build and maintain themselves.
- Our genetic code is unique to us.

We see these properties of the genome, for example, in the way we inherit physical features from our parents, in the existence of genetic diseases, and in the uses forensic science makes of genetic markers for identification purposes.

The development of technologies to read the genetic sequence has opened the door to new uses. For example:

- A medical practitioner can prescribe a genome sequencing for a patient to diagnose a disease.
- A researcher can compare the genome of many individuals to find the common traits related to a specific illness.
- An individual can use a service that establishes the origins of this particular person based on the distribution of known mutations around the world.

In the past years, a sharp decrease in the price of genome sequencing has led to an increase in the amount of generated information. The quantity of information generated during a genome sequencing depends on its purpose, but it can realistically range from some gigabytes to some hundreds of gigabytes. The result of this dynamic is the need for new compression mechanisms which will make it feasible to sustain the current growth rate.

As we have seen, there are various uses for the data of a sequenced genome: it can be relevant to multiple research studies, to diagnostics purposes, and to personalized medicine. To maximize the reusability potential, there is the need to have a standardized way of representing genomic information. Two of the working groups within the International Organization for Standardization (ISO) [1], MPEG [2] and the Data processing and integration group within the Biotechnology technical committee [3], have identified the necessity for a new standardized compressed representation of genomic information.

The usefulness of the genomic information is also a source of danger. If a bad actor gains access to an individual's genomic information, facts which have to remain secret such as diseases or predisposition to them can be revealed. In fact, due to the inheritability of the genomic information, not only does this attack affect the individual but also the individual's blood relatives. The sensitivity of the genomic information has led the MPEG group to include the protection of the genomic information as a requirement for the new standard.

The here presented work contributes to answering how to protect the genomic information in the framework of the new standard. The framework places specific requirements on the answer:

- The main goal is the usability of the file. Due to the nature of the data, in order for the data to be usable, the standard must provide the best possible compression. We will consider the compression strategy as fixed.
- The standard must be usable in current genomic pipelines. We cannot remove the ability to perform actions that are available in current formats.
- The standard shall only rely on existing technologies that have already been specified in a standard.

The question of how to protect the genomic information depends on other questions that we must address, although they are not the primary target. We need to understand the nature of the data to protect, how to represent it, and which actions a user can perform on it.

Chapters 2, 3 and 4 analyze the context of the work and the parts of the specification we will consider as prior work. These chapters present the nature of the data and how MPEG intends to represent it.

Once the data is encoded, a file format must structure it. Chapter 5 describes our proposal for the format.

To facilitate further reusability of the data, it must also be accompanied by metadata. At the onset of the here presented work, there is no proposed representation for the metadata. The metadata can be as sensible as the genomic information, for example, due to information about the diagnosed diseases. Chapter 6 explains our proposal for metadata representation, and for a mechanism to secure it.

Chapter 7 also analyzes a possible attack on a security mechanism that was contributed to MPEG. Furthermore, we present a mechanism to enable leaker identification. We can use this mechanism to protect any genomic format. This leaker identification solution addresses a problem not covered by the other parts of this work.

Finally, Chapter 8 analyzes how the new standard is used: we discuss our proposal for an Application Programming Interface (API) and for a way to control who has access to which action. Furthermore, we propose an alternative structuring of the data which optimizes API methods while maintaining region-based encryption.

Chapter 2

Context

This chapter presents some of the characteristics of sequenced genomic data. It includes the main aspects of sequencing technologies and the alignment of the information to a reference genome. It summarizes the representation of the resulting information.

At the macroscopic level, our human level, we are used to bodies. Bodies are the sum of systems (e.g., locomotive system, digestive system), which are each composed of multiple organs. We can further divide each organ into tissues, which are in their turn formed by cells. In order for the body to perform correctly, the different systems need to perform metabolic processes. The cells need to be programmed to perform these actions. Deoxyribonucleic acid (DNA) molecules store the code specifying these actions. We refer to these DNA molecules as chromosomes. In a healthy human body, there are 23 pairs of chromosomes. At specific time points of the lifecycle of the cell, the chromosomes are observable: at this point, we can take a picture of them and group them by type. Figure 2.1 shows such a picture, a so-called *karyotype*. In the example, it is from a female human. In diploid lifeforms, such as the human species, there are two chromosomes of each type: one from the mother, the other from the father. In a healthy patient, all cells share the same chromosomes. However, depending on the circumstances (e.g., gender, type of cells), only certain regions of the chromosomes will be active.

A succession of nucleotides forms the DNA molecule. For this work, we can simplify our conception of nucleotides and consider them as elementary building blocks of the DNA. The sequence of these building blocks represents the code that our cells are following. There are four different nucleotides: adenine, cytosine, guanine and thymine. These nucleotides are represented usually by their initials: A, C, G and T. During its lifecycle, the cell reads and translates the DNA into other molecules. The cell does not translate the entire DNA into molecules: certain regions are non-coding.

Organisms translate the DNA into molecules by following the genetic code. The DNA molecule is read in units of three nucleotides, referred to as *codon*. Each codon is translated into one amino acid, as shown in Figure 2.2. The resulting succession of amino acids is the result of translating the DNA into a new molecule (e.g., insulin).

2.1 Inheritance of genetic material and mutations

Organisms using sexual reproduction produce sexual cells through a process known as *meiosis*. The genetic material is duplicated and then divided twice. The result is four sexual cells. Sexual cells have only one copy of each chromosome, either a copy of the genetic material of the mother or of the father. Upon fecundation, this will lead to an egg with the usual pair of chromosomes.

The inherited copies are, however, altered. During the duplication process of the genetic material, errors can occur: an error can substitute individual nucleotides (e.g., a nucleotide A replaces a nucleotide C), can insert nucleotides, can delete nucleotides, or can copy nucleotide subsequences multiple times. The process can even relocate nucleotide subsequences to other locations. Due to the quantity of nucleotides, it must be assumed that errors will occur.

During the division process, the copies exchange genetic material. The copies are not any more entirely from the father or the mother: it is still mainly a copy of the genetic code from either one, but it contains genetic material from the other.

As the cell translates the genetic code into molecules, alterations in the code can cause alterations in the resulting molecules. These differences could imply advantages or disadvantages as the mutations improve or deteriorate the sequenced molecules.

By analyzing the genome of as many individuals as possible [4, 5, 6], researchers have published, and continue publishing at regular intervals, a reference genome of the human species. This reference genome

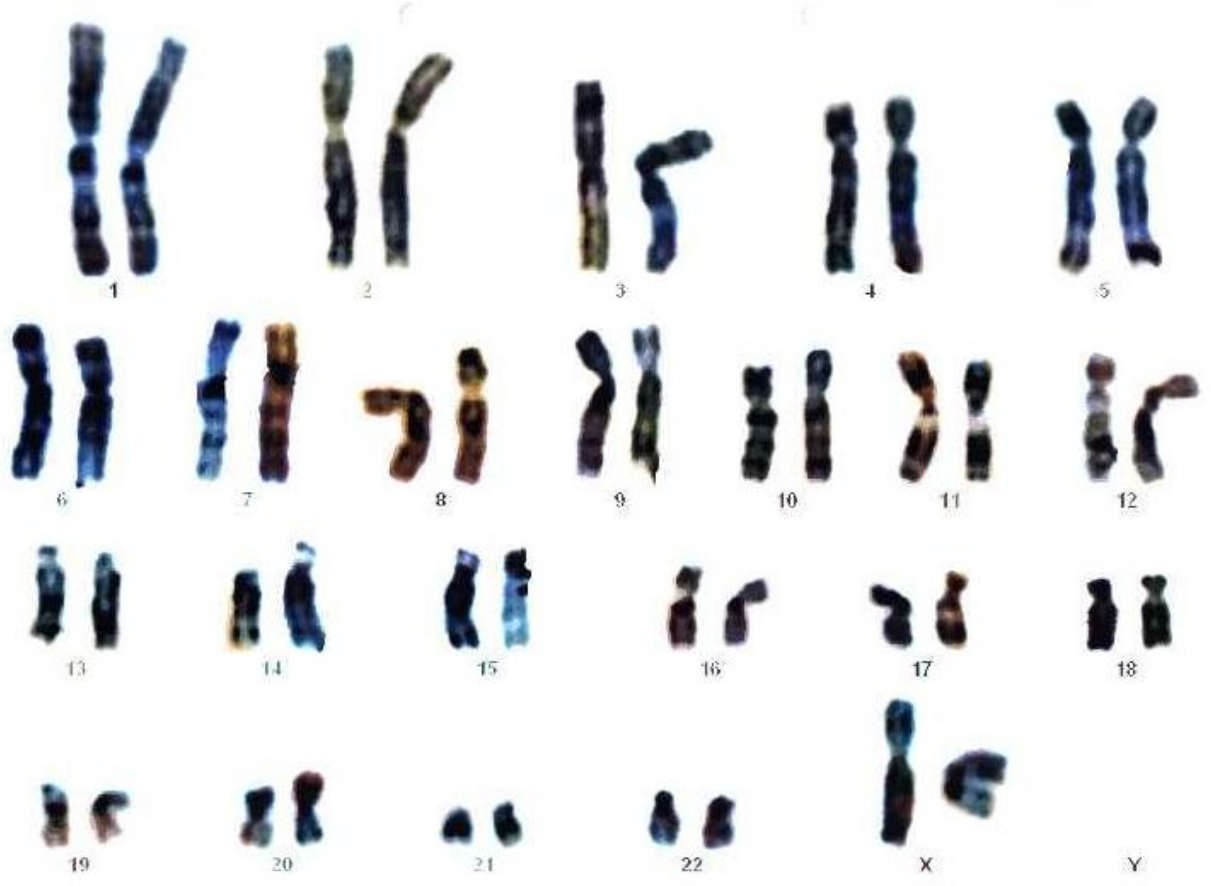


Figure 2.1: Karyotype of a female human From the wikimedia https://commons.wikimedia.org/wiki/File:Mapa_genético_o_cariograma.jpeg

Standard genetic code

1st base	2nd base								3rd base
	T		C		A		G		
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA	Stop (*Ochre*)[B]	TGA	Stop (*Opal*)[B]	A
	TTG[A]		TCG		TAG	Stop (*Amber*)[B]	TGG	(Trp/W) Tryptophan	G
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG[A]		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

Figure 2.2: Genetic code From the wikipedia https://en.wikipedia.org/wiki/DNA_codon_table

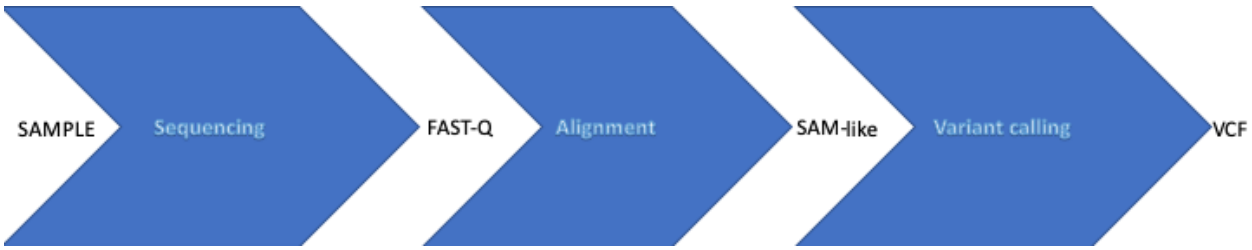


Figure 2.3: The pipeline to analyze the genome

represents the most usual sequence of nucleotides. Any human will have almost the reference genome, but with some key differences. The set of differences an individual presents is unique. It corresponds to the combination of the differences inherited from the mother and the father and any new mutations.

In order to understand a genetic disease, we will compare the genome of each patient with the reference genome. We can also compare the genome of a cell involved in a disease with the genome of a healthy cell of the same patient. In this case, we discover a somatic change in the genome: a difference with the reference genome not present in other cells. The exposure to external factors (e.g., radiation, smoke) can cause these new differences. The study of these differences is relevant to understand the cause of certain cancers [7].

2.2 Genes and alleles

We have seen how we inherit mutations from our parents. Furthermore, we have seen that the genome encodes the metabolism of the cell: the proteins the cell generates and at which rate. Geneticists have defined the concept of gene: a unit of inheritable material the cell can transcribe into a protein or which the cell uses to regulate the expression of other genes. Researchers have estimated that there are 21000 genes in the human genome [8].

Since we inherit some of our parents' mutations, there are, throughout the population, multiple instances of the same mutation. Therefore, there are well-known variants of the nucleotides sequence corresponding to a gene. We will refer to each of these variants as an *allele*. The effects of some of these alleles are familiar to us: some of the common examples are the shape of the thumb, the hair color, the blood type. In some cases, both copies of the gene (a copy corresponding to a chromosome) must share the same allele so that the effect appears (we refer to this as *homozyguous*), in other cases the effects of the mutation are already visible if there is only one copy of the mutated allele (we refer to this as *heterozyguous*).

The frequency at which each allele appears throughout the population is different. The study of the different alleles, their effects and frequencies has led to the creation of databases [9] documenting the currently known mutations in the human population and the diseases to which they might be linked.

2.3 Sequencing the genome

In order to understand the genome of a specimen, we follow steps in a pipeline as represented in Figure 2.3. First we must sequence its genome [10, 11]. To do that, we use genome sequencing devices, such as Illumina devices [12].

A biological sample is treated chemically in order to obtain many fragments of DNA. A sequencing machine then reads these fragments. This type of device sequences the genome by reading subportions of it: from the previously mentioned DNA fragments, the machine reads contiguous nucleotides. The sequencing technology used determines the length of this subsequence. It is unknown to which genomic region the subsequence corresponds. Some sequencing technologies generate two of these subsequences at a time. In that case, we refer to them as *paired reads*, and we know that the two subsequences correspond to close locations on the genome. Besides identifying the nucleotides, the sequencing machine also attributes a quality grade to each nucleotide. This grade corresponds to the confidence with which the sequencing machine has called the nucleotide.

The result of sequencing the genome is a list of (paired) reads with their qualities. This information is represented usually in a FASTQ file [13], which is formatted as follows.

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=36
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGA
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=36
IIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBI
```

For each read, there is:

1. a line starting with '@' stating the name of the read followed by some other optional information
2. a line containing the called nucleotides in order
3. a line with a '+' marking the end of the nucleotides line and the beginning of the quality information (if applicable, accompanied by optional information)
4. a line containing the quality symbols, one per nucleotide.

2.4 Aligning the genome

Once the sequencing machine has generated enough reads, the information can be combined to reconstruct the genome of the individual. We refer to this step as *alignment*. A program aligns each of the reads to the reference genome. The process is performed either without a reference genome (a *de novo alignment*) or with a reference genome. The objective is to determine each read's position (i.e., the chromosome and the position on it to which the read corresponds).

Locating the read on the genome also allows to obtain further information. We might detect that the read aligns well to different locations of the genome, thus requiring multiple alignments. We can give a quality grade to each of the mappings. The aligner can also recalibrate the sequencing qualities of each nucleotide when taking into account all reads mapped to the same location.

The default format to represent aligned information is the Sequence Alignment/Map (SAM) format [14], which also has a binary and compressed counterpart, the Binary Alignment Map (BAM) format. This format extends the information provided in the FASTQ file with the information generated in the alignment phase. This new information includes, but is not limited to, the read's position, the position of the read paired to the current read if such is the case, the quality of the mapping, and any differences with the original genome.

These differences are represented in a list, encoded in a so-called *CIGAR string*[14] (Compact Idiosyncratic Gapped Alignment Report). Each entry in the list is composed of a number (the operation's length) and a letter (the operation's type). The usual types of operations are the following:

- A mapped operation is represented with the character 'M'. Each nucleotide associated with the operation maps to a nucleotide in the reference genome. The nucleotides in the read and in the reference genome are not necessarily the same. In order to differentiate between the two cases (i.e., the case where there is no evidence of mutation and the case where there is evidence for a mutated nucleotide), new versions of the CIGAR string specification have subdivided this operation into an operation '=' and an operation 'X', corresponding to equal and different to reference genome respectively.
- An insertion operation is represented with the character 'I'. The corresponding nucleotides of the read do not appear on the reference genome, i.e., there is evidence that a mutation has inserted nucleotides in the individual's genome when compared to the reference.
- A deletion operation is represented with the character 'D'. The corresponding nucleotides of the reference do not appear in the read, i.e., there is evidence that a mutation has deleted nucleotides in the individual's genome.
- A softclip operation is represented with the character 'S'. It can only appear as the first or the last operation and indicates that the corresponding nucleotides of the read are different from those of the reference.

Table 2.1 shows an example CIGAR. In this instance, we have aligned a fake read to a fake reference. The meaning of the values of the CIGAR operations row in Table 2.1 are as follows:

- The first two nucleotides of the sequence match the ones from the reference (i.e., we have two M operations represented by =, because they are equal to the reference).

- A nucleotide is inserted (i.e., we have an I operation).
- A nucleotide matches the reference (M operation).
- A nucleotide from the reference is not present in the read (D operation).
- A nucleotide matches the reference (M operation).
- A nucleotide in the read replaces the nucleotide in the reference (M operation represented with X because the nucleotide differs from the reference).

Finally, all operations are collapsed into one representation indicating the type of operation and the number of nucleotides to which the operation applies, as shown in the 'Read CIGAR' row in Table 2.1.

Table 2.1: CIGAR example

Reference (example chromosome)	A	C	T		G	A	C	T	G	A	C	T	G
Read		C	T	A	G		C	A					
CIGAR operations		=	=	I	=	D	=	X					
Read's CIGAR (as stored in SAM)	2=1I1=1D1=1X												

An example of SAM file is provided next.

```
HS25_09827:2:2215:4133:22216#49 99 1 10001 2 100M = 10028 126
    ↪ TAACCCTAACCCTAACCCGAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAAC AA>C=?F?I=GGEEGFBA\%?
    ↪ GFHEHFFHGDEHCGEE8AFCCHGIFCDC;HBFFD7EGG:FDI-D8DF@G?FB6JFE6E7CEH\&C55FB@@@)C3)6)37E X0:i:1 X1:i:333 a3:i:3 BC:Z:NGTCTATC MD:Z:18T81 RG
    ↪ :Z:1#49 XG:i:0 AM:i:0 NM:i:1 SM:i:0 XM:i:1 XO:i:0 QT:Z:!1:BBA?D XT:A:U
HS25_09827:2:1212:15822:94146#49 163 1 10002 17 2S12M1I12M1I71M1S = 10358 455
    ↪ TAAACCCTAACCCTAAACCCTAACCCTAAACCCTAACCCTAACCCTAACCCTAACCCTAGCCCTAGCCCTAGCCCTAGCCCTAGCCCTAGCCCTAACCCC ;
    ↪ CBCDDGHBGFGBGEGHIE7DFHIIDFIIEGH>6G7HFD5HFHDCDEGDBF+AD>E4,B7=F7?5+555+A+,B=55=,C4+=+;;=4+4+5F++>,G42 MD:Z:55A5A5A5A5A5A9 RG:Z:1#49 XG
    ↪ :i:2 AM:i:17 NM:i:8 SM:i:17 XM:i:6 XO:i:2 XT:A:M
HS25_09827:2:1209:9304:17097#49 99 1 10004 15 18M1I6M1I6M1I58M4D9M = 10266 361
    ↪ CCCTAACCCTAACCCTAACCCCTAACCCCTAACCCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTCTAACCCTA
    ↪ CAEEFGEFIFGFGJEEGGGHGHIGHJHHGHEHHHEFF97JEHEE@GHCCFEICF@EFGGF8AGHFD7G7HFG,CCDFG7?DGID5F>GEE4G45+FBHA* BC:Z:NGTCTATC MD:Z:88^AACC9 RG:
    ↪ Z:1#49 XG:i:7 AM:i:15 NM:i:7 SM:i:15 XM:i:0 XO:i:4 QT:Z:!1=BDDFF XT:A:M
HS25_09827:2:2309:1998:52482#49 99 1 10004 29 2S98M = 10311 406
    ↪ ACCCCTAACCCTAACCCTAACCCTAACCCTAACCCTAGCCCTAGCCCTAGCCCTAGCCCTAGCCCTAGCCCTAGACCTAGCCCTACCCCTAACCCTAACC B@BCF?FGJFGE@E?EG>FFGF>EECDB@-
    ↪ ECG:E:F\&77E9G-F8@.;:,;C7F,,9C@@9-?@(C,-'7,6,(??,-(,GICH,>B@,4,\$@=+(F,B BC:Z:NGTCTATC MD:Z:35A5A5A5A5A5A5A0C4A5A14 RG:Z:1#49 XG:i:0
    ↪  AM:i:29 NM:i:10 SM:i:29 XM:i:10 XO:i:0 QT:Z:!11BDFFF XT:A:M
HS25_09827:2:2311:5542:13577#49 99 1 10005 25 100M = 10028 123
    ↪ CCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTACCCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTA AA>
    ↪ CFGDGAFEDCHFHGEGEGHIEEE@I8HHDGIEGFEBCDHHDFFD?C?AD,HF8C@:G-AC78EF@7E7FA77DBB77,G7,55,C6@@7+55=4=F; X0:i:331 BC:Z:NGTCTATC MD:Z:52A47
    ↪ RG:Z:1#49 XG:i:0 AM:i:0 NM:i:1 SM:i:0 XM:i:1 XO:i:0 QT:Z:!1:ADDAF XT:A:R
```

If we take the first line, we see, among other things, the following elements:

1. the read name: HS25_09827:2:2215:4133:22216#49
2. the chromosome to which the read is aligned: 1
3. the position to which the read is aligned: 10001
4. the CIGAR string: 100M
5. the chromosome to which the mate read is aligned: in this case it is the same and is therefore marked with '='
6. the position to which the mate read is aligned: 10028
7. the nucleotides of the read : 'TAACCCTAACCCTAACC[...]CCTAACCCTAACCCTAAC'.

2.5 Calling variants

Based on the information gathered during the alignment phase, we can analyze the genome in search of mutations. We refer to this process as *calling variants*. The main issues faced during this operation are possible errors in the alignments (the aligner wrongly placed a read on the genome), the sequencing machine misread a position [15, 16], or there is not enough information for a given position.

By combining all available information, the variant caller determines which mutations are present (i.e., their position and type) and whether the mutations are present on both chromosomes or just one.

Chapter 3

State of the art

In this chapter we analyze the state of the art in the different fields relevant to this thesis: current encoding strategies and their respective file formats; encryption strategies, repository protection strategies, and watermarking technologies for genomic data; and for each of the main genomic data repositories its respective metadata schemas and the associated semantics.

3.1 Current formats

3.1.1 Binary Alignment Map

The BAM format [14] is the binary and compressed counterpart of SAM. Each read is stored separately in binary format. The binary information is to be read from a unique stream compressed using Blocked GNU Zip Format (BGZF), a compression mechanism similar to GNU Zip (GZIP) [17]. This compression mechanism creates blocks of data which are each compressed independently of one another.

Although this should mean that BAM offers out-of-the-box capabilities to skip portions of the file, the bytes corresponding to a given read can end up being stored in more than one block: if placed close to the end of a block, the information will be stored there until the block is full; the remaining information will be stored in the next block. If a read is too long, i.e., if its binary representation exceeds the maximal size of uncompressed information for a block, it might be stored in three or more blocks.

In other to circumvent this problem and offer random access capabilities, a BAM file can be accompanied by a Binary Alignment Index (BAI) file [14]. This file indexes the information stored in the BAM file by creating a tree of bins. The read is associated to the finest grained bin where the read is entirely contained. For each read the BAI file then stores the virtual offset position: this virtual offset comprises both the real offset within the file (i.e., the position of the first byte of the block relative to the beginning of the file) and the first byte of the read within the decompressed block storing it. This approach avoids the issue of not knowing where each read starts if not reading from the beginning of the file.

3.1.2 Compressed ARchiving for GenOmics

In the SAM structure, there are multiple fields per record. Each one of these fields uses different types of data: ASCII-character arrays, integers, nucleotide strings among others. By compressing all of this information into one unique stream, we are losing the possibility to use our knowledge of the different fields to optimize the compression.

Compressed ARchiving for GenOmics (CARGO) [18] addresses this issue by allowing the user to define how the information is interpreted. A metalanguage is used to specify the different fields of information, their elementary data types and the algorithms used for their compression. This metalanguage is then translated to an actual program. CARGO is not a specification per se, but upon sharing the configuration defined in the metalanguage, the content of the file can be understood and returned to its original format.

As mentioned previously, the encoding process divides the input information into the different fields defined using the metalanguage. Each of these new streams of information is then compressed as specified. The compression occurs in blocks.

3.1.3 Compressed Reference-oriented Alignment Map

As we have said, the majority of any individual genome is the same as the reference genome. Therefore only those nucleotides which differ from the reference genome should be stored. The Compressed Reference-oriented Alignment Map (CRAM) format [19, 20] takes advantage of this fact. As only the differences with

the reference genome are stored, the original reference genome must be provided to decode the content of the file.

The CRAM specification segregates the information into different blocks, and each block can be compressed with one of multiple compressors available.

CRAM does lose part of the adaptability of CARGO as there is no metalanguage to tailor-fit the file according to the needs, but the information is already segregated in order to reduce heterogeneity, there is no issue in interoperability as everybody uses the same segregation in types and CRAM further reduces the need for memory storage by allowing to remove all the information equal to the reference genome.

3.2 Securing the genomic information

As we have seen, the genomic information reveals facts about our parents, our identity and the diseases we might face. Furthermore, as the genome is inherited by the offspring, a breach of confidentiality concerning the genome of one generation also carries consequences for the next and for the previous generations. The research community is carrying out studies on possible attacks and ways to protect against them.

Currently, one of the approaches to protect genomic information is by the use of *beacons* [21]. In a beacon, the genomic information of multiple individuals is stored in the form of the diagnosed mutations. This then allows to query information about the mutations: a query to a beacon might for example concern the frequency of a given mutation in the part of the population with a given disease, or the question whether at least one patient presents the mutation. The beacon approach is intended to protect the information of any given patient as the information of one individual cannot be obtained, only information relating to the whole population is accessible.

3.2.1 Proposed attacks

3.2.1.1 Based on correlations within the genome

One approach which has been considered for the release of genomic information while maintaining certain levels of confidentiality is to simply not publish one part of the genome. This approach was taken in 2008, when the genome of Dr. James Watson was published without the data corresponding to the APOE gene linked to Late Onset Alzheimer Disease. However, using statistical information about which mutations are usually inherited together, the non-published information can be obtained by inference, as was done in [22]. The proposed attack led to erasing 2-Mb worth of information around the region of interest in Dr. Watson's published genome.

In 2015, Samani et al. [23] proved that such an attack could also be carried out using another mathematical approach. The authors built a Markov model which tries to represent the probability to see a given allele on the basis of the knowledge about the previously seen alleles.

The danger of such techniques can be lessened by taking into account those same statistics, as proposed in [24], where the authors propose to maximize the utility of the published information (e.g., publish as much relevant information for research as possible) while ensuring that certain features which have to remain private cannot be inferred.

3.2.1.2 Based on population statistics

Some attacks have been proposed against the beacon approach. In [25] a technique for detecting the presence of an individual in a group is proposed. In the publication, the authors consider the case of forensic analysis, but they also prove the validity of their contribution regarding DNA sequences from the HapMap repository. The objective of the contribution is to detect the presence of the genome of one individual in a sample containing the genome of many individuals. By sequencing the DNA in the sample, they have access to the allele frequencies within the group. They also have allele frequencies for a reference population. In order to determine whether the genome of the individual is in the sample they compare whether his or her frequencies are more similar to the reference population or to the sample. With a statistical test they give the probability that the individual is in the sample.

As the frequency of Single Nucleotide Permutations (SNP) might be affected by ancestry, the authors propose solutions to reduce this influence. One of them consists in using SNPs which are known to be less tied to ancestry. The other solution is to adapt the reference population to the individual who is being searched. By knowing, or discovering through SNPs, which reference is better suited, we are better equipped to correctly assert whether the individual is in the sample or not. Since blood-relatives share at least some DNA, this attack can be reformulated to detect someone close to the studied individual in the mixture.

This technique can be generalized to the case of beacons. In this configuration, there is no sample any more, but instead, the mixture of genomes is the beacon. The frequency of each mutation is not measured

in the sample, but rather queried from the beacon. This technique is considered as privacy threatening since being in a beacon might be an indication of a disease or similar. A straightforward implementation of a beacon where every frequency request is answered without any control enables such an attack.

The publication by Homer et al. [25] led to further improvements on the strategy. For example in [26], a mathematical reformulation of the attack was proposed which also allowed to include prior knowledge to further enhance the reliability.

Due to various factors, the probability of finding a certain mutation is possibly not independent of finding another. Jacobs et al. stated in 2009 [27] that at that point, for their method to behave correctly, they needed to be in linkage equilibrium. In other words, the mutations had to be uncorrelated until the underlying dynamics were better understood. Wang at al. prove in [28] that linkage disequilibrium is in fact a powerful tool to further improve re-identification attacks in Genome Wide Association Studies. According to the authors, just with the results which are usually published after such a study it could be possible to discover the presence of one individual in the case group. Wang et al. describe the likelihood of such an attack as "even more realistic than expected".

The previous reference methods need inputs which are not trivially obtained. Besides access to the beacon, the genome of the target of the attack and a reference population statistics are required. Moreover, in case of just having access to an anonymized genome, the negative effects of the attack are greatly reduced.

In [29], the authors show that the attack presented in [25] has good sensitivity, but that it lacks in specificity. False positives are too likely, which undermines the effectiveness of [25]'s method.

The attack published in [25] is based on frequencies. In [30], a variation is proposed which is based on answers to yes or no questions. The idea is that certain beacons allow to query whether, for a given position, they have a certain mutation in the aggregation, whereas the question was previously about the minimum allele frequency for that position (i.e., what is the probability to see the least common variation). The attack proposed in [30] can be interpreted as an adaptation of a Bloom filter to the attack proposed in [25].

3.2.2 Differential privacy

A key element to succeed in attacks like those proposed in [25] is the accuracy of the frequency measurements. Differential privacy [31] defends against such attacks by adding noise to the results. There is a trade-off in the amount of noise added: the more noise the system adds, the more it preserves privacy, but the less useful the information becomes. We want to achieve a situation where the result of a query over the aggregated data is virtually the same as the result of the same query over the same data with one individual more or one individual less. We achieve this by adding noise to the result obtained from the aggregated data.

We define a parameter ϵ which gives a sense of the privacy we are striving for. For two neighboring sets, the results of the aggregation function should not differ more than by a ratio of $\exp \epsilon$. We then need to balance the levels of noise in order to reach a compromise between privacy and utility. This balance is the critical point of differential privacy (see Figure 3.1). An example of this can be found in the pharmacogenetics study published in [32]. In this publication, the authors try to establish what the correct dosing of warfarin would be, using differential privacy as a protective method for the privacy of the patient. Their conclusion is that if the trade-off between privacy and utility is set in such a way that we do prevent attacks, the dosing is so far off that "patients would be exposed to increased risk of stroke, bleeding events, and mortality".

One solution to ensure high privacy and high utility at a time is to increase the number of participants in the study, but this might not be feasible. In [33] we are introduced to the idea that by better assessing the attackers' knowledge we will be able to use less noise: we have the required privacy levels, without the loss of accuracy.

As we have seen, differential privacy has been used to protect minor allele frequencies, but also chi-square statistics as in [34] (an idea which was further extended in [35] with the introduction of χ^2 statistics with variable numbers of individuals in the case and control groups).

3.2.3 Homomorphic encryption

We are considering how to execute some computation on information which has to be private due to its sensitivity. This use case is not a specificity of the genomic data. The ability of executing a computation over private data is a feature needed in many different fields: for example a web search engine where the server shall not know what the submitted query is, or just a database with very limited access to the query being run and the data which is stored (e.g., CryptDB [36]).

Homomorphic encryption's goal is to allow the execution of a computation over encrypted data. The result of the computation is obtained when the result is decrypted. In other words, the final decrypted result is the same as the result obtained when the computation is executed over the plaintext data, yet the data is never decrypted.

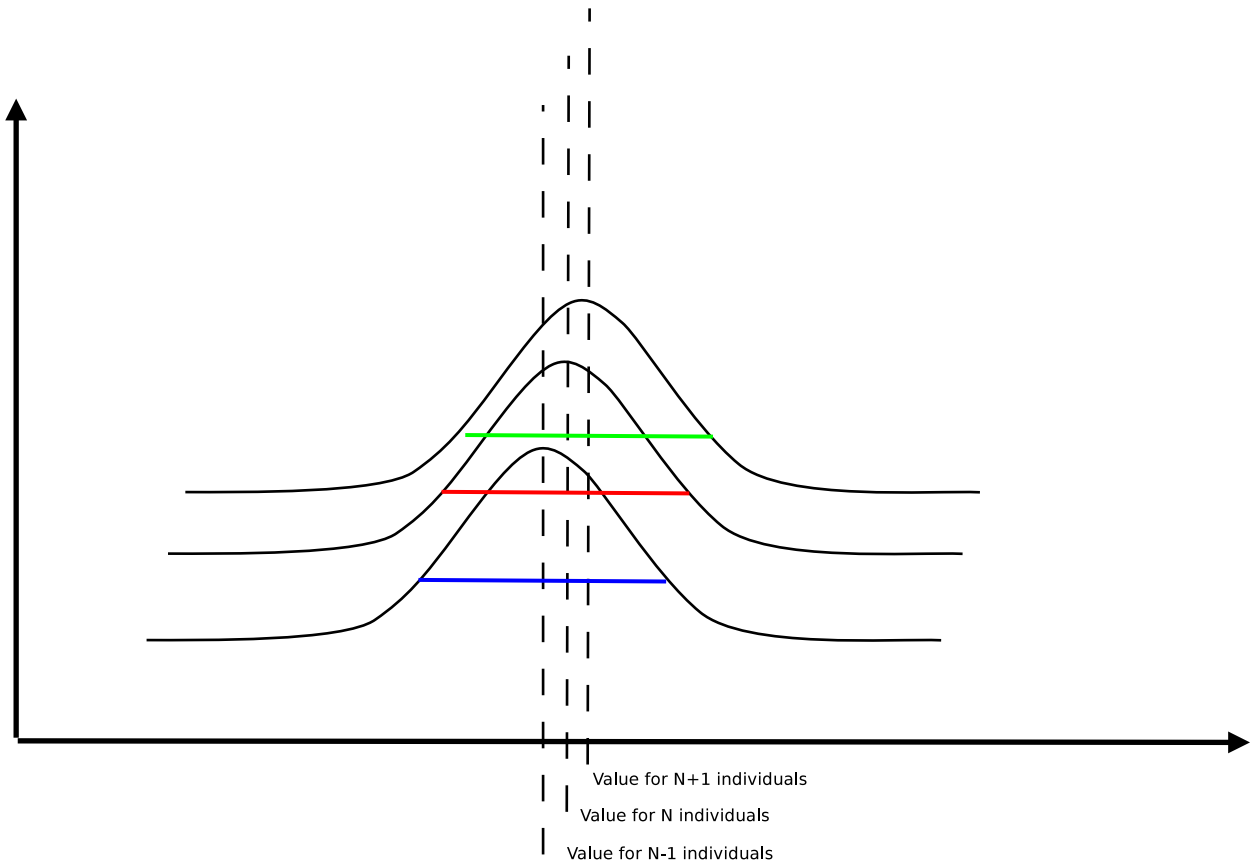


Figure 3.1: Visual representation of differential privacy. Three aggregation values are represented: one for N individuals and two for the neighboring sets (i.e., removing one individual or inserting one). The noise which is added creates a likely segment of values. As can be seen, these segments overlap for the neighboring sets: this makes it harder to guess from which set the result originated. This illustrates the idea of privacy. We also lose track of the real value: this is the loss of accuracy.

In [37] homomorphic encryption is used to perform pattern matching over genomic data. The method returns the Hamming distance, executing the computation securely on the cloud using Somewhat Homomorphic Encryption. In the proposed system the genomic data and the pattern to search for are encrypted using the same key.

A similar problem is the question whether there is a specific marker in a given DNA string. [38] presents a secure computation to detect the presence of a given marker at a given location in the genome. On the one hand nobody other than the person in possession of the DNA should learn which markers were detected, and on the other hand nobody other than the provider should know the test being performed (i.e., which substring is searched for, and at which position). The motivation behind protecting the test is plainly defending the interest of the Intellectual Property (IP) of the company whose test is being executed.

[39] also contemplates the need to protect the IP. In this proposal there are two parties: a Data Center attempts to defend the patient's data it holds, and a Medical Center executes a request to this data in order to obtain a measure for a given patient but wants the request to remain secret. In the proposed construction, the Medical Center sends an array of weights: the response of the Data Center is the scalar product of the SNPs and the weights. In this model, both players can be the attacker or the defender. The attack of the Data Center is logging which SNPs are requested. In order to defend itself against this, the Medical Center can add dummy weights. The Medical Center could perform an attack on the Data Center by only including one non-zero weighted SNP, thus discovering its value, or use a sequence of powers of the same number, revealing multiple SNPs. When sending the request, the Medical Center generates, for each weight, what the authors call a commitment: this can be viewed as a hash of the weight. The Data Center then randomly selects two commitments and asks for the weights: the Data Center checks whether the numbers might indicate an attack by the Medical Center. If the Data Center has suspicions, this step can be repeated. Ultimately, the scalar multiplication is executed homomorphically. As the authors state, the Data Center could deviate from the protocol and request more verification rounds, but the Medical Center can detect this and abort the procedure. Similarly, if too many weights appear to be non-legitimate, the Data Center can decide to stop the query.

[40] describes a method of computing the χ^2 statistic securely on the cloud. The proposal uses a secret and a public key: every participant in the study encrypts his data with the public key and sends it to the cloud. The cloud then computes the result which is subsequently decrypted by the research team using the secret key. The researchers generate the key-pair used. Therefore, there is a risk that they could obtain the data sent by the patients: if the cloud provider and the researchers collude, the cloud could send the encrypted inputs to the researchers, who could easily decrypt the data using the secret key.

3.2.4 Multiparty computation

[41] proposes two solutions for computing the χ^2 test for genomic data. The proposal considers multiple computation parties and multiple parties holding genomic information that must remain private. Each party holding information computes a contingency table summarizing the number of instances for a specific combination of genome and phenotype. All contingency tables are secret-shared with all computation parties. The computation parties then compute whether the χ^2 test is positive when compared to a given threshold. The computation parties combine all computed secrets to reveal a table of true/false results indicating whether, for each position, there seems to be a correlation with the phenotype.

[42] also uses multiparty computation. Multiple individuals share their genome with multiple computation parties using secret-share. The parties perform the Genome-Wide Association Study (GWAS) by retrieving multiple statistics and computing Primary Component Analysis (PCA). Each computation parties computes over the secret-shared information. The final solution is revealed to the computation parties when they combine the results of their computations. The contribution considers the computation parties to be curious but honest: the computation parties are expected not to diverge from the protocol, i.e., not to collude to retrieve the original input.

3.2.5 Proposed secure formats

The most frequent formats [14, 43] for storing genomic information do not provide confidentiality mechanisms, and thus no attacks can be proposed against them. To palliate this shortcoming, other formats have since been introduced: Cryfa [44] for unaligned data, Global Alliance For Genome and Health (GA4GH) File Encryption Standard [45] and Selective retrieval on Encrypted and Compressed Reference-oriented Alignment Map (SECRAM) [46] for aligned data. To the best of our knowledge no attack has been proposed against these.

3.2.5.1 Cryfa

Cryfa uses an approach of dividing the input data into different streams of information. Each of these streams are treated separately in their representation. In a step the authors refer to as compacting they represent multiple symbols of the stream with one output symbol. The result is then shuffled and encrypted. The rationale behind the shuffling is to increase the difficulty of a brute-force attack. If the first bytes to be decrypted were a known fixed sequence of bytes, it could be used to identify the moment when a brute-force attack has found the correct key. However, as the content is shuffled, the first bytes to be expected are not known, therefore such an approach cannot be straightforwardly used to identify the successful end of a brute-force attack.

Although the publication presenting Cryfa is turned towards protecting FASTQ, the approach could also be used for the information provided in an aligned file.

3.2.5.2 SECRAM

As previously mentioned, SECRAM [46] is designed for aligned data. While SAM [14] stores the alignment information for each read, as already described, SECRAM [46] stores the information per position: for each position, all alignment results are stored together. In other words, the information provided in a SAM read is broken down into the alignment information for each of the mapped positions, and each of these subunits of information is appended to the list of alignment results per position. This restructuring of the information, akin to the transposition of a matrix, allows the encryption of the information for certain positions. To the best of our knowledge, no attack has been proposed against this. However, using a database such as [9] and assuming that the user will only encrypt a mutated position, the original data might be recovered.

SECRAM does not use a compression approach as in CRAM that avoids to give information already present in the reference genome. However, the restructuring of the information increases the likelihood of having the same nucleotide repeated multiple times, thus improving the compressibility, as reflected in the results.

3.2.5.3 GA4GHCrypt

The Global Alliance For Genome and Health (GA4GH) is specifying a new container ensuring the privacy and the integrity of the information during file transports. This new container, called GA4GHcrypt [45], relies on dividing the content of the original file into smaller units of information. These smaller units do not need to correspond to semantical units of the original file. They correspond instead to the units of encryption and integrity verification.

This approach allows encrypting any file format, regardless of its content or representation. From this point of view, GA4GHCrypt is similar to other formats, such as Pretty Good Privacy (PGP) [47].

GA4GHcrypt relies on the combination of different keys, which shall yield, through a specified computation, the key used to encrypt the different blocks of data. There are four different asymmetric keys: both for the writer and for the reader, the algorithm uses both the public and the secret keys. Whereas in PGP, the public key of the sender must be made available at an earlier stage and outside the scope of the file transmission, or using open registries, in GA4GH, the writer's public key is stored within the file.

On the writer's side, the tool encrypts the content with a random key K_S. The tool then stores the key within the output file in an encrypted format. The algorithm obtains the key used to encrypt K_S in a two-step computation: first, it combines the secret key of the writer with the public key of the reader using Diffie-Hellman, and then hashes the result of this computation with the two public keys. Figure 3.2 shows the process.

On the reader's side, the tool executes a similar algorithm. The Diffie-Hellman algorithm now takes the reader's secret key and the writer's public key as inputs, and the tool does not generate a random key. Instead, it decrypts the key to read the content of the file. Figure 3.3 shows the process.

GA4GHcrypt defines that the file format can transport multiple keys. In that case, the tool can encrypt any one of the file portions with any one of the keys. On the reader side, the tool will attempt to decrypt the portion until one of the keys verifies the Message Authentication Code (MAC).

3.2.6 Access rules

3.2.6.1 ADA-M

Currently, each repository of genomic information must decide how to resolve each access request.

GA4GH aims at providing a format to represent the access rules. They propose a format which encapsulates for a given resource the different requirements which have to be met to authorize a request. They refer to this format as Automatable Discovery and Access Matrix (ADA-M) [48].

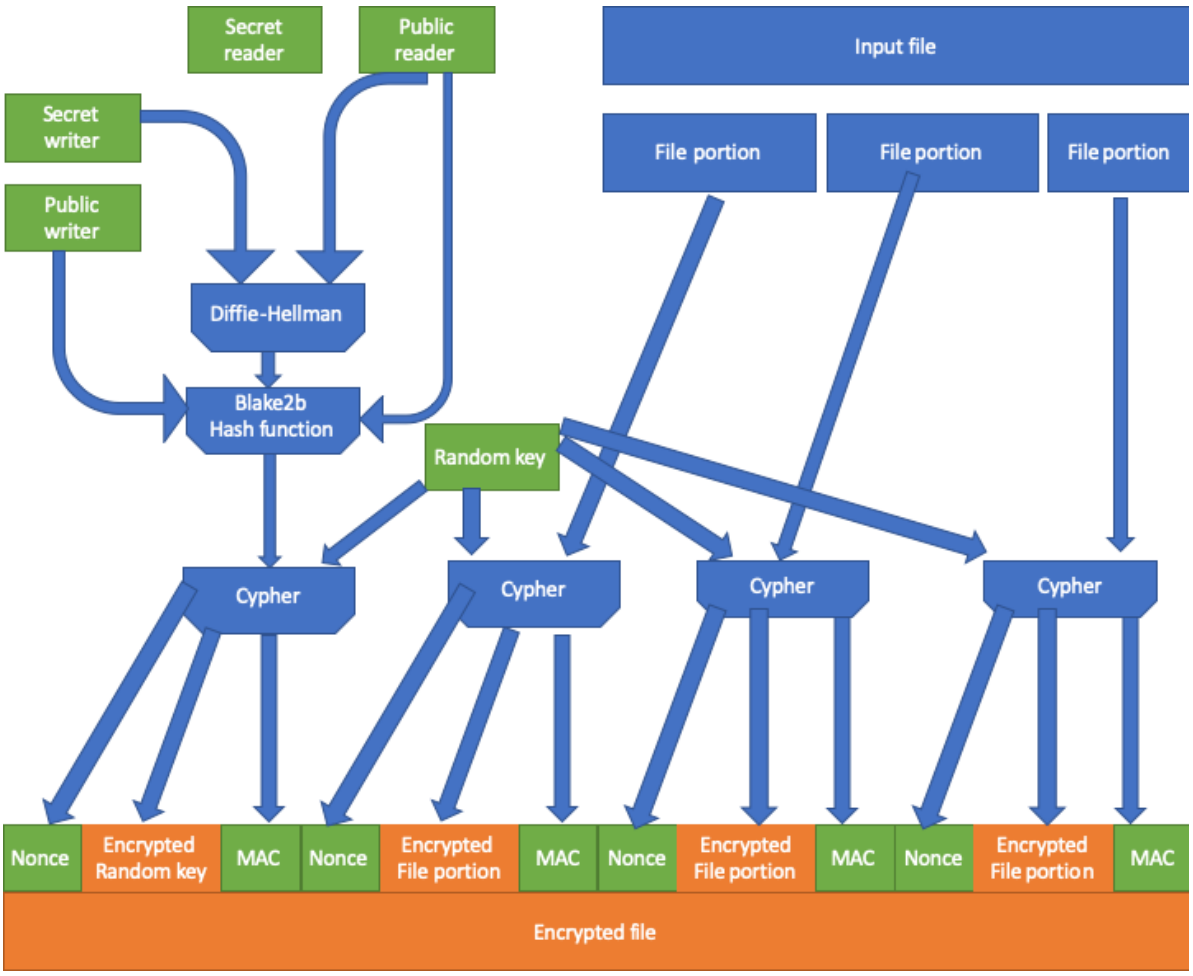


Figure 3.2: Encryption process used in GA4GHCrypt

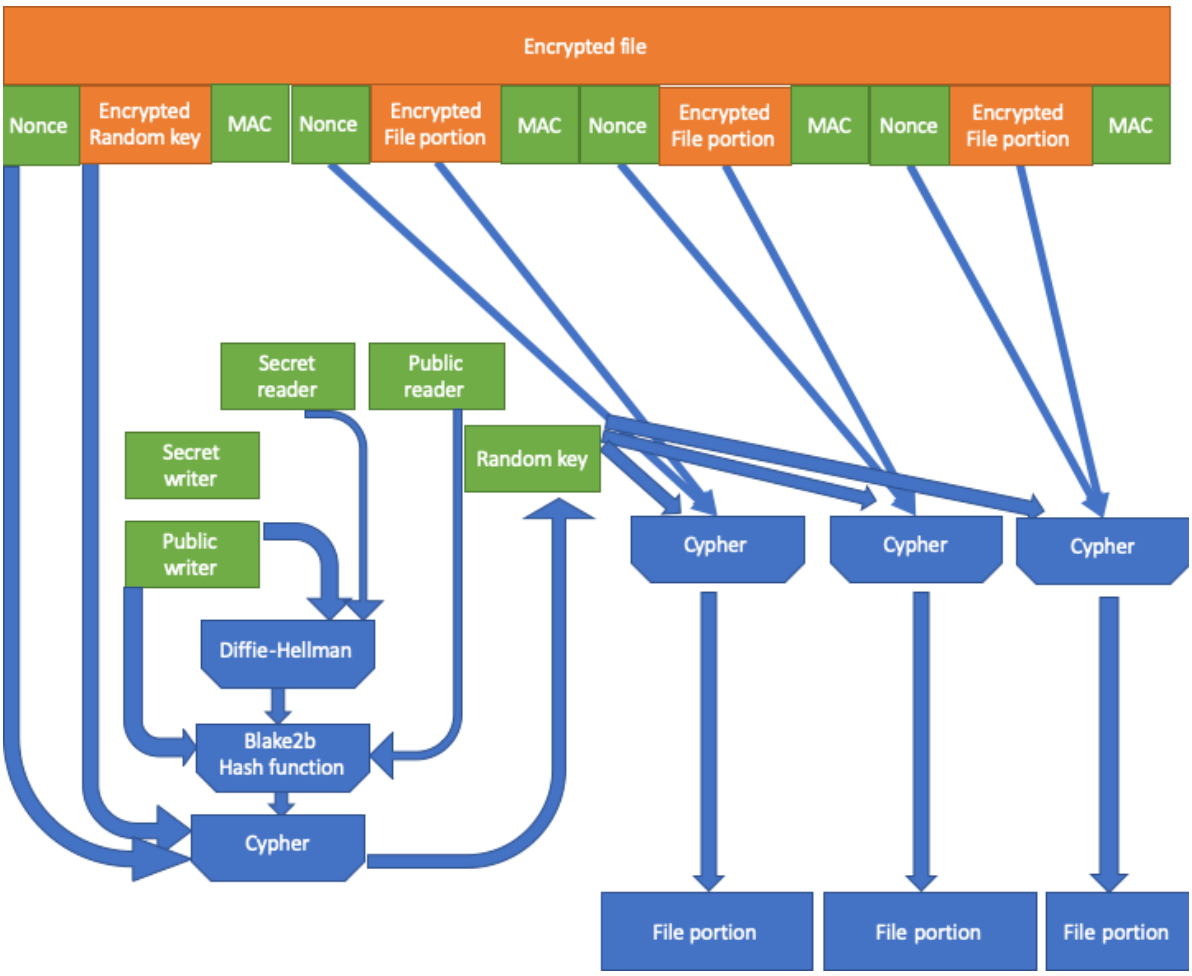


Figure 3.3: Decryption process used in GA4GHCrypt

ADA-M's authors list five main goals:

1. Through the definition of a broadly applicable access rule description format, they intend to consolidate the approach.
2. Creating one unified approach also helps in the creation of storage tools.
3. Transmission is also simplified, as the two parties can use ADA-M as lingua franca between them.
4. Having access rules following the same format bound to every resource simplifies querying the rules governing a specific one.
5. ADA-M entries allow a tool to resolve, potentially, a request automatically.

The specification divides the structure of an ADA-M profile into two main parts: a header and a body. The header conveys information on the resource to which the ADA-M profile regulates access. The specification further divides the body into three sections:

1. permissions
2. terms
3. meta-conditions.

3.2.7 Watermarking

3.2.7.1 Watermarking in genomics

Watermarking techniques can be used for a broad set of tasks. In [49], the authors list different use cases which could be covered with the use of watermarking in the context of healthcare, specifically when the watermark is applied to an image. These use cases include:

- simplifying the access control (If the image's metadata – e.g., information about the patient – is stored within the image by the means of the watermark, protecting the image and controlling the access to it also protects and controls access to the metadata.)
- moving identification information present in the image to a watermark only accessible through a key
- improving the captioning of the image in order to open new ways of attaching information to it
- using a watermark to carry information identifying the origin, such as a signature.

In [49], the new channel obtained thanks to watermarking is used to merge an identification of the receiver into the genomic information.

As in the case of multimedia information, the watermarking may be more or less obtrusive (depending on the significance of the introduced changes). There are different criteria to compare watermark methods, for example:

- the easiness to be recognized by a computer
- the decrease in value of the data
- the resistance of the detection of the watermark to file alterations.

Following with the multimedia analogy, these points would correspond to the following questions:

- Is the watermark visible at plain sight, or is it hidden, for example, using steganography?
- Is the watermark adding known content which can be easily searched for, or is it just adding noise?
- Is the watermark still present if the image is downscaled, rotated or otherwise modified?

On the other hand, the modifications can be more or less intrusive: methods can limit their effects to less significant regions, or, on the contrary, modify the content without considering its impact. In [50] we find an example of modifications in images with limited effect, as the proposed algorithm only modifies the least significant bits of certain elements.

The modification of the least significant bit is also employed in [51]. The authors include a new signal to the wireless transmission of an electrocardiogram (ECG): the proposed new signal is to indicate the identity

of the patient within the ECG signal. The effect of the least significant bit modification is mitigated by upscaling the values of the signal.

Another strategy is the one employed by the authors of [52]: their contribution addresses the need for watermarking images. In order to preserve the clinically relevant portions of the image, they discriminate between regions of interest (Region Of Interest, ROI) or not (Non-Region Of Interest, NROI). By applying the watermarking techniques to the NROI regions only, the potential drawbacks due to the file modification are mitigated.

In order to know whether the watermark will be resistant to file modifications, we first identify a number of actions which can be performed when modifying a genomic file and then consider their implications on the security method:

- exporting portions of the file, e.g., data for one chromosome
  The effects of the watermark should be present in the entirety of the file, so that it is unlikely that any significant portion has not been modified.
- importing other (portions of) files
  The watermark should not be invalidated by the presence of new non-watermarked data.
- modifying without semantic changes (i.e., the file is changed but not its meaning, e.g., the read identifiers are changed)
  The watermarking method should rely on a minimal set of information, unlikely to be modified.
- modifying with semantic changes (i.e., certain mutations have been added or deleted)
  The watermarking method should limit the input from the file's data, so that sparse modifications have as little effect as possible.

Additionally, the watermarking strategy could be defeated by collusion: if multiple instances of the same file (with different watermarks) are obtained, they can be compared in order to determine the non-watermarked version.

As previously referred to, a watermarking technique might alter the original host signal. The magnitude of the modification could render the signal unusable. One approach to mitigate this issue, further than just constraining the magnitude of the modification, is to ensure that the modification as a whole is reversible. This is done for example in [53], where two images are merged together, and both are recoverable. The host signal is an image to which a logo is associated, without notably altering the original image. When received, and if the necessary side information is also available, both the image and the logo can be recovered.

3.2.7.2 Use of watermarking in genomic information

The process to sequence and align the genome is important for many use cases. It can be used to perform research on diseases, to identify the best treatment or even to find better crops. Thus, it is important to be able to share the genomic information, but we might want to retain some control on the shared data: either to respect intellectual property or to hold bad actors accountable in case of data leaks. To this end, it is interesting to be able to modify the data in some way (i.e., to watermark it), in order to identify leakers in case of an audit. We have to make clear the difference between watermarking a DNA molecule of a living cell and including a mark in the result of sequencing such molecules.

[54] and [55] describe an algorithm to introduce content in the genome of a living organism. The information is introduced by changing some nucleotides (specifically the last nucleotide of each group of three nucleotides). Each changed nucleotide allows to encode two bits of information (as there are four different nucleotides). The nucleotides to be changed are decided according to the consequences of changing the nucleotide. In the process of translating the DNA into a protein, each group of three nucleotides is translated into an amino acid, and for certain combinations of the two first nucleotides, the translation will result in the same amino acid, no matter the value of the third nucleotide. Only those changes which do not change the protein synthesized from the edited portion of the DNA are possible. This procedure ensures that the living organism will still produce the same proteins. As the modifications are meant to be present in a living cell, and DNA replications might occur, the authors propose to integrate error detection and correction schemes into the message being encoded in the DNA: the method and the length of the correction strategies are determined by the likelihood of a mutation affecting the modified region. Therefore, the authors of [54, 55] focus on ensuring that the modifications do not affect the living cell.

On the contrary, we are focusing on watermarking the representation of the result of sequencing DNA molecules. There are other authors with similar objectives, as those of [56]. They modify sequential data before sending it to every recipient in such a way that the possibility to reconstruct the original data is minimal, the modifications are unique for each recipient, and it is hard for the recipients to collude and revert the modifications. Furthermore, the number of modified nucleotides is a constraint given as an input.

[56] uses the concept of sequential data as a sequence of data points where each point can take one value from a fixed pool of values. The data points (nucleotides which could be modified) they use are the locations of well-known mutations, and the pool of values comprises the situations where the mutation is present in none, in one or in the two chromosomes of the individual. [56] does not address one specific genomic file format but rather the genomic data as an array of properties. This array of properties resembles more the data represented in a Variant Call Format (VCF) file (limited to mutations affecting only one nucleotide - Single Nucleotide Polymorphism, SNP), but it could also be used for sequenced data files (such as a SAM file), if all reads covering one position modified by the watermark are modified. Based on the knowledge of what has been shared before and the modifications applied, the authors use an optimization problem represented as an Integer Linear Program (ILP) to decide what modifications should be operated on the data before sending it to the next recipient. The optimization minimizes the probability of the recipients being able to collude and deduce the watermarked positions, under the constraints guaranteeing certain levels of utility. Each modification is deliberate (the ILP minimization can select each data point independently). This allows the authors to perform such actions as to repeat certain modifications across multiple watermarked instances to fight off collusions. The authors consider the utility as a fraction of modified data: if the number of modified data points compared to the number of total points is low, the authors consider the utility to be high. A limitation of this algorithm is that the watermark cannot be undone.

3.3 Metadata

In order to make sense of the information, it must be accompanied by metadata. This metadata can range from facts about how the information was obtained and how it was processed to facts about the study in which it was generated and information about the individual (such as phenotypical information).

3.3.1 SAM header

The SAM header, which is also present in BAM and CRAM, provides certain information in the form of metadata. This information includes:

- the version being used and information on the sorting and grouping of reads
- information on the sequences used for the alignment (their name, their length, hashes to identify them in the future)
- description of each of the read groups:
  - the library from which the group has been sequenced
  - the name of the center where the group has been sequenced
  - a description of the sample
  - the programs used in the processing of the information
  - information about the platform used for sequencing (technology name, platform model, platform unit)
- each of the programs used in the post-processing of the information, with the parameters used and the order in which the pipeline was executed.

3.3.2 EGA

As we have seen, the SAM header covers many of the specific needs required to document an aligned file. However, we have also seen that there are other formats involved in analyzing the genome. In the case of a whole study (i.e., a purposefully built collection of individual genomes) there is more information to be represented which, instead of affecting each one of the genomes, concerns the collection of them. This might for example be the centers involved in the study. These might not coincide with the centers performing the actual sequencing of the genomes. To this end, the European Genome Archive (EGA) [9] has created a broad set of metadata specifications. The metadata is encoded with eXtensible Markup Language (XML) and specified using XML schemas [57].

The metadata specification used by the EGA aims at documenting each one of the following aspects:

- data access committee
  It lists the contact information for each member of the data access committee. In case of wanting access to the dataset, the corresponding data access committee must be contacted.

- dataset
  It groups different run and analysis results. The dataset can have a type attached to it (e.g., whole genome sequencing, exome sequencing).

- policy
  It marks which usages are allowed for the data contained in the dataset.

- analysis
  It represents a sequence analysis result (sequence alignment, sequence variation or sequence annotation).

- experiment
  It specifies how the sequencing will be performed (this includes the experiment's design, the platform being used, and the processing strategy).

- run
  It represents a group of reads generated.

- sample
  It specifies the sequenceable material being used (information on the taxonomy, its provenance if not anonymized or the anonymization procedure).

- study
  It defines the purpose of the study and the framework in which the experiments are conducted.

Figure 3.4 depicts the dependency relation between the different elements. This structure allows researchers to upload datasets with the results they have obtained, alongside information on how the datasets can be reused (dataset access committee). This structure also permits the creation of studies which will use one or more of these datasets to obtain the data required for the research.

This metadata specification also documents the evolution of the genomic material (FASTQ) by logging how the information was sequenced in a 'run', and how this information has further evolved into more complete material (SAM or VCF files) through 'analysis'. The 'run' is then linked to a sample with a documented phenotype, and the different 'analyses' are linked to the original material.

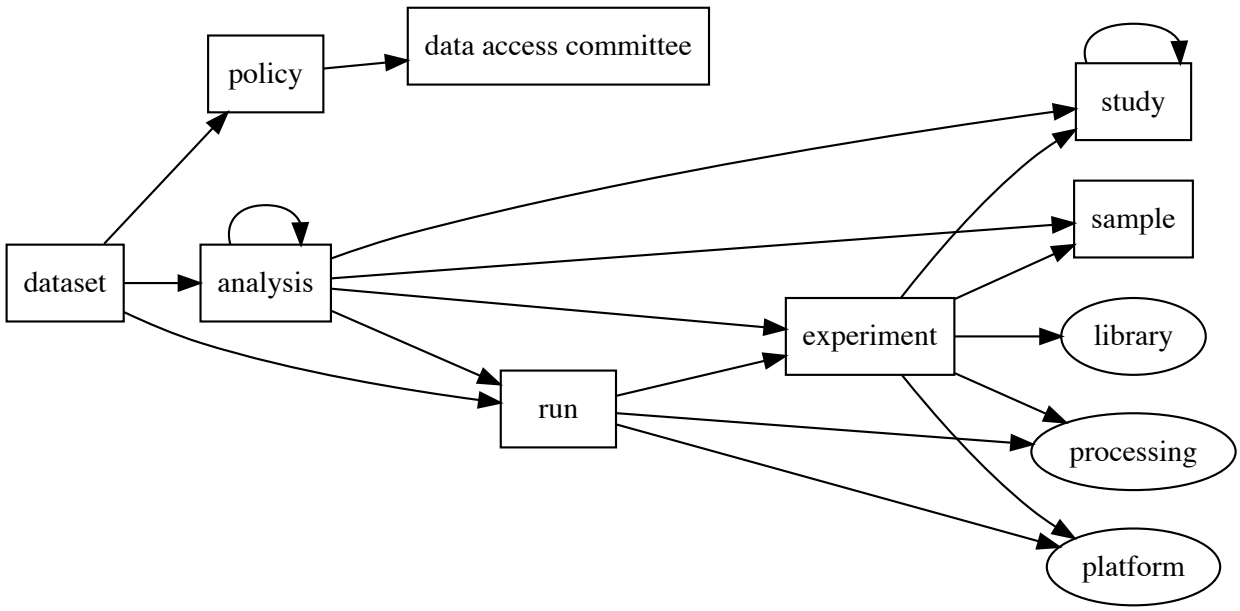


Figure 3.4: Relation between concepts used in the EGA metadata specification

3.3.3 National Cancer Institute: Genomic Data Commons

The Genomic Data Commons (GDC) [58] portal helps to navigate the library of medical results obtained in the context of different projects investigating cancer. The results are available as files, and multiple formats coexist. The GDC portal contains, among other things, alignment results in the BAM format, variation calls in the VCF format, and even images of the different samples.

The portal groups different results according to multiple criteria. Each result originates from a project. The portal identifies and references each project through a unique ID. For each project, the portal provides us with the main characteristics (the anatomical site under investigation, the type of cancer, and the employed strategy).

The repository ties each result to a case. Each case belongs to a clinical entry. A clinical entry is a combination of different factors:

- demographic (for example, the gender and the indication whether the patient is still alive)
- diagnostic/treatments (the diagnostic and further information such as the affection's severity, the age of the patient when diagnosed, or the date of a relapse)
- exposures (a grouping by external factors, such as alcohol consumption).

The Genomic Data Commons relies on having the provided information interlinked. A Universally Unique Identifier (UUID) identifies every entry regardless of the type: the project, the case, the demographic, the diagnostic and the exposures are all identified. The IDs enable searches such as finding all cases belonging to the same demographic.

3.3.4 Genomic Standards Consortium

The Genomic Standards Consortium [59] has specified a metadata representation for genomes (MIGS), metagenomes (MIMS) and marker genes (MIMARKS). The specification defines a list of fields which have to be present, or could be present. For each field, its name, meaning, expected format and occurrences (among others) are provided. The encoding format is not specified: the specification only defines the fields to provide. The authors refer to this as checklists. For each one of the broader goals, the specification defines multiple environments. For example in the case of metadata for genomes (MIGS), there are five different environments defined (eukarya, bacteria or archaea, plasmid, virus, organelle). Whether a field has to be present or not depends on the environment.

The specified set of fields is what the consortium considers to be the minimal set of information required. In order to extend the capabilities of the format, there are so-called environmental packages which add new fields to the checklist in order to document, for example, where the sample was obtained (e.g., the human gut).

This basic metadata representation can be extended in two ways: either a new core checklist can be introduced, or new extensions can be defined.

Chapter 4

MPEG-G compression

This chapter presents the encoding strategy for sequenced genomic data adopted by the Moving Picture Expert Group (MPEG), resulting in ISO/IEC 23092, Genomic Information Representation, MPEG-G [60]. In the context of this thesis, we have not contributed with new technologies to this encoding, but we have pointed out errors in it (cf. [61]). We have also contributed to the development of two tools implementing MPEG's encoding, as published in [62, 63]. We comment on the adopted approach and propose possible future improvements.

4.1 Standardization procedure

The International Organization for Standardization (ISO) is a body whose aim is the definition and promotion of standards. ISO divides itself into different technical committees that dedicate themselves to different subjects of standardization (e.g., the technical committee ISO/TC 17 publishes standards on steel, ISO/TC 117 on fans). The committee dedicated to information technology is a joint committee (ISO/IEC JTC 1) resulting from the merging of ISO's and the International Electrotechnical Commission's (IEC) groups on information technology.

If the area of standardization is too broad, technical committees can structure themselves into subcommittees, each corresponding to a finer-grained area of standardization.

Within a technical committee or subcommittee, there are multiple working groups. Each one has an even finer-grained topic of standardization. This thesis describes work carried out in the Moving Picture Expert Group (MPEG), the working group dedicated to "Coding moving picture and audio". MPEG is the 11th working group within the subcommittee for "Coding of audio, picture, multimedia and hypermedia information" (SC29) of the previously mentioned joint committee ISO/IEC JTC 1. As such, MPEG can also be referred to as ISO/IEC JTC 1/SC 29/WG11.

Working groups are composed of delegates. Delegates are specialists on their topics who are sent by their respective national bodies to participate in the crafting of the international standard.

In some cases, different working groups cooperate in the crafting of a standard, if the standard in question intersects their respective working areas.

When the need for a new standard is perceived, a new standardization activity is opened. The relevant working group starts by collecting requirements. Once it has identified the needs, a Call for Proposals is issued. The Call for Proposals specifies the requirements which all propositions must meet. The process expects the different actors from both industry and research who have relevant technologies for the upcoming standard to answer the call.

The working group builds the standard from the different received proposals. In order to make the most informed decision, the working group can define a Core Experiment: the delegates evaluate the different proposals against the defined criteria.

From the evidence gathered during the Call for Proposals and the Core Experiment, the delegates start writing the actual specification. Each one can submit new evidence and material to be reviewed by the working group during the next meeting. Based on the prior work and these contributions, the working group refines the specification.

The standard passes through different phases. The initial one is the Working Draft (WD), which successively becomes a Candidate Draft (CD), then a Draft International Standard (DIS), and finally an International Standard (IS). There might even be an additional stage in between DIS and IS, called Final Draft International Standard (FDIS).

For the standard to pass to the next stage, the national bodies must agree to it in a voting process. They can vote yes or no, or abstain. A vote can be accompanied by comments (in the case of a negative vote, the

comment is obligatory.) Comments can be either general, technical or editorial. At the next meeting, the working group gathers all comments and addresses each one of them. A comment can be taken into account or discarded, and the working group must state what has been changed (or not) in the draft and why.

As previously mentioned, the here presented work has been performed within the scope of MPEG and the Data processing and integration group within ISO TC 276 Biotechnology. We contributed to the standard as delegates of the Spanish national body. The standard focuses on genomic information representation. As this information must be compressed to be usable, the standardization activity corresponds to the area of two working groups: MPEG due to the need for compression, and the Data processing and integration group within the Biotechnology technical committee for the bioinformatics component.

MPEG usually divides its standards into different parts. Each part undergoes the previously described stages. In the case of the here described standard for genomic information representation (MPEG-G [60] or ISO/IEC 23092), there are currently five different parts:

- Part 1 Transport and storage of genomic information
  It describes the file and transportation format.
- Part 2 Coding of genomic information
  It describes the decompressing and decoding of the genomic information.
- Part 3 Metadata and application programming interfaces (APIs)
  It describes the metadata, the security mechanisms and the programmatic interfaces offered by the standard.
- Part 4 Reference software
  It provides a reference implementation of the standard.
- Part 5 Conformance
  It provides testing material to check the conformity of new implementations of the standard.

We have mainly contributed to Parts 1 and 3.

4.2 Data structuring in Part 2

In order to decode the information we need it to be represented in a form priorly agreed upon. MPEG-G Part 2 defines this representation as a sequence of Data Units of which there are three different types:

1. Parameters Data Unit
   It provides the parameters required during decompression (e.g., binarization used and its configuration) and decoding (e.g., maximal number of reads per record, whether the reads have constant length, and if so which length).
2. Reference Data Unit
   It provides the reference information to be used while decoding.
3. Access Unit Data Unit
   It provides the actual information to be decoded. This structure contains the compressed information, alongside indexing information on the content of the Access Unit.

Thus, similarly to CRAM, MPEG-G uses both the idea of breaking the information to be decoded into independent units and the use of an externally provided reference genome to storing redundant information.

4.3 Information decoding

Similarly to how CARGO and CRAM compress the information, the different fields of the record are separated into streams. We refer to each of these streams as descriptors. Some of the descriptors are:

- the position descriptor (referred to as 'pos')
  As the name indicates, it is meant to store the position of the record. But as we previously indicated, in a record there might be multiple reads. This descriptor indicates the position of the read within the record appearing first in the genome.
- the pair descriptor (referred to as 'pair')
  This descriptor is used for multiple purposes. On the one hand it allows to determine whether there are multiple reads within the record. In the case there are, it stores the information of where the paired read is placed (i.e., the position of the read within the record appearing second in the genome).

- the read name descriptor (referred to as 'rname')

- the read length descriptor (referred to as 'rlen')
  In the case where the reads have variable lengths, this descriptor is used to decode the length of each one of them.

- the mutation position and type descriptors (referred to as 'mpos' and 'mtype')
  These streams are used to store the differences with the reference genome: the combination of both descriptors allows to recover the original sequence. The process to recover the read's nucleotides sequence is to execute each of the operations deduced from the mutation position and mutation type streams: change one nucleotide, introduce new nucleotides, or delete nucleotides.

- the quality value stream (referred to as 'qv')
  It stores for each of the nucleotides the quality value associated to it.

We illustrate how the different streams and substreams are used by referring back to the instance provided in Table 2.1. In our example we would have the following Descriptor Streams:

- the position of the read ('pos')
  This stream contains the value 1, as the first and only read begins at position 1 of the reference.

- the operation positions ('mpos')
  This stream contains the values 2, 4 and 6, as these correspond to the positions where the read differs from the reference.

- the operation types ('mtype')
  This stream uses multiple substreams:

  - The actual operation is stored in a dedicated substream. This substream contains the values Insertion, Deletion and Mutation.
  - The inserted nucleotides are stored in another substream. This substream contains the value 'A', as it is the only inserted nucleotide.
  - The mutated nucleotides are stored in yet another substream. This substream contains the value 'A', as it is the only mutated nucleotide.

However, depending on the record not all descriptors are required: for example, in the case where all reads in a record are identical to the reference genome, the streams dedicated to operations are not required. Based on this approach the decoding process specified by MPEG-G Part 2 defines different classes of data, which allows to avoid providing certain streams. The class of the data is defined for the entire Data Unit Access Unit, and the approach is meant to segregate the information. For each region of the genome, the data is separated in as many classes as required, with one Data Unit Access Unit per class. The different classes defined are:

1. Class P
   It contains only reads which are mapped to the reference genome with only an equal operation (i.e., perfectly mapped, leading to the class being named P). This implies that there is no need to store the position of the operation, its type, and which nucleotides are inserted or which substitutions there are.

2. Class N
   It contains only reads where some nucleotides could not be identified represented with the symbol 'N' (leading to the class being named N). With respect to class P, we only need to add the descriptor for the operation positions (as we already know that all operations will be a mutation to 'N').

3. Class M
   It contains only reads where there are only mutations (leading to the class being named M). With respect to class N, we only need to add the descriptor for the mutated nucleotide.

4. Class I
   It contains all remaining reads (i.e., reads containing insertions and deletions, also known as indels, leading to the class being named I). In this class, all descriptors must be present.

We should also note that, for those cases where information is unmapped or half-mapped, MPEG-G provides the classes U (Unmapped) and HM (Half-Mapped), but they are not relevant for the present case.

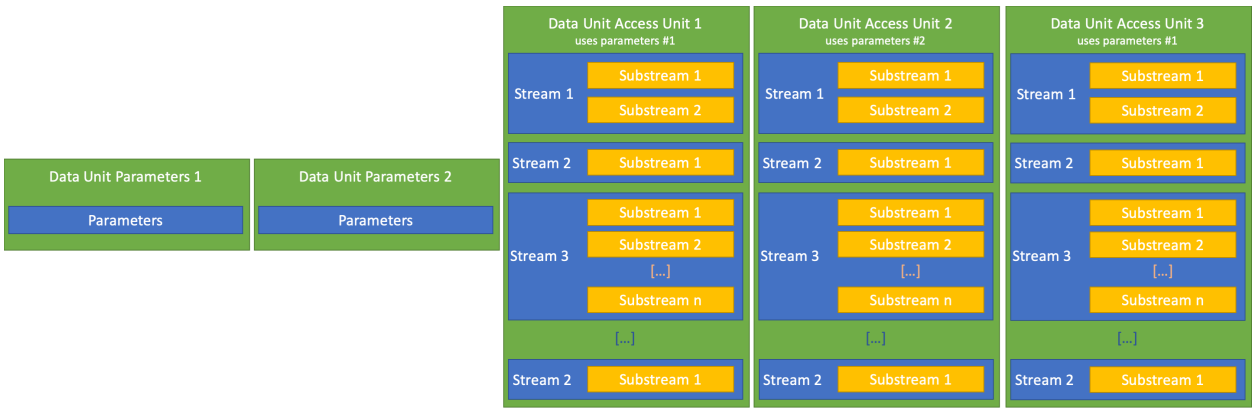


Figure 4.1: Structure of encoded information

4.4 Information decompressing

In each of the Data Unit Access Unit, the information of each of the streams is compressed. The decompressor used in MPEG-G is the Context-Adaptive Binary Arithmetic Coding (CABAC) [64] entropy decoder, an entropy decoder devised and still used for the video formats developed by MPEG.

The CABAC entropy decoder needs to be initialized with certain parameters which are provided partly in the Parameters Data Unit and at the beginning of each of the streams. CABAC only works with binary information (i.e., the symbols being decoded are bits), therefore a binarization is also parametrized in order to recover from the bits the original symbols to be used while decoding the records.

Finally, and in order to improve the compression, additional transformations and processes are defined in MPEG-G Part 2. Some of the processes are based on a rewrite of the information: for example in the case of the position descriptor, the information is not stored in absolute values but as delta values (i.e., the distance to the previous value) as this helps to increase the repetition of symbols.

Another type of process used in MPEG-G Part 2 to improve the compression of information is to break the information into substreams or subsequences. An example of this is the pair descriptor: Its first substream indicates the type of pairing information (whether the record has paired information or not and if so, whether the paired read is close by, on the same chromosome but further away, or whether it is on a different chromosome). Then there are substreams for each of the cases (one substream to provide the position delta in case of close-by pair, one substream to provide the absolute position in case of same sequence but distanced pair, and one substream for the sequence and one substream for the position for the remaining case). The subdivision into streams and subsequences is shown in Figure 4.1.

4.5 Thoughts on adopted approach

Certain aspects of the decoding process of the encoded MPEG-G information can be convoluted. Let us address two of the origins of this complexity.

4.5.1 Data classes

One of the complexities arises from the existence of multiple classes. The motivation for using classes was to improve both random access capabilities and compression. Let us reflect on both aspects.

4.5.1.1 Random access

The MPEG-G specification supported classes before supporting multiple alignments. After the inclusion of secondary alignments, the group did not redefine the classes: the class still indicates the type of the first alignment pair stored in the record. Depending on the record, this could correspond either to the primary or the secondary alignment. Thus, if the parameters indicate the presence of multiple alignments, the decoder cannot rely on the class of the Access Unit to determine all types of alignment present in it.

We have also seen that MPEG-G defines mechanisms to store multiple segments in one record, yet it does not enforce the use of this feature. In some instances, e.g., when the aligner has mapped the segments to different sequences, the segments have to be stored separately. When stored together, the Access Unit containing the record must be of the most general class required. Yet, the same logic is not applied when storing the information in different records. Let us clarify this with an example: We have two paired segments, the first maps without mutations to the reference (this would correspond to a class P), the second

maps to the reference, but has an insert (this would correspond to a class I). If the two segments were stored together, the record would belong to an Access Unit of class I (dictated by the needs of the second segment). If the encoder represented the two segments in two separate records, then the first one would be of class P, and the second of class I. Since the encoder has no means to signal the encoding strategy to the decoder, the decoder is forced to access all Access Units to retrieve the class P specific segments.

Although in the first versions of the specification, the separation in classes might have been useful in creating new options for random access, this is not the case anymore.

4.5.1.2 Encoding

The primary motivation behind the creation of classes was to reduce the needs of descriptors when encoding. For example, if we encode all reads mapping without any mutation to the reference, we can spare the need to include the descriptors for mutation position, mutation type and similar. Although this reasoning seems sound, the reduction in required memory is minimal. In the current specification, the first subsequence of the descriptor for mutation positions encodes whether there are further mutations for the segment. If there are, the subsequence returns the symbol 1, otherwise the symbol 0. If all segments stored in an Access Unit were to have no mutations, the first subsequence of the mutation position would only contain a repetition of the symbol 0. Such a sequence of symbols is trivially encoded using repetitions. We can apply this reasoning to the other streams. Therefore, the gain in compression is not due to the existence of classes in the specification. Instead, the segregation of the information is the root of the gain in compression.

4.5.1.3 Conclusion

We have seen that the benefits obtained from the specification defining classes are debatable. The use of classes does not provide real gains in terms of Access Units. We would also obtain the same compression result just by segregating the data without explicitly relying on classes. Yet, the concept of classes is central to the decoding process: the specified process increases in complexity as the classes become less specific and requires frequent conditional jumps based on the class.

We proposed to remove the concept of classes, but we made our contribution too late to have the specification changed.

4.5.2 Stream decoding process

The MPEG group performed the majority of late-stage corrections to the MPEG-G specification on the sections describing the decoding processes. Most notably, the corrections have focused on the less tested parts of the standard, such as the decoding of multiple alignments. We have also contributed to some of the corrections, for example, by proving that some ill-formed but still compliant input could cause errors in the specified decoding process. We must therefore face the possibility that through seeking decoding processes that lead to a better compression across different test cases, the group has created a hard to debug specification.

Let us propose an alternative approach to the decoding process which should obtain if so wanted the same compression levels, yet allows us to define a much easier decoding process. We must first see some of the techniques currently used. To do so, we review the respective decoding processes of two of the streams.

4.5.2.1 Current specification

4.5.2.1.1 Quality values For each nucleotide of each read there is a metric of the quality of the sequencing (i.e., the degree of confidence with which the nucleotide has been sequenced). The encoding strategy specified in MPEG-G relies on the idea that by grouping qualities by position, the encoder can detect similar quality distributions, which leads to better compression results.

The encoder can specify multiple quality books. The first descriptor subsequence indicates which quality book is used for which position of the genome. On the basis of this information, the decoder identifies the corresponding subsequence and reads the symbol encoding the quality value.

Figure 4.2 shows this process and highlights the division between the information provided by the file and the decoding process defined by the specification. We can see that the file is just a conveyer of encoded symbols.

4.5.2.1.2 Mutation types When decoding the mutation type, we must, in fact, recover two pieces of information: the mutation type and, if applicable, the nucleotide detected in the read (in case of a deletion, per definition, there is no detected nucleotide).

The process of decoding the detected nucleotide relies on the decoded mutation type. If the mutation is a deletion, no further reading is required. If the mutation is an insertion, the decoder reads the nucleotide

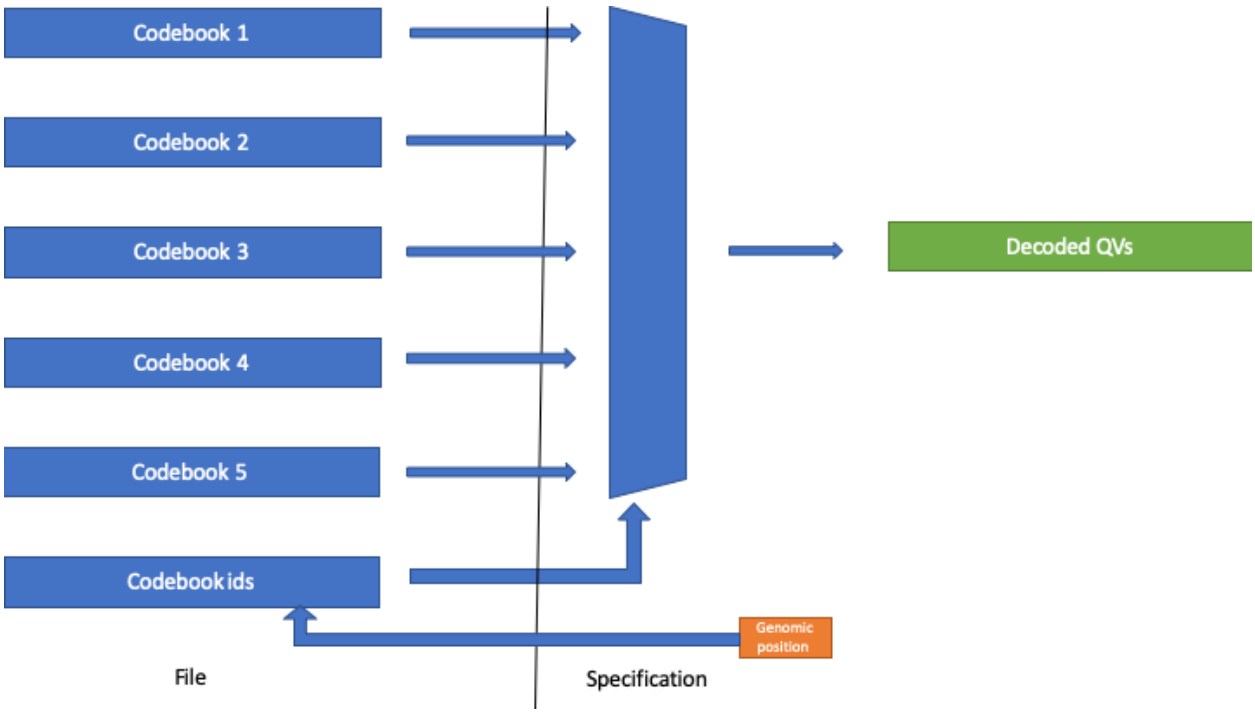


Figure 4.2: Decoding of quality values

in the corresponding subsequence. If the mutation implies a change of the nucleotide, the decoder must identify the stream which corresponds to the current genomic position: to obtain better compression, the encoder has grouped the nucleotides according to the reference. Figure 4.3 shows this process.

4.5.2.2 Proposed changes

In Figures 4.2 and 4.3, the specification uses the decoding process of the stream to retrieve the element marked in green. A valid solution, yet costly in terms of compression, would be storing all information for the stream in one subsequence. Figure 4.4 shows this approach.

The benefit of this type of encoding is to lower the complexity of implementing an encoder. But this makes it impossible to use any strategy based on the nature of the data (as we have seen, grouping specific values improves the compression).

Let us draw a list of the building blocks of the decoding process we have seen:

- substream selection (as seen in Figures 4.2 and 4.3, we select from multiple substreams the correct one on the basis of a value external to the decision)
- use of a contextual value highlighted in orange in Figures 4.2 and 4.3 (the decoding process has access to specific genomic information used in the process)
- lookup tables (as in the case in the decoding process of the quality values, the decoding process retrieves specific values from lookup tables).

Let us imagine a specification with substreams of two types. The first type, similar to the current ones, represents encoded symbols. The second one, which has no equivalent in the current specification, corresponds to a logical decision. Regardless of the type, the subsequence is identified. Some of the identifiers are well-known, for example the 'identifier for quality value stream'. The streams with well-known identifiers correspond to the elements highlighted in green in Figures 4.2 and 4.3.

Like regular streams, the logical streams also return symbols. But as the name implies, the choice of the returned symbol relies on a logical decision. To execute this decision, we need variables. Variables are also identified. A subset of these identifiers is well-known. They correspond to the values represented in orange in Figures 4.2 and 4.3.

A logical decision can take the following form: "if variable A is true, read from stream B; otherwise, from stream C". Similarly, logical streams can be defined to represent stream selection and lookup tables.

With these mechanisms in place, we can shift the complexity of the decoding process to the file. Figures 4.5 and 4.6 show the result of this rewriting. As we are still using the same encoding approach, we obtain the same compression result. Yet, as the complexity of the decoding process is not defined in the specification, the latter could be more straightforward, which simplifies its debugging.

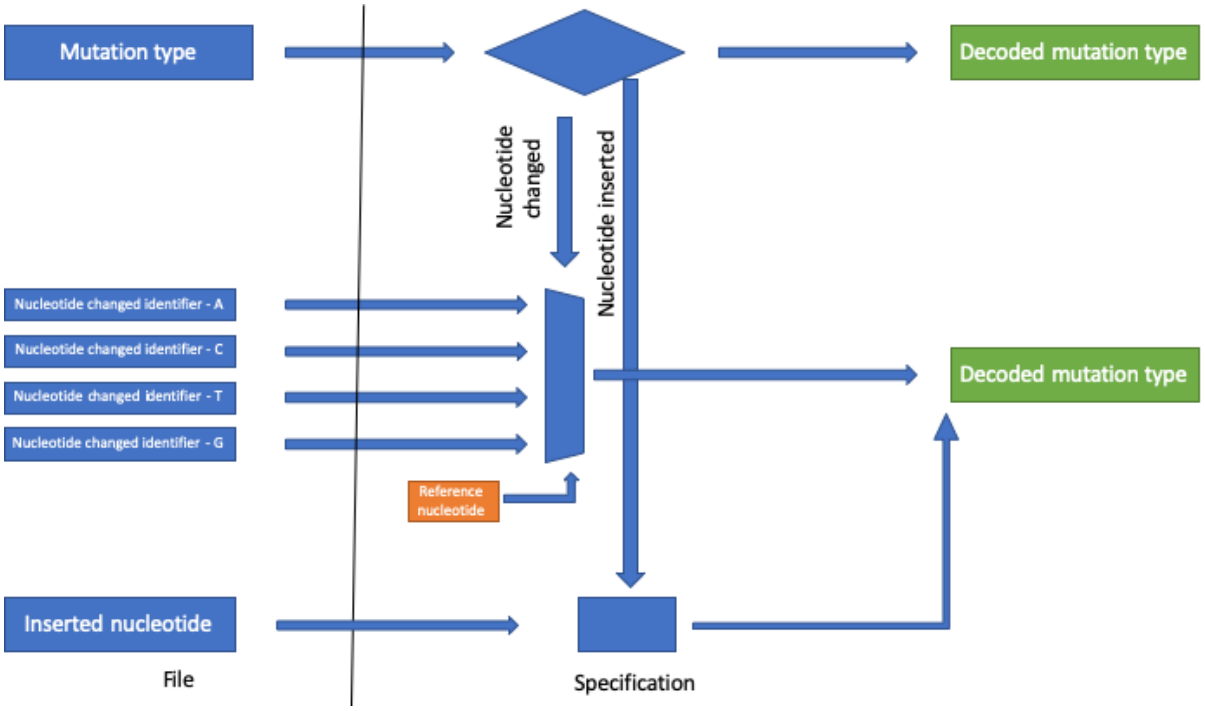


Figure 4.3: Decoding of mutations

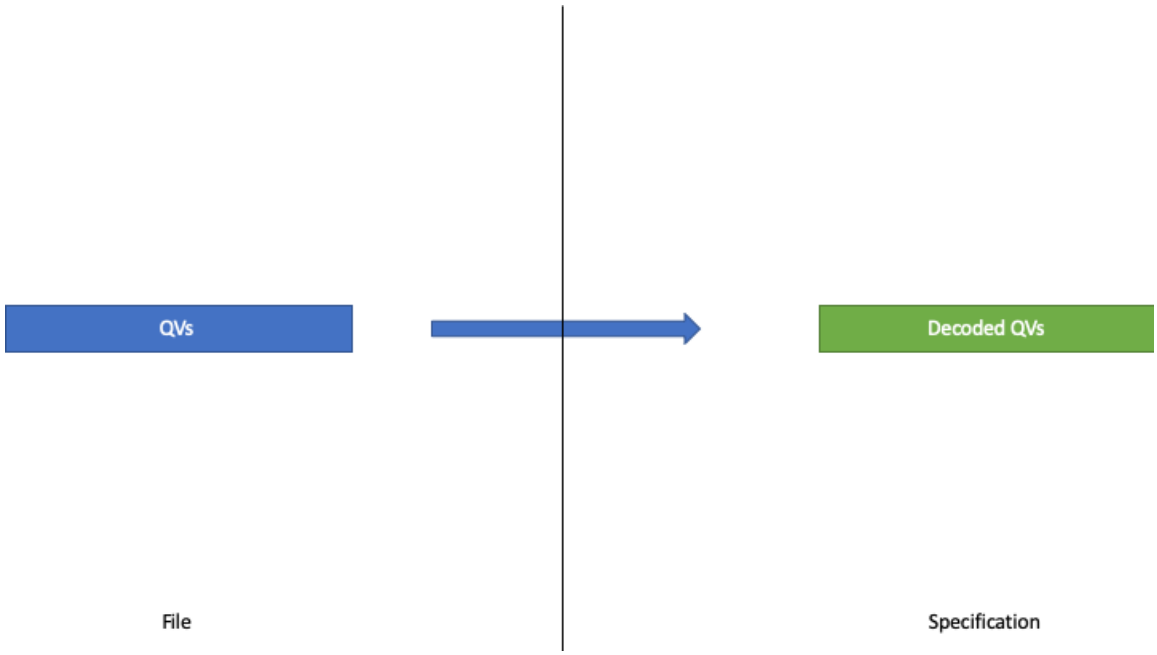


Figure 4.4: Proposed easy decoding of quality values

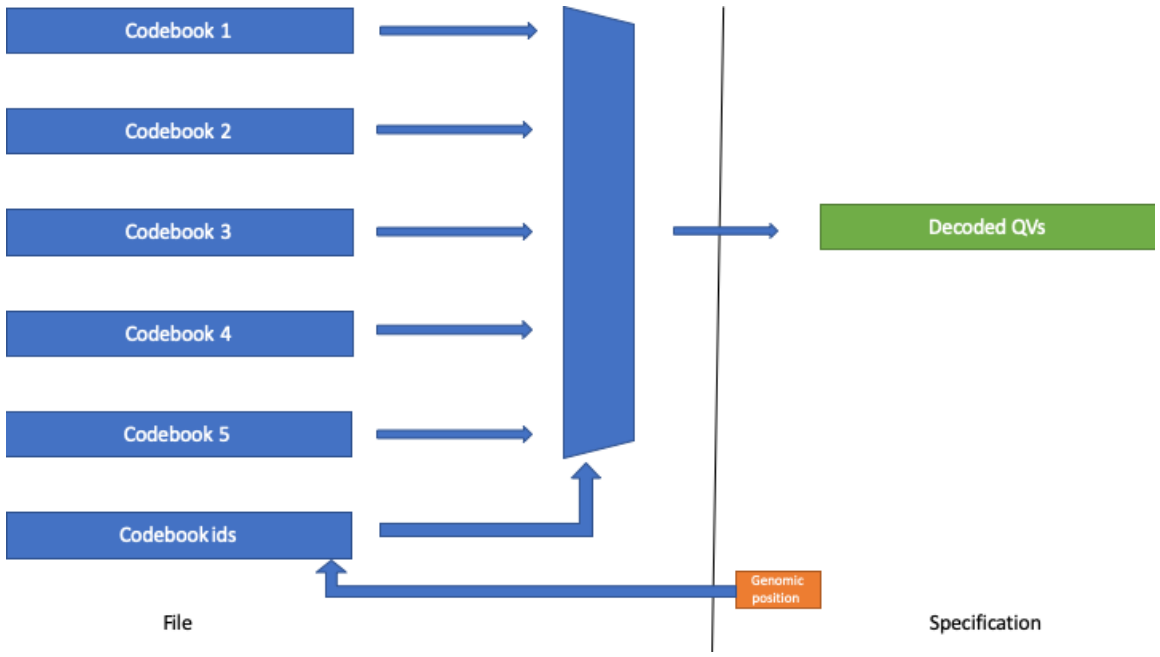


Figure 4.5: Proposed decoding of quality values

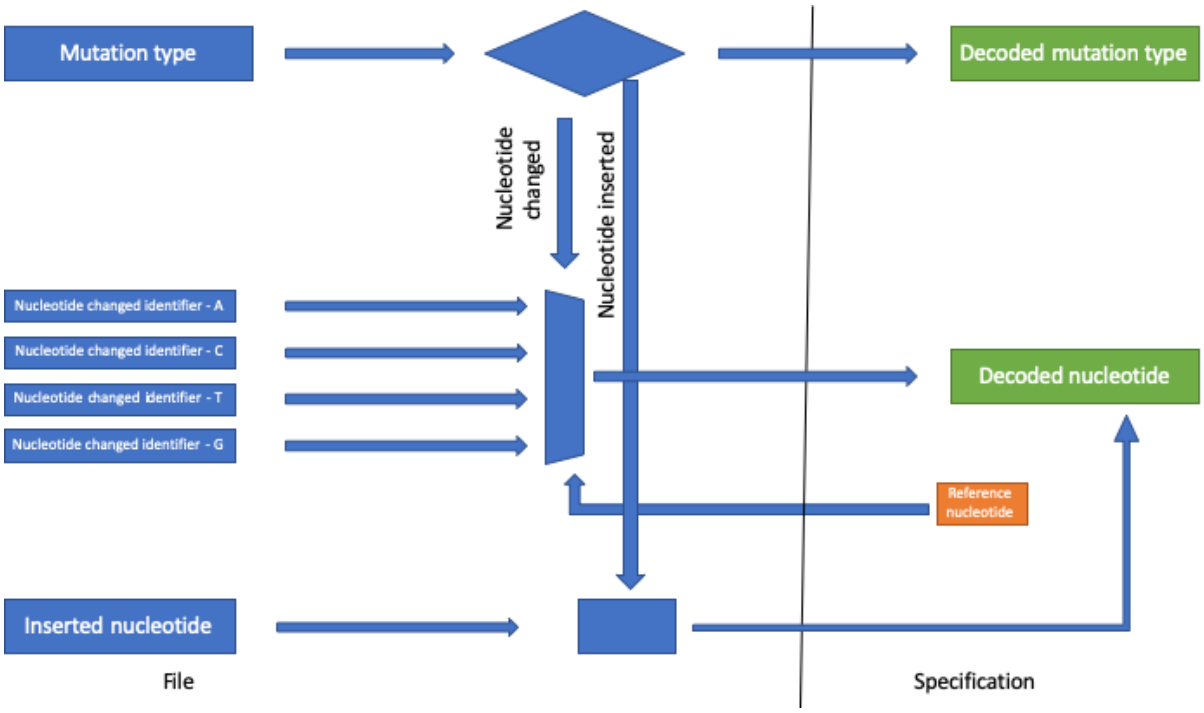


Figure 4.6: Proposed decoding of mutations

Chapter 5

File format

MPEG identified the requirements for MPEG-G´s file format. This chapter presents the result of our work to meet these requirements. Our proposal is a container-based file format which creates a layering in the information and allows to encapsulate sequenced genomic data (cf. Chapter 4), the metadata (cf. Chapter 6) and the security parameters (cf. Chapter 7). We describe the finally adopted approach, which is a combination of our proposal with other inputs. We comment on the adopted approach and propose possible future improvements. The result of our work has been published in [65, 66].

5.1 Requirements

Alongside the definition of the compression mechanism, the MPEG-G group called for proposals on how to transport the information. Transport refers both to the local representation of the data (e.g., as a file in the local file system) and to the streaming of the genomic information over the network. The first identified requirements were related to the needs of ensuring the integrity of the information, its privacy, and, generally, the access control mechanism.

The initial requirements were later on extended to encompass other features required by the use cases: random access based on position, random access based on certain features of the information and random access for unaligned information.

5.2 Our proposed solution

Our first proposal submitted to MPEG in October 2016 was based on the ISO Base Media File Format [67]. Using this technology as the basis for our proposed file format had the main advantage of creating an easy-to-parse file, with a clear tree-like structure of elements where containers for different types of information could coexist. The ISO Base Media File Format (ISO BMFF) defines a structure based on a Key Length Value (KLV) approach. The specification defines the syntax of the different containers, for example, whether a given container holds another container.

We used the features of ISO BMFF to propose a file format able to carry different types of information: uncompressed or compressed information of diverse nature (FASTQ, SAM, VCF). In this proposal, we introduced the idea of grouping multiple files in one container: this allowed us to have one study constituted, for example, by multiple alignments (i.e., SAM-like data), accompanied by VCF files. Our proposal thus had three different layers: the Study level, dedicated to grouping together different sets of information; a second level (which we called 'Dataset level') corresponding to a sequencing, an alignment or a variant call; and a third level allowing the subdivision of the information into multiple Streams akin to what has proven effective in CRAM and CARGO.

Our subdivision of the information into multiple layers also proved key to meet the requirements defined by MPEG. At each of these levels, we included the required elements to provide information on the privacy, integrity and access rules settings. Metadata relevant for the Studies, Datasets or Streams also accompanied this additional information.

5.3 The adopted approach

5.3.1 Overview

The solution finally adopted by MPEG-G adopts many characteristics from our approach. Although the final approach is not based anymore on ISO BMFF, the final approach takes its main characteristics: a tree of containers is the basis for the format, containers which are represented, as in our contribution, with a KLV syntax. Also taken over from our proposal, different layers structure the format: a Dataset represents each SAM-like or FASTQ-like file to be contained in the MPEG-G file, and Datasets can be grouped in Dataset Groups, akin to what we proposed with the Studies. The final proposal also includes our approach of taking leverage on the hierarchical structure of the file when including metadata and security information.

5.3.2 Representation of the data

The file format defined for MPEG-G only transports MPEG-G encoded information, diverging from our proposal. This fact streamlines the requirements faced by the transport format. We know that the encoder structures the genomic information in a conceptual grid: the columns would be the different streams used for the encoding, the rows the different regions of the genome. In other words, each cell in this grid corresponds to one encoding block: for example, this block could correspond to the stream 'pos' of the class P for the region corresponding to chromosome 1 from nucleotide 1000 to 50000. Figure 5.1 shows the grid-like understanding of the encoded information.

		Class P			Class N			
		pos	pair	rname	pos	pair	rname	mpos
Chr 1	0 - 1000000	Block-P-AU1-chr1-pos	Block-P-AU1-chr1-pair	Block-P-AU1-chr1-rname	Block-N-AU1-chr1-pos	Block-N-AU1-chr1-pair	Block-N-AU1-chr1-rname	Block-N-AU1-chr1-mpos
	1000000 - 2000000	Block-P-AU2-chr1-pos	Block-P-AU2-chr1-pair	Block-P-AU2-chr1-rname	Block-N-AU2-chr1-pos	Block-N-AU2-chr1-pair	Block-N-AU2-chr1-rname	Block-N-AU2-chr1-mpos
	2000000 - 3000000	Block-P-AU3-chr1-pos	Block-P-AU3-chr1-pair	Block-P-AU3-chr1-rname	Block-N-AU3-chr1-pos	Block-N-AU3-chr1-pair	Block-N-AU3-chr1-rname	Block-N-AU3-chr1-mpos
Chr 2	0 - 1000000	Block-P-AU1-chr2-pos	Block-P-AU1-chr2-pair	Block-P-AU1-chr2-rname	Block-N-AU1-chr2-pos	Block-N-AU1-chr2-pair	Block-N-AU1-chr2-rname	Block-N-AU1-chr2-mpos
	1000000 - 2000000	Block-P-AU2-chr2-pos	Block-P-AU2-chr2-pair	Block-P-AU2-chr2-rname	Block-N-AU2-chr2-pos	Block-N-AU2-chr2-pair	Block-N-AU2-chr2-rname	Block-N-AU2-chr2-mpos
	2000000 - 3000000	Block-P-AU3-chr2-pos	Block-P-AU3-chr2-pair	Block-P-AU3-chr2-rname	Block-N-AU3-chr2-pos	Block-N-AU3-chr2-pair	Block-N-AU3-chr2-rname	Block-N-AU3-chr2-mpos

Figure 5.1: MPEG-G encoding represents the information as a grid, dividing both in regions of the genome and descriptor sequences.

MPEG-G proposes two different ways to structure the information provided in the grid:

1. Access Unit Contiguous (AUC)
   An Access Unit Container groups all blocks which correspond to it. Figure 5.2 shows how Access Unit Containers group the blocks represented in Figure 5.1.

2. Descriptor Stream Contiguous (DSC)
   A Stream Container groups all the blocks corresponding to it. Figure 5.3 shows how Stream Containers group the blocks represented in Figure 5.1. We store the blocks contained in the Stream Container without a header: to understand where a specific block starts in a Stream Container, we need external information. The indexation table indicates for each Access Unit the offset at which each block starts.

When the user requests the decoding of a set of regions, the tool must restructure the information in Data Unit Access Units. Each of these Access Units must contain all required blocks. We have seen that

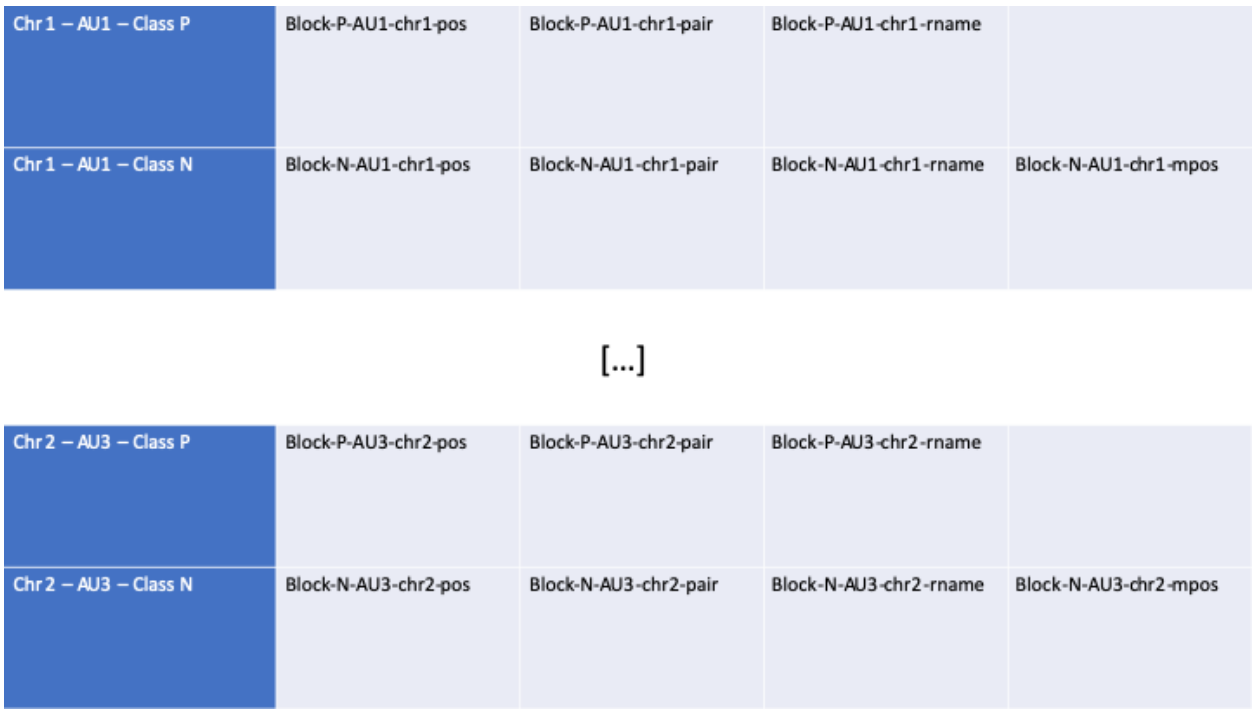


Figure 5.2: MPEG-G encoded data grouped in Access Units

blocks associated with an Access Unit are either grouped as a container when in Access Unit Contiguous mode (AUC), as shown in Figure 5.2, or the indexation information is used to perform this grouping when in Descriptor Stream Contiguous mode (DSC).

To retrieve the information relevant to a specific query, we need to know which set of blocks corresponds to which region of the genome. When the file is in DSC, the indexation information structure also provides this information. In AUC mode, we either use similar indexation information, or we provide a more comprehensive header with the Access Unit Container to store the mapping on the genome.

However, due to subsequent changes in the requirement list, the file format was modified. The MPEG group decided that MPEG-G should include some of the auxiliary tags defined by SAM. The encoding did not reflect this change. Instead, a new file format element, the Access Unit Information, now stores this data. This Access Unit Information unit must be stored in an Access Unit Container, consequently there must be an Access Unit Container independently of the file mode (AUC or DSC)

5.3.3 Format summary

We have seen that the MPEG-G file format relies on a Key Length Value mechanism, as we proposed. The specification defines a set of containers that can, in turn, aggregate other containers. As in our proposal, the MPEG-G file can aggregate multiple SAM-like contents (to which we refer as 'Datasets') in a higher-level element we call 'Dataset Group' (equivalent to the study in our proposal).

In our proposal, if the encoding was stream-based, the encoded information was always stored in streams. The MPEG-G format leaves the choice between the AUC and DSC mode, which leads to two different file structures (cf. Figures 5.4 and 5.5). The syntactic level reflects the concept of the Access Unit, regardless of the mode used.

5.3.4 Indexation mechanisms

MPEG-G uses a subdivision of the information based on regions of the genome, in the case of aligned data. In the case of unaligned data, it offers the user the possibility xto give a list of nucleotides subsequences that will be associated with the Access Unit: all records belonging to the Access Unit should then share these subsequences.

The information to be stored in the indexing tables and for each Access Unit is thus the following:

- Which information does the Access Unit store? In the case of aligned data, this means the tuple of the chromosome, start and end position on the chromosome and the class of the Access Unit (e.g.,

Stream Class P- pos	Stream Class P- pair	Stream Class P- rname	Stream Class N- pos	Stream Class N-pair	Stream Class N-rname	Stream Class N-mpos
Block-P-AU1-chr1-pos	Block-P-AU1-chr1-pair	Block-P-AU1-chr1-rname	Block-N-AU1-chr1-pos	Block-N-AU1-chr1-pair	Block-N-AU1-chr1-rname	Block-N-AU1-chr1-mpos
Block-P-AU2-chr1-pos	Block-P-AU2-chr1-pair	Block-P-AU2-chr1-rname	Block-N-AU2-chr1-pos	Block-N-AU2-chr1-pair	Block-N-AU2-chr1-rname	Block-N-AU2-chr1-mpos
Block-P-AU3-chr1-pos	Block-P-AU3-chr1-pair	Block-P-AU3-chr1-rname	Block-N-AU3-chr1-pos	Block-N-AU3-chr1-pair	Block-N-AU3-chr1-rname	Block-N-AU3-chr1-mpos
Block-P-AU1-chr2-pos	Block-P-AU1-chr2-pair	Block-P-AU1-chr2-rname	Block-N-AU1-chr2-pos	Block-N-AU1-chr2-pair	Block-N-AU1-chr2-rname	Block-N-AU1-chr2-mpos
Block-P-AU2-chr2-pos	Block-P-AU2-chr2-pair	Block-P-AU2-chr2-rname	Block-N-AU2-chr2-pos	Block-N-AU2-chr2-pair	Block-N-AU2-chr2-rname	Block-N-AU2-chr2-mpos
Block-P-AU3-chr2-pos	Block-P-AU3-chr2-pair	Block-P-AU3-chr2-rname	Block-N-AU3-chr2-pos	Block-N-AU3-chr2-pair	Block-N-AU3-chr2-rname	Block-N-AU3-chr2-mpos

Figure 5.3: MPEG-G encoded data grouped in Descriptor Streams

perfectly matched, matched with some unread nucleotides), whereas in the case of unaligned data, this could mean a list of subsequences of nucleotides.

- Where are the blocks corresponding to the Access Unit? In AUC mode, the Access Unit Container stores the blocks. In the case of DSC mode, the tool must retrieve the blocks associated with the Access Unit from the stream. Therefore we need to know the offset at which the block starts in the stream. The indexing table provides this information, in case of the file being in DSC mode.
- Where is the Access Unit Container stored? Even in DSC mode, the Access Unit Container is still needed to retrieve some of the auxiliary tags. The indexing table must establish the link between the region on the genome and the Access Unit Container. An integer corresponding to the offset in the file where the Access Unit Container associated with the region begins constitutes the link.

To parse and decode the contents of the Access Unit, we must identify each block. In AUC mode, the block header helps to identify the content. The block header gives the size of the block and the nature of its data (to which descriptor it corresponds). In DSC mode, there is no need for block headers. All blocks stored within the same stream store the same type of information. Therefore, the syntax records the nature of the information only once. This difference introduces an overhead: the file size in AUC mode is higher than in DSC mode, due to the block headers.

5.3.5 Extending the format to other use cases

From the beginning, the security requirements were the first identified. We then also introduced the requirements of transporting metadata and of labeling regions of the genome. The MPEG-G specification handles these requirements by introducing new structures at each level:

1. Dataset Group level
    - A Dataset Group Protection element encapsulates the required information regarding the privacy of the Dataset Group's specific information, the authenticity of this information, and the access control rules applied to it.
    - A Dataset Group Metadata element encapsulates all metadata information relevant to the Dataset Group.
    - A set of Label elements encapsulates collections of regions that the user has associated with a specific label represented with a string.

2. Dataset level

    - A Dataset Protection element encapsulates the required information regarding the privacy of the Dataset's specific information, the authenticity of the same information, and the access control rules applied to it.
    - A Dataset Metadata element encapsulates all metadata information relevant to the Dataset.

3. Access Unit level

    - An Access Unit Protection element encapsulates the required information regarding the privacy of the Access Unit information and the authenticity of this information.

4. Descriptor Stream level

    - A Descriptor Stream Protection element encapsulates the required information regarding the authenticity of the Descriptor Stream's information.

As access control rules specify which API actions are allowed to be executed by a given user under specific consequences, and API actions only apply to Dataset Groups and Datasets, there would be no use for action rules at the Access Unit and Descriptor Stream levels. Therefore, access control rules are only specified at the Dataset Group and Dataset levels.

5.4 Thoughts on the adopted approach

5.4.1 Removing the DSC mode

One can give two main reasons to justify representing the information in the stream approach. The first reason is that since the encoder and the decoder operate using Descriptor Streams as outputs and inputs respectively, the Descriptor Streams are a natural choice for the construction of the file format. The other reason is that it might be useful to access some subset of streams without accessing the others. For example, if one had decoded only the information of the 'pos' descriptor, in the early versions of MPEG-G, this would have yielded a list of distances between reads. This information could then have been used to draw statistics for the coverage. However, after several reworks of the MPEG-G encoding specification, the Descriptor Streams are interleaved: for example, it is not possible anymore to decode the position descriptor correctly without knowing how many alignments there are for a specific read. Therefore, the previously presented use case is not useful anymore.

Furthermore, to decode each block of data, the parameters used for the entropy encoder must be known, but this information is only available through the parameter set bound to the Access Unit. It is now impossible to retrieve information from the streams without accessing first information on the Access Unit and possibly also other streams relevant to the one which shall be decoded.

Based on this observation, we proposed twice to the MPEG group to remove the possibility of representing the file in Descriptor Stream Contiguous mode. Our proposal was not accepted. On the first occasion, the group rejected it because the use case was still valid (this was before modifications in the encoding hindered the use case). On the second occasion, the group rejected the modification as it seemed too significative for a late change. Nevertheless, the concerns we expressed are more valid than ever. Removing the Descriptor Stream Contiguous mode would simplify the standard by removing a mode which intrinsically makes access to the information more complicated and would, in turn, simplify the development of software for the MPEG-G file format.

The fact that, in AUC mode, each block signals its nature causes overhead. We can also address this issue. The specification could be modified to force the order of the blocks to be always the same and to use always the same list of blocks. If we introduced these criteria into the standard, there would be no further need to indicate the nature of each block, thus removing one reason for the overhead.

5.4.2 Creating new structure options

Another issue that was not yet raised in the MPEG group is the point that the hierarchical structure might not be sufficiently refined in some instances. Let us imagine a study that compares, for each subject, the DNA observed in healthy tissue and the DNA observed in cancerous tissue. In that case, we would prefer the following structure: Dataset Group (study), patient container, Dataset (DNA). In the current state, this hierarchy is not possible as the grouping by patients does not exist. This lack of information representation features obliges to use metadata in order to obtain the intended grouping.

Similarly, we can imagine a study where control and case patients should be separated. This requirement could also be combined with the previous one. This highlights the need for an even more adaptive file structure.

Meeting this requirement could be as easy as creating a new grouping container, which would be placed within the Dataset Group as a substitution of the Dataset. A grouping container could contain either a collection of further grouping containers or a collection of Datasets. The semantic meaning of the grouping could then be discovered by using specific metadata fields.

(a) AUC mode without indexing information

(b) AUC mode with indexing information

Figure 5.4: Basic MPEG-G file structure when in AUC mode

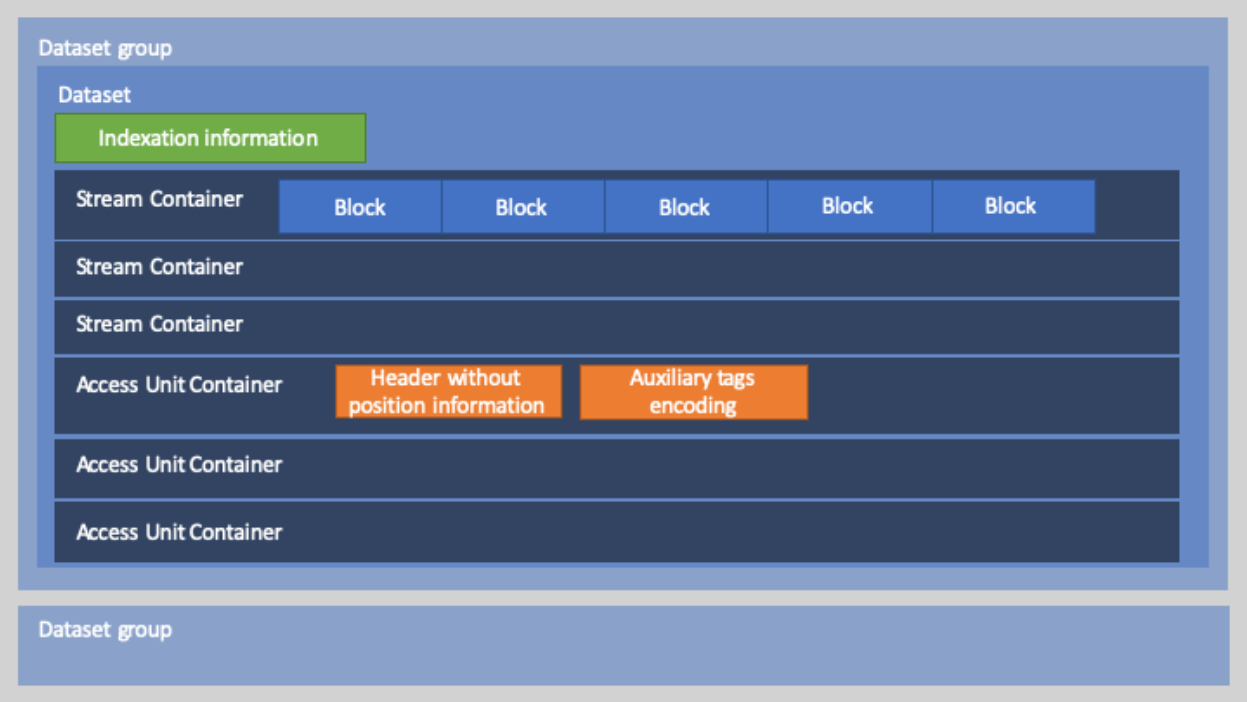


Figure 5.5: Basic MPEG-G file structure when in DSC mode

Chapter 6

Metadata

As we have seen in Chapter 3, the sequenced genomic data must be accompanied by its metadata. We present our approach for metadata encoding. We rely on a core schema which can be extended to match each repository's requirement. We simplify the compatibility with the genomic data repositories by defining a profile mechanism. MPEG has adopted our work (published in [68]) in the standard.

6.1 Requirements

The MPEG-G file format defines a layered structure for the representation of the information. At the highest level of the hierarchy, there is the Dataset Group. Within the Dataset Group, MPEG-G stores Datasets. Datasets contain access unit containers. Only the Dataset is equivalent to currently existing formats: it represents either aligned or unaligned data, similar to SAM and FASTQ, respectively.

The access unit is a construction intended to provide a smaller working unit for codification, random access and security. The access unit is not a unit designed to exist outside the scope of MPEG-G. Its existence should be as transparent as possible to the user. Thus there are no requirements to document it.

The Dataset is the representation, in an MPEG-G file, of the content stored currently in a SAM or FASTQ file. We know that the SAM specification provides a schema for metadata relevant to its content. We have also seen that databases such as EGA specify metadata schemas for such a type of information. There is only a partial overlap between the SAM header and the EGA schema. There is thus the need to document, with metadata, the content of the Dataset. It is, however, unclear which fields the specification should include in the documentation. Requiring too many entries might make it impossible to use MPEG-G documentation if the information is not available (for old genomic files which have been generated without this information) or non-existent (the genomic data has been sequenced with another use case in mind, a use case with different documentation needs).

The Dataset Group does not have an equivalent in current file formats. As we have seen, the Dataset Group is intended to group multiple Datasets into a unit of similar Datasets. The MPEG-G specification does not define which criteria of similitude to use. To inform the recipient about the clustering criteria, we require means to document the Dataset Group with metadata information. Databases such as EGA also specify similar groupings of results. In these databases, the cluster itself also requires its documentation. As in the case of the Dataset, it is unclear which set of fields to use, as the relevant specification might vary between use cases.

We have seen that two of the three layers (the Dataset Group and the Dataset) in the hierarchy of the MPEG-G file require documentation. In the case of the Dataset level, in the absence of a metadata structure, information currently provided in the SAM header would be lost when converting from a SAM-like file to MPEG-G. To be compatible with existing databases of genomic results, MPEG-G also requires to be consistent with their metadata specification. Nevertheless, this metadata specification might not be relevant to every use case, or the information might not be available in all instances.

The specification for metadata representation must meet the following requirements:

- The Dataset Group and Dataset containers must provide structures for the encoding of metadata.
- Compatibility use cases with current metadata specifications both from current formats and existing databases must be representable in the metadata specification.
- The user must not be forced to meet another unrelated compatibility use case to meet one compatibility use case.

6.2 Our proposed approach

6.2.1 Concept

We have proposed a metadata representation based on XML. In our proposal, there are two XML schemas, each one defining one of the metadata structures required, i.e., one for the metadata of the Dataset Group, the other for the metadata of the dataset. As we have seen, the metadata specification needs to be adaptable to the different use cases. We achieve this by defining, in our XML schemas, a standardized list of metadata fields and a way to provide extensions. The motivation behind the shared standardized list of metadata fields is to have only one representation of the most common metadata fields across the different use cases. Our proposal represents the specificity of each use case with extensions. An extension is a new element, specified outside the definition of the core metadata representation.

As the Dataset Group is intended to group related datasets, we expect to have redundancy across the metadata of the different datasets. For example, the same sequencing center may have sequenced all datasets. If we want to store this information in the metadata of the dataset, we will be forced to store the same sequencing center name in each dataset's metadata. To overcome this, we have included in our proposal a mechanism to represent the shared field's value only once.

6.2.2 Inheritance

To limit the overhead introduced by the metadata representation, we want to reduce the number of repetitions of values across the datasets. We propose to resolve this issue by specifying a mechanism through which the dataset's metadata inherits values from the Dataset Group's metadata.

In the previously considered example, all datasets' metadata elements had to include a field specifying the sequencing center where the information stored in the dataset was sequenced. Let us imagine the case where the value of this field is the same across all datasets. With our inheritance mechanism, our proposal stores the sequencing center's name only once. The Dataset Group metadata provides the value for this field, and each dataset inherits this value.

The syntactical level does not represent the inheritance. Only the semantic level takes into account the inherited values. If we were to retrieve the bytes encoding the metadata of any one of the datasets, we would not see the sequencing center's name. Yet an MPEG-G compliant tool shall, when retrieving the metadata of the dataset, extract all inherited fields from the Dataset Group and return them within the remaining dataset's metadata XML document.

The inheritance mechanism would not allow that one or more datasets differ from the metadata defined in the Dataset Group. Therefore, we included in our proposal the idea of overwriting. If a dataset and a Dataset Group both provide a value, the dataset does not inherit it, but rather the one provided by the dataset is used.

6.2.3 Core metadata specification

With 'core metadata', we refer to the set of metadata fields that belong to the base schema. There are two instances of core metadata: one for the Dataset Group and one for the dataset. Our objective when defining the core was to enforce the presence of necessary information and to provide one storage location for the most common fields in the different databases. As can be seen from the specification of the core metadata shown in XML source 6.1, we mark the difference between the mandatory information and the common but non-mandatory fields by the optionality of the element in our schema. In the case of the dataset, as the information provided by the Dataset Group is inherited, we do not place any fields as mandatory, as can be seen in XML source 6.2.

XML source 6.1: Specification of the core metadata for the Dataset Group

```
1  <xs:element name="DatasetGroup" type="mpg-meta-data-gr:DatasetGroupEncryptedType"/>
2
3  <xs:complexType name="DatasetGroupEncryptedType">
4   <xs:sequence>
5    <xs:element name="Title" type="xs:string"/>
6    <xs:element name="Type">
7     <xs:complexType>
8      <xs:attribute name="existing_study_type" use="required">
9       <xs:simpleType>
10       <xs:restriction base="xs:string">
11        <!--[...]-->
```

```
      </xs:restriction>
     </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="new_study_type" use="optional" type="xs:string"/>
   </xs:complexType>
  </xs:element>
  <xs:element name="Abstract" type="mpg-meta-data-gr:AbstractString" minOccurs="0"/>
  <xs:element name="ProjectCentre" type="mpg-meta-data-gr:ProjectCentreType"
   ↪ minOccurs="0" />
  <xs:element name="Description" type="mpg-meta-data-gr:DescriptionString" minOccurs="0"
   ↪ />
  <xs:element name="Samples" type="mpg-meta-data-gr:SamplesType"/>
  <xs:element name="Extensions" type="mpg-meta-data-gr:ExtensionsType"/>
 </xs:sequence>
 <xs:attribute name="profile" type="xs:anyURI" use="optional"/>
</xs:complexType>

<xs:simpleType name="AbstractString">
 <xs:restriction base="xs:string">
  <xs:maxLength value="1024"/>
 </xs:restriction>
</xs:simpleType>

<xs:complexType name="SamplesType">
 <xs:sequence>
  <xs:element name="Sample" type="mpg-meta-data-gr:SampleType" minOccurs="1"
   ↪ maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="SampleType">
 <xs:sequence>
  <xs:element name="TaxonId" type="xs:int" minOccurs="1" maxOccurs="1"/>
  <xs:element name="Title" type="xs:string" minOccurs="0" maxOccurs="1"/>
  <xs:element name="Extensions" type="mpg-meta-data-gr:ChildExtensionsType" minOccurs="0"
   ↪ maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="ProjectCentreType">
 <xs:sequence>
  <xs:element name="ProjectCentreName" type="xs:string"/>
  <xs:element name="Extensions" type="mpg-meta-data-gr:ChildExtensionsType" minOccurs="0"
   ↪ maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>
```

XML source 6.2: Specification of the core metadata for the dataset

```
<xs:complexType name="DatasetType">
 <xs:sequence>
  <xs:element name="Title" type="xs:string" minOccurs="0" maxOccurs="1"/>
  <xs:element name="Type" type="xs:string" minOccurs="0" maxOccurs="1"/>
  <xs:element name="Abstract" type="xs:string" minOccurs="0"/>
  <xs:element name="ProjectCentre" type="mpg-meta-data-gr:ProjectCentreType"
   ↪ minOccurs="0" maxOccurs="1"/>
  <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs="1"/>
  <xs:element name="Samples" type="mpg-meta-data-gr:SamplesType" minOccurs="0"
   ↪ maxOccurs="1"/>
  <xs:element name="Extentions" type="mpg-meta-data-gr:ChildExtensionsType" maxOccurs="1"
   ↪ minOccurs="0"/>
```

```
 </xs:sequence>
 <xs:attribute name="profile" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

6.2.4 Extensions

Certain genomic repositories, such as EGA (see Clause 3.3.2), require a broader set of fields than the ones provided in the core metadata sets. In order to address this issue, we propose to use the concept of extensions, which the Genomic Standards Consortium proposal (see Clause 3.3.4) also considers. As the name indicates, the objective of an extension is to extend the scope of the metadata structure by providing new fields. We specify an extension (XML source 6.3) as an element providing a type identifier and a value. MPEG-G compliant tools are to use the type identifier to relate the provided value with the corresponding semantics. We expect MPEG-G tools to implement well-known extensions, such as the ones we provide as informative material in our proposal, yet there is no requirement on supporting every existing extension. To mitigate the possible issues which could be caused by this, we have also added a documentation pointer to the schema of the extension. If a party generates an MPEG-G file with an extension which is not well-known, a human-readable documentation will maintain the usability of the field.

We have seen that in order to reduce the overhead, we have specified an inheritance mechanism. As some extensions provided in the Dataset Group's metadata structure might be only relevant to the Dataset Group, we define an attribute in the extension schema to indicate whether said extension is to be inherited by the dataset or not. Extensions can also appear at the dataset level (to correct information provided in the Dataset Group level) and within certain elements (as in the sample description). The schema does not define the 'inheritable' attribute in these cases, as the inheritance mechanism does not apply for them.

XML source 6.3: Specification of the core metadata Dataset Group

```
 <xs:complexType name="ExtensionType">
        <xs:sequence>
            <xs:element name="Type" type="xs:anyURI"/>
            <xs:element name="Documentation" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="Inheritable" type="xs:boolean"/>
            <xs:any minOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
```

Alongside the specification of the schemas and the definition of the mechanism, our proposal also integrates examples of extensions allowing an MPEG-G file to meet the requirements of specific use cases. We define one extension to maintain the SAM header in the case of a round-trip from a SAM-like file to MPEG-G and back. The extension is a translation of the SAM header specification in XML format, thus making it compatible with our proposal.

We also propose a specification of extensions to include information to make the metadata of an MPEG-G file compliant with the EGA repository. This specification consists of two schemas: one for the information relative to the sample, the other relative to the experiment to which the current information belongs.

6.2.5 Profiles

The extension mechanism allows us to increase the representation capacity of the core metadata set, yet at the possible cost of harming the interoperability of the file. We have seen that our proposal integrates identifiers of the extension and signaling mechanism to convey the semantics of the extension. However, to meet specific use cases, such as uploading an MPEG-G file to EGA, multiple extensions are required.

Our proposal defines the concept of profile to signal the compatibility with the more complex use cases. Both the Dataset Group and the dataset can signal that they comply with a profile's specification. The profile attribute must be equal to the profile identifier. When a Dataset Group or a dataset indicates the compliance with the profile, all extensions necessary to the profile must be present.

The MPEG-G file does not define the profile. The MPEG-G compliant tool must retrieve the requirements of the profile and its semantics through other channels.

In our proposal, we define the profile for the EGA compatibility. The definition specifies which extensions the Dataset Group and the dataset's metadata must provide to meet the requirement. However, our discussion with the EGA team has highlighted that extensions are not enough to guarantee full compatibility with existing services. Moreover, additional requirements must be placed on the schema for the core metadata set. In the case of the EGA profile, the sample's taxonomy has to be equal to that of the human species and

the description element of the core metadata set must be populated. We also define a profile for the GDC compatibility.

6.3 The adopted approach

The retained solution was our proposal, and it has been taken over in section 6 of Part 3 Metadata and application programming interfaces of ISO ISO/IEC 23092-3 [60].

Chapter 7

Security

After having presented the encoding of the sequenced genomic data and its metadata in chapters 4 and 6, we present now the main contribution of this thesis: how to protect both the sequenced genomic data and its metadata in their respective encoded representations. In the case of the metadata, we take advantage of its XML encoding to protect it: we use XML encryption and XML signature. This approach was adopted by MPEG. In the case of sequenced genomic data, we present the evolution of our approach. We initially proposed mechanisms for key transportation and for encryption parameters transportation for every container in the file. When MPEG formulated new requirements, we reviewed our proposal to specify the encryption parameters in terms of genomic regions and store them at the Dataset level. MPEG adopted our final approach. The encryption is partly covered in [66].

In this chapter, we also prove the deficiencies of the initial encryption proposal: in case the encryption is done per stream, using the properties of MPEG-G's encoding permits to deduce the content of a theoretically encrypted stream on the basis of the contents of other unencrypted ones. A paper on this attack has been submitted [69].

Encryption strategies cannot prevent information leakage: a user who has been granted access to a file could unlawfully publish an unencrypted copy. To address this issue, we propose a watermarking algorithm for genomic data. The modifications introduced by this algorithm are virtually undetectable and reversible if the data holder reveals the watermarking key. However, if multiple users were to collude, they could unlawfully reverse the watermarking. We present a variation of the algorithm which addresses this issue. This watermarking approach is compatible with any current sequenced genomic data encoder. We published this algorithm in [70].

7.1 Protecting the metadata information

7.1.1 Use cases to meet

We have seen that the information in the metadata can be sensitive, either because it might affect the private sphere, or because it affects the intellectual property. In order to address this issue, we need to have a mechanism to protect the privacy of the metadata information.

The metadata might be a key element in the process executed in a pipeline. Modifying a Metadata element might be used as the vector of an attack affecting the correct functioning of services. Therefore we need to protect not only the privacy but also the integrity of the information.

One of the objectives of MPEG-G is to provide a file whose information is entirely encrypted. A party can offer a service with the MPEG-G file without having any access to it. Such services might be the hosting of the file and offering access to it to the intended users. Such services could be offered, for example, by a company recollecting the data from sequencing centers and giving access to them to a practitioner. In such a situation, the company would not have any rights to access the file, but it is in everybody's interest that the company can determine if an adversary has compromised the integrity of the metadata information. The service provider should be able to perform the integrity test without having to unencrypt the content in order to meet the requirement.

We have also seen that multiple entities might have an interest in the same MPEG-G file. For example, a sequencing center might generate an MPEG-G Dataset and sign the contents of its metadata. Upon reception, the researcher will validate the metadata and then sign it, alongside the metadata of the other Datasets, and maybe even of the Dataset Group grouping all this information together. We can extract from

this that the signature mechanism of the metadata content should allow the information to be signed any number of times and by as many parties as required.

7.1.2 The solution we have proposed

7.1.2.1 The selected technologies

As MPEG-G stores the metadata information in XML format, the most straightforward approach is to use XML security [71, 72] to implement both encryption and signature needs.

XML signatures allow signing information from an XML document. The standard defines how to identify the keys, how to reference the content, and which settings are available (e.g., the hashing method used, the signature algorithm employed). We have proposed, and it has been accepted, to use XML signatures to provide the signature use cases. We do not place any requirements on the number of provided signatures: each signature can be used independently of the others. Furthermore, thanks to the internally referencing mechanisms provided in the specification of XML signature [72], our approach requires minimal modifications to the defined metadata schemas.

XML encryption [71] enables us to substitute an element with an encrypted element in an XML document. This encrypted element provides the ciphertext and the parameters which have to be used to decrypt the content: the encryption algorithm, the padding algorithm and the identifier of the key used. Upon reception, the receiver follows the specification of XML encryption to revert the changes: the encrypted elements are decrypted, yielding the original element, which is then placed within the XML document in substitution to the encrypted element. We thus proposed to MPEG to use XML encryption for the handling of the encryption of metadata, proposal which was accepted.

We should also note that the signature element can refer to the encrypted content, thus allowing us to sign encrypted content.

7.1.2.2 The implementations

We materialized our proposal in an updated schema for the metadata content. For the different elements of the metadata schema, we had to enable the choice between the regular element or an encrypted element. We then added new *signature_ elements* to provide the place holders for the signature mechanism. There are no restrictions on the number of signature elements, allowing as many parties as necessary to provide their confirmation on the quality of the information. In order to make the signing of the content as easy as possible, we included in each metadata element an optional 'id' attribute, which can then be used within the signature element to refer to it while signing.

We have chosen not to specify the mechanism by which the mapping between the key names used in the XML signature and the encryption elements and the key value is signaled. The distributor of the file has to give the key and its identifier to the end user, using an external signaling mechanism.

The solution we have proposed conforms to every use case we have identified. The solution is also easy to integrate into future MPEG-G tools, as there are libraries available to handle these mechanisms.

7.2 Protecting the data

7.2.1 Requirements

The information obtained from the DNA is extremely sensitive as it not only reveals sensible information on the patient, but also on his family. The same information can also be useful when understanding a specific genetic disease: by comparing the genome of a pool of individuals not presenting a given disease and a pool of individuals which do present this disase, we can draw conclusions on which mutation might be at the origin of the illness. We see that there are incentives both to protect and to release the genetic information. We have met the requirement for fine-grained privacy settings.

Some regions might be of interest for a doctor, other regions might be of interest to a researcher, and some others to both. Therefore we need to be able to select which set of recipients has access to which region of the file, and the set of recipients might differ from region to region.

Harm could also be done if the results came to be altered. For example, if the results obtained by a sequencing center are altered before being received by the medical team, the results of the diagnosis could be modified. Therefore we also have the requirement of being able to guarantee the integrity and authenticity of the information. The requirement for fine-grained authenticated information is less clear than in the case of privacy. Nevertheless, we should consider the case of a party receiving a file, or transmitting a file to another party, such that the final recipient will have access to a file with some encrypted portion for which the key is unknown and will remain like that. In this case, there is no purpose in keeping the encrypted information

or transmitting it, and it would be preferable to remove it altogether in order to lessen the burden on the storage system or the network. This should not affect the signatures for the accessible information. Therefore we also have the requirement for fine-grained units to be authenticated.

7.2.2 Our proposed solution

7.2.2.1 Early proposal

In our first proposal, we used the subdivision into blocks to provide both encryption and signatures. As a specific region of the file was being accessed, the corresponding blocks were to be retrieved from the respective streams (i.e. one block per stream). While accessing each block, the signature had to be checked, and, if required, the privacy settings had to be followed to retrieve the block as plaintext.

This approach readily complied with the identified requirements: as the settings were provided at the block level, the authenticity and privacy of the information were already provided at the genomic width of the block (i.e. the size of the genomic region associated to the block). It was thus a matter of the encoding process to generate encoding units, materialized with their corresponding blocks, which would correspond to the privacy and authentication requirements. In order to guarantee that only the correct set of recipients came to have access to the region, the key used to guarantee the privacy was to be adapted for each region: a key known to only those recipients who were to have access to it.

Due to the fact that we proposed a file format with a hierarchy in multiple layers — and at each layer supplementary material such as metadata, indexation information and similar was present, we had also the need to protect those elements, both in terms of privacy and authenticity of the information. The same techniques apply: in the protection element, for example of the Dataset, we specified that the Dataset Metadata element was encrypted. We also used the protection element to protect the protection element of a layer below. For example, the protection element at the study level was used to encrypt and sign, if needed, the protection element of the Datasets. The encryption and signing of each element could be done independently.

7.2.2.2 Revised proposal

As the idea of having one block header per block in the stream was not taken over in MPEG-G, the foundation of our proposal was removed. Instead of creating additional structures at the block level, the structuring of information in MPEG-G is such that there are additional elements at the Access Unit level. This level, which was semantically but no syntactically present in our first proposal, is equally relevant to the currently discussed use case.

We modified our proposal to indicate at the level of the Access Unit the parameters to guarantee the privacy and authenticity of the information. This information was stored in an Access Unit Protection box, stored within the Access Unit Container element. In the case of the file being in Access Unit Contiguous mode, the Access Unit Protection box would indicate how to interpret the blocks stored in the same Access Unit Container. In the case of Descriptor Contiguous Mode, the Access Unit Protection box would provide the configuration on how to interpret the blocks associated to the Access Unit, albeit stored in the Descriptor Stream Container.

Whereas in the case of protecting the metadata we relied entirely on existing technologies, namely XML encryption and XML signatures, it was not possible to do the same here, at least not at the full extent. Since XML encryption is meant to encrypt elements of an XML document, and not external resources such as, in our case, the blocks stored externally we had to propose an alternative. Some of the aspects of XML encryption were interesting. Most notably the fact that it relies on a list of well-known ciphers, just provides a container for the required parameters, and defines ways to derive keys from other keys. We have thus proposed an alternative which takes over these concepts, but which was intended to specify the encryption parameters which have been used for an external resource.

As in the case of XML encryption, our approach represented the configuration in XML, for which we show the schema in XML source 7.1. From the schema we can see the similarities with XML encryption: there is a field to specify which cypher has been used (Advanced Encryption Standard -AES- 128, 192 or 256 bits in Counter -CTR- mode or Galois/Counter Mode -GCM-), which key has been used (identified with a string), the Initialization Vector (IV) used, if required, and the TAG used for integrity check if supported by the cipher. The location of the cyphertext, and thus the semantics of the encrypted content, is derived form the URI attribute 'encryptedLocation'. The specification defines a list of possible URIs, each mapping to a specific Value of one of the elements in the file internal tree structure.

For example, at the Dataset level, the encryptedLocation URI could point to one of the parameter elements, if the latter needed to be encrypted.

XML source 7.1: Encryption parameters schema

```
<xs:complexType name="EncryptionParametersType">
 <xs:sequence>
  <xs:element name="cipher" type="CipherURIType" minOccurs="0">
  <xs:element type="xs:string" name="keyName"/>
  <xs:element type="xs:base64Binary" name="IV"/>
  <xs:element type="xs:base64Binary" name="TAG" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="encryptedLocations" type="xs:anyURI" use="required"/>
</xs:complexType>
```

7.2.2.3 Key transport

One of the requirements for the privacy settings was that the right set of recipients was granted access to the content of an element. The entity in charge of distributing the file will have to distribute the keys used to the entitled parties. However, in order to simplify the process of distributing keys, we have defined multiple structures which can help reduce the need to distribute keys: from a shared secret, the remaining keys can then be recovered.

Let us imagine the case where an entire Dataset Group is meant for a specific recipient, individual A. One of the Datasets within the Dataset Group can also be accessed by individual B. The party generating the file will encrypt the entire Dataset Group except the Dataset also meant for B with key K_1, while the specific Dataset which should be accessible to both A and B is encrypted with key K_2. Key K_1 is then revealed to individual A, and K_2 to individual B. The file then also indicates how to obtain the value of K_2 based on the value of K_1. This can be done either by deriving the value of the key or decrypting. In order to transport this configuration, we have proposed the schema for an XML element (element 'KeyTransport'), which ties together a name for the newly obtained key ('keyName') with the value of this new key. In our proposal, we define two main ways to retrieve the value of the new key;

1. computation
   The Password-Based Key Derivation Function 2 (PBKDF2) algorithm defined by the Internet Engineering Task Force (IETF) in Release For Comment (RFC) 8018 [73] allows to generate deterministically a new secret from a previously known secret. An example of the proposed settings to transport this configuration is shown in XML source 7.2. The secret used as the input for generating a new secret is the key identified with 'passwordName', the remaining elements used correspond to the different inputs to the algorithm defined in the RFC.

2. decryption
   The key can also be provided in an encrypted form. There are standards both for symmetric and asymmetric key wraps, and we include in our proposal the schema to transport the settings for both solutions as shown in XML sources 7.3 and 7.4.

In order to allow more complex structures we do not place any requirements on which keys can be used to retrieve a new key. A derived key can be used to derive a second new key, as shown in XML source 7.5.

XML source 7.2: Key transport: key derivation

```
<KeyTransport xmlns="urn:mpeg:mpeg-g:protection:dataset-group:2019">
 <keyName>derivedKey1</keyName>
 <KeyDerivation>
  <PRF>urn:mpeg:mpeg-g:protection:hmac-sha256</PRF>
  <passwordName>password</passwordName>
  <salt>AQIDBAUGBwg=</salt>
  <iterations>10000</iterations>
  <length>32</length>
 </KeyDerivation>
</KeyTransport>
```

XML source 7.3: Key transport: key symmetric wrap

```
<KeyTransport xmlns="urn:mpeg:mpeg-g:protection:dataset-group:2019">
 <keyName>wrapped1</keyName>
 <KeySymmetricWrap>
  <kek>sharedSecretKek</kek>
  <wrappedKey>1FcMIMa0QhOTgNMpkn+j3THvWp7bgCwFJ64FEn+/D6z6VoCLx6xKeA==</wrappedKey>
 </KeySymmetricWrap>
</KeyTransport>
```

XML source 7.4: Key transport: key asymmetric wrap

```
<KeyTransport xmlns="urn:mpeg:mpeg-g:protection:dataset-group:2019">
 <keyName>wrapped2</keyName>
 <KeyAsymmetricWrap>
  <hashFunction>urn:mpeg:mpeg-g:protection:sha384</hashFunction>
  <maskGenerationHashFunction>urn:mpeg:mpeg-g:protection:sha1</maskGenerationHashFunction>
  <publicKeyName>testRSAPublic</publicKeyName>
  <wrappedKey>IMHE5mugWDokW7ljA1ERdu9Ey6OpWk6Xw0J0VFCHC7IQUW2v1c3Fnjnqhbs4f79iGpMwcNHkzjz/ad1NaNBkAky
 </KeyAsymmetricWrap>
</KeyTransport>
```

XML source 7.5: Key transport: key transport chaining

```
<KeyTransport xmlns="urn:mpeg:mpeg-g:protection:dataset-group:2019">
 <keyName>derivedKey2</keyName>
 <KeyDerivation>
  <PRF>urn:mpeg:mpeg-g:protection:hmac-sha256</PRF>
  <passwordName>derivedKey1</passwordName>
  <salt>AQIDBAUGBwg=</salt>
  <iterations>10000</iterations>
  <length>32</length>
 </KeyDerivation>
</KeyTransport>
```

7.2.2.4 Final proposal

As we have seen, our proposal was based on the idea of using the Access Unit as the minimal unit of encryption and signature when it came to protecting the actual genomic data. Due to the process of sequencing genomic information, the data might overlap over different regions of interest. One reason for this, is the size of the actual read, which already creates an atomic unit which can span over regions intended for different sets of recipients, and the fact that two reads can be paired together, increasing the chances of the record belonging to two different regions. In our proposal this was handled by using three different Access Units: the first Access Unit would be encrypted so that the first set of recipients would have access to it, the third Access Unit so that the second had access to it, and the second Access Unit would be encrypted as desired by the patient, for example, so that the intersection of the two sets would have access to the data. This approach is summarized in Figure 7.1.

The MPEG group decided, however, that this approach was too complicated for the end user. The intention of the group was that if the encoding party submitted encryption requirements as in the first row of Figure 7.1, then a decoding party should be able to retrieve the same settings, as this could be considered more meaningful than the end configuration of each Access Unit, as shown in the last row of Figure 7.1. In order to avoid contradictions, and obtain a unique place to retrieve the privacy configuration, the encryption by regions would be defined at the Dataset level. As we have seen, the main problem when encrypting regions of the genome is which action should be taken when a record belongs to two regions with different recipient sets. One solution we proposed was to forbid region definitions which meant that Access Unit would be addressed to two sets of recipients or more. In other words, we proposed to force the user to give as input to the encryption process the depiction of the last row of Figure 7.1, instead of the first row. Our solution was rejected as this would imply that the configuration might not match the biological meaning of the region (e.g. encrypting the content of a gene). The group preferred rather that if an Access Unit intersects the region authorized for one user, then the user shall have access to the Access Unit. We did not favor this approach, as if the entity encrypting the file has no means to modify the content of the Access Unit, this

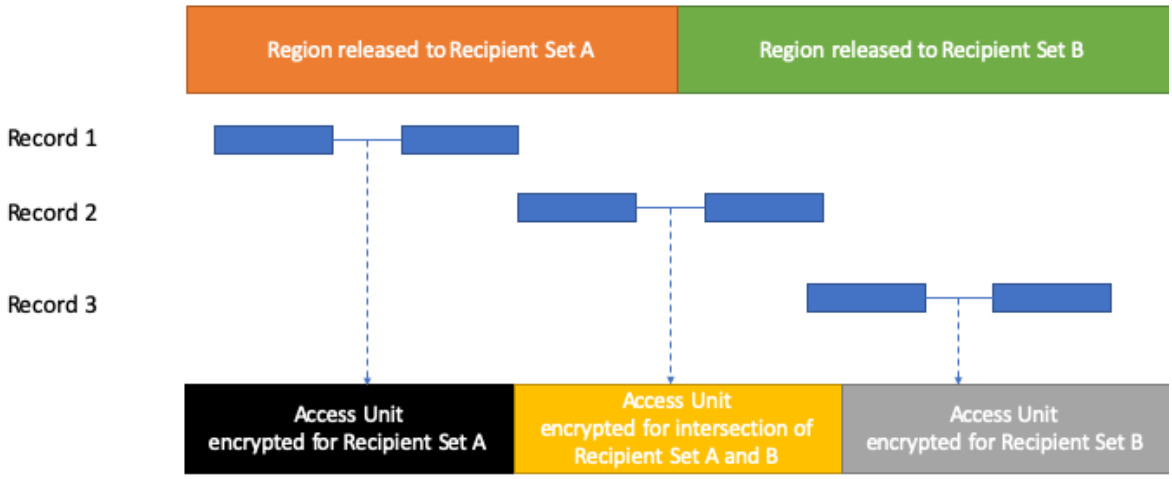


Figure 7.1: In our approach the configuration of recipients for each Access Unit is given at the Access Unit level, independently of any other configuration.

could force to release more information than originally intended. In other words, if in our approach the encryption of the second unit shown in Figure 7.1 was not fixed (the entity encrypting the file could choose any configuration including the intersection of the two sets, their union, or any other alternative), the new requirements fixed the encryption to be addressed to the union of the two sets.

Our final proposal thus has to meet a new additional set of requirements:

- There shall only be one place where the configuration per region is defined.
- The settings used during encryption should match exactly what is stored in the configuration and not be just semantically equal.
- Any Access Unit belonging to multiple regions should be accessible by any user belonging to one of the sets the Access Unit maps to.

Our proposal maintained our idea of encrypting any element within the same container. We used this approach, for example, to encrypt the indexation table, or to encrypt the Access Unit Header (in the case of Access Unit Contiguous without indexation table), in order to hide where on the genome are placed the encrypted elements. We also enabled the possibility to encrypt the list of labels, as the name given to the regions could be a vector of attack: if only the regions of interest of the patient are named, the fact that a region is named and encrypted could prove sensible. The MPEG group placed, however, yet another additional requirement in the form that the security settings should not have any negative effects on the extraction of the data. In other words, a tool implementing only Part 1 of the standard should be able to extract the required blocks, albeit the blocks might be encrypted.

We thus had to revise our proposal in order to meet the new requirements. The first step is to address the issue of letting adversary parties discover which regions are encrypted. We define new metadata extensions to this end: the names given in the MPEG-G Part 1 structures might be syntactically correct but meaningless, and the extension would give the translation between the meaningless information and the real information. For example, an Access Unit could be mapped to the sequence 'k5f32', from position 0 to 10000, and the metadata would then indicate that the sequence 'k5f32' does in fact correspond to the sequence of chromosome 2, starting from position 10 000 000. Of course this implies that the reference sequence used for encoding and decoding the data must reflect these fake names. Similarly, the true content of the label can be hidden by providing a fake name, which is then corrected in the metadata. As we have mechanisms to encrypt the metadata, we can maintain this information secret and still maintain the MEPG-G Part 1 functionalities. We should note, however, that queries such as retrieve records from chromosome 2, will not

yield results from our fake sequence, even if the Access Unit is stored as plaintext, in the case the mapping is unknown, as would happen in an MPEG-G Part 3 agnostic tool.

In order to match the new requirements, we also move the definition of the encryption regions from the Access Unit Protection element to the Dataset Protection element. As requested by the group, the definition is expressed in terms of regions. The 'encryptedLocation' URI will, in the case of encrypting the encoded genomic information, match the following syntax: "genomic_region/{auType}/{sequenceName}/{start}/{end}". As we can see, there are multiple degrees of freedom: the class which shall be encrypted, the sequence which shall be encrypted, and between which positions on the genome shall the data be encrypted. As expressed in the new requirements, there can be as many such entries as necessary, and there is no restriction on how these can intersect or not. As we have seen in XML source 7.1, these URI will be provided in conjunction with other parameters: most notably the Key to be used while decrypting.

However, due to the new requirements, we could face a situation where there are multiple EncryptionParameters elements which provide contradictory information on how to decrypt one or multiple Access Units. In order to do so, we must disambiguate the situation. We update our proposal by modifying the interpretation of the settings. The Access Unit will be encrypted with a key, which might be independent of all other keys. Every party who has access according to what is defined in the EncryptionParameters encrypting the Access Unit shall have access to the key. In order to ensure this, we use the symmetric key wrap approach we had already implemented. The Access Unit is encrypted with a key, which is provided multiple times in symmetrically wrapped form, once per EncryptionParameters with an URI intersecting the range associated to the Access Unit.

In order to achieve this, we have to update our structures, as shown in XML sources 7.6 and 7.7. As the information provided at the Dataset level might not be used directly to decrypt anymore (but only to unwrap the key provided at the Access Unit Protection level), the Initialization Vector (IV) element and the 'cipher' element become optional. And, as there might be multiple different keys used for the unwrapping, we add a 'configurationID' attribute which will be referenced from the Access Unit Protection element.

In the Access Unit Protection element, we define the structure shown in XML sources 7.7, with the following elements:

- an unbound list of wrapped keys
  Each of the wrapped keys is extended with a new 'configurationID' attribute, which links to the key specified for the range at the Dataset level. In the case where only one encryption range intersects the Access Unit range, there is no need to wrap the key: in this case there is no wrapped key provided, and the key belonging to the configuration is directly used.

- cipher
  As at other levels of the file, the encryption parameters specify which cipher is in use.

- initialization vectors
  Two IVs are provided, one for the block encryption, the other for the Access Unit Information's value encryption. This simplifies the specification as to how the ciphertext must be obtained prior to decryption.

- TAG
  Similarly to the IVs, and if required by the cipher, the TAGs are provided both for the blocks and the Access Unit Information's value.

The result of the key wrap is shown in Figure 7.2, where we show the case when the user requests three Access Units to be encrypted according to two regions. Two of the Access Units map to only one region, thus there is no need to wrap the key: the key of the region can be directly used. The third Access Unit maps onto the two regions, therefore a new specific key for the Access Unit is generated, and provided in wrapped form twice, once for each of the privacy regions.

In order to perform the decryption of the ciphertext, the ciphertext in itself must be defined. When we are encrypting the value portion of one of the Key-Length-Value elements of the file format, the ciphertext is already well defined: while encrypting, the bytes of the value are replaced with the value of the ciphertext, whereas during decryption, the bytes of the Value are the ciphertext which is to be decrypted. In the case of Access Unit Contiguous mode, although the blocks of data do not correspond to the bytes of a Value, they do correspond to a clear syntactical group, which can be referred to in the specification. However, in Descriptor Stream Contiguous mode, the blocks are not stored within the Access Unit. Furthermore, there are multiple syntactically independent blocks, yet we need to have just one ciphertext to decrypt. Our proposal thus specifies how the different blocks are concatenated together in one ciphertext, which is then encrypted and stored at their original location. As the cipher we use in our proposal does not cause a variation in size between the plaintext and the ciphertext, there is no issue in modifying the indexing mechanisms. Figure

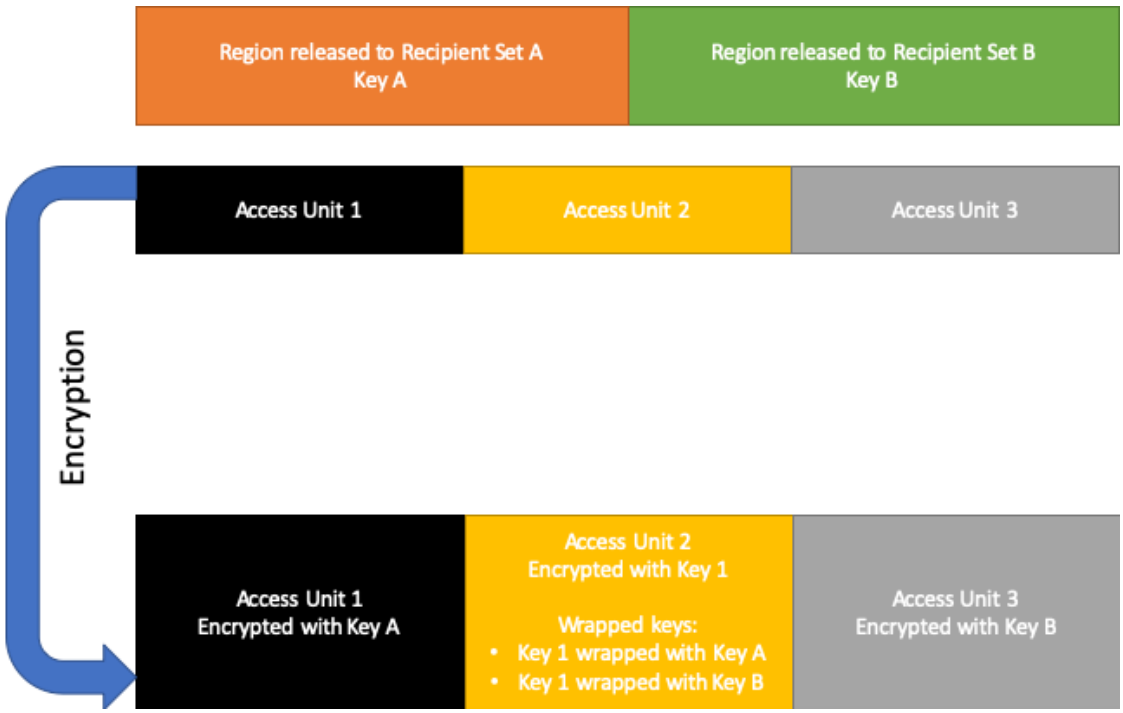


Figure 7.2: Visualization of the key wrapping result

7.3 shows how the different blocks are grouped to form one plaintext which is then encrypted, and again broken down in order to replace the original blocks.

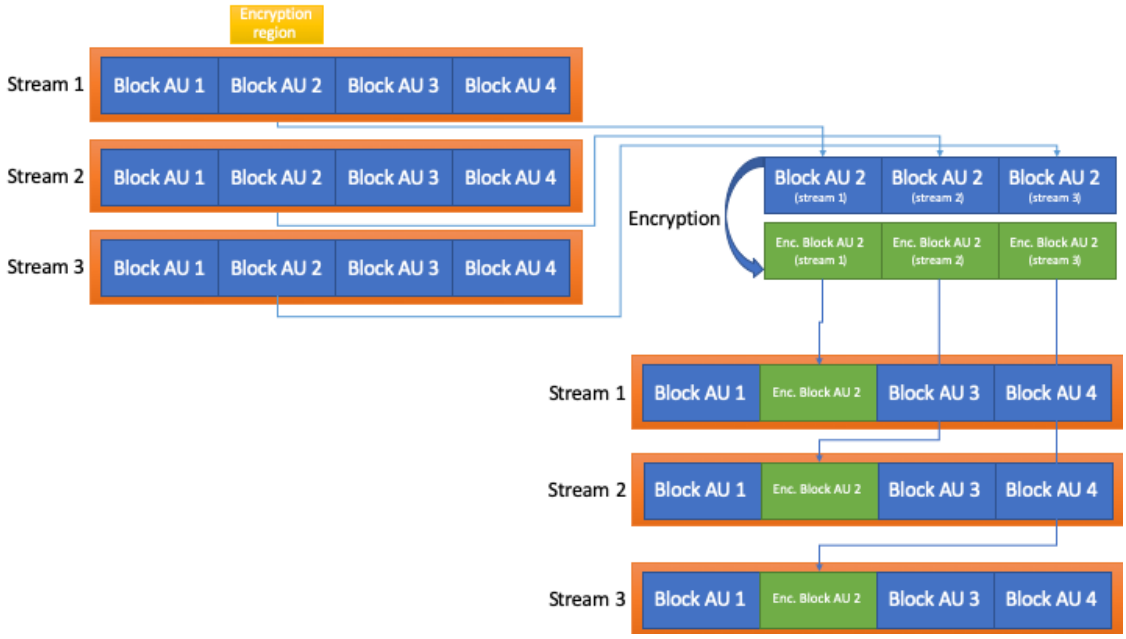


Figure 7.3: Reordering of blocks in Descriptor Stream Contiguous in order to perform encryption

XML source 7.6: Revised encryption parameters schema

```
<xs:complexType name="EncryptionParametersType">
 <xs:sequence>
  <xs:element name="cipher" type="CipherURIType" minOccurs="0">
  <xs:element type="xs:string" name="keyName"/>
  <xs:element type="xs:base64Binary" name="IV" minOccurs="0"/>
  <xs:element type="xs:base64Binary" name="TAG" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="encryptedLocations" type="xs:anyURI" use="required"/>
 <xs:attribute name="configurationID" type="xs:integer"/>
</xs:complexType>
```

XML source 7.7: Revised Access Unit encryption parameters schema

```
<xs:complexType name="AccessUnitEncryptionParametersType">
 <xs:sequence>
  <xs:element name="wrappedKey" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType >
    <xs:simpleContent>
     <xs:extension base="xs:base64Binary">
       <xs:attribute name="configurationID" type="xs:integer"/>
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
  </xs:element>

  <xs:element type="dg:CipherURIType" name="cipher"/>
  <xs:element type="xs:base64Binary" name="aublockIV"/>
  <xs:element type="xs:base64Binary" name="auinIV" minOccurs="0"/>
  <xs:element type="xs:base64Binary" name="aublockTAG" minOccurs="0"/>
  <xs:element type="xs:base64Binary" name="auinTAG"  minOccurs="0"/>
 </xs:sequence>
</xs:complexType>
```

The proposed solution for authenticity of the information, which was not modified since our original proposal, relies on XML signature. We define in our proposal a set of URIs which can be used from the XML signature elements to resolve the signed content. As in the case of the URI used for encryption, these will resolve to either the value element of the KLV structures used in the file structures, or in the case of the Blocks, to the result of a defined process which restructures the collection of related blocks in a determined way.

7.2.3 Retained solution

The retained solution was our proposal, and it has been taken over in clause 7.1 and 7.2 of Part 3 Metadata and application programming interfaces of ISO/IEC 23092-3 [60].

7.2.4 Thoughts on final solution

The additional requirements imposed by MPEG, albeit the quorum decision of the group, are hard to justify. Let us address each one of them.

- There should be only one place where the configuration per region is defined.
  Although persons more concerned with privacy might be interested in encrypting as much as possible, it is fair to assume that for the less engaged user, an encrypted region will be equivalent to a region with a mutation in it. If there is only one place to provide the encryption settings, there is no way to hide this fact to an attacker without blocking access to much more information: as the settings will not be accessible anymore, the decryption will not be possible and we are not meeting the requirements. As we have seen, we have proposed a work-around to this issue. However, our work-around increases the complexity of the random access. Therefore, in order to preserve the user experience on one front, we have to deteriorate it on another front.

- The settings used during encryption should match exactly what was stored in the protection elements and not be just semantically equal.
  Our understanding of the genome improves as research is carried on. If a patient requests the genes related to a given disease to be encrypted, this will lead to a given set of regions to be encrypted. If in the future another patient has the same request, it is possible that the set of regions will be different, as our knowledge of the genome has increased. These two different settings are, however, motivated by the same requirement. Nevertheless, when analyzing which regions are encrypted in the first file, we might not identify the expressed requirement, as it no longer matches the current knowledge. We thus have an example of a situation where the encryption boundaries should not be used to draw conclusions, yet the MPEG group wants to maintain this possibility.

- An Access Unit belonging to multiple encryption regions should be accessible by any user having access to at least one of the regions the Access Unit maps belong to.
  With this requirement the group is forcing to have some information being leaked (information which should only be accessible to one set, and now has to be accessible to other sets of recipients because it also intersects those regions). The magnitude of the leak can be minimized by creating additional Access Units which are meant for the overlapping information, but the leakage is only mitigated, not resolved.

We have also seen how complying with these new requirements has pushed us to increase the complexity of the signaling mechanism, which in turn makes misunderstandings of the standard likelier, so that erroneous files being produced or incorrectly decoded.

7.2.5 Side-channel attack

7.2.5.1 Introduction

In our initial file format proposal, we used a container for each encoded block. For each one, there was a block container grouping all related information, including information on security settings. The MPEG group rejected the concept of a container per block. Instead, in early versions of MPEG-G, security settings for an entire Stream replaced the per-block security settings.

Whereas our file format was intended to transport different encoding, each one with different properties, the MPEG-G file format is meant only to address the MPEG-G encoding. We know that one specificity of this encoding is to generate, for the same Access Unit, interleaved blocks. With interleaved, we refer to the fact that the information provided by one block is dependent on the information provided by another one. For example, the descriptor 'mmpos' will inform of the number of mutations and their positions. In order to decode the type of mutations from 'mmtype' correctly, we need to know the number and position of the mutations.

As the encoder interleaves the information provided by the different Streams, we can assume that by combining the information of some of the Streams, we can infer the content of another Stream. Inferring the content of one Stream is not useful for decoding purposes, but it would prove that encrypting only individual Streams is not enough and even provides a false sense of security. We will attempt to defeat the security mechanism of encrypting one Stream by using the information provided by side-channels.

The rest of this clause will show how we tested the hypothesis that at least one type of side-channel attack is possible. We presented the result of this experiment to the MPEG group. As a consequence, the specification abandoned the mechanism to encrypt individual Streams for the final specification. Instead, encryption encompassing all blocks related to the same Access Unit, regardless of the file format mode chosen, is used.

7.2.5.2 Method

7.2.5.2.1 Available information We will attempt to revert to the first proposed privacy mechanics of MPEG-G. We will be playing the role of an actor having received a copy of an MPEG-G file, with two Access Units (AU), one of class P and the other of class M, both encoding information from the same portion of the genome. This consideration is not unrealistic: if we are able to intercept a file, we will receive multiple Access Units, and it is reasonable to assume that many among those will be of class P or M, and that, in many cases, we will be able to pair them according to the encoded region.

We also suppose that all reads have the same known length. As previously introduced, the privacy-relevant informations are mainly the differences of each read with the reference. Therefore, we will assume that the information stored in the AU of class P is stored in plaintext and the information stored in the AU of class M is partly encrypted: the information storing the position of the reads is stored as plaintext, whereas the information storing the position and type of the mutations is encrypted. At first glance, this

approach of partial encryption is secure, as without the position of the mutation the original information cannot be reconstructed. However, we will attempt to discover where the mutations are and if the mutations affect one or the two copies (chromosomes) of the genomic region.

7.2.5.2.2 Signal processing interpretation We know that for each read stored in an AU of class P, all nucleotides of the read are equal to the reference genome. Therefore, each read in an AU of class P gives us evidence that there are no mutations for a sub-region of the genome in one of the copies. This sub-region is delimited by the first mapped position and the known length of the read. In order to obtain the best compression, a read should only be stored in an AU of class M if there is at least one changed nucleotide (but no other operations such as insertions, deletions, or clips). Therefore, we will take the presence of a read in an AU of class M as proof that there is at least one mutation in one of the copies for the same sub-region (as per our assumed initial state, this information is available).

Our attempt at deducing mutations will now revolve around contrasting all provided evidence and deducing which hypothetical mutations could best explain the observed results. Let us propose an interpretation of the attack using signals. To do this, we will diverge from nature's reality and imagine for a moment that there is only one copy of the information, i.e. there is only one chromosome. Furthermore, we will pretend that there cannot be any insertion or deletion of nucleotides. We will interpret this copy, i.e. the chromosome, as a discrete time signal which can take only two possible values. Each value of the signal is mapped to a nucleotide on the reference genome. If for one position there is no mutation, then the signal is equal to 0, but if there is a mutation, the signal is equal to 1. We will refer to this signal as the $genome_{signal}$. We can now define to which class a read belongs, given its initial position. We sum all values of the signal for the region the read corresponds to. If the sum is greater or equal to 1, there are one or more mutations with respect to the reference, hence the read belongs to class M as otherwise it could not be represented due to the lack of Descriptor Streams. If the sum is equal to 0, we will assume that the read belongs to class P, as this is the choice yielding the best compression. Let us define a new signal $class_{signal}$ as the signal indicating to which class a read starting at that position would belong: we will use 1 as class M and 0 as class P. See Figure 7.4 for an illustration.

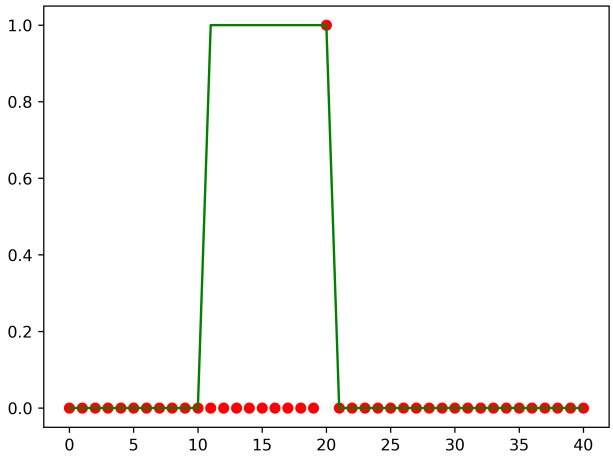


Figure 7.4: Illustration of $class_{signal}$ value (plotted as a solid line), based on a $genome_{signal}$ (plotted as dots), in the case where the read length is equal to 10. $class_{signal}$ is equal to one if a read starting at that position would include at least one mutation, i.e. a position at which $genome_{signal}$ is equal to 1.

We can see how $class_{signal}$ can be obtained by convoluting $genome_{signal}$ with a rectangular signal integrating over the current and future values. However, the result of the convolution is clipped. Thus we lose the information of how many mutations are present.

The last step will be to calculate the expected ratio of class M and class P information for a given position. We know the reads' length, therefore we can compute a signal $ratio_{signal}$, such that for each position p we store the average value of $genome_{signal}$. The average is computed on all reads whose starting position implies that they cover the position p. This computation can be understood as a convolution with a rectangular filter, integrating over the read length position prior to p. See Figure 7.5 for an illustration. This signal does not have any equivalent in nature, but it gives a metric on the likelihood of there being a mutation at position p: if the value of $ratio_{signal}$ at a position p is 0, it is very unlikely that there is a mutation close-by, and if the value is close to 1, it is very unlikely that there is none.

As previously introduced, a read belongs to class M if there is one or more mutations, therefore $class_{signal}$

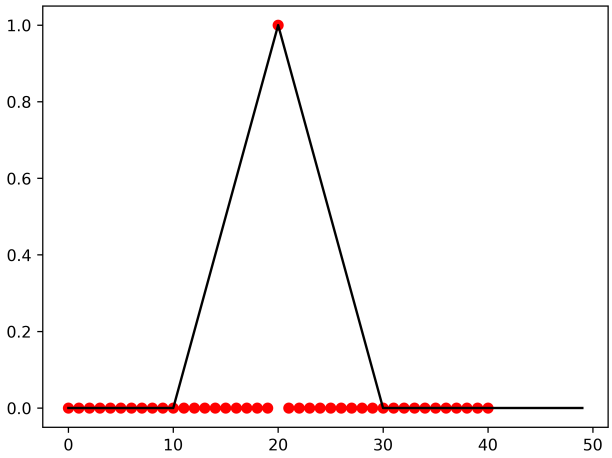


Figure 7.5: Illustration of $ratio_{signal}$ value (plotted as a solid line), based on a $genome_{signal}$ (plotted as dots), in the case where the read length is equal to 10.

loses the information on how many mutations are involved. This implies that there are cases where different $genome_{signal}$ generate the same $ratio_{signal}$. See Figure 7.6 for an illustration of the case where two mutations are separated by a length inferior to the read length, leading to the possibility of a third mutation being hidden.

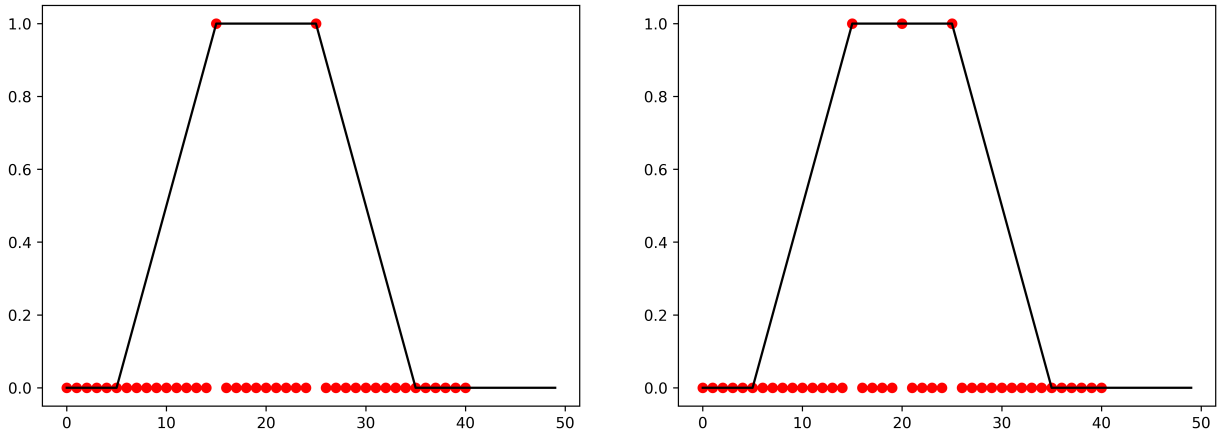


Figure 7.6: If the distance between two mutations is inferior to read length, other mutations might be present between the two without altering the value of $ratio_{signal}$

If we now go back to our initial assumptions, we can see that we know the first position of each read, and the class it belongs to. In other words, we can compute an experimental version of $ratio_{signal}$. By deriving the inverse operation allowing us to go from $ratio_{signal}$ to $genome_{signal}$, we should be able to apply the same procedure to our experimental signal and retrieve a guess for the mutations in the genome, which was supposed to be protected through the encryption of certain fields. Of course, in nature there are two chromosomes, thus we need to add some complexity to our schema to allow for this. We will assume that sequencing technologies sequence from either chromosome with the same probability. Thus, from $genomeMother_{signal}$ we can derive $ratioMother_{signal}$ as we have shown, and the same for the father. The observed $ratio_{signal}$ will then be equal to the average of both.

In order to reverse the convolution, we will need to derive which signal reverts the convolution with a rectangle. We will first derive the representation of a rectangle signal in the z domain. To do this, we defined

the rectangle signal as the difference of two Heaviside signals, with the second signal offset by t_1 time steps:

$$\mathfrak{Z}\{u[n] - u[n-t_1]\} = \left(\frac{z}{z-1}\right) - \left(\frac{z}{z-1}\right)\frac{1}{z^{t_1}} \tag{7.1}$$

$$= \frac{z}{z-1}\left(1 - \frac{1}{z^{t_1}}\right) \tag{7.2}$$

$$= \frac{z}{z-1}\frac{z^{t_1}-1}{z^{t_1}} \tag{7.3}$$

We need to find which signal $y[n]$ reverts the convolution with the rectangle signal, in other words signal $y[n]$ must verify the following equation:

$$(u[n] - u[n-t_1]) * y[n] = \delta[n] \tag{7.4}$$

We translate the problem into the z domain and resolve the equation:

$$\mathfrak{Z}\{(u[n] - u[n-t_1]) * y[n]\} = \mathfrak{Z}\{\delta[n]\} \tag{7.5}$$

$$\frac{z}{z-1}\frac{z^{t_1}-1}{z^{t_1}}Y(z) = 1 \tag{7.6}$$

$$Y(z) = \frac{z-1}{z}\frac{z^{t_1}}{z^{t_1}-1} \tag{7.7}$$

$$\mathfrak{Z}^{-1}\{Y(z)\} = \mathfrak{Z}^{-1}\left\{\frac{z-1}{z}\frac{z^{t_1}}{z^{t_1}-1}\right\} \tag{7.8}$$

$$y[n] = \mathfrak{Z}^{-1}\left\{\frac{z-1}{z}\right\} * \mathfrak{Z}^{-1}\left\{\frac{z^{t_1}}{z^{t_1}-1}\right\} \tag{7.9}$$

$$= \mathfrak{Z}^{-1}\left\{1 - \frac{1}{z}\right\} * \mathfrak{Z}^{-1}\left\{\frac{z^{t_1}}{z^{t_1}-1}\right\} \tag{7.10}$$

$$= \left[\mathfrak{Z}^{-1}\{1\} - \mathfrak{Z}^{-1}\left\{\frac{1}{z}\right\}\right] * \mathfrak{Z}^{-1}\left\{\frac{z^{t_1}}{z^{t_1}-1}\right\} \tag{7.11}$$

$$= [\delta[n] - \delta[n-1]] * \mathfrak{Z}^{-1}\left\{\frac{z^{t_1}}{z^{t_1}-1}\right\} \tag{7.12}$$

In order to resolve the equation we perform the polynomial division:

$$\begin{array}{rl|l}
z^{t_1} & & z^{t_1}\ -1 \\
\cline{3-3}
-z^{t_1}\ +1 & & 1\ +z^{-t_1}\ +z^{-2t_1}\ +z^{-3t_1}\ +... \\
\cline{1-1}
1 & & \\
-1\ +z^{-t_1} & & \\
\cline{1-1}
z^{-t_1} & & \\
-z^{-t_1}\ +z^{-2t_1} & & \\
\cline{1-1}
z^{-2t_1} & & \\
-z^{-2t_1}\ +z^{-3t_1} & & \\
\cline{1-1}
z^{-3t_1} & & \\
... & &
\end{array} \tag{7.13}$$

With this derived we finish the resolution of the equation.

$$= [\delta[n] - \delta[n-1]] * \mathfrak{Z}^{-1}\left\{\frac{z^{t_1}}{z^{t_1}-1}\right\} \tag{7.14}$$

$$= [\delta[n] - \delta[n-1]] * \mathfrak{Z}^{-1}\left\{1 + z^{-t_1} + z^{-2t_1} + z^{-3t_1} + ...\right\} \tag{7.15}$$

$$= [\delta[n] - \delta[n-1]] * (\delta[n] + \delta[n-t_1] + \delta[n-2t_1] + \delta[n-3t_1] + ...) \tag{7.16}$$

$$y[n] = \delta[n] - \delta[n-1] + \delta[n-t_1] - \delta[n-1-t_1] + \delta[n-2t_1] - \delta[n-1-2t_1] + ... \tag{7.17}$$

With this solution we are now able to reverse the convolution with the rectangle signal, i.e. to derive the genome from the observed signal ratios. This process is summarized in Figure 7.7.

In our experiments, however, this approach is too sensible to irregularities in $ratio_{signal}$. Figure 7.8 shows how the noise increases with each convolution. The higher the coverage, the less noise there is, and

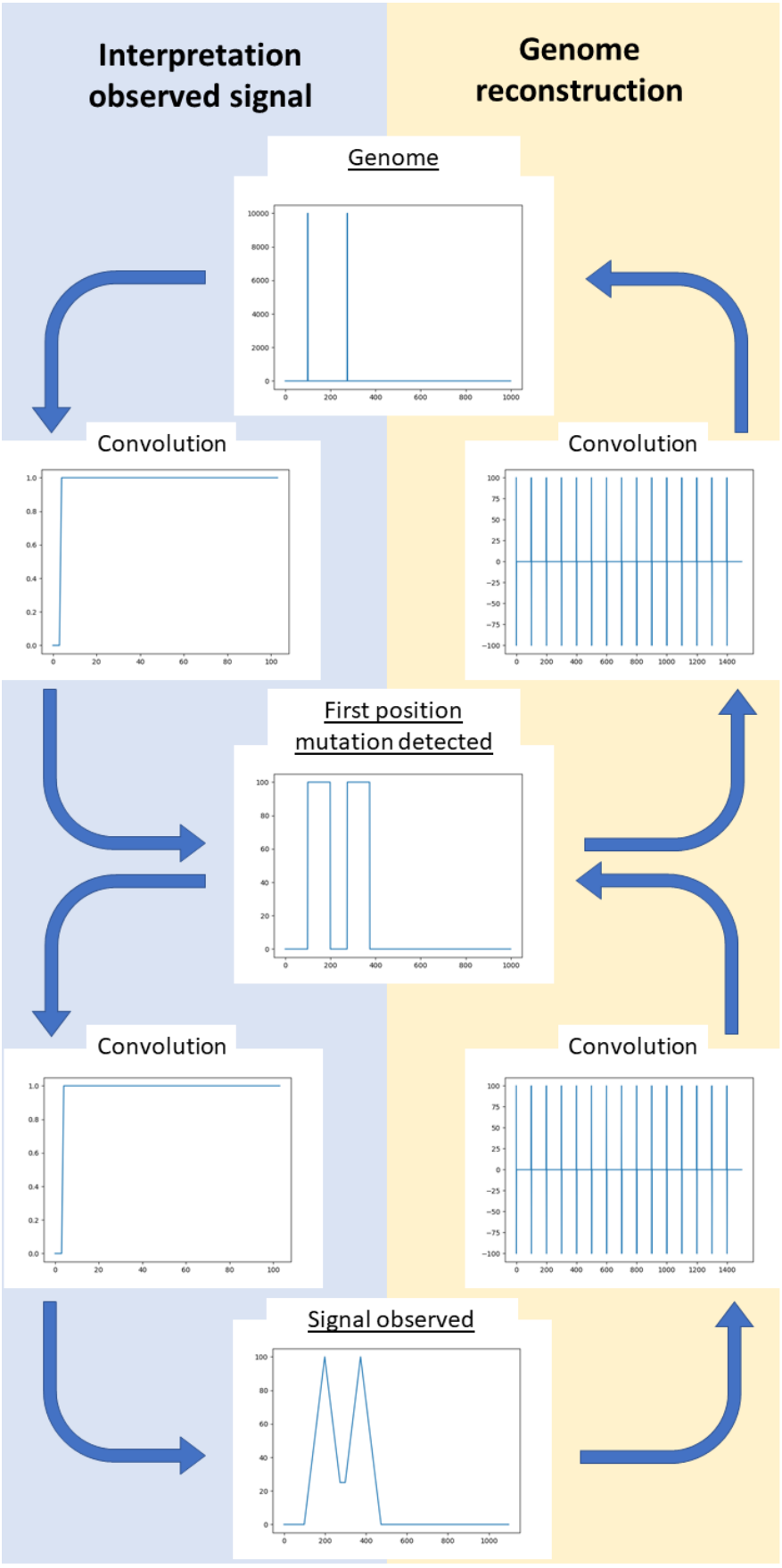
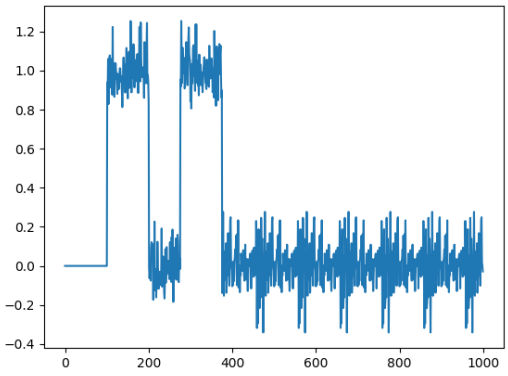
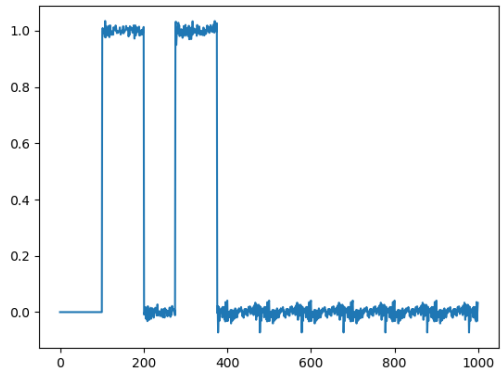


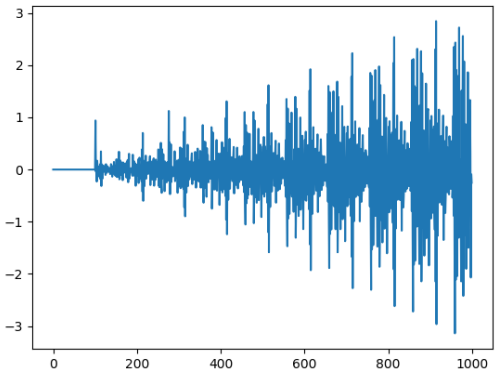
Figure 7.7: Summary of convolution interpretation of the attack



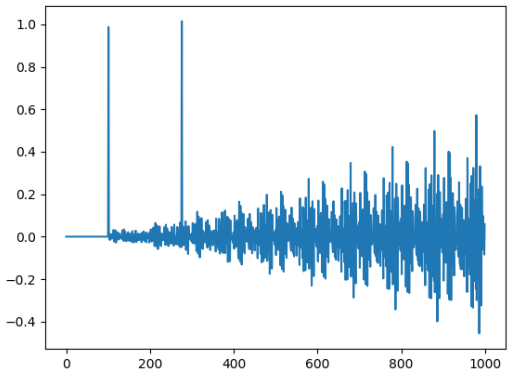
(a) Recovered first position mutation detected, coverage 10000



(b) Recovered first position mutation detected, coverage 50000



(c) Recovered genome, coverage 10000



(d) Recovered genome, coverage 50000

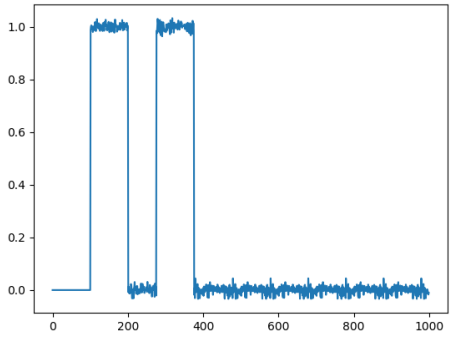
Figure 7.8: Examples of genome reconstruction using convolutions

the better the prediction is. We can further improve the prediction by reducing the noise by rounding to the closest possible value as shown in Figure 7.9. Using this approach, we test the performance of the attack across different coverage values. The results of the performance, represented with the f1-score, are provided in Figure 7.10. The results obtained are not good enough to present a significant risk for the privacy, even with unrealistically high values of coverage.

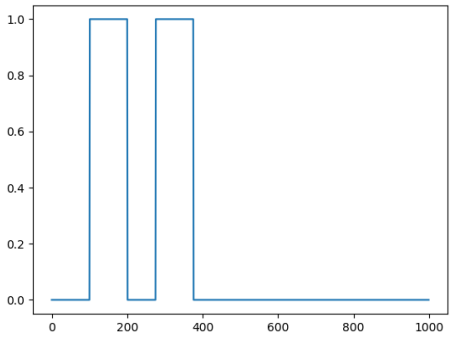
7.2.5.2.3 Neural network In Figures 7.4, and 7.6, we can see that through visual inspection the mutations in the genome can be inferred. The problem we try to solve is a classification problem: for each position, we use a signal as input to determine to which class the test data belongs.

We have constructed a neural network and trained it to retrieve, from a shorter experimental version of the $ratio_{signal}$, whether there is a mutation at the central position of the corresponding genome. For a given $read_{length}$, the neural network receives as input $read_{length} * 3 * 2 + 1$ floating point values bounded in the range $[0; 1]$: this correspond to the $read_{length} * 3$ ratio values prior to the point of interest, to the $read_{length} * 3$ ratio values after the point of interest, and to the ratio value at the point of interest. Furthermore, it also receives $read_{length} * 3 * 2 + 1$ floating point values bounded in the range $[0; 1]$ and corresponding to the likelihood of a mutation at that position. The output of the neural network is a value equal to either 0, 1 or 2. Value 0 indicates that the neural network considers that there is no mutation, 1 indicates a mutation on one of the copies of the chromosome and 2 indicates a mutation on both copies.

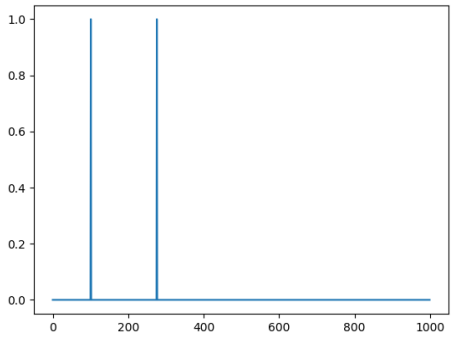
As previously defined, the $ratio_{signal}$ concerns the entire genome, and we want to obtain the entire $genome_{signal}$. Therefore, to accomplish the initial goal, we would need to apply the neural network to every position. Furthermore, Figure 7.6 shows that we can see that there are possible transient effects. In order to ensure that these effects are as limited as possible (and thus closer to what is possible using the entire $ratio_{signal}$) we use an input window size big enough to guarantee that no mutation outside the window could have an effect on the currently studied position.



(a) Prior to noise reduction



(b) Noise reduced



(c) Recovered genome

Figure 7.9: Proposed noise reduction

The information we use for training and testing the neural network is obtained by randomly generating material from real information. The process is the following: from the online database ensembl.org [9], a gene is randomly chosen, and the list of known mutations for the gene is obtained. For each position, we retain only the most frequent mutation. We also consider all mutations to be independent of one another.

For each generated case, we select randomly one position p with a known mutation and generate two fake chromosomes: the chromosomes have size $read_{length} + read_{length} * 3 * 2 + 1$ and are an array of 0's, except for those positions corresponding to a location on the reference genome with a known mutation, where we randomly put either 0 or 1, following the distribution indicated in the retrieved statistics. The mapping between the fake chromosome and the statistics is done in such a way that after dropping the first $read_{length}$ values, the central position of the signal corresponds to p. The data dropped ensures that we have the transient effect of data prior to the input window present in the input window.

The distribution of reads is then simulated: using as input a target *coverage* (i.e. an average number of reads mapped to a given position), we determine how many reads are to be generated. Then, for each of these reads, we select one of the two fake chromosomes, randomly select an initial position for the read and determine whether it belongs to class P or class M. The neural network is trained to classify whether the given position includes no mutation, one mutation in one of the chromosomes or a mutation in both. The neural network uses an output layer of three neurons, while there are five internal layers of sizes 256, 128, 64, 16, and 8 using the ReLu activation function (Rectified Linear Unit) [74].

We trained the neural network against different situations by varying the value of read length and coverage.

7.2.5.3 Results

As we have two chromosomes for each position (i.e. each entry in our testing material), there are three different possible outcomes: neither chromosome has a mutation, one has a mutation nor both have it. We take the arbitrary decision to refer with 'positive' to the fact that there is at least one mutation. However, when measuring accuracy we consider necessary to report the correct number of mutations for a given position (either 0, 1 or 2, depending on the mutation being present on neither chromosome, on one or on both). We report here the obtained results after training the neural network on 16 batches of 200000 cases

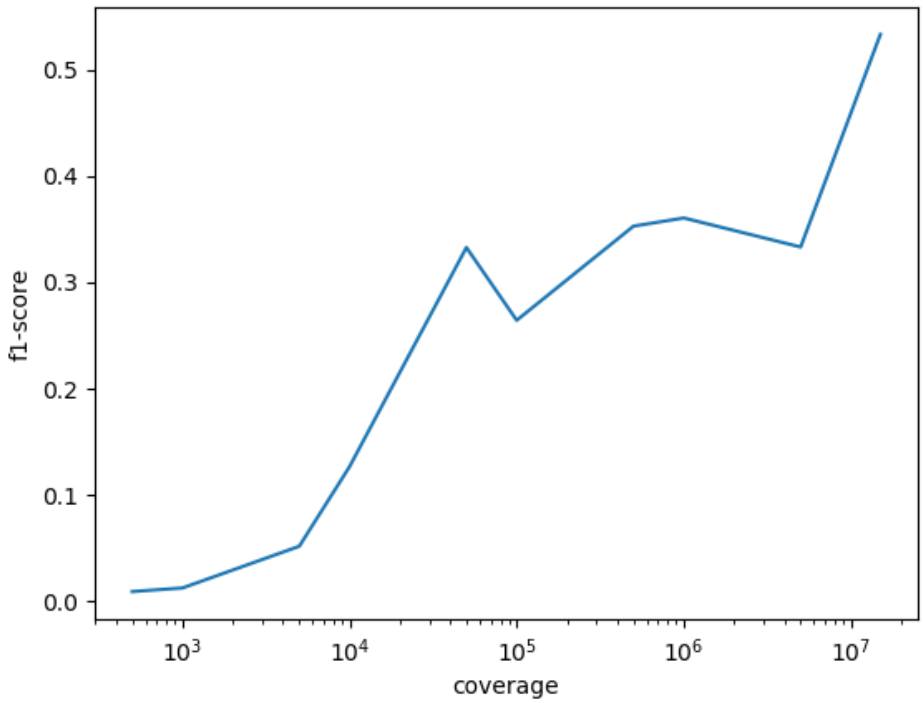


Figure 7.10: F1-score obtained by the convolution attack

for 240 iterations each. The reported numbers are obtained by testing the neural network on an additional batch of 50000 cases.

Figure 7.11 shows the fraction of accurate results obtained across the different read lengths and coverages. However, as the input statistics we use leads to a low probability of having mutations, we should not take into account the correctly predicted negative results, as this will be correct in approximatively 93.44% of the cases. Therefore we turn ourselves to the f1-score as shown in Figure 7.12.

7.2.5.4 Discussion

It is worth noting that the smaller the read length, the fewer doubts there are about the position of a mutation. For example, in the case of a read having a length of two, there are only two possible positions which can be mutated. Results need to be compared with our longer simulated reads of 100, where the evidence applies to 100 different positions. We can expect to obtain better results with smaller reads, and indeed it is the case.

Similarly, if the coverage is higher, i.e. the average number of reads mapping to a position is higher, we can expect to have more evidence in the form of more reads. For example, in the case of testing a region with only one mutation which is not at the point of interest but where both positions (the one of interest and the one with the mutation) could be mapped with one unique read, it is unclear for the classifier if the detected mutation originated from a mutation at the position of interest or another one. A higher coverage will translate in more evidence, which will increase the likelihood of having evidence for the absence of a mutation at the position of interest, thus simplifying the decision. We expect the results to improve with higher coverage, and this is a trend we observe in the results.

Nevertheless, we do not deduce every result correctly. As previously mentioned, some of these errors can be linked to the problem of two mutations possibly shadowing a middle mutation. In this kind of situation, the network can only do a random guess. Figure 7.13 shows an example of this situation, where the neural network wrongly predicts a mutation (the circle indicates the input genome, the cross the classification output for the position of interest, and the plotted line the ratio of mutated reads for the given position). As there are more negative results than positive results, the neural network is likelier to return a negative result, as shown in Table 7.1.

Furthermore, this type of input where the position of interest is windowed, appears only in 0.67% of the cases with read length equal to 100, and even less for data with shorter reads. Therefore this cannot be the main source of false categorizations.

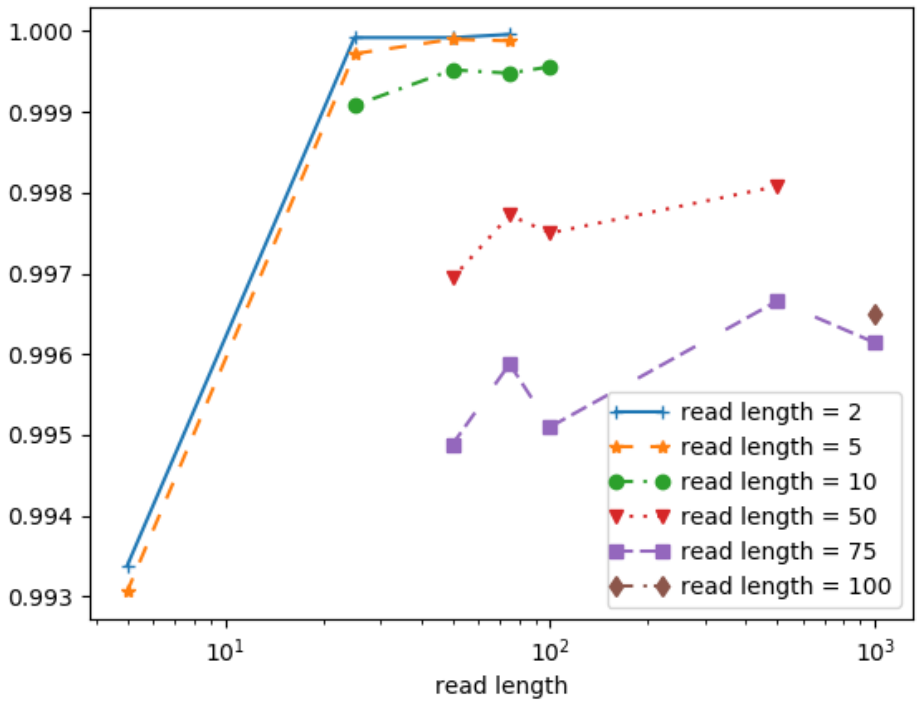


Figure 7.11: Accuracy results obtained

Table 7.1: Results for read length = 100, when the position of interest was within the window formed by two mutations

	Prediction is correct	Prediction is incorrect	Total
Predicted no mutation	90.20%	2.61%	92.81%
Predicted mutation	1.63%	5.56%	7.19%
Total	91.83%	8.17%	

If we observe the results which were misclassified, we can see that many of them share the same characteristics of having a mutation close by, as shown in Figure 7.14. If we take all misclassified outputs and observe the distribution of the distance between the closest mutation and the point of interest, we notice that the majority of incorrect results can be linked to close mutations: in more than 90% of the cases the closest mutation was 4 positions away, as can be observed in the distance histogram presented in Figure 7.15.

Although we are not recovering all the data, we believe that these results are proving that, in the studied settings, the privacy of the information is at risk. On the one hand, we have obtained high F1-scores, and, on the other hand, we have proven that, if we wrongly classify an input, it is likely that there is indeed a mutation in the neighboring positions. A read length value of 100 would be what is generated with an Illumina sequencer [12].

The recommended coverage can vary depending on the objectives. However, it appears that the recommendations [12, 75, 76, 77] for a broader set of tests are between 33 and 100, as summarized in [78]. At these recommended coverage levels, we consider the attack to recover too much of the supposedly protected information to consider the initial settings as privacy preserving.

7.2.5.5 Future work

In order to improve on this method, other neural networks could be employed. For example, and if the memory requirements can be met, a neural network predicting the classification at multiple locations at a time and using a prior round of predictions to refine the results of the next one could improve the obtained results.

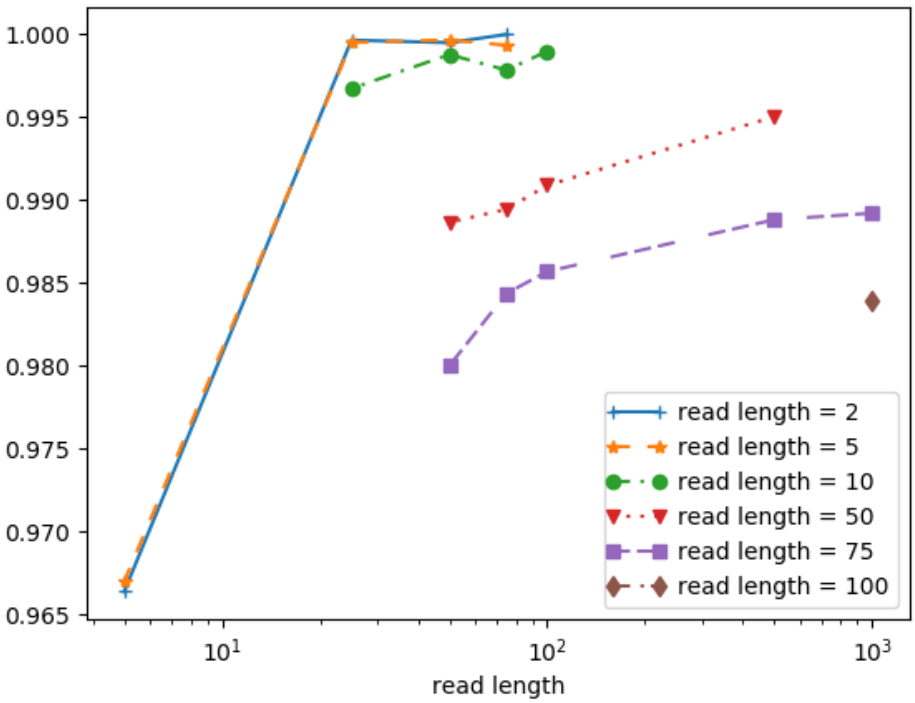


Figure 7.12: F1-score obtained

In the exercise presented here we have not considered linkage disequilibrium: this metric measures the likelihood that two mutations appear together. If this information could be represented as input to the neural network alongside the classification of a linked mutation, the point of interest could be better predicted.

Finally, this work makes the assumption that the read length is constant and known. However, the read length might be variable. In order to take this into account, the Stream giving the read length should either be available as plaintext or a maximum value for the read length should be assumed. Furthermore, we have not taken into account the possibility of paired reads: in this configuration two reads are paired together in the most general class necessary: if the two reads would have belong to class P, then the pair belongs to class P, if the two belong to class M then the pair belongs to class M, and if one of the reads belongs to class P and the other to class M, then the pair belongs to class M. This could be modelled in the neural network presented here by considering the two reads as one very long read, but as we have seen the length of the read reduces the amount of information recovered. Alternatively, both reads could be treated separately, but in this case one read might be mislabeled as providing evidence for mutations, increasing the noise present in the input.

7.2.5.6 Conclusion

In the initial approach of the privacy protection of MPEG-G, it was possible to encrypt only some of the blocks of information for a given Access Unit. We have here shown that taking advantage of this option, one could create a file where the mutation types and mutation positions are hidden, and consider the data to be private. We have shown, however, that this information can be partly recovered. The magnitude of the recovery depends on the read length and the coverage.

In order to defeat the here presented attack, the easiest path is to encrypt also the first position of each read. However, other attacks might be possible: other non-encrypted information might be used to recover other encrypted information. This fact led the privacy approach of MPEG-G to be redefined in order to impose all blocks in an Access Unit to be encrypted.

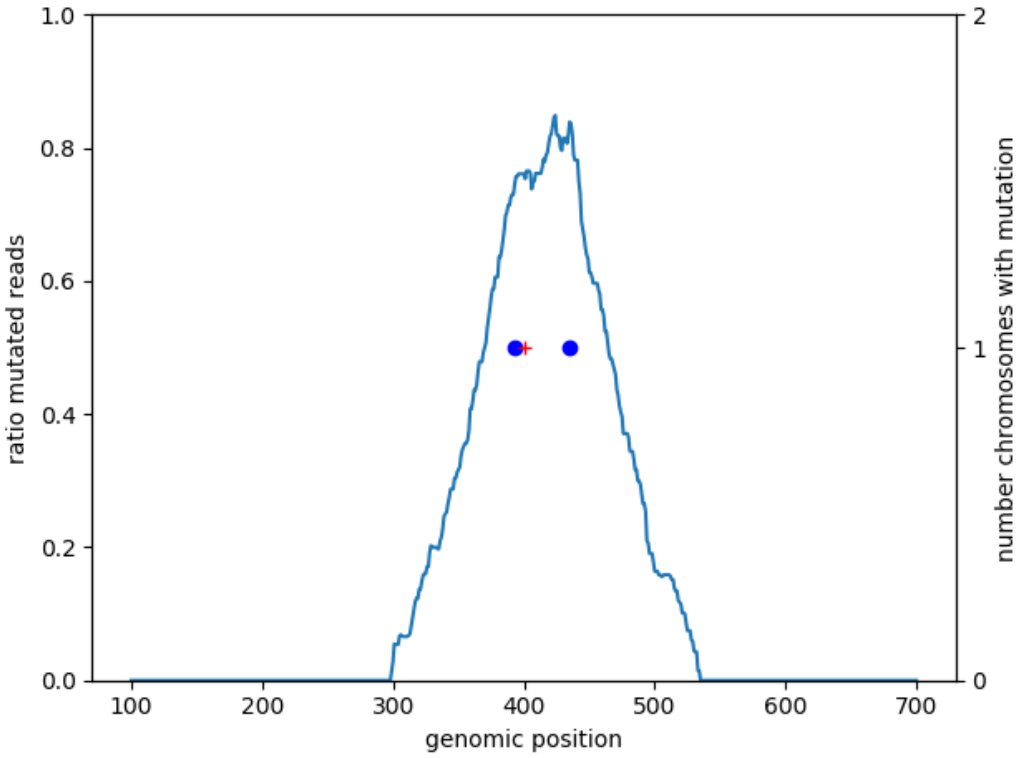


Figure 7.13: Example of a wrongly predicted mutation within the window formed by two mutations.

7.3 Possible wrongdoings: Fingerprinting

7.3.1 Introduction

MPEG-G defines multiple mechanisms to protect the security of the conveyed information: both for the genomic data and the metadata, the specification provides privacy and integrity for the information. We will also see how a tool can determine which actions a user is allowed to execute. These mechanisms ensure that only those recipients who are authorized to receive the information stored in the MPEG-G file are able to decode it.

These mechanisms do not protect against wrongdoings posterior to the transmission and decoding of the file. A party could receive a copy of the file and retransmit its content to new, unauthorized recipients. This situation could happen if the party is an adversary, or also if due diligence is not respected: for example, the MPEG-G transmitted file might be converted to BAM, thus losing all security mechanisms, and be recovered by an external actor who has gained access to the BAM file.

Once a party has leaked the information, no mechanism will be able to remove all unauthorized copies. In order to prevent the leak in the first place, it can be useful to integrate into the file a mechanism to identify the origin of the data.

We here present a possible algorithm intended to modify the genomic data in a controlled way. The objective is to send each recipient a slightly modified version of the file. In the case where a leak is detected, the file owner can conduct an audit. The audit consists in detecting which unique version has been leaked, thus leading to the recipient responsible for it. The goal is that the threat of the audit will force the users into compliance.

As this algorithm alters the genomic information, we have not proposed its inclusion to the MPEG-G specification. The result of the algorithm is aligned genomic information. Thus, any encoder for aligned data, including an MPEG-G encoder, can be used to represent the output of the algorithm.

7.3.2 Fingerprinting of reads with mutations

7.3.2.1 Introduction

The aim of the presented method is to be as less intrusive as possible. We focus on watermarking reads with some mutations, considering them more likely candidates for an exportation of data. The reads with

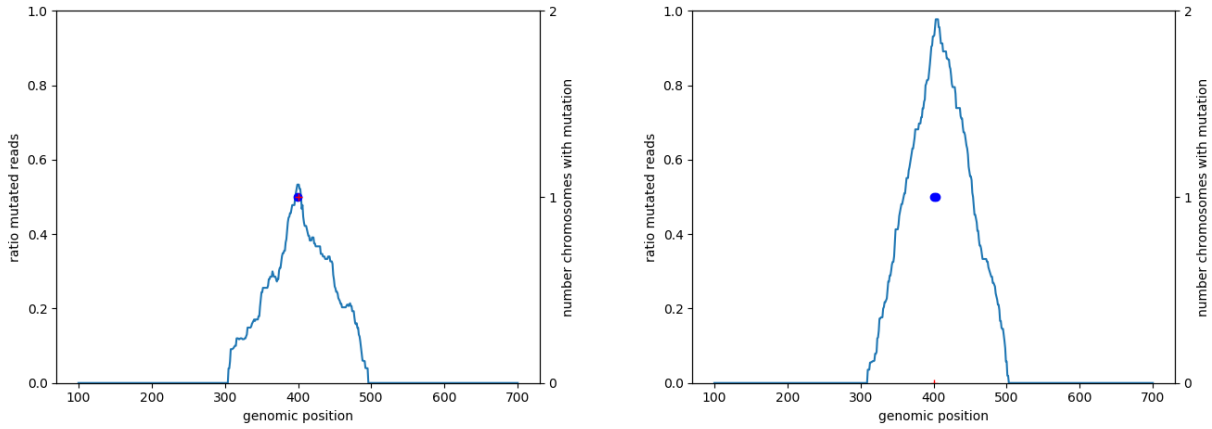


Figure 7.14: Example of two misclassifications probably caused by the close proximity of a mutation

Table 7.2: Example of description construction

Read CIGAR	5M	2I	3M	4D	1M
Aggregated position	0	5	7	10	14
Information stored in description		5-I-2		10-D-4	

mutations are independent subsequences of DNA read by the sequencing machine where the alignment has detected differences between the read and the reference genome, as introduced in chapter 2. Each read is treated separately, so importing other information will not make the watermark disappear.

On the other hand, the proposed method results are hard to detect if the reads are modified. This, however, is limited by the cost of the modification: introducing or removing mutations could alter possible conclusions drawn from the file, thus rendering this approach as a non-suitable solution to defeat the fingerprinting. An important feature from our approach is the fact that the user holding the watermarking key can reverse the modifications.

The proposed method consists of four steps for each read:

- generate a description for it
- verify that it is a suitable candidate for the fingerprint method
- decide whether we can perform the modifications based on a secure transformation over the description
- perform the modifications in case of a positive result.

The four steps are described in detail in the following four subsections.

7.3.2.2 Description

The assumption on which this method is built is that the valuable information within a genomic file are the positions of the modified nucleotides (i.e. the positions of an insertion, deletion or skipped nucleotide base). Therefore, we assume that no such operation will be added or removed, and that we can use them to construct the description.

To do so, we propose to use a 256 bits long description. For every modification (insertion I, deletion D, skipped N and mutation X) and in the order of the read, we append to our description the position of the modification, for example as an 8 bits unsigned integer (which is enough to store the position in the case of a file with a maximum read length of 150 base pairs), followed by another byte encoding the modification operation (first four bits) and the number of nucleotides to which the operation applies (last four bits). The remaining bits are left with a 0 value. An example is shown in Table 7.2, where we separate with hyphens each field used for the description.

7.3.2.3 Suitability of read

As we have seen, the method works on the information contained in the CIGAR sequence of each read (i.e. the list of operations in respect to the reference such as insertion or deletion).

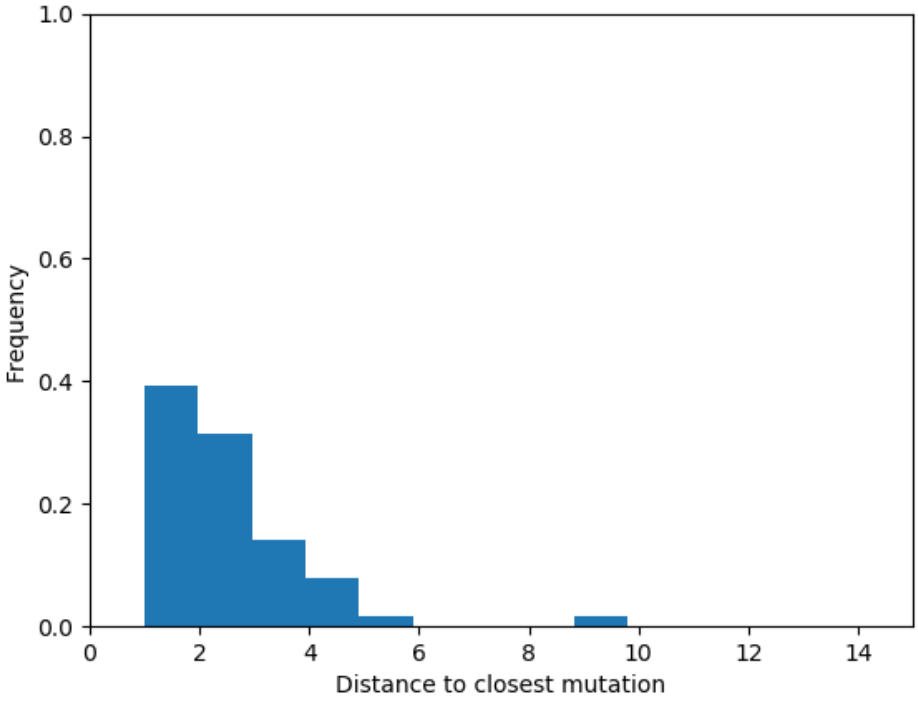


Figure 7.15: Histogram of distance between the point of interest and the closest mutation, when the mutation was wrongly predicted and the read length is 100

Under the conditions introduced in the previous subsection, we can neither overflow the 256 bits of description (limiting to 16 operations) nor the four bits operation length (limiting to 16 the maximum length of each operation). We consider suitable those reads for which we can construct the description. Non-suitable reads are disregarded: by changing the size of the description and its fields the number of reads taken into account varies.

7.3.2.4 Secure transformation

In order for this algorithm to be secure, we need to obtain a secret from the previously constructed description. This secret will define how to apply the fingerprint. Furthermore, in order to be able to undo the modification, the result of this secret must always be the same given the description. With secure transformation we refer to such an operation which, given a description, generates deterministically a secret.

Both during the watermark auditing and the watermark removal we want to obtain the same result from the secure transformation. We do not consider the numbering of the read to be a suitable input to this end since this input is lost if, for example, the user generates at some point a new file containing only a subset of the reads (perhaps all the reads aligned to only one reference sequence). As we would require an initialization vector per read (in order to maintain the independence of reads), such a transmission would not be reasonable. Therefore, the only input to the transformation will be the previously constructed description.

One candidate for the transformation is the AES cipher in electronic codebook mode. In this mode, the cipher is stateless and takes only the plaintext block and the key as input: the consequence is that the same input always returns the same output. There are flaws in this secure transformation which has repercussions on the fingerprinting method. This is discussed in clause 7.3.3.

7.3.2.5 Perform the modifications

The result of the fingerprinting will be a modification of the soft clipping at the beginning of the read (see clause 2 for details). As soft clipping is not considered in the description's formulation, the process will not affect the description. The method can be readily extended to support also modifications at the end of the read.

As explained later, we interpret the result of the secure transformation as the conditions to be met and the output of the fingerprinting operation. We want the obtained secret to convey the necessary information

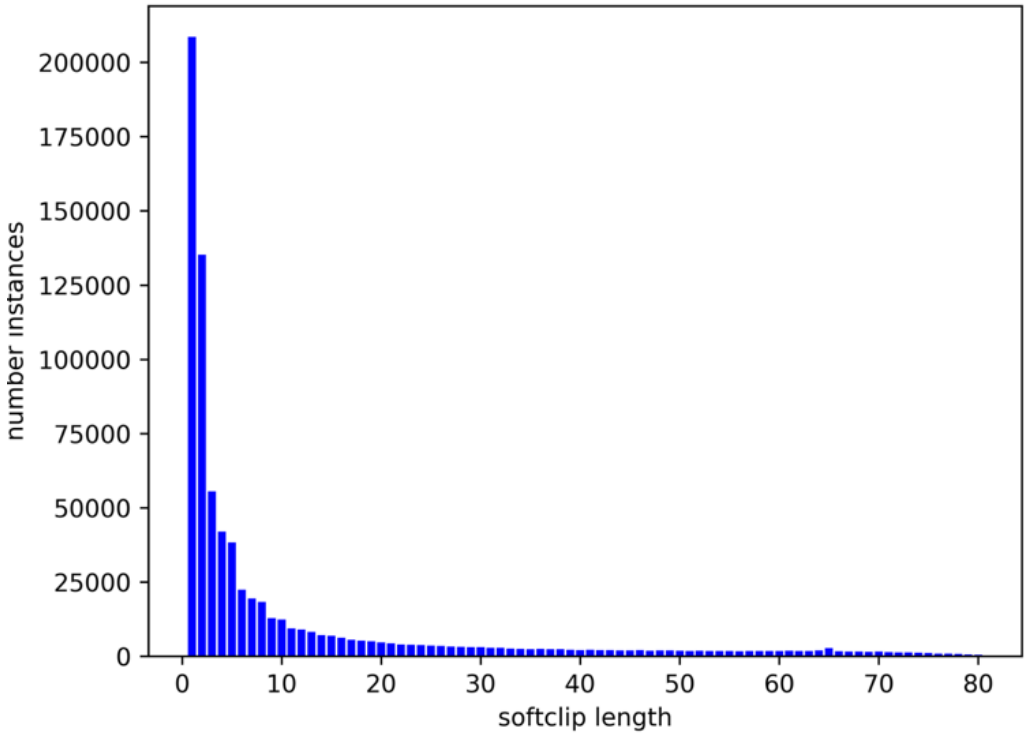


Figure 7.16: Histogram of softclip lengths

to perform and reverse the fingerprinting process. The modification process at the beginning or the end of the read is the same, the only difference is that the effect is mirrored. The basic idea is that the secure transformation result will encode the state before and after fingerprinting, the state being the length of the soft clip operation and its content.

We define a number of bits encoding the length of the soft clip operation. In this case, it is preferable not to use a straightforward encoding as this would lead to sampling uniformly the pool of possible lengths. On the contrary, if we plot the histogram of soft clip lengths (Figure 7.16), we observe that short lengths are far more likely. Therefore, it is preferable to construct the decoding of the length of the soft clip operation using the observed Cumulative Distribution Function.

With this mechanism, we can now first read from the result of the secure transformation a guess for the current soft clip operation size ($\text{guess}_{\text{size}}$), and then the length of the one into which we should transform it ($\text{transformation}_{\text{size}}$). This distribution function is likely to return the value zero for the two random variables, thus reducing the likelihood of executing a transformation. Additionally, we also decode two nucleotide sequences: $\text{guess}_{\text{string}}$ and $\text{transformation}_{\text{string}}$. For the interpretation of the string we use only the four base pairs found in nature ('A','C','T','G'). As such, each byte of the secure transformation encodes up to four base pairs. In Figure 7.17 we show how the bytes are interpreted.

The operation is performed if $\text{guess}_{\text{size}}$ and $\text{guess}_{\text{string}}$ match the observed operation. This ensures that if we perform the reverse operation we will have the original soft clip content encoded in the output of the secure transformation. One exception to this is when the guess is not equal to the observed operation, but the proposed result of the transformation is. In other words, $\text{transformation}_{\text{size}}$ and $\text{transformation}_{\text{string}}$ match the observed soft clip. In this case, if we do not apply any modification, during the reverse operation we would be wrongfully induced to believe that the watermark operation was performed. In order to avoid this, we swap the values of guess and transformation. In doing so, we signal to the un-fingerprinting routine this special case and we are able to correctly reverse the modifications (see Algorithm 1 for an illustration of this).

The operation is only applied when both $\text{guess}_{\text{size}}$ and $\text{transformation}_{\text{size}}$ are smaller than the position of the first non-softclip and non-equal operation. This is to ensure that all the information required to undo the operation is available in the reference.

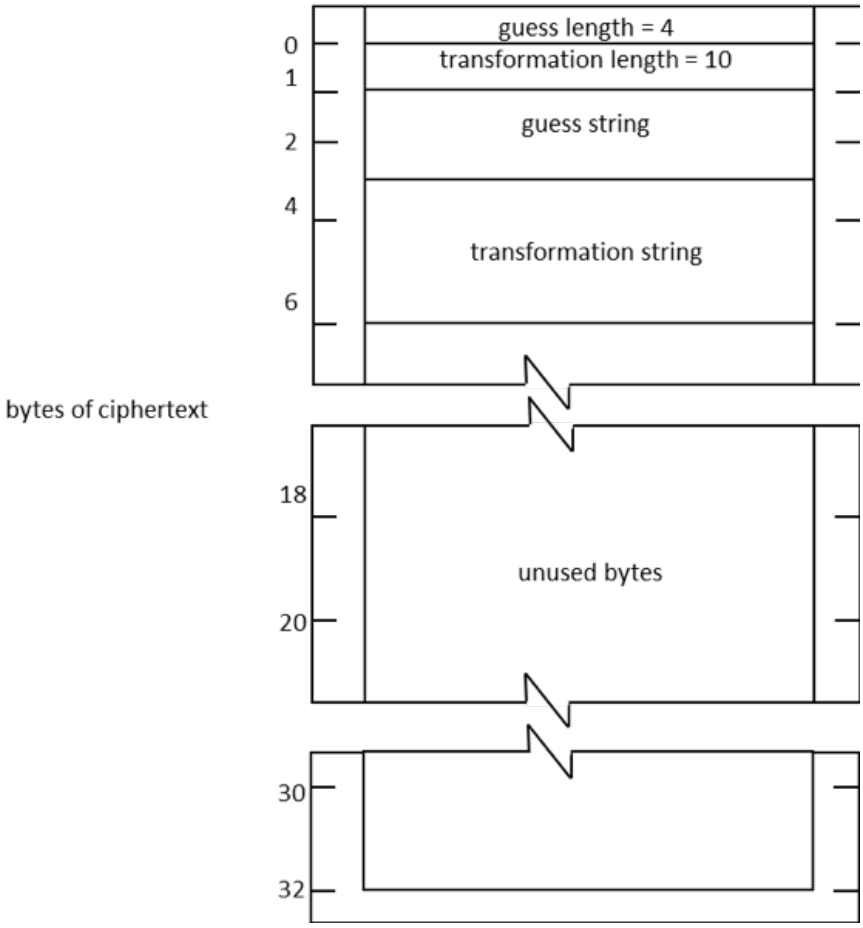


Figure 7.17: Interpretation of the secure transformation output

Data: read, secure_transformation_output
Result: Watermarked read
$real\_soft\_clip_{length}$ = getSoftClipLength(read);
$real\_soft\_clip_{string}$ = getSoftClipString(read);
$guess_{size}$ = readSoftClipLength(secure_transformation_output[0]);
$transformation_{size}$ = readSoftClipLength(secure_transformation_output[1]);
$guess_{string}$ = readSoftClipString(secure_transformation_output[2]);
$transformation_{string}$ = readSoftClipString(secure_transformation_output[3]);
if $real\_soft\_clip == guess$ **then**
  fingerprint(read, from = $guess$, to = $transformation$);
else
  if $real\_soft\_clip == transformation$ **then**
    fingerprint(read, from = $transformation$, to =$guess$);
  end
end

Algorithm 1: Pseudocode for reading watermarking function parameters

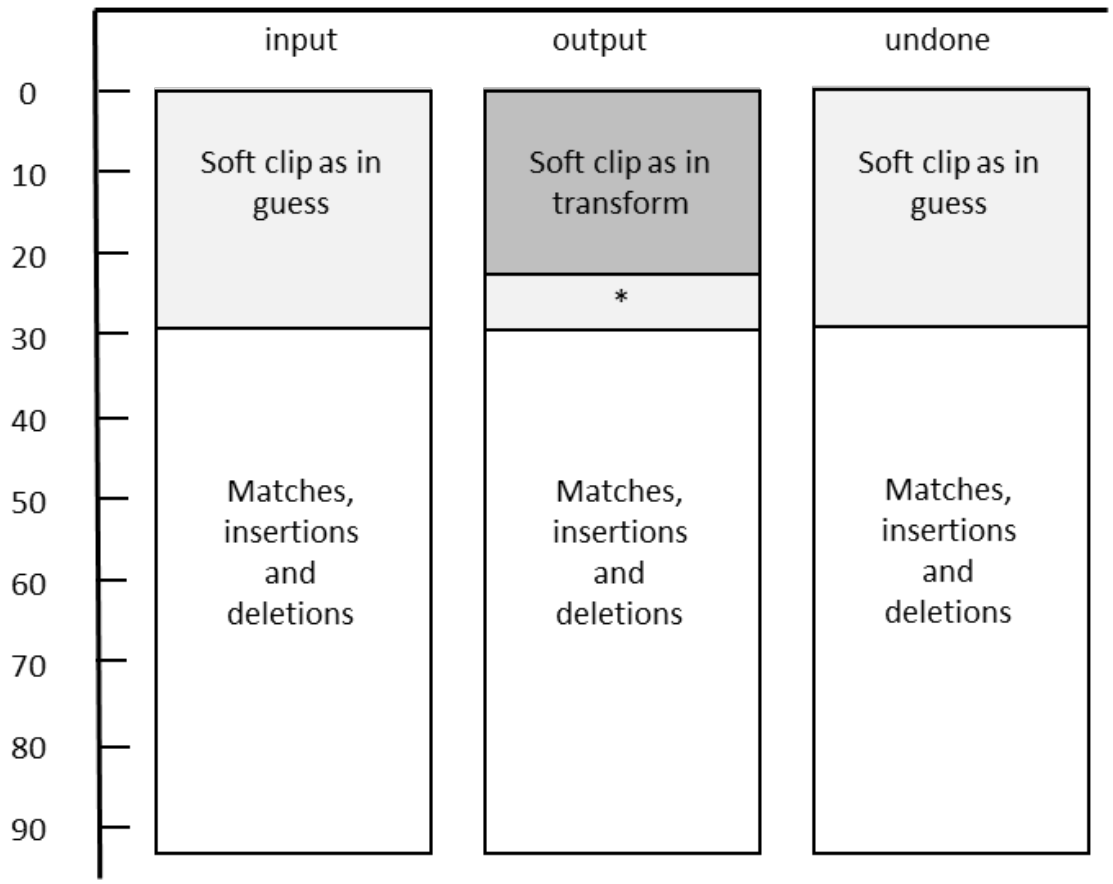


Figure 7.18: Soft clip contraction operation

7.3.2.6 Perform the modifications

In a nutshell, the fingerprinting operation will replace the initial guess content with the transformation input. This gives us new criteria for the suitability of the read to undergo a fingerprinting operation: there cannot be a base pair involved in the description in the first $\max\left(\text{guess}_{\text{length}}, \text{transformation}_{\text{length}}\right)$ base pairs of the read. This ensures that we will only replace soft clip operations (for which the content is encoded in transformation) or match operations (in which case the content is straightforwardly present in the reference genome).

See Figure 7.18 for a visual representation of the case where $\text{transformation}_{\text{length}} < \text{guess}_{\text{length}}$. In the inverse case, we would copy the reference genome in the undoing operation. Figure 7.19 shows how a read, its soft-clipping and its corresponding CIGAR string are modified after applying the transformation operation.

7.3.2.7 Undoing fingerprints

The process to undo the modifications is the same as for modifying. If the read starts with transformation, we modify back to guess. In the case where it starts with guess, we change the beginning with transformation.

7.3.3 Results and discussion

7.3.3.1 Basic tests

The fingerprinting method needs a clear mapping between each of the file's recipients and watermarking keys. In case of auditing leaked data, we would test for each key whether the leaked data matches the modified result. In case of granting access to the unmodified version, the recipient only the fingerprinting key to reverse the modification process. If one of the keys is leaked, the corresponding file could be reversed, but the other files do not lose their watermarking. In fact, using an incorrect key to remove the watermark from a file would add a new watermark to it.

In order to test the algorithm, we use a file containing the sequencing of a human genome, generated with an Illumina device [79]. The file has a low coverage (2.3). Coverage refers to the average number of

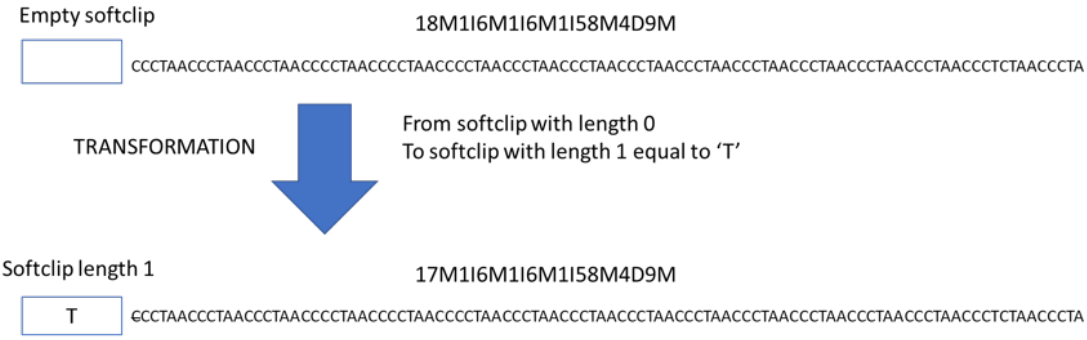


Figure 7.19: Example of application of transformation operation to a read and its associated CIGAR String

identifications for a single nucleotide, as described in [80]. The file is part of the database of test material [81] used in the standardization of MPEG-G.

The file contains $5.5 \cdot 10^7$ reads, from which $3.8 \cdot 10^7$ are perfect matches and are therefore discarded. From the remaining reads, the description could be built for $6.2 \cdot 10^5$.

We discard those cases where the change in soft clip length would be too big (arbitrarily defined at 10): this discards $2.8 \cdot 10^5$ reads. The algorithm can only be applied if both $\text{guess}_{\text{length}}$ and $\text{transform}_{\text{length}}$ are less than the position of the first operation which is not a soft clip or a match. This is the case in $3.4 \cdot 10^5$ reads, from which for $3.1 \cdot 10^5$ either guess or transform match the beginning. However, as it is highly likely to read a null length from the secure transformation, in $2.0 \cdot 10^5$ instances, the ciphertext does not represent a real transformation. In the end, $1.2 \cdot 10^5$ reads are transformed.

This number has to be compared to the $6.2 \cdot 10^5$ reads which could have been possibly modified. Someone trying to circumvent the mark would need to work on these $6.2 \cdot 10^5$ in order to reverse the modifications purposely introduced in 18.6% of them.

7.3.3.2 Resistance to collusion attacks

7.3.3.2.1 Vulnerability Let us assume a collusion attack where all parties suppose that the most frequent version of a read has not been modified. Such situation is more likely if a watermarked read is modified differently for multiple recipients. In order to test such a risk, we reduce the previous test data to only those reads mapped to the first chromosome. We then simulate sharing the file with multiple parties: we do this by executing the fingerprint method with different keys. For each read several outcomes are possible: the read could not be fingerprinted with any of the keys, it could be fingerprinted with some of the keys, or with all of the keys. As the probability of a read being modified is fairly low, the most likely outcome is that if across multiple watermarked instances of the same file there are different variants of the same read, then the most common variant is the original, non-watermarked one.

In this test, we reduced the number of reads to $4.1 \cdot 10^6$, and we used seven different keys for watermarking. On average, for each key there were around $1.8 \cdot 10^4$ reads watermarked, for a total of $6.62 \cdot 10^4$ reads being watermarked at least once across all instances. For a total of $6.59 \cdot 10^4$ reads, the most frequent variant was the original, non-watermarked one.

Figure 7.20 shows how the risk of collusion increases as the number of watermarked files rises: as the number of watermarked files increases, the ratio of cases where the non-watermarked variant is the most frequent one augments, thus simplifying a collusion attack which only selects the most frequent variant. The here presented version of the algorithm is vulnerable to such attacks. However, some of the concepts from [56] could be applied to the algorithm. Although we cannot select each read to be watermarked independently, due to the way the reads are selected, the algorithm could be executed multiple times, once for each of the provided keys. As soon as a read is watermarked with one of the keys, the iterations stop.

As for each read, the likelihood of being modified is low, we can think of each key as modifying a distinct subset of reads, without any overlapping with any other set. In these conditions, we want a configuration such that, for each recipient, we have selected a unique set of keys, where none of the sets is empty (as it would imply that the recipient would receive a non-modified version), and that each key is present in a majority of sets (thus we ensure that the modified version is the most frequent one, even if all recipients collude). This selection of configuration also needs to minimize the number of keys used in each set, as each key implies modifications to the file, thus decreasing its utility and possibly harming conclusions. If the number of recipients is known beforehand, the task is trivial, and the collusion strategy of selecting the most frequent version will always fail.

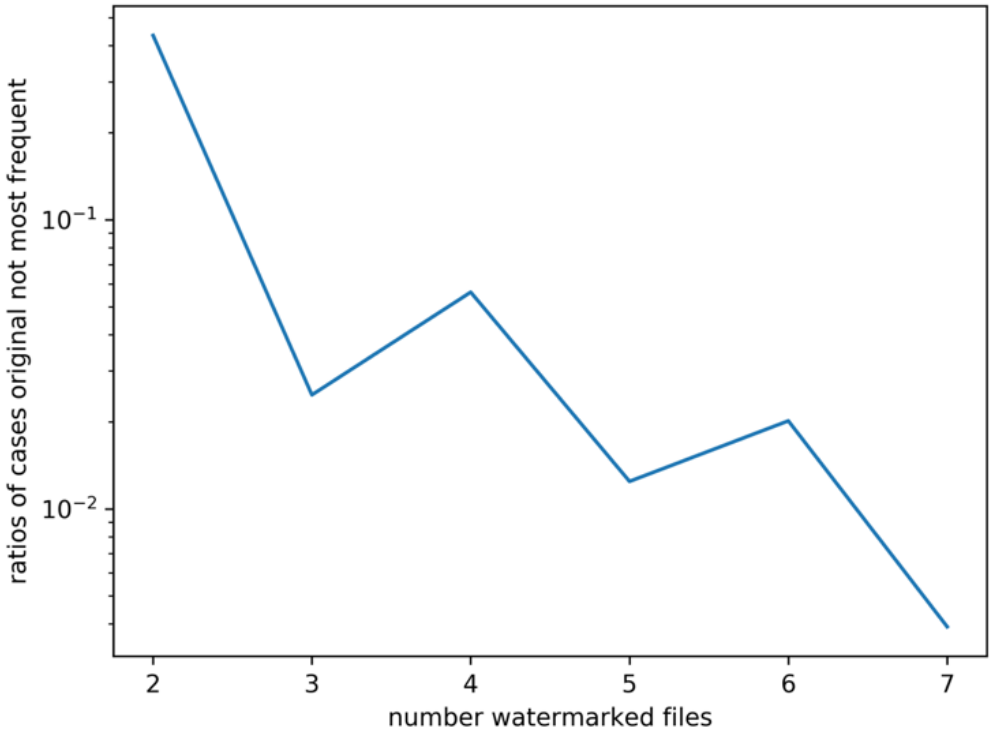


Figure 7.20: Evolution of the ratio of cases where the original non-watermarked variant is not the most frequent across all watermarked files

7.3.3.2.2 An improvement using an ILP In the case where the number of recipients is unknown, some concessions have to be made. Let us see how the set of keys for a new user is decided upon after receiving the new request. We only know which keys were used for the previous recipients.

Similar to [56], we construct an Integer Linear Program (ILP), which aims at minimizing the cost associated with the decision. We construct the objective function as a lexicographic search: first we minimize the risk of leakage, then we minimize the sum of the sizes of sets, and finally the size of the biggest set. As previously explained, each time a key is used at least once, but is not present in a majority of set, the reads associated to that key will be recoverable. The ILP decides the set for the new recipient, which has to be different from all sets previously employed.

Table 7.3 shows the result of this iterative process: in order to obtain a column, all previous columns are provided as input. We can observe how the solver attempts to maintain the number of keys used as low as possible, but as soon as all combinations are used, a new key appears in the pool of used keys. This key is then a potential source of leaking, as it appears in a minority of sets, but in subsequent calls the key is always selected and eventually reaches a point where it is not a source anymore. However, this happens when all combinations are used, therefore the problem arises again at the next iteration. The process can be observed for recipient 2, 4, 8, and 16.

We generate the fingerprints for each of the recipients decided by the ILP; the corresponding results are summarized in Figure 7.21. We can see that the average number of modified reads increases as the number of recipients increases. Furthermore, and as previously explained, the introduction of a new key to the pool creates situation of leaking, which are corrected as soon as the file has been shared with a sufficient number of recipients.

There is a trade-off between the use of multiple keys to avoid collusion attacks and the computation needed to generate such files with multiple keys: each key increases the time required to generate such file as depicted in Figure 7.22.

7.3.3.3 Generalization of the description

Our next step is to slightly modify the behavior of the algorithm in order to improve it. As a result a greater variety of cases is accepted. We use a file with variable read length, some of which have a length greater

Table 7.3: ILP iterative decisions

	Recipient 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Key 1	Y		Y		Y	Y			Y	Y			Y		Y	
Key 7		Y	Y		Y		Y		Y		Y	Y				
Key 3				Y	Y	Y	Y		Y		Y		Y	Y		
Key 2								Y	Y	Y	Y	Y	Y	Y	Y	
Key 5																Y
Key 4																
Key 6																

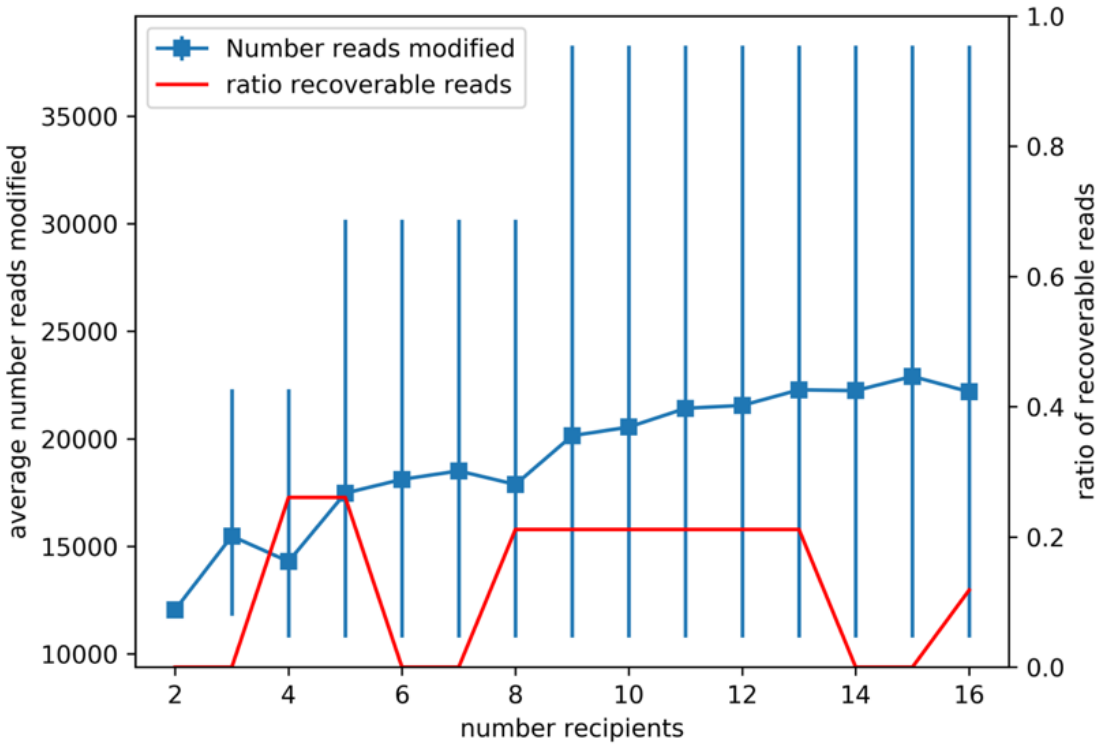


Figure 7.21: Number of reads modified and leakage ratio versus number of recipients

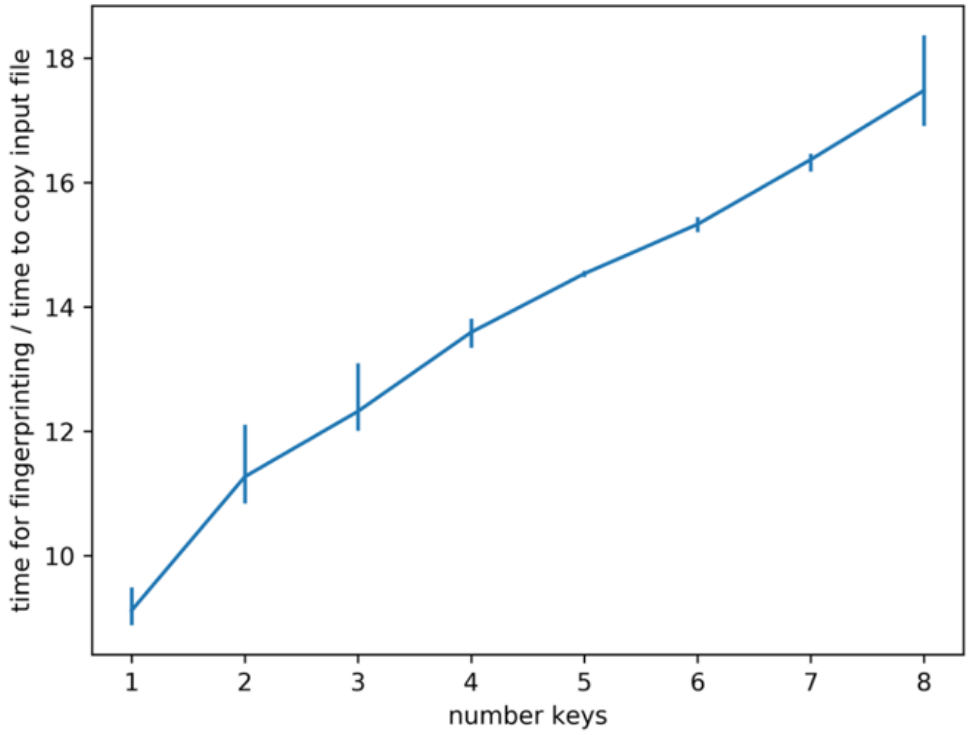


Figure 7.22: Time required to fingerprint a file compared to copying it

than 255 nucleotides (average depth of 8.1 reads per base). Thus, the byte used to store the position of the mutation is overflowed: we convert its representation to a two bytes system. The maximal length of the description is kept at 32 bytes; therefore, we are not able to store as many mutations anymore. The observed likelihood of a read being marked, provided that it had at least one mutation and the description could be constructed, is 25.5%.

Finally, we test a file with $1.7 \cdot 10^8$ reads [82] for an average depth of 7.74. In this file, there are $1.5 \cdot 10^8$ candidates. For $6 \cdot 10^7$ of them, a description could be built, and in 25.3% the constructed description resulted in a modification of the read.

The proposed watermarking method has the positive side to be reversible and to limit the alteration of the file.

The negative aspect is that an attacker can render it useless by modifying the reads. For example, by shifting all reads one position to the left, or inserting new modifications, the process is broken. Furthermore, by modifying the length of every soft clip operation the result of the watermarking is removed. On the other hand, this same approach decreases greatly the value of the published information, as it modifies the data blindly, thus introducing more noise to the data. In order to have successfully defeated the security method, the attacker would have to publish a version of the file as close as possible to the original non-fingerprinted file, without the modifications introduced to identify that instance of the data.

An attacker could also try to reconstruct partially the original file. As the secure transformation will always give the same output for the same description, the attacker knows that for all reads whose descriptions are the same, some will share the same soft clip operation due to the watermarking. If there are enough instances, it would be possible to infer the values of guess and transformation allowing to reverse the process for those reads.

7.3.4 Conclusions

We have presented a method of modifying a file of aligned genomic information in order to introduce recognizable changes, which are, however, hard to identify as such without the required parameters. These changes can be undone in case the original parameters of the method are known.

The proposed method is designed with some assumptions in mind: mainly that the reads to be marked by the fingerprint operation are those carrying information about mutations, and that a file recipient has

no incentive to modify blindly and broadly the content of the file showing mutations, as this decreases the value of the information.

The inner workings of the fingerprint operation can be viewed as separate entities. First, special features of each data entry are combined, this combination is then used as input of a cryptographic function, and lastly the ciphertext is interpreted as the modifications to be applied to the data entry. By modifying each one of these three steps, the proposed method could be adapted to other needs or another set of assumptions. For example, if modifying the soft clip operations is considered too harmful to the value of the file, a new interpretation of the ciphertext could be devised to modify the quality values instead.

The algorithms that come closest to what we propose are watermarking strategies as used in the audio-visual world, e.g. in the case of images (still and video) and audio. Alongside the familiar strategy of clearly modifying the values of certain regions of an image, other approaches have been proposed as surveyed in [83]. As in the case of image watermarking, we have to be concerned with the possible transformations applied to the file. However, despite the similarity in the objectives (either mark the ownership of the intellectual property or identify a specific copy), the data types are quite different. For example, in the case of genomic information we have multiple reads storing multiple copies of what should be the same information, with no clear equivalent in the case of an image file. Similarly, a modification not visible to the human eye or perceptible to the ear is acceptable in the audio-visual world; however, in the case of genomic information it might be wrongly interpreted as a mutation.

Furthermore, another path which could be explored in order to create a fingerprinting method for genomic information is exploiting the less significant bits for the qualities. Each read's base pair comes with a quality score, a representation of the sequencer device's confidence when identifying that base pair. Using the less significant bits could however fail if a new format used lossy compression for the quality scores [60].

Some requirements used in this version of the algorithm could be lifted in case the reversing properties were not to be used. This could allow a new version of the proposed method where more reads are changed.

Chapter 8

Application Programming Interface

Initial requirements formulated by MPEG included privacy rules. Early on, a need for an Application Programming Interface (API) was also identified.

In this chapter, we present our API approach, which was not adopted, but was publicly presented at [84]. We also present our proposal for privacy rules. It relies on XACML to parameterize who is under which circumstances allowed to execute a given action specified in the API. To avoid repetitions, we introduce algorithms that allow the user to define default behaviors and exceptions. MPEG adopted this approach. It was published in [85, 86]. MPEG-G's API defines filtering criteria which are not supported by MPEG-G's indexing. We propose an alternative file format allowing to adapt the indexing tables to the foreseen uses: the end-users indicate which criteria they will apply and our new file format adapts the indexing accordingly. The indexing relies on multiple output units, each of which corresponds to a permutation of the filtering criteria. If required, the output unit is also encrypted. We have prepared a draft paper to be submitted [87].

8.1 Requirements

The MPEG group identified no requirements on the Application Programming Interface (API) in the preliminary work. However, to make MPEG-G consistent with other MPEG standards, MPEG-G has to include an API.

8.2 Our proposed solution

We based our initial proposal on the foreseeable lifecycle of the MPEG-G encoded information. We identified six groups of actions:

1. access
   The user submits an identification of the content to be retrieved, and the tool returns the data either in its MPEG-G encoded form, or in the decoded form.

2. modification
   The user submits the identification of the content to be modified and the new value of this content. The tool operates the requested modifications.

3. authorization
   The user wants to discover whether a given action is allowed. The tool returns if the action is permitted or denied based on the user's context information.

4. verification
   The user requests the tool to confirm that the information stored in the file does not appear to have been altered.

5. conversion
   The user requests to convert the information contained in the MPEG-G file into another format or to include the information provided in another format in the MPEG-G file.

6. beacon-like
   The user requests statistics computed across multiple entries in the file (such as multiple Datasets).

We wanted a consistent approach for each of the actions. We wanted our approach to be usable both with local and remote tools and to be agnostic to the implementation characteristics of each tool, for example, its user identification mechanism and the overall work context created and maintained.

We decided to propose a REST-like API, as this addresses our goals:

- REST forces each action to be atomic.
- A tool can use a REST-like API both to work with a local server or a remote one.
- Each actions is independent of the channel transmitting the action request: the channel might be created using HTTPS, cookies, or another technology, but the specification of the action remains agnostic.

Each one of the actions is atomic to the user. Although to perform a given action, the tool might require to perform multiple steps, however to the eyes of the user the action is atomic.

Let us imagine that the user requests some of the reads stored in a given Dataset. In order to perform this, the tool will need to perform a succession of actions:

- accessing the Dataset Group where the Dataset is stored
- identifying the Access Units corresponding to the requested information
- determining whether the Access Units' information is encrypted and decrypting it if required
- decoding the information.

For the user, the action is atomic, yet it is composed of a succession of operations.

The usual HTTP action also corresponded to the set of actions we envisaged for MPEG-G. The actions of requesting data or converting information from or to MPEG-G maps to the HTTP method GET; the modification action can map to PUT, PUSH, DELETE, POST; the verification action to HEAD. The remaining can also be mapped to GET methods.

Our approach defined actions of different magnitude (for example, it could retrieve an entire structure or just a field of said structure), and in order to obtain homogeneity a given action could be used at different levels of the MPEG-G structure (for example obtaining header information from the Dataset Group, the Dataset, or the Access Unit level).

8.3 The adopted approach

The MPEG group had several issues with our proposal. The group summarized these issues in three main requirements:

1. The specification should limit the API to information retrieval.
2. The specification of the different methods should be consistent with the other specifications of the file.
3. The API should not use concepts foreign to the current genomic data representation formats: the API should, for example, not refer to the stream structure, as this does not have an equivalent in a SAM-like file.

The final approach adopts two main traits from our proposal. The standard does not specify how the API creates an execution context. As in our proposal, the user and the tool have started a context through an external channel. It is within this context that the tool executes the actions. During the instantiation of the context, the user and the server (either local or remote), shall agree on the content on which the tool performs the actions, and both parties must exchange the required security information (the user reveals his identity and provides the keys to the tool if required).

As in our proposal, the final approach uses the idea of atomic operations. As the requirements restrict the operations to the use of concepts that are similar to the ones available in current genomic formats, the API focuses on the structure that is equivalent to the current genomic formats, the Dataset. Due to the features present in MPEG-G, some exceptions must be accepted. The security capabilities of MPEG-G introduce new outcomes that are, in current formats, impossible. These outcomes include:

- the lack of the proper key to retrieve the information
- an error while checking the integrity of the information
- denial of the query due to privacy rules.

When performing queries on a Dataset, the user must identify the Dataset which is the object of the query. To avoid the situation where the user has to find the Dataset's ID through trial and error, the API also provides queries to retrieve the structure of the file, i.e., the different IDs of the Dataset Groups and Datasets present in the file.

The API finally adopted to retrieve information from the Datasets offers more fine-tuning than current methods. Alongside the usual region definition (i.e., from which chromosome and from which position to which position shall the tool retrieve data), the API specifies that a conformant tool shall also be capable of filtering according to other criteria. These criteria include, but are not limited to, the class of the read, the quality of the mapping, the existence of multiple alignments, or even the size of specific operations.

8.4 Thoughts on adopted approach

The indexation mechanism defined in the file format uses the sequence employed to align the information, the range on the sequence to which the data belongs, and the encoding class to optimize access times. MPEG-G has adopted a query interface allowing to use other characteristics when identifying the information to return. The combination of these facts leads to a situation where, if the user uses these new filtering fields, the tool will need to fetch, decode and filter many reads that will not be part of the final result.

As we have seen, the issue originates in the difference between the indexation mechanism and the filtering criteria. One solution would be to include every field present in the filtering criteria into the indexation mechanism. There are some issues with this straightforward solution. The first problem is that each indexation field will be bound to additional signaling: including all filtering fields will lead to an increase in overhead, and some of these fields might even not interest any of the foreseen recipients. The second problem is the feasibility of this approach. Some of the criteria can take multiple values per record, and some of the criteria might have an infinite set of possible values (criteria represented as a ratio).

Therefore, we are interested in a solution that provides additional indexation mechanisms, but which limits the increase in overhead. We present next an approach which addresses these needs.

8.4.1 Approach and methods

In our approach all parties who will be working with the sequenced genome are to be asked which subset of the information they intend to use; for example, all the reads having more than 30 inserted nucleotides, or all the reads having a mapping quality beyond a certain threshold. These intended queries are then translated into a set of discrimination criteria. If the original information is reordered and indexed taking into account these criteria, the time required to perform each one of the queries can be greatly reduced, especially for fine-grained searches with just a few compliant results. Furthermore, the privacy requirements are respected by separating and encrypting subsets of the information which are to remain private.

We use the API specified for MPEG-G as the collection of queries we want to be able to optimize. Although this API was intended to be used with MPEG-G files, the query interface is agnostic to the underlying data representation. For the purpose of our experiment, we will be using the Binary Alignment/Map (BAM) [14] format for the information encoding, due to its easiness. However, as in MPEG-G the information is stored in records, we use a similar approach by ordering our information by name; thus all the information related to a same sequencer output is contiguous.

We envision the case where one or multiple shareholders have already a prior guess on which query they want to perform. Therefore, and prior to distributing the sequencing result, we want to reorder the content and index it in order to optimize all of the foreseen queries. Furthermore, we want to respect the privacy concerns of the patient: during the reorganization process, the records are distributed according to the persons who should be granted access and are encrypted accordingly.

8.4.1.1 Processing pipeline

We propose a single interface where all shareholders can contribute their discrimination criteria (e.g. presence of mapped information, length of insertions). Furthermore, a listing of access rights (recipient identification to genomic region) is expected.

Based on these settings and the given input, an output is generated. The output is composed of two files: a database and one file corresponding to the concatenation of multiple BAMs. These output BAMs are possibly encrypted.

The content of the original BAM is divided into different output BAMs. Figure 8.1 shows the pipeline we propose for this. Each record (collection of all alignments of all segments of a given sequencer output) of the original file is given as input to an annotation process which consists in testing against each filtering rule whether one or more tags should be associated to the record. Upon completion, all records are grouped in units sharing exactly the same tags: each one of these groups corresponds to one of the output BAMs.

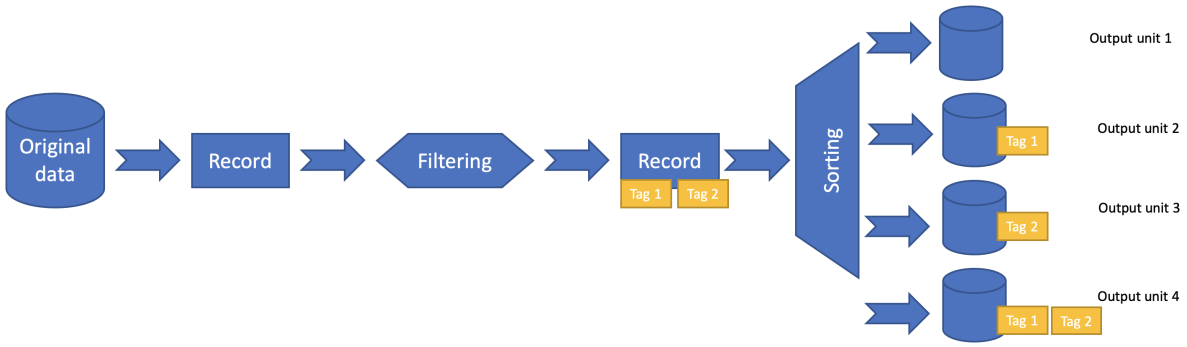


Figure 8.1: Record classification pipeline

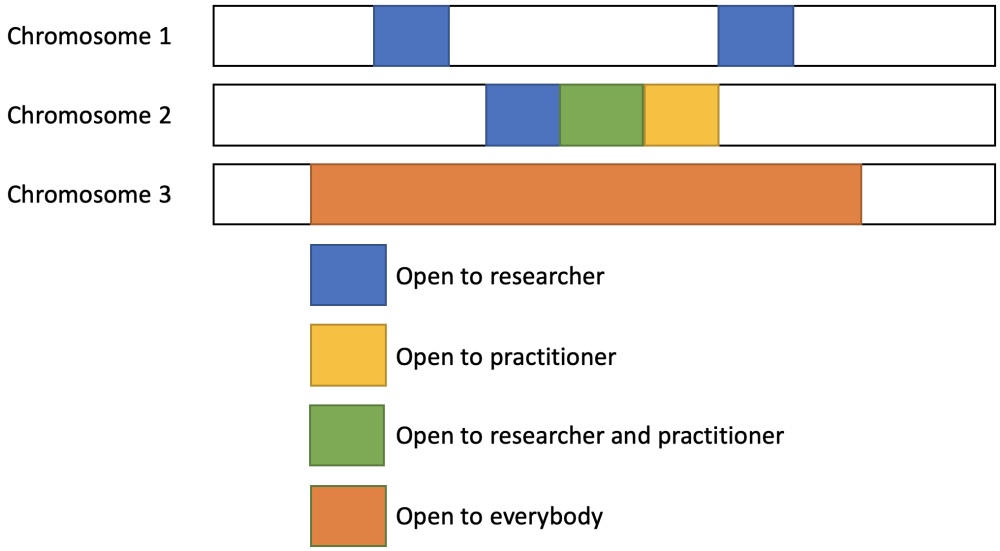


Figure 8.2: User's privacy requirements

The content of each output unit is such that all records stored in it share the same tags. Nevertheless, there might be multiple independent units encoding records with the same tags. The database being generated alongside the different BAM files allows the querying of the different units: for each one of them, there are entries documenting each tag related to it.

8.4.1.2 Collaboration among multiple users

For this example, we will suppose that there are three different actors: the owner of the file, a practitioner and a researcher. The owner of the file wants to distribute a copy, but the practitioner and the researcher shall have access to only selected parts of the file, as shown in Figure 8.2. Let us also suppose that the practitioner intends to use those records where one of the reads is unmapped, and the researcher intends to use those records where a long insert appears.

The presented approach will combine the requirements of all three actors and divide the input into multiple output units such that each one of the output units corresponds to only one of the permutations of the criteria. One possible result is shown in Figure 8.3, where both the privacy settings and the filtering criteria have been taken into account. The number within the cells is the identifier of the output unit.

						Has one read unmapped	Has long insert
Chromosome 1	1	2	3	4	5	False	False
	6	7	8	9	10	False	True
	11	12	13	14	15	False	False
	16	17	18	19	20	True	True

Figure 8.3: Division in Access Units matching the filtering queries and the security settings

The foundation of the sorting of records in output units is the attachment of tags. Each tag corresponds to a specific trait which is expected to be of relevance for the future pipelines.

8.4.1.2.1 General tags The criteria are represented as rules, which, for each record, assign a tag: for example, one rule will add a tag *hasMapped*, with either a true or a false value, and another rule will add a tag *numInsertions*, with the number of insertions. Source code 8.1 shows one example of rules. We expect every user who will be using the distributed file to contribute to these rules. Each one of these contributions can be translated to more tags, or finer grained bins.

XML source 8.1: Example of tagging rules

```
<ns:Rules xmlns:ns="es.upc.dmag.dispatcher">
    <ns:markMapped>true</ns:markMapped>
    <ns:markUnmapped>true</ns:markUnmapped>
    <ns:markHasPairedEnded>true</ns:markHasPairedEnded>
    <ns:markHasSingleEnded>true</ns:markHasSingleEnded>
    <ns:markHasOpticalDuplicateRule>true</ns:markHasOpticalDuplicateRule>
    <ns:markGroup>true</ns:markGroup>
    <ns:deletionsLengthFractionBins>
        <ns:boundary>0.25</ns:boundary>
        <ns:boundary>0.5</ns:boundary>
        <ns:boundary>0.75</ns:boundary>
    </ns:deletionsLengthFractionBins>
    <ns:deletionsLengthBins>
        <ns:boundary>25</ns:boundary>
        <ns:boundary>50</ns:boundary>
        <ns:boundary>75</ns:boundary>
    </ns:deletionsLengthBins>
    <ns:insertionsLengthFractionBins>
        <ns:boundary>0.25</ns:boundary>
        <ns:boundary>0.5</ns:boundary>
        <ns:boundary>0.75</ns:boundary>
    </ns:insertionsLengthFractionBins>
    <ns:insertionsLengthBins>
        <ns:boundary>25</ns:boundary>
        <ns:boundary>50</ns:boundary>
        <ns:boundary>75</ns:boundary>
    </ns:insertionsLengthBins>
    <ns:mismatchedBasesFractionBins>
        <ns:boundary>0.25</ns:boundary>
        <ns:boundary>0.5</ns:boundary>
        <ns:boundary>0.75</ns:boundary>
    </ns:mismatchedBasesFractionBins>
    <ns:mismatchedBasesBins>
        <ns:boundary>25</ns:boundary>
        <ns:boundary>50</ns:boundary>
        <ns:boundary>75</ns:boundary>
    </ns:mismatchedBasesBins>
    <ns:numDeletionsBins>
        <ns:boundary>1</ns:boundary>
        <ns:boundary>3</ns:boundary>
        <ns:boundary>5</ns:boundary>
        <ns:boundary>10</ns:boundary>
    </ns:numDeletionsBins>
    <ns:numDeletionsFractionBins>
        <ns:boundary>0.01</ns:boundary>
        <ns:boundary>0.03</ns:boundary>
        <ns:boundary>0.05</ns:boundary>
        <ns:boundary>0.1</ns:boundary>
    </ns:numDeletionsFractionBins>
    <ns:numInsertionsBins>
        <ns:boundary>1</ns:boundary>
        <ns:boundary>3</ns:boundary>
        <ns:boundary>5</ns:boundary>
        <ns:boundary>10</ns:boundary>
    </ns:numInsertionsBins>
    <ns:numInsertionsFractionBins>
        <ns:boundary>0.01</ns:boundary>
        <ns:boundary>0.03</ns:boundary>
        <ns:boundary>0.05</ns:boundary>
        <ns:boundary>0.1</ns:boundary>
    </ns:numInsertionsFractionBins>
    <ns:numSplicesBins>
        <ns:boundary>1</ns:boundary>
        <ns:boundary>2</ns:boundary>
```

```
        <ns:boundary>3</ns:boundary>
        <ns:boundary>4</ns:boundary>
    </ns:numSplicesBins>
    <ns:numSplicesFractionBins>
        <ns:boundary>0.01</ns:boundary>
        <ns:boundary>0.02</ns:boundary>
        <ns:boundary>0.03</ns:boundary>
        <ns:boundary>0.04</ns:boundary>
    </ns:numSplicesFractionBins>
    <ns:spliceLengthBins>
        <ns:boundary>1</ns:boundary>
        <ns:boundary>5</ns:boundary>
        <ns:boundary>10</ns:boundary>
        <ns:boundary>30</ns:boundary>
        <ns:boundary>50</ns:boundary>
    </ns:spliceLengthBins>
    <ns:spliceLengthFractionBins>
        <ns:boundary>0.01</ns:boundary>
        <ns:boundary>0.05</ns:boundary>
        <ns:boundary>0.1</ns:boundary>
        <ns:boundary>0.3</ns:boundary>
        <ns:boundary>0.5</ns:boundary>
    </ns:spliceLengthFractionBins>
</ns:Rules>
```

The currently implemented rules are the ones needed to provide an API similar to the one MPEG-G offers. Although this API has been designed to work with MPEG-G files, it could be readily applied to other formats. Its main focus is on allowing more specific filtering on the data: whereas the SAMtools command line allows to filter per position, the MPEG-G API uses structures allowing to filter according to certain properties of the genomic information (e.g., the minimal or maximal quality, the number or length of insertions, deletions, mismatches, among others). Our tagging rules are a translation of these MPEG-G API's filtering structures. As using the real value of these properties would result in many output units, the values are binned together. The size of each bin is user-specified. For example, in one file, all reads having between 1 and 25 inserted nucleotides can be marked as belonging to the bin 2.

The different rules are translated into tables in the indexation database. This allows to keep track of which tags are related with which output units, where to find the output unit and whether it is encrypted or not. The database schema can be seen in Figure 8.4.

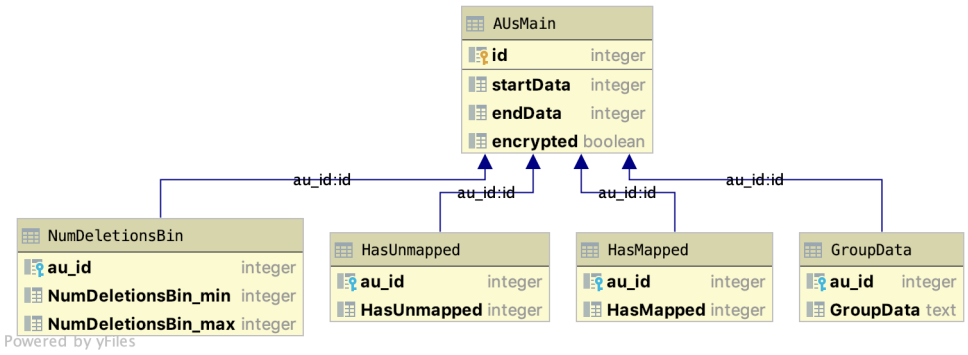


Figure 8.4: Indexation database schema

8.4.1.2.2 Security tags Alongside the rules concerning the query optimization, we provide the owner of the file with a mechanism to set privacy requirements on the data.

The security rule generates the list of recipients to whom access to a given record is granted. The result can span from no identifier whatsoever (in this case, the data is encrypted with only the patient as recipient) to open to everybody (in this case, the data is stored unencrypted). Source code 8.2 shows an example of such a configuration.

XML source 8.2: Security rules: regions open for some recipients

```
<ns:SecurityRules xmlns:ns="es.upc.dmag.dispatcher">
<ns:securityRule>
        <ns:region>
            <ns:referenceId>14</ns:referenceId>
            <ns:start>38727275</ns:start>
```

```
            <ns:end>38728481</ns:end>
        </ns:region>
        <ns:recipients>
            <ns:Recipient>Researcher</ns:Recipient>
        </ns:recipients>
    </ns:securityRule>
    <ns:securityRule>
        <ns:region>
            <ns:referenceId>11</ns:referenceId>
            <ns:start>17392498</ns:start>
            <ns:end>17476879</ns:end>
        </ns:region>
        <ns:anybody/>
    </ns:securityRule>
    <ns:securityRule>
        <ns:region>
            <ns:referenceId>9</ns:referenceId>
            <ns:start>136673143</ns:start>
            <ns:end>136687457</ns:end>
        </ns:region>
        <ns:anybody/>
    </ns:securityRule>
    <ns:securityRule>
        <ns:region>
            <ns:referenceId>9</ns:referenceId>
            <ns:start>21994139</ns:start>
            <ns:end>22128103</ns:end>
        </ns:region>
        <ns:recipients>
            <ns:Recipient>Researcher</ns:Recipient>
            <ns:Recipient>Physician</ns:Recipient>
        </ns:recipients>
    </ns:securityRule>
</ns:SecurityRules>
```

In order to generate the list of recipients for a given record, the security rule matches each record against each region parametrized in the security configuration. The final list of recipients is the list of recipients common to each of the regions. If the list is empty, it is understood that just the owner of the file is to have access to it. We use PGP [47] to encrypt the output units. This format allows us to encrypt the unit only once, yet leaving it available to the entire list of designated recipients.

As the security tag is treated like any other tag, it does not affect the previously described pipeline: all records being collected in one output structure share the same general and security tags.

8.4.1.3 Filtering

The representation of filtering criteria is similar to the one of tagging rules. For each of the filters, it has to be indicated whether the filtering criteria applies to one or all reads. In the case where the filtering criteria is used in conjunction with the *query* command, the filtering criteria are translated into a database query, allowing to recover only those output units which seem to be relevant. In case a parameter was not foreseen as relevant during the encoding phase is, therefore, not included in the indexation information, the parameter is left out of the query.

XML source 8.3: Filter rules for perfect match

```
<ns:AndFilter xmlns:ns="es.upc.dmag.dispatcher">
    <ns:HasMapped>
        <ns:value>true</ns:value>
        <ns:forAllReads>true</ns:forAllReads>
    </ns:HasMapped>
    <ns:HasOptical>
        <ns:value>false</ns:value>
        <ns:forAllReads>true</ns:forAllReads>
    </ns:HasOptical>
    <ns:HasUnmapped>
        <ns:value>false</ns:value>
        <ns:forAllReads>false</ns:forAllReads>
    </ns:HasUnmapped>
    <ns:NumDeletionsBin>
        <ns:lowerBound>0</ns:lowerBound>
        <ns:upperBound>1</ns:upperBound>
        <ns:forAllReads>true</ns:forAllReads>
    </ns:NumDeletionsBin>
    <ns:NumInsertionsBin>
```

```
        <ns:lowerBound>0</ns:lowerBound>
        <ns:upperBound>1</ns:upperBound>
        <ns:forAllReads>true</ns:forAllReads>
    </ns:NumInsertionsBin>
    <ns:NumMismatchesBin>
        <ns:lowerBound>0</ns:lowerBound>
        <ns:upperBound>1</ns:upperBound>
        <ns:forAllReads>true</ns:forAllReads>
    </ns:NumMismatchesBin>
    <ns:NumSplicesBin>
        <ns:lowerBound>0</ns:lowerBound>
        <ns:upperBound>1</ns:upperBound>
        <ns:forAllReads>true</ns:forAllReads>
    </ns:NumSplicesBin>
</ns:AndFilter>
```

8.4.2 Results and discussion

We use the genomic query API defined in MPEG-G as it is convenient for testing the approach. It provides us with a well-defined collection of criteria which can be used when searching. We generate multiple files, from coarse-grained to fine-grained output units, and test four different queries against these files. The filtering criteria are as follows:

- no rules
  The number of records is the maximum one which could be held in memory.

- separate types
  The only rules used concern the number of operations: the records without any operation of a given type are separated from the rest, thus recreating the concept of classes from MPEG-G.

- separate types 25 and separate types 5
  The tags are created based on bins of width 25 and 5 respectively (or 0.25 and 0.05 if decimal). Those reads without any operation of a given type are still separated from all other records.

We test four different queries:

1. retrieving all records having an exact match with the reference
2. retrieving all records having one read mapped and the other unmapped
3. adding to the prior query the requirement of the mapped record being an exact match with the reference
4. querying for those records where at least one record has a number of inserted nucleotides in a given range.

The time required to perform each query is summarized in Table 8.1. The entry 'reference' corresponds to the results obtained when filtering the original BAM file and serves as the benchmark to beat.

As it can be observed from the reference results, there are two main factors contributing to the computation time: the time to read the entire file and the time to write the results, evidentiated by the variance of times for the reference, directly imputable to the number of records to be written. The proposed methodology allows to reduce the time spent in reading the file. However, there is a trade-off between the granularity and the overhead caused by accessing more output units.

Table 8.1: Seconds to extract records for filter

	Reference	No rules	Separate types	Separate types 25	Separate types 5
Perfect match	387	388	398	391	395
Half-mapped	112	114	7	8	9
Half-mapped perfect match	108	113	4	4	5
Long inserts	109	107	1	0,623	0,746
Mid-long inserts	110	106	4	5	1

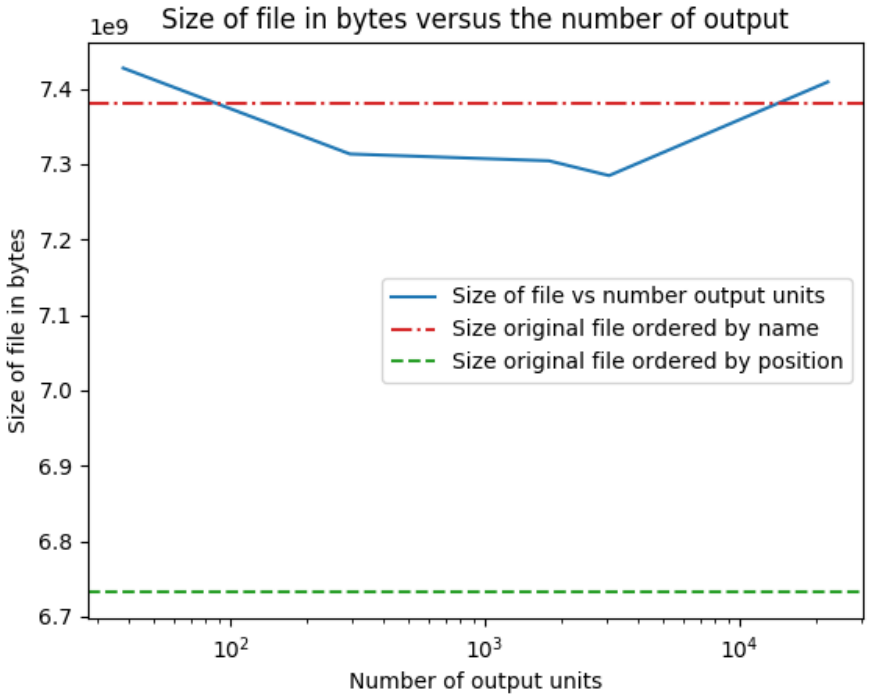


Figure 8.5: Size of output versus number of output units

As previously stated, we are aiming at recreating the MPEG-G API and thus require all information relevant to a given template to be contiguous. In order to achieve this, we use, both as input and output, files which are ordered by query name. This does automatically translates into an increase in the file size, as the information is harder to compress.

We include the header information in each of the output units, and this represents approximately 6 kilobytes per output unit, thus creating an overhead. However, the result of clustering similar reads is an increase in the compressibility. This leads to a trade-off between a lower entropy and more output units, as can be observed in Figure 8.5. Although we have not tested it, it can be assumed that the size of the output file could be further lowered if the ordering was changed to something more akin to position ordering and if the header were not included in each of the output units.

8.4.3 Conclusion

We have developed a tool that, given a BAM file as input, generates a new file with optimized querying mechanisms. We also take into account the file owner's privacy settings to protect the corresponding file accordingly. At the moment, our implementation only addresses the queries defined in MPEG-G's APIs, but can be readily extended to other use cases.

Our results show that there are important optimizations to be achieved by using our technique. However, there are caveats: the speed-ups can only be observed if it is possible to avoid reading many records, and the time required to write can render any improvements insignificant.

8.5 Privacy rules

8.5.1 Requirements

As we have seen in clause 5.1, one of the earliest identified requirements, is the need to control who is allowed to do what with the content of the file. In clause 7.2, we have defined a method to limit access to some aspects of the information to individual users. We have done so by encrypting specific regions of the files with specific keys. This technology gives us the vocabulary to create a key-distribution infrastructure that controls who has access to what information.

However, the requirement specifies the need to control what the user is allowed to do with the content. Being allowed to perform a particular task with information of a specific region is different from having access to the region. For example, the file's user might be allowed to add the observed mutations to a frequency database, yet not allowed to deduce information about the patient from this (such as genetic lineage). For

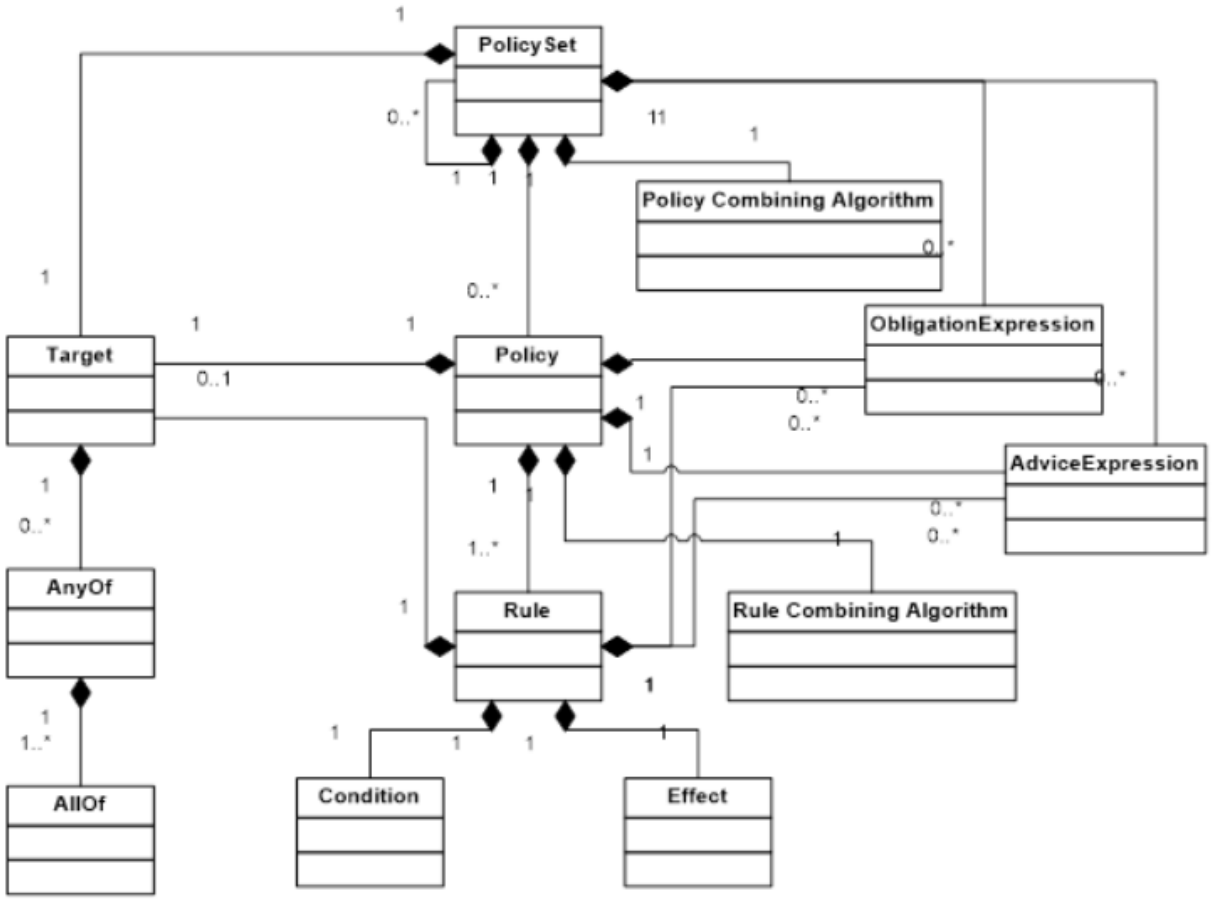


Figure 8.6: Policy Language Model, figure from [88]

both actions, the user must have access to the content of the file in plaintext, but the patient has given the researcher permission to perform only one of them: having access to information through the decryption mechanism and being allowed to perform a given task on this information is not equivalent.

We thus need a mechanism to specify who, and under which circumstances, is allowed to perform a given task.

8.5.2 Proposed solution

8.5.2.1 XACML

8.5.2.1.1 Definition of Privacy Rules using XACML The eXtensible Access Control Markup Language (XACML) standard version 3.0 [88] was specified by the OASIS organization in 2013 [89] and reviewed in 2017. It is an XML-based language that allows to express authorization policies which can be applied to digital objects.

The main motivation for the definition of XACML was the big amount of proprietary and application-specific access control policy languages used to define policies, which once defined cannot be shared across different systems. To this end, XACML specifies a Policy Language Model, shown in Figure 8.6. The three top-level elements defined for this model are Rule, Policy and PolicySet. The rule element is the basic unit of management within an XACML Policy Administration Point (PAP). The policy element consists of rule elements and mechanisms for combining the results of their evaluation. The PolicySet element enables the combination of separate policies into a single one. In order to ask for authorization, the Policy Enforcement Point (PEP) and the Policy Decision Point (PDP) have to exchange some information in the form of XACML request and response. Both information structures are also defined in the standard and contain the required information to perform the authorization and send its result. Figure 8.6 shows the Policy Language Model as defined in the XACML standard. It also presents the relationships between Rule, Policy and PolicySet, as well as the elements that compose them. We briefly summarize them next.

A rule is the most elementary unit of policy. The main components of a rule are:

1. a target
   It defines the set of attributes identifying subjects or roles, resources and actions needed to request

authorization. If this element is not present, it has to be taken from the parent policy.

2. an effect
   It indicates what happens when the rule applies. Its value can be Permit or Deny.

3. a set of conditions
   Each one represents a boolean expression that refines the applicability of the rule according to the target. It is an optional element.

4. obligation expressions
   They represent actions to be performed after application of the rule; for instance, send an email to the owner of the information when access to it has been granted.

5. advice expressions
   They are similar to obligations, but can be ignored.

In order to be evaluated, rules must be encapsulated into policies, which may contain more than one rule. The main components of a policy are the following:

1. a target
   It defines the set of attributes identifying subjects or roles, resources and actions needed to request authorization applying to the policy.

2. a rule-combining algorithm-identifier
   The algorithm-identifier specifies the procedure by which the results of evaluating each rule have to be combined to provide a final result at the policy level.

3. a set of rules

4. obligation expressions
   They represent actions to be performed after evaluation of the policy.

5. advice expressions
   They are similar to obligations, but can be ignored.

Finally, a policy set comprises the following components:

1. a target
   It defines the set of attributes identifying subjects or roles, resources and actions needed to request authorization applying to the policy set.

2. a policy-combining algorithm-identifier
   The algorithm-identifier specifies the procedure by which the results of evaluating each policy have to be combined to provide a final result at the policy set level.

3. a set of policies

4. obligation expressions
   They represent actions to be performed after evaluation of the policy set.

5. advice expressions
   They are similar to obligations, but can be ignored.

To ask for authorization, an XACML request has to be provided, which contains several attribute values that have to be checked against the attributes defined in the different rules in order to provide an authorization decision. These attributes have an associated category and identifier that must be identical both in the request and in the rules of the policy. After checking whether the authorization is granted, an XACML response is generated which informs of the result. The possible values in the response are: Permit, Deny, Indeterminate or Not Applicable.

As XACML follows an XML schema, syntactical correctness is implicit, as any policy must follow the schema to be valid. Regarding the quality and correctness of policies and rules, other aspects must be considered, as [90] analyzes. In any case, other considerations regarding the validity of XACML rules are out of the scope of this work.

The XACML architecture presented in Figure 8.7 consists of the following building blocks:

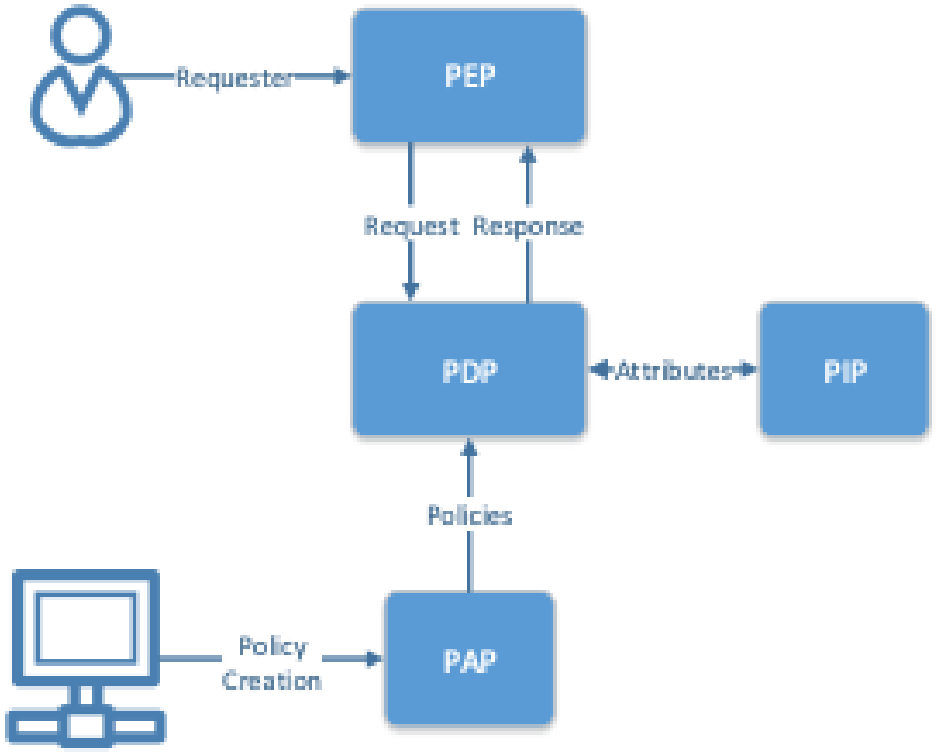


Figure 8.7: XACML Architecture

1. Policy Enforcement Point (PEP)
   It performs access control by making decision requests and enforcing authorization decisions. This is the entity that sends the XACML request to the Policy Decision Point (PDP) and receives an authorization decision.

2. Policy Decision Point (PDP)
   It evaluates an applicable policy and returns an authorization decision.

3. Policy Information Point (PIP)
   It acts as a source of attribute values. Basically, if there are attributes missing in the XACML request that is sent by the PEP, the PIP will find them for the PDP to evaluate the policy.

4. Policy Administration Point (PAP)
   It creates policies or policy sets and manages them.

Our approach is to use XACML to define privacy rules which can be automatically checked, in order to establish whether a particular use of the data is permitted in a given situation.

8.5.2.2 Using privacy rules to protect the API

We propose to use the methods defined in the API as the list of possible actions a user could perform. We propose to convey privacy rules at the Dataset Group and the Dataset levels. We include privacy rules neither at the Access Unit nor at the Stream level, as there are no actions specific to them. We include a privacy rule by placing an XACML policy element in the schema for the Dataset Group and Dataset Protection.

The file structure already provides the relationship between the policy and the structure to which it refers. Therefore, there is no need to link both in the policy.

XACML defines a range of attributes to use in the resolution of a policy. A central attribute is the one identifying the requested action: the attribute identified as "`urn:oasis:names:tc:xacml:1.0:action:action-id`". In our proposal, this attribute shall take the name of one of the API methods as value in the request. In this case we use an attribute with an ID defined by the XACML specification and a value defined by the MPEG-G specification. XML source 8.4 shows an example using this attribute.

XML source 8.4: XACML basic deny all getData requests

```
<Rule
 RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleMPEGG"
 Effect="Deny"
>
 <Description>Denying all getData</Description>
  <Target>
   <AnyOf>
    <AllOf>
      <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema\#string">getData</AttributeValue>
     <AttributeDesignator
      MustBePresent="true"
      Category=
      "urn:oasis:names:tc:xacml:3.0:attribute-category:action"
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema\#string"
     />
    </Match>
   </AllOf>
  </AnyOf>
 </Target>
</Rule>
```

The methods defined in the API also use other attributes. For example in the case of "getData", the user can select whether multiple alignments should be considered in the returned result. This filtering can be relevant to the privacy rules (must notably when used to delimit the returned region of the genome), therefore in our proposal this shall also be included in the request. In our example of multiple alignments, this translates into an attribute with its ID equal to " `presence_of_multiple_alignments`". To limit the effects of the rule, the file creator can use attributes whose IDs and values are defined in the MPEG-G specification. XML source 8.5 shows how the previous example has been refined to take into account the request for multiple alignments.

XML source 8.5: XACML basic deny getData request, if the presence of multiple alignments is required

```
<Rule
 RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleMPEGG"
 Effect="Deny"
>
 <Description>Denying all getData</Description>
 <Target>
  <AnyOf>
   <AllOf>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema\#string">getData</AttributeValue>
     <AttributeDesignator
      MustBePresent="true"
      Category=
      "urn:oasis:names:tc:xacml:3.0:attribute-category:action"
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema\#string"
     />
   </AllOf>
  </AnyOf>
 </Target>
 <Condition>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-one-and-only">
     <AttributeDesignator MustBePresent="true" Category="alignment"
      AttributeId="presence_of_multiple_alignments"
      DataType="http://www.w3.org/2001/XMLSchema#boolean"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
   </Apply>
  </Apply>
 </Condition>
</Rule>
```

We expect file creators to require other attributes to fully represent the desired rules. New attributes can be identified by using identifiers defined in the XACML specification (for example the role of the user is identified "`urn:oasis:names:tc:xacml:3.0:example:attribute:role`"). Other attributes could require new identifiers and semantics to be defined (for instance to cover the topic of the research).

XML source 8.6: XACML basic deny getData request if requested by a researcher and if the presence of multiple alignments is required

```
<Rule
 RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleMPEGG"
 Effect="Deny"
>
 <Description>Denying all getData</Description>
 <Target>
  <AnyOf>
   <AllOf>
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema\#string">getData</AttributeValue>
     <AttributeDesignator
      MustBePresent="true"
      Category=
      "urn:oasis:names:tc:xacml:3.0:attribute-category:action"
      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema\#string"
     />
    </Match>
    </Match>
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">researcher</AttributeValue>
     <AttributeDesignator MustBePresent="false"
      Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
      AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Match>
   </AllOf>
  </AnyOf>
 </Target>
 <Condition>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-one-and-only">
     <AttributeDesignator MustBePresent="true" Category="alignment"
      AttributeId="presence_of_multiple_alignments"
      DataType="http://www.w3.org/2001/XMLSchema#boolean"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
   </Apply>
  </Apply>
 </Condition>
</Rule>
```

8.5.2.3 Avoiding repetitions

The MPEG-G file format creates multiple semantic levels within the file: there are multiple Dataset Groups that group multiple Datasets that collect multiple Access Units. We have seen that the MPEG group has expressed the requirement that the security mechanisms should not involve concepts foreign to SAM-like files: therefore, our proposal does not provide privacy rules at the Access Unit level (at least not explicitly).

The API defines methods that are similar for both the Dataset Group and the Dataset. For example, the user can use the getData method to retrieve information either from the Dataset Group or the Dataset. There are in fact two different methods, one for the Dataset Group, the other for the Dataset. As we have seen, this would lead to multiple rules: one for the Dataset Group and one per Dataset. However, we can expect a certain homogeneity in permissions. For example, in a Dataset Group for research purposes on Alzheimer, we can expect all requests to Alzheimer related regions of the genome to be granted. To simplify the rules, we want a mechanism that delegates the permission for the Dataset to the Dataset Group. This mechanism should not impede one Dataset to diverge from the rules applied to the Dataset Group. In other words, the permission for a Dataset must be able to rely on a default Dataset Group-wide policy, but the Dataset Group-wide policy should not hide specificities of the Dataset.

We propose an algorithmic solution to this situation. There are two algorithms: the case where access to one Dataset is requested and the case where access to all data in the Dataset Group is requested.

As seen in Figure 8.8, when the user requests data from the Dataset, the Dataset-specific rules are checked. If the rules grant permission, the data is returned. If not, the algorithm checks whether the equivalent request to the Dataset Group is granted. If so, the algorithm extends the original request to the Dataset by one additional attribute: grantedByDatasetGroup with value 'true'. The algorithm then checks the updated request against the Dataset policy. The resulting response is then the definitive answer.

The creator of the rules can use this additional attribute to define new behaviors. In order to meet the previously defined default policy, a rule could simply return the permission based on this additional

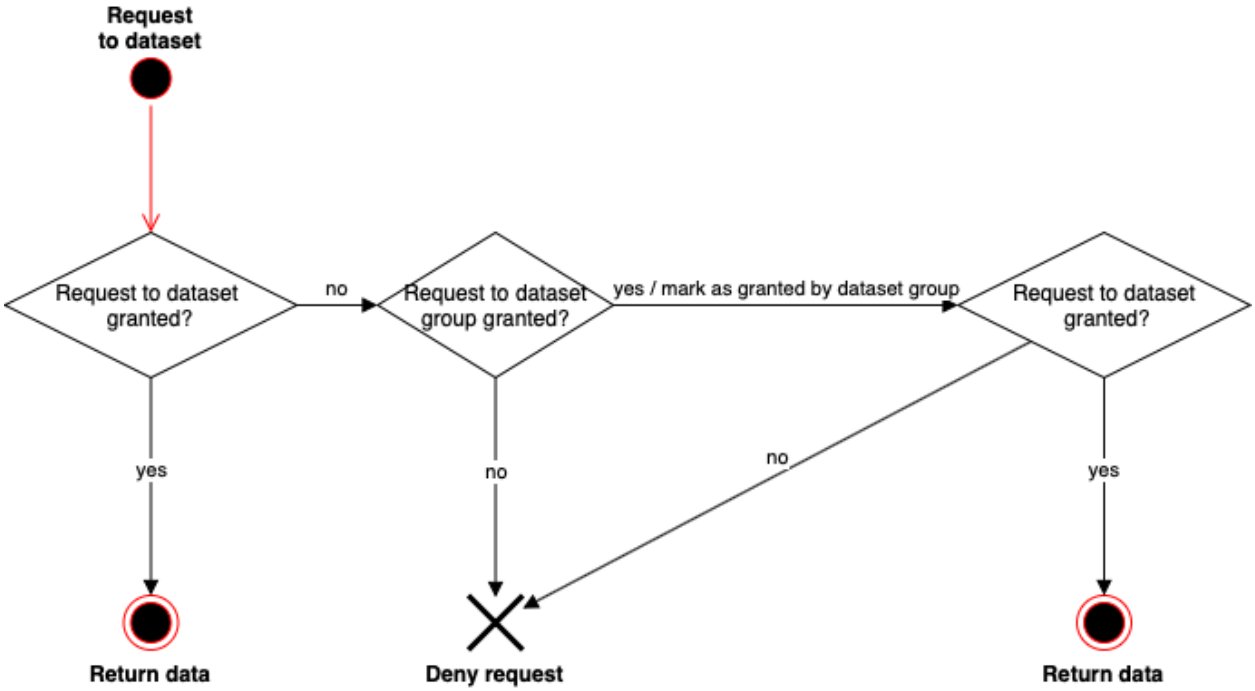


Figure 8.8: Resolving Dataset permissions

attribute.

We do not want to return information to which the access is denied. However, due to the Dataset Group policy we might see a request granted: when the user requests genomic information from the Dataset Group, the API returns the corresponding and accessible information from each Dataset. We must point out the difference between the request being granted and the information being returned. For example, there might be no reason at a Dataset Group level to deny a certain request; yet, no patient might be willing to share this information. An MPEG-G compliant tool shall verify for each Dataset whether its information can be included. This process is shown in Figure 8.9.

As the Access Unit width is established during the encoding which might not be when the privacy rules are being generated, we might face a situation where multiple access rules affect one Access Unit. When performing a getData method on a Dataset, a tool must decode every Access Unit which might contain relevant information.

One solution would have been to test whether the record's position conforms to the access rules. Given the number of records, this would be computationally costly.

Our approach returns information from the Access Unit only if access to its entirety is granted. We determine this by checking whether a getData request is granted within the boundaries of the Access Unit. This process is shown in Figure 8.10.

8.5.3 Adopted approach

The retained solution was our proposal and it has been taken over in clause 7.3 of Part 3 Metadata and application programming interfaces of ISO/IEC 23092-3 [60].

8.5.4 Thoughts on adopted approach

In our approach, we use XACML to provide the privacy rules. There are similar representation methods such as ADA-M. As we have seen, ADA-M might require the use of natural language, whereas XACML forces that the rule be represented in a computer treatable way. In order to support all features of ADA-M, new attributes for the XACML rules might have to be defined by the repository.

Our proposal does not use XACML to protect the data: a non-conformant tool could circumvent the privacy rule and access the encoded information. In order to mitigate this issue, the cryptography mechanisms must be used. They limit the range of persons who have access to certain units of information, but they cannot protect against the case where a user has access to the information under certain conditions but does not respect them. For example, a researcher having access to a given gene for the purpose of studying a disease but in fact uses it to determine the genetic lineage of the patient.

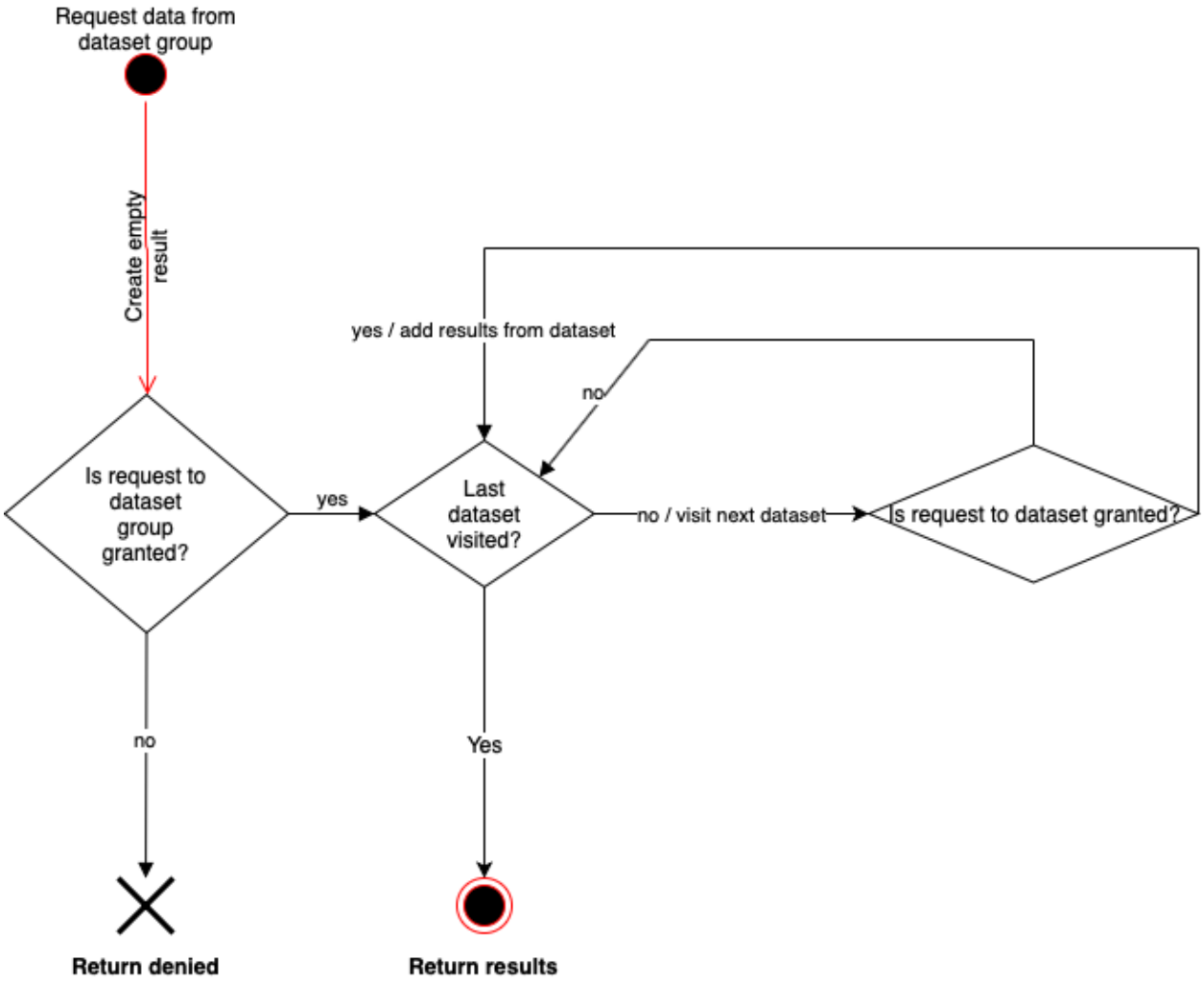


Figure 8.9: getData Dataset group resolution

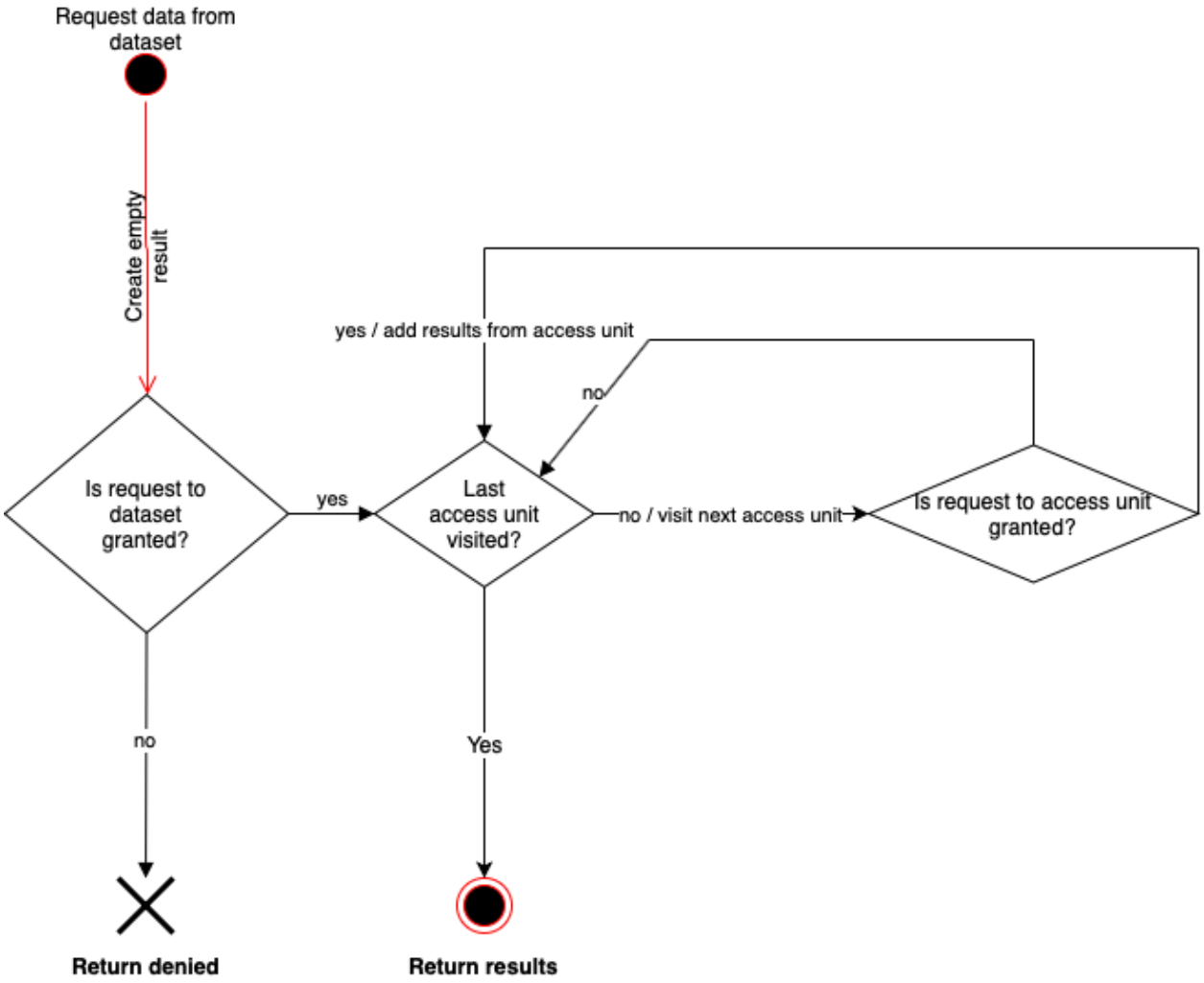


Figure 8.10: GetData Dataset resolution

Some alternatives are under development which might further mitigate this issue: for example Ciphertext Policy - Attribute-Based Encryption (CP-ABE) [91] and functional encryption [92].

CP-ABE is a type of encryption where each key has a set of attributes associated to it. The ciphertext can be decrypted by a key if its attributes verify the policy tree integrated into the cipher text. Using CP-ABE would for example allow encrypting the key used for the digital content: each user holding a key associated to the required parameters would be able to obtain the key used to encrypt the rest of the file. This solution has, however, some drawbacks: to the best of our knowledge, there is yet no standardized cipher to rely on, and the challenge consists in distributing the pool of keys.

Chapter 9

Conclusion

9.1 Results

Genomic information is sensible. It can reveal the genetic origins of the patient and possible diseases. However, what makes this information so sensible also makes it so valuable for research and medical purposes. By sequencing many genomes and cross-comparing the results, we will be able to understand new biological mechanisms, which will lead to new diagnostic tools and treatments. Due to the magnitude of the required information, this process requires new compression mechanisms.

The ISO working group MPEG started, in 2016, a new standardization process for genomic data. MPEG's goal is to use its knowledge in compression techniques to define the new state of the art compression specification for genomic information. However, the new standard should not just address the requirements for compressing genomic data: it shall also represent the information metadata and provide security mechanisms for the data and the metadata.

Our main work has consisted in answering the question of how to protect the data and the metadata. We have explained how we have taken advantage of the granularity specified by the encoding process to segment the genomic data and to provide region-specific encryption and signature mechanisms. We have also analyzed the level of security of other solutions, thus proving the danger of allowing too fine-grained security mechanisms.

To convey the security parameters alongside the encoded genomic information, a new file format is needed, to which we have also contributed. Our solution is inspired by the ISO Base Media File Format. Our format allows the combination of multiple genomic sources in one file: it thus enables researchers to represent an entire study in one file, for example.

We have defined the metadata representation and how to secure it. We have adopted an approach based on a core profile, whose coverage the user can improve through extensions. We have defined this metadata structure with XML schemas. The XML format has, in turn, allowed us to use existing security mechanisms to protect the metadata (XML signature and XML encryption).

We have also proposed a mechanism for privacy rules. These rules specify who, and under which circumstances, can execute a specific action specified in the MPEG-G's API. Our definition of privacy rules does not replace the encryption mechanism but rather extends it. The privacy rules can easily be circumvented by a non-MPEG-G-compliant tool. Therefore the encryption mechanisms must be used to limit the possible reach of the circumvention. Our proposed privacy rules mechanism relies on the XACML standard. We use the file format to identify the resource being accessed and the API to define the actions and the attributes which can be requested. The mechanism allows to extend the definition easily with new attributes corresponding to new needs. Finally, our solution avoids to repeat rules by defining two algorithms which combine rules from different file hierarchies.

Finally, we have proposed two new technologies which go beyond the scope of the current MPEG-G specification. Presently, if a file is unauthorizedly published, it is not possible to discover who published it. In order to mitigate this issue, we have proposed a fingerprinting mechanism: for each one of the recipients of the file, the content is uniquely modified. In case of a publication, the recipient responsible for this copy of the information can thus be discovered. In order not to harm the utility of the genomic information, the modifications can be reverted if the key is made available.

Our second proposed technology aims at optimizing user queries. We propose to segregate information according to the criteria of each user. This segregation then allows to retrieve specific subsets of the information. We also segregate the information on the basis of the protection policies: our approach allows specific segregated subsets to be encrypted for specific recipients.

9.2 Outputs

The result of the research carried out in the context of this work has taken mainly two shapes: the contributions to journals and conferences and the input documents submitted to MPEG.

9.2.1 Publications

9.2.1.1 Journals

1. "Reversible fingerprinting for genomic information" published in *Multimedia Tools and Applications* journal [70]
   It presents our work on the fingerprinting technique for genomic information. This contribution is relevant to clause 7.3 of this work.

9.2.1.2 Submitted

1. We submitted the draft "Side-channel attack on a partially encrypted MPEG-G file" to *Multimedia Tools and Applications* journal [69].
   This draft presents our work on the side-channel attack. This contribution is relevant to clause 7.2.5 of this work.

2. We submitted the draft "Providing FAIR principles while preserving patient's privacy" to Methods of Information in Medicine [93].
   This draft presents how XACML can be used to provide privacy within the context of the FAIR principles [94]. This contribution is relevant to clause 8.5 of this work.

9.2.1.3 Draft to be submitted

1. We have prepared a draft "Improving queries on genomic information by segregation" to be submitted to *Patterns* journal [87].
   This draft presents our work on an alternative indexing system to optimize query times. This contribution is relevant to clause 8.4.

9.2.1.4 Conferences

1. "Adding security and privacy to genomic information representation" published in *Studies in Health Technology and Informatics*, presented at the EFMI Special Topic Conference (Hannover, 2019) [85]
   This contribution presents how the XACML approach we previously presented can be applied to MPEG-G. This contribution is relevant to clause 8.5.

2. "xACS: Management of privacy in genomic and medical information" presented at 6th International Workshop on Genome Privacy and Security (Boston, 2019) [95]
   This contribution presents a generalization of the Picture Archiving and Communication System to any type of resources, and maintaining access control using rules expressed in XACML.

3. "Metadata to Describe Genomic Information" published in *Studies in Health Technology and Informatics*, presented at the Medical Informatics Europe conference (Gothenburg, 2018) [68]
   This contribution presents our work on metadata extensions and profiles, first in the context of GENIFF and then MPEG-G. This contribution is relevant to Chapter 6 of this work.

4. "Protecting Privacy of Genomic Information" published in *Studies in Health Technology and Informatics*, presented at the Medical Informatics Europe conference (Manchester, 2017) [86]
   This contribution introduces the use of XACML in the context of genomics. This contribution is relevant to clause 8.5.

9.2.1.5 Posters

1. "Genie: an MPEG-G Conformant Software to Compress Genomic Data" presented at The International Conference for High Performance Computing, Networking, Storage, and Analysis(Denver, Colorado, 2019) [62]
   It presents the results obtained by the demonstrator for unaligned compression, in the development of which we have collaborated.

2. "mpegg 1.0 – the pure java implementation of the MPEG-G ISO/IEC 23092 standard" presented at ELIXIR All Hands (Lisbom, 2019)[63]
   It presents our pure Java implementation of MPEG-G, alongside compression results we have obtained.

3. "Representing compressed and secure genomic information" presented at the Medical Informatics Europe conference (Gothenburg, 2018) [65]
   It summarizes our proposal for a file format that conveys multiple genomic results in one file, alongside metadata and protection information. This contribution is relevant to Chapter 5.

9.2.1.6 Preprints

1. *An introduction to MPEG-G, the new ISO standard for genomic information representation* on bioRxiv (2018) [66]
   This pre-print introduces MPEG-G: the different parts, the underlying technology and the features the standard offers. This contribution is relevant to the entire work. It has been referenced 5 times, downloaded 2690 times (on 03/02/2020).

9.2.2 Contributions to ISO

MPEG organizes four meetings per year. The national bodies' delegates can submit, before each meeting, documents to be reviewed by the working group. The contents of the contribution is possibly approved and included in the output results of the group. We provide here a list of the contributions which submit results from our research to the working group. Figures 9.1 and 9.2 summarize our main contributions, the main outputs of each meeting, and the stage of the standard in the overall process. Figures 9.1 and 9.2 show our contributions in green and the meeting's output, to which we have contributed, in blue. The stages are indicated as requested by MPEG: the result of the vote and thus the possible official change of stage happened later than indicated.

1. *M39175 GENIFF (GENomic Information File Format), a proposal for a Secure Genomic Information Transport Layer (GITL) based on the ISO Base Media File Format* [96]
   This initial description of GENIFF introduces the idea of using the ISO BMFF. It covers the idea of a file being able to containerize any type of compression (uncompressed, ORCOM, QVZ, CARGO) but specifically CARGO compressed files. The document introduces the idea of combining multiple genomic results in one file; the idea of having an XACML document within another file; the idea of having a SAM Header translated into XML. It contains first considerations about EBI metadata and MIAME metadata and aims at being also able to contain VCF files. Furthermore, it includes first considerations about encryption. This contribution is relevant to Chapters 5, 6 and clause 7.2.

2. *M39939 Genomic Information Compression and Storage CE4: DMAG-UPC's GENIFF v.2 implementation*[97]
   This document is a summary of the work submitted to Core Experiment 4 (CE4, the core experiment dedicated to comparing the received proposals for file format). The submitted proposal is able to encapsulate the inputs of the Core Experiment. It proves the feasibility of encryption: each stream can be encrypted separately. As each stream might be a BAM file, a user can treat a specific subset of data differently. We obtain granularity by splitting a file into multiple streams (the example given is splitting a BAM into one stream per sequence). Besides, we propose the use of XACML for the definition of privacy rules. This contribution is relevant to Chapters 5, 6, 7 and 8.

3. *M39940 GENIFF (GENomic Information File Format) v2* [98]
   This document is a summary of changes applied to the GENIFF format. Most notably, and to differentiate from the ISOBMFF, we change the key's length to 8 characters, and we fix the length's field to 64 bits. We propose a draft API. The interface is differentiated in hierarchy levels and allows us to retrieve entire datasets, headers, or just fields. This version of the API has interfaces to work with records. We propose a new metadata structure that is inspired by the metadata in use at EGA. We split information in title, type, abstract, project centers, description, and samples. It introduces the idea of more complex types to represent such things as Project centers and samples. We introduce the idea of extensions: if the proposed schema is not sufficient, the elements can be extended to cover more information. We introduce the idea of encrypting individual blocks of BAM file using AES CTR, but also splitting a BAM file into multiples subfiles (each one treated as a stream and independently encrypted). We provide examples of XACML rules. This contribution is relevant to Chapters 5, 6, 7 and 8.

4. *M39941 User's Guide for DMAG-UPC's GENIFF v.2 software used for Genome information CEs*[99]
   The document describes the usage of the submitted software. It describes how to specify every file which we need to containerize and how to retrieve the data. The intermediate manifest to serve a GENIFF file is shown: with it, a server can know where to find the privacy rules and the corresponding information (which the server might have to decrypt). This contribution is relevant to Chapters 5, 7 and 8.

5. *M39943 Genomic Information Compression and Storage CE4: Cross-check of HEGIF* [100]
   This contribution analyzes the HEGIF proposal. The testing has focused on the generation of the file, its random access capabilities, the ability to transport the information and the privacy features. This contribution is relevant to Chapters 5 and 7.

6. *M39944 GENIFF (GENomic Information File Format) v2: Security and signature issues* [101]
   This document describes the challenges when dealing with signatures and the shortcomings of the current solutions. We introduce the need for the ability to sign the same information multiple times and with different keys. We also introduce the need for being able to sign only portions of the file. This contribution is relevant to clause 7.2.

7. *M39945 Towards a WD for a format on genomic information compression and storage* [102]
   This document analyzes the similarities between GenomSys' file format proposal and ours. This contribution is relevant to Chapter 5.

8. *M40494 Genomic Information Representation. Proposal for Part 3 on Protection, Application Programming Interfaces and Metadata* [103]
   This document is the first draft of Part 3. It introduces the idea of security parameters represented in XML, including information on encryption, privacy rules, and signatures. It presents the idea of having a "security tree" where a protection box can encrypt the encryption box of the layer below. It also introduces the idea of using XML encryption and signature, pointing to resources with URI. This contribution is relevant to Chapters 6 and 7.

9. *M41069 Genomic Information Representation. Proposal for metadata representation* [104]
   This document is a proposal on how extensions should work (i.e., including a URI, a description, and a value). It introduces to elaborate a proposal to specify whole ranges of required extensions (the example given is for EGA). This contribution is relevant to Chapter 6.

10. *M41070 Comments and issues in current MPEG-G Working Drafts (Parts 1 and 2)* [105]
    This document analyzes possible mistakes in both Part 1 and Part 2 and proposes solutions for them. This contribution is relevant to Chapters 5 and 4.

11. *M41072 Description of DMAG-UPC's MPEG-G generation and decoding tool* [106]
    This document is a user guide for the tool we are developing at the UPC. The guide explains how to retrieve the contents of a file, how to decode it and access its security features. Furthermore, the guide lists the implemented operations. This contribution is relevant to Chapters 5 and 7.

12. *M41073 Use case for referencing external existing genomic information from MPEG-G* [107]
    The document proposes a new use case: expanding the features of MPEG-G to be able to document existing files (e.g., providing the metadata in MPEG-G for a SAM file), in order to use the new features with old files. This contribution is relevant to Chapter 6.

13. *M41730 MPEG-G metadata: Issues and mapping with EGA profiles* [108]
    This document analyzes how to represent EGA metadata information in MPEG-G files. It details the differences in approach (EGA has an aggregation level missing in MPEG-G), which fields are missing, and which fields we have to constrain in order to ensure compatibility. The document also analyzes how a tool could work to upload and download MPEG-G files to and from the EGA repository. This contribution is relevant to Chapters 5 and 6.

14. *M41731 Genomic Information Representation Metadata* [109]
    This document presents all the modifications which have been deemed necessary for the metadata. It covers the core fields of the dataset group and dataset metadata, and the mechanism for extensions (and the schema of an extension). Additionally, we introduce the profile for EGA: which extensions are required to cover the EGA's provided schemas. This contribution is relevant to Chapter 6.

15. *M41733 Genomic Information Representation. Proposal for WD of Part 3 on Protection, Application Programming Interfaces and Metadata* [110]
    The document shows how the work we have carried out is integrated into Part 3. This contribution is relevant to Chapters 6, 7 and 8.

16. *M42106 Proposal for CD of MPEG-G Part 3: Genomic Information Metadata and Application Programming Interfaces (APIs)* [111]
    This document is our proposal for the Candidate Draft (CD). In it, we separate the API into two sets: core and extended. This contribution is relevant to Chapter 8.

17. *M42588 Proposal for a Study of ISO/IEC CD 23092-3 Genomic Information Metadata and APIs* [112]
    The document corrects minor terminology and formatting issues in Part 3. This contribution is relevant to Chapters 6, 7 and 8.

18. *M42590 Comments on Draft Text of ISO/IEC DIS 23092-1 Transport and Storage of Genomic Information* [113]
    The document corrects minor terminology and formatting issues in Part 1. This contribution is relevant to Chapter 5.

19. *M43542 MPEG-G Part 3: Proposed Draft DIS* [114]
    This document is our proposal for the Draft International Standard (DIS), where we include our proposed algorithms to have default access rules, and not revealing confidential information. This contribution is relevant to clause 8.5.

20. *M43543 MPEG-G Part 3 ballot: Pre-analysis of ballot comments* [115]
    This document analyzes the different votes received concerning Part 3. We indicate in each case whether we have included the content of the vote into our proposed draft and justify our choice. This contribution is relevant to Chapters 6, 7 and 8.

21. *M43545 Side-channel attack on MPEG-G in Descriptor Stream Contiguous mode* [116]
    This document proves that there is a way to attack and recover data from an MPEG-G file if it is stored in DSC mode. The attack proves that if the pos descriptor is not encrypted, then the position of the SNPs can be inferred. This contribution is relevant to clause 7.2.5.

22. *M44845 Text proposal for ISO/IEC DIS 23092-3 Genomic Information Metadata and APIs* [117]
    This documents is our proposal for the Final Draft International Standard (FDIS), which now includes metadata extensions for GDC, an encryption mechanism that is not explicitly based on access units, and mechanisms to encrypt sequence and label information. This contribution is relevant to Chapters 6 and 7.

23. *M45601 Comments on ISO/IEC DIS 23092-1 and ISO/IEC DIS 23092-2* [61]
    The document analyzes different issues in the MPEG-G specification, especially the following:

    - issue of the need for classes
    - issues in the decoding process of mmtype when multiple mutations are given for the same position
    - issues in the correct decoding of the cigar and the sequence when mutations are provided through the rftp and rftt streams
    - issues in the identification of the primary read and the computation of the number of alignments.

    This contribution is relevant to Chapters 4 and 5.

24. *M47222 Draft text proposal for ISO/IEC FDIS 23092-3 Genomic Information Metadata and APIs* [118]
    We propose an FDIS draft which includes our latest modifications to the security mechanisms: the encryption parameters are no longer given at the access unit level, but at the dataset level for the different regions. In order to decrypt an access unit, the encryption parameters for the relevant regions must be checked. This contribution is relevant to Chapters 6 and 7.

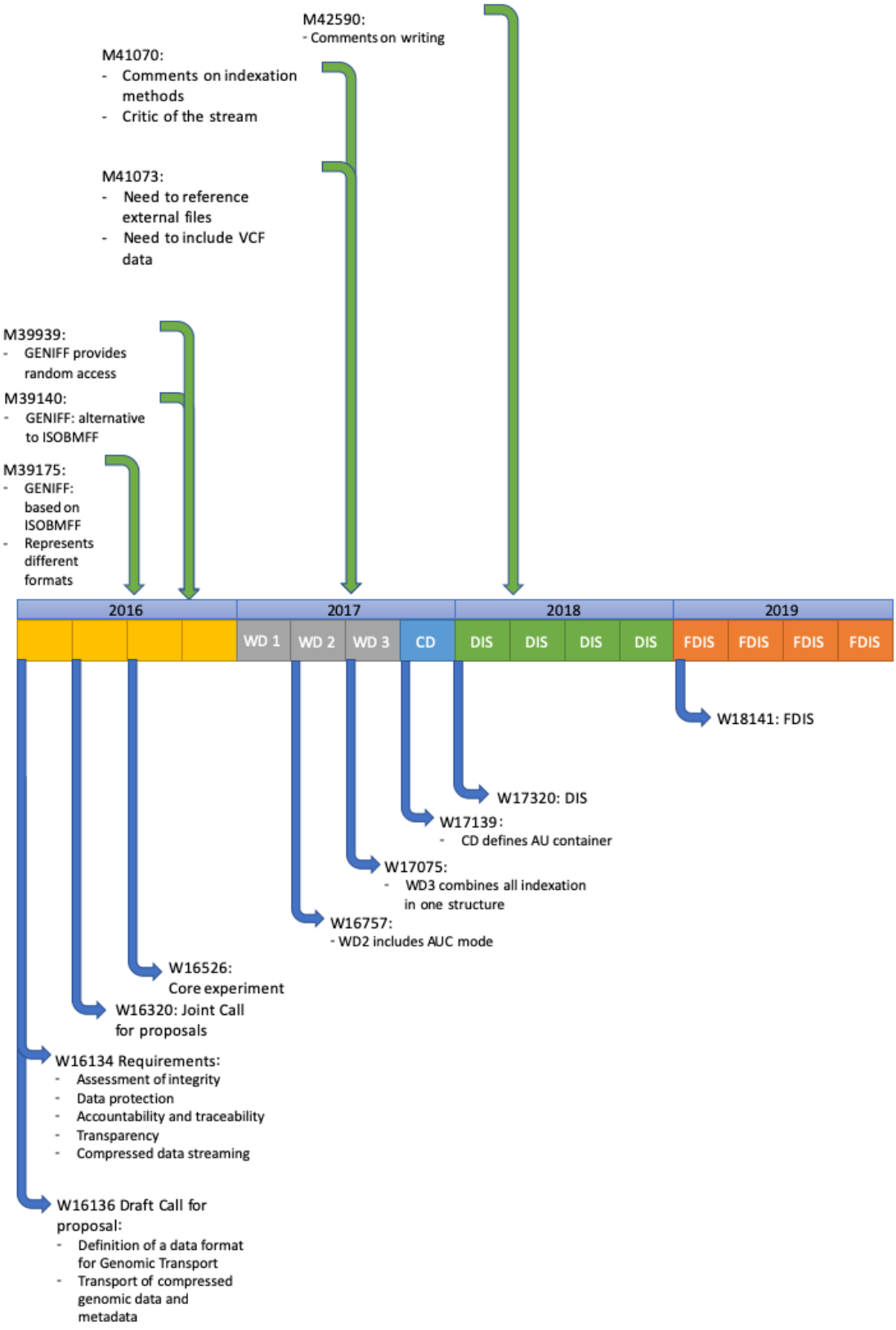


Figure 9.1: Evolution of MPEG-G Part 1

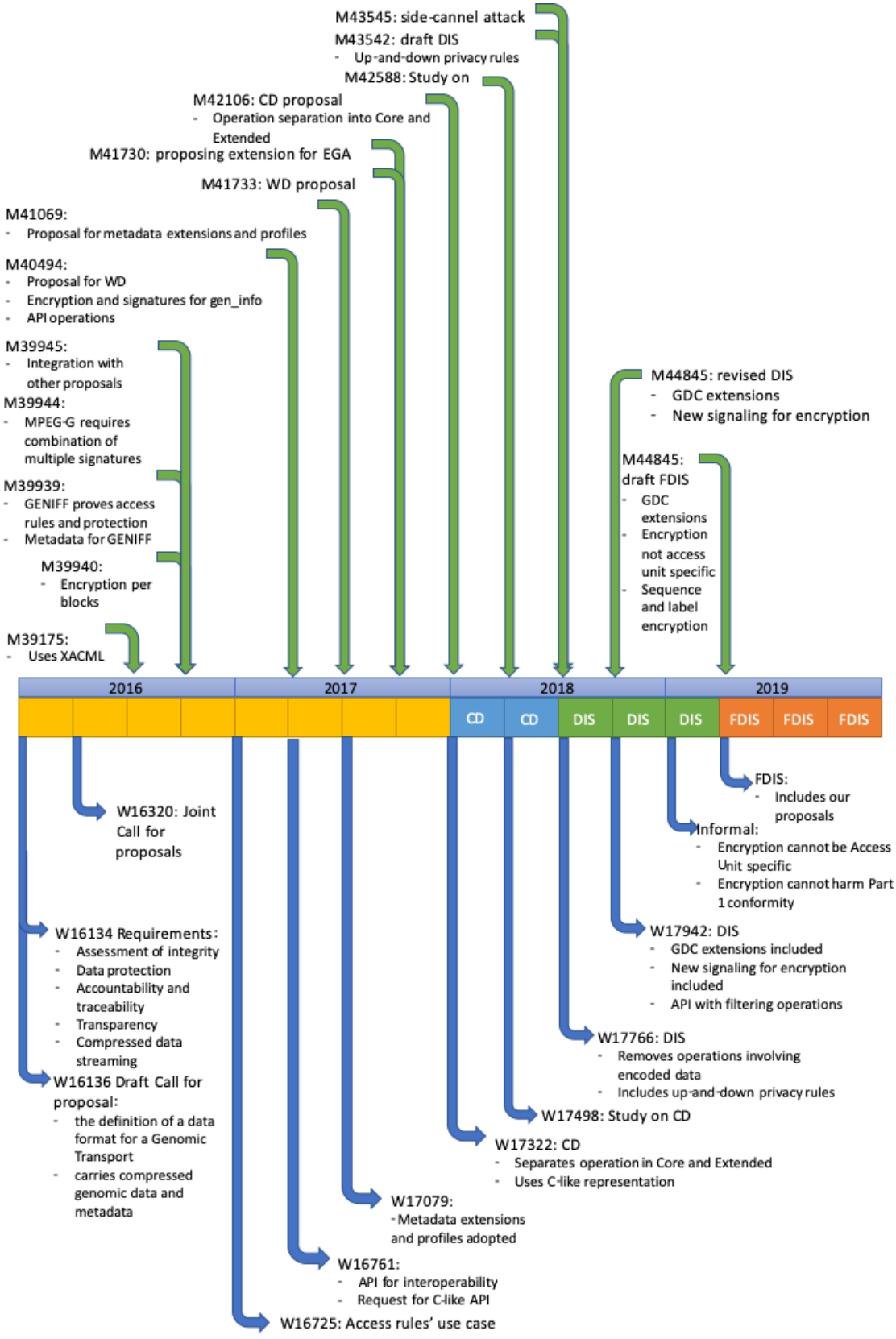


Figure 9.2: Evolution of MPEG-G Part 3

Chapter 10

Future work

We can view the future work as two distinct sets of tasks. On the one hand, we have the objectives related to the current version of the standard. On the other hand, we have the aims associated to the future of the standard.

10.1 The current version of the standard

We have created a specification that can adapt to multiple configurations. As in other parts of the standard, e.g. the ability to encode the information either with a reference or a computed reference, we have seen that we have a high potential of adaptiveness regarding the metadata representation, the security parameters, and the privacy rules.

10.1.1 Improving user experience for security parameters

Tools must offer a more accessible experience, both to represent the intentions of the user and to understand the content of the file. For example, we have defined how to encrypt according to positions on the reference genome. Yet, for the average user, these positions will not be representative. New tools have to provide a user experience such that the patient can choose between an opt-in or opt-out approach (i.e., only specific regions are more protected or less protected, respectively). The decisions regarding protection regions also affect the keys used (i.e., the number of keys used and their association with different parts of the genome), how to transport them (i.e., using the key derivation and wrapping solutions of the standard or relying on external channels) and how to represent them as privacy rules. Furthermore, these decisions would probably have to be integrated within existing repositories of genomic data.

We have seen that there are databases of genomic regions related to specific diseases (e.g. ensembl.org [9]). We could use this information to extract the first list of regions to protect. For example, let us imagine that we see the following information in the database:

- Disease 1 is related to the region on chromosome 1 going from 1'000'000 to 1'060'000.
- Disease 2 is related to the region on chromosome 1 going from 1'030'000 to 1'060'000.
- Disease 3 is related to the region on chromosome 1 going from 1'060'000 to 1'090'000.

We could create encryption regions corresponding to:

- 1'000'000 to 1'030'000
- 1'030'000 to 1'060'000
- 1'060'000 to 1'090'000.

Each one of these regions would correspond to a new and unique key. If the patient decides to share information regarding one or more of these diseases, the configuration provides the corresponding keys by wrapping these pieces of information with a secret shared with the recipients.

This solution provides an opt-out approach: everything is inaccessible unless the patient indicates otherwise. This approach can also adapt to future configurations, as the tool has created the units according to ranges meant to accommodate future variations. Furthermore, this approach is easily adaptable to entire groupings of diseases (e.g., all cancers). Although this seems to address the initial objective of simplifying the user experience regarding the generation of the protection configuration, there are many open questions.

For example, the user must be made aware of the risks related to linkage disequilibrium: if a given mutation is usually inherited in combination with another, knowing one of them helps to infer the other.

The proposed solution might also have an impact on the compression efficiency, as there are too many output units. Therefore, the user would have to be made aware of the possible drawbacks. The user might, for example, decide to opt for a coarser-grained output if this improves the compression even if it harms the granularity of the protection.

Proposed tools should also handle the creation of the privacy rules and the key distribution based upon the user settings. Tools can offer these features without any modification to MPEG-G. The difficulty would not consist in how to provide the configuration but rather in how to translate the user's requests into the corresponding settings.

10.1.2 Improving user experience for metadata

We have provided a mechanism for the metadata definition and inclusion inside MPEG-G files. We have done so by defining a core metadata set, which we can enhance through the use of extensions. The definition of a profile permits us to indicate which set of extensions must be present to meet the requirements. We have developed profiles both for EGA and GDC, and we expect other profiles to be defined.

There is yet no tool that represents the metadata information engagingly. The main challenge with the development of this tool will be its adaptability to new and unknown extensions. Although the standard defines a way to convey the documentation related to the extension, this documentation is normally in human-readable form, which hinders the automatic adaptation of its rendering in a user interface.

10.1.3 Deploying MPEG-G

The main challenge with the current specification of MPEG-G will be to deploy it. We have contributed to some part of its deployment, but more work is required. Current libraries should support this new standard, both to read data from it and to write data to it.

We can expect multiple challenges on this front, related to the features that MPEG-G introduces. For example, current libraries will not expect files which contain multiple results within: the equivalent of a SAM file is an MPEG-G's dataset, not an MPEG-G file. Similarly, when writing an MPEG-G file multiple configurations are expected: regarding the threshold used while creating records, the shape of the access units or the encryption strategies.

In order to obtain the most out of MPEG-G, this new specification cannot be implemented as a SAM replacement in existing libraries. Otherwise, many features will be lost.

Genomic archives such as EGA or GDC share several characteristics with MPEG-G, for example, the combination of multiple results in a larger structure, the use of metadata, or even the use of protection mechanisms. In order to foster the adoption of MPEG-G, it will be required to define mappings between all MPEG-G elements and their counterparts in the archival structure. We have contributed to this task in the form of a mapping between the MPEG-G metadata and the metadata used at EGA, but there are still other aspects to deal with ; for example, how to translate the privacy rules from ADA-M to XACML or from XACML to ADA-M.

The deployment of MPEG-G will also require to have it included in existing libraries and tools. We have contributed to such initiatives (jMPEGG [63] and Genie[62]): these tools offer ways to compress from existing formats to MPEG-G, or to decompress MPEG-G files into existing formats. However, in order to further foster the adoption of MPEG-G, the new standard needs to be included in existing libraries and tools such as htsjdk [119]. This integration will require to consider how to interact with MPEG-G in the context of libraries originally intended for FAST-Q, SAM or similar file formats. One of the main issues is that within an MPEG-G file there can be multiple dataset groups, with multiple datasets each. Each one of these datasets is conceptually equivalent to a FAST-Q or SAM-like file. Therefore, an integration should handle an MPEG-G dataset as the equivalent of an existing format, rather than considering the entire MPEG-G file as an equivalent, possibly having to modify the paradigm of simply opening a file to read it.

10.2 Future versions of the standard

In the preceding chapters, we have identified multiple points that might be of concern with the current standard. We have seen, for example, that the file format might require new layers to accommodate the structure used in the requirements. We have also seen that the finally adopted solution for protection forces the decision in edge cases. The return of experience will provide evidence whether these concerns are justified.

10.2.1 File lifecycle

The current file format does not allow to modify the content of a file easily. In the context of the security mechanism, this is especially relevant. For example, if new access rules are generated for the file or a new recipient has to be granted access to specific regions, the entire file would have to be rewritten in order to update the content of the protection element. The return of experience will determine whether this is an aspect that needs correction.

For example, we can imagine two situations for the sources of MPEG-G encoded data. We can imagine the case where, upon reception of a request for a given file, the original file returned is the one submitted by the patient. This approach simplifies the logic of the repository, but it would require that the copy stored at the repository be updated to reflect the newest user's preferences. This situation would benefit from a new file format allowing modification of the content. Such a file format could rely on buffers of unused bytes. These unused byte buffers would store the new settings: instead of rewriting a new file with a new size, we assign bytes originally from the buffer to the structures holding the settings, thus maintaining the file size.

We could also imagine the case where the repository generates the MPEG-G file on the fly. In such a situation, the repository would store the file in some intermediate state, alongside the privacy settings. As the repository generates a new file on each occasion, there is no need to update the content of the file.

We have seen that GA4GH is working on multiple activities relevant to different aspects of MPEG-G: for example, encryption strategies, access rule specifications and API to access genomic content. Concepts can be contributed in both directions between GA4GH and MPEG-G, in order to improve the different proposals and ensure compatibility and interoperability.

10.2.2 Future of protection

The MPEG working group will have to update the standard according to the evolution of the different specifications used (e.g., for key derivation and key wrapping) and the security environment (e.g., the discovery of new attack vectors).

New technology might also be of interest to the development of the MPEG-G standard. Currently, there is no mechanism to force the user to comply with the privacy rules. We can improve the situation by encrypting the information and ensuring that the key management is consistent with the privacy rules. For example, if we want to give access to region A to only researchers working on cancer, we can encrypt region A with a key only known to researchers working on cancer. In this configuration, the encryption acts as an enforcer of privacy rules.

With new technologies such as Ciphertext-Policy Attribute-Based Encryption (CP-ABE [91]), we could combine both the encryption and the privacy rules. Only those recipients who have a key matching the policy would be able to access the encrypted data. We could also use CP-ABE as the groundstone for a new wrapping mechanism. This approach would ensure that the user addresses the policy before accessing the data.

However, both in the current approach and in a future approach using CP-ABE, we are unable to protect the information against a user accredited for one use case, but executing another one. For example, a researcher could be interested in both cancer and cosmetics and have the keys corresponding to the two use cases. Let us imagine a file open for cancer research but closed for cosmetics research. No mechanism can ensure that the user who gets access to the data using the 'cancer key' does not use it for cosmetics research.

For MPEG-G to remain a secure and up-to-date solution, further research needs to be conducted, both to include new features and to secure the standard against newly discovered threats.

Acronyms

ADA-M Automatable Discovery and Access Matrix. 20

AES Advanced Encryption Standard. 53

API Application Programming Interface. 81

AUC Access Unit Contiguous. 38

BAI Binary Alignment Index. 15

BAM Binary Alignment Map. 10

BGZF Blocked GNU Zip Format. 15

CABAC Context-Adaptive Binary Arithmetic Coding. 32

CARGO Compressed ARchiving for GenOmics. 15

CD Candidate Draft. 29

CIGAR Compact Idiosyncratic Gapped Alignment Report. 10

CP-ABE Ciphertext Policy - Attribute-Based Encryption. 98

CRAM Compressed Reference-oriented Alignment Map. 15

CTR Counter. 53

DIS Draft International Standard. 29

DNA Deoxyribonucleic acid. 7

DSC Descriptor Stream Contiguous. 38

EGA European Genome Archive. 25

FDIS Final Draft International Standard. 29

GA4GH Global Alliance For Genome and Health. 19

GCM Galois/Counter Mode. 53

GDC Genomic Data Commons. 27

GWAS Genome-Wide Association Study. 19

GZIP GNU Zip. 15

IETF Internet Engineering Task Force. 54

ILP Integer Linear Program. 25

IS International Standard. 29

ISO International Organization for Standardization. 29

ISO BMFF ISO Base Media File Format. 37

IV Initialization Vector. 53

KLV Key Length Value. 37

MPEG Moving Picture Expert Group. 29

PBKDF2 Password-Based Key Derivation Function 2. 54

PCA Primary Component Analysis. 19

PGP Pretty Good Privacy. 20

RFC Release For Comment. 54

SAM Sequence Alignment/Map. 10

SECRAM Selective retrieval on Encrypted and Compressed Reference-oriented Alignment Map. 19

SNP Single Nucleotide Permutations. 16

UUID Universally Unique Identifier. 27

VCF Variant Call Format. 25

WD Working Draft. 29

XML eXtensible Markup Language. 25

Bibliography

[1] International Organization for Standardization. International Organization for Standardization. https://www.iso.org/home.html.

[2] ISO/IEC JTC 1/SC 29/WG 11. MPEG ISO/IEC JTC 1/SC 29/WG 11. https://mpeg.chiariglione.org.

[3] International Organization for Standardization. ISO/TC 276 Biotechnology. https://www.iso.org/committee/4514241.html.

[4] David L. et al. Altshuler. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, oct 2010.

[5] David M. et al. Altshuler. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, nov 2012.

[6] Adam et al. Auton. A global reference for human genetic variation. *Nature*, 526(7571):68–74, oct 2015.

[7] Valentí Moncunill, Santi Gonzalez, Sílvia Beà, Lise O. Andrieux, Itziar Salaverria, Cristina Royo, Laura Martinez, Montserrat Puiggròs, Maia Segura-Wang, Adrian M. Stütz, Alba Navarro, Romina Royo, Josep L. Gelpí, Ivo G. Gut, Carlos López-Otín, Modesto Orozco, Jan O. Korbel, Elias Campo, Xose S. Puente, and David Torrents. Comprehensive characterization of complex structural variations in cancer by directly comparing genome sequence reads. *Nature Biotechnology*, 32(11):1106–1112, nov 2014.

[8] Elizabeth Pennisi. GENETICS: Working the (Gene Count) Numbers: Finally, a Firm Answer? *Science*, 316(5828):1113a–1113a, may 2007.

[9] Andrew Yates, Wasiu Akanni, M. Ridwan Amode, Daniel Barrell, Konstantinos Billis, Denise Carvalho-Silva, Carla Cummins, Peter Clapham, Stephen Fitzgerald, Laurent Gil, Carlos García Girón, Leo Gordon, Thibaut Hourlier, Sarah E. Hunt, Sophie H. Janacek, Nathan Johnson, Thomas Juettemann, Stephen Keenan, Ilias Lavidas, Fergal J. Martin, Thomas Maurel, William McLaren, Daniel N. Murphy, Rishi Nag, Michael Nuhn, Anne Parker, Mateus Patricio, Miguel Pignatelli, Matthew Rahtz, Harpreet Singh Riat, Daniel Sheppard, Kieron Taylor, Anja Thormann, Alessandro Vullo, Steven P. Wilder, Amonida Zadissa, Ewan Birney, Jennifer Harrow, Matthieu Muffato, Emily Perry, Magali Ruffier, Giulietta Spudich, Stephen J. Trevanion, Fiona Cunningham, Bronwen L. Aken, Daniel R. Zerbino, and Paul Flicek. Ensembl 2016. *Nucleic Acids Research*, 44(D1):D710–D716, jan 2016.

[10] J. Craig et al. Venter. The Sequence of the Human Genome. *Science*, 291(5507):1304–1351, feb 2001.

[11] Eric S. et al. Lander. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, feb 2001.

[12] David R. et al.. Bentley. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456(7218):53–59, nov 2008.

[13] Peter J A Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6):1767–1771, dec 2009.

[14] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, aug 2009.

[15] Michael G. Ross, Carsten Russ, Maura Costello, Andrew Hollinger, Niall J. Lennon, Ryan Hegarty, Chad Nusbaum, and David B. Jaffe. Characterizing and measuring bias in sequence data. *Genome Biology*, 14(5):R51, 2013.

[16] Travis C. Glenn. Field guide to next-generation DNA sequencers. *Molecular Ecology Resources*, 11(5):759–769, sep 2011.

[17] Peter L. Deutsch. RFC 1952: GZIP file format specification version 4.3. Technical report, Internet Engineering Task Force, 1996.

[18] Łukasz Roguski and Paolo Ribeca. CARGO: effective format-free compressed storage of genomic information. *Nucleic Acids Research*, 44(12):e114–e114, jul 2016.

[19] Markus Hsi Yang Fritz, Rasko Leinonen, Guy Cochrane, Ewan Birney, M. Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane, and Ewan Birney. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research*, 21(5):734–740, may 2011.

[20] James K. Bonfield. The Scramble conversion tool. *Bioinformatics*, 30(19):2818–2819, jun 2014.

[21] Global Alliance for Genomics & Health. Beacon Network. https://beacon-network.org/#/.

[22] Dale R Nyholt, Chang-En Yu, and Peter M Visscher. On Jim Watson's APOE status: genetic information is hard to hide. *European journal of human genetics : EJHG*, 17(2):147–149, mar 2009.

[23] Sahel Shariati Samani, Zhicong Huang, Erman Ayday, Mark Elliot, Jacques Fellay, Jean-Pierre Hubaux, and Zoltan Kutalik. Quantifying Genomic Privacy via Inference Attack with High-Order SNV Correlations. In *2015 IEEE Security and Privacy Workshops*, pages 32–40. IEEE, may 2015.

[24] Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux, and Amalio Telenti. Reconciling Utility with Privacy in Genomics. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society - WPES '14*, pages 11–20, New York, New York, USA, 2014. ACM Press.

[25] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS genetics*, 4(8):e1000167, aug 2008.

[26] David Clayton. On inferring presence of an individual in a mixture: a Bayesian approach. *Biostatistics (Oxford, England)*, 11(4):661–73, oct 2010.

[27] Kevin B Jacobs, Meredith Yeager, Sholom Wacholder, David Craig, Peter Kraft, David J Hunter, Justin Paschal, Teri A Manolio, Margaret Tucker, Robert N Hoover, Gilles D Thomas, Stephen J Chanock, and Nilanjan Chatterjee. A new statistic and its power to infer membership in a genome-wide association study using genotype frequencies. *Nature Genetics*, 41(11):1253–1257, nov 2009.

[28] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers. In *Proceedings of the 16th ACM conference on Computer and communications security - CCS '09*, page 534, New York, New York, USA, nov 2009. ACM Press.

[29] Rosemary Braun, William Rowe, Carl Schaefer, Jinghui Zhang, and Kenneth Buetow. Needles in the haystack: identifying individuals present in pooled genomic data. *PLoS genetics*, 5(10):e1000668, oct 2009.

[30] Suyash S. Shringarpure and Carlos D. Bustamante. Privacy Risks from Genomic Data-Sharing Beacons. *The American Journal of Human Genetics*, 97(5):631–46, oct 2015.

[31] Cynthia Dwork. Differential Privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[32] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, San Diego, CA, 2014. USENIX Association.

[33] Florian Tramèr, Zhicong Huang, Jean-Pierre Hubaux, and Erman Ayday. Differential Privacy with Bounded Priors. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, pages 1286–1297, New York, New York, USA, oct 2015. ACM Press.

[34] Caroline Uhler, Aleksandra B. Slavkovic, and Stephen E. Fienberg. Privacy-Preserving Data Sharing for Genome-Wide Association Studies, may 2012.

[35] Fei Yu, Stephen E. Fienberg, Aleksandra B. Slavković, and Caroline Uhler. Scalable privacy-preserving data sharing methodology for genome-wide association studies. *Journal of Biomedical Informatics*, 50:133–141, 2014.

[36] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11*, page 85, New York, New York, USA, 2011. ACM Press.

[37] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Secure pattern matching using somewhat homomorphic encryption. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 65–76, nov 2013.

[38] Emiliano De Cristofaro, Sky Faber, and Gene Tsudik. Secure genomic testing with size- and position-hiding private substring matching. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 107–117, New York, New York, USA, 2013. ACM Press.

[39] Ludovic Barman, Mohammed-Taha Elgraini, Jean Louis Raisaro, Jean-Pierre Hubaux, and Erman Ayday. Privacy Threats and Practical Solutions for Genetic Risk Tests. In *2015 IEEE Security and Privacy Workshops*, pages 27–31. IEEE, may 2015.

[40] Wenjie Lu, Yoshiji Yamada, and Jun Sakuma. Efficient Secure Outsourcing of Genome-Wide Association Studies. In *2015 IEEE Security and Privacy Workshops*, pages 3–6. IEEE, may 2015.

[41] Charlotte Bonte, Eleftheria Makri, Amin Ardeshirdavani, Jaak Simm, Yves Moreau, and Frederik Vercauteren. Towards practical privacy-preserving genome-wide association study. *BMC Bioinformatics*, 19(1):537, dec 2018.

[42] Hyunghoon Cho, David J. Wu, and Bonnie Berger. Secure genome-wide association analysis using multiparty computation. *Nature Biotechnology*, 2018.

[43] Markus Hsi Yang Fritz, Rasko Leinonen, Guy Cochrane, and Ewan Birney. Efficient storage of high throughput DNA sequencing data using reference-based compression. *Genome Research*, 2011.

[44] Morteza Hosseini, Diogo Pratas, and Armando J. Pinho. Cryfa: a secure encryption tool for genomic data. *Bioinformatics*, 35(1):146–148, jan 2019.

[45] GA4GH. GA4GH File Encryption Standard. Technical report, Global Alliance for Genomics and Health, 2019.

[46] Zhicong Huang, Erman Ayday, Huang Lin, Raeka S. Aiyar, Adam Molyneaux, Zhenyu Xu, Jacques Fellay, Lars M. Steinmetz, and Jean-Pierre Hubaux. A privacy-preserving solution for compressed storage and selective retrieval of genomic data. *Genome Research*, 26(12):1687–1696, oct 2016.

[47] Jon Callas, Lutz Donneerhacke, Hal Finney, David Shaw, and Rodney Thayer. RFC4880: OpenPGP Standard. Technical report, Internet Engineering Task Force, 2007.

[48] J. Patrick Woolley, Emily Kirby, Josh Leslie, Francis Jeanson, Moran N. Cabili, Gregory Rushton, James G. Hazard, Vagelis Ladas, Colin D. Veal, Spencer J. Gibson, Anne-Marie Tassé, Stephanie O. M. Dyke, Clara Gaff, Adrian Thorogood, Bartha Maria Knoppers, John Wilbanks, and Anthony J. Brookes. Responsible sharing of biomedical data and biospecimens via the "Automatable Discovery and Access Matrix" (ADA-M). *npj Genomic Medicine*, 3(1):17, dec 2018.

[49] Aggeliki Giakoumaki, Sotiris Pavlopoulos, and Dimitris Koutsouris. Multiple Image Watermarking Applied to Health Information Management. *IEEE Transactions on Information Technology in Biomedicine*, 10(4):722–732, oct 2006.

[50] Niels Provos and Peter Honeyman. Hide and seek: an introduction to steganography. *IEEE Security & Privacy*, 1(3):32–44, may 2003.

[51] Ayman Ibaida, Ibrahim Khalil, and Ron van Schyndel. A low complexity high capacity ECG signal watermark for wearable sensor-net health monitoring system. In *2011 Computing in Cardiology*, pages 393–396, sep 2011.

[52] Digvijay Singh Chauhan, Amit Kumar Singh, Basant Kumar, and J P Saini. Quantization based multiple medical information watermarking for secure e-health. *Multimedia Tools and Applications*, 78(4):3911–3923, feb 2019.

[53] Hamidreza Zarrabi, Mohsen Hajabdollahi, S M Reza Soroushmehr, Nader Karimi, Shadrokh Samavi, and Kayvan Najarian. Reversible Image Watermarking for Health Informatics Systems Using Distortion Compensation in Wavelet Domain. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 798–801. IEEE, jul 2018.

[54] Dominik Heider and Angelika Barnekow. DNA-based watermarks using the DNA-Crypt algorithm. *BMC Bioinformatics*, 8(1):176, dec 2007.

[55] Dominik Heider and Angelika Barnekow. DNA watermarks: A proof of concept. *BMC Molecular Biology*, 9(1):40, 2008.

[56] Arif Yilmaz and Erman Ayday. Collusion-Secure Watermarking for Sequential Data, aug 2017.

[57] Shudi Gao, Michael Sperberg-McQueen, and Henry S. Thompson. *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. W3C, 2012.

[58] Carolyn Hutter and Jean Claude Zenklusen. The Cancer Genome Atlas: Creating Lasting Value beyond Its Data. *Cell*, 173(2):283–285, apr 2018.

[59] Genomic Standards Consortium. MIxS Extensions and Profiles, 2019.

[60] ISO/IEC JTC 1/SC 29/WG 11. MPEG-G, ISO/IEC 23092 Genomic Information Representation, 2019.

[61] Daniel Naro, Jan Voges, Tom Paridaens, Dmitry Repchevsky, Jaime Delgado, Paolo Ribeca, Idoia Ochoa, and Mikel Hernaez. M45601 Comments on ISO/IEC DIS 23092-1 and ISO/IEC DIS 23092-2. Technical report, ISO/IEC JTC1 SC29 WG11, Marrakech, 2019. http://dmag.ac.upc.edu/downloads/mpegg/m45601_Comments_on_ISO_IEC_DIS_23092_1_and_ISO_IEC_DIS_23092_2.pdf.

[62] Brian E. Bliss, Joshua M. Allen, Saurabh Baheti, Matthew A. Bockol, Shubham Chandak, Jaime Delgado, Jan Fostier, Josep Lluís Gelpí, Steven N. Hart, Mikel Hernaez, Matthew E. Hudson, Michael T. Kalmbach, Eric W. Klee, Liudmila S. Mainzer, Fabian Müntefering, Daniel Naro, Idoia Ochoa, Jörn Ostermann, Tom Paridaens, Christian A. Ross, Jan Voges, Eric D. Wieben, Mingyu Yang, Tsachy Weissman, and Mathieu Wiepert. Genie: an MPEG-G Conformant Software to Compress Genomic Data. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis*, Denver, 2019.

[63] Dmitry Repchevsky, Daniel Naro, Romina Royo, Silvia Llorente, Jaime Delgado, and Josep Lluís Gelpí. jmpegg 1.0 – the pure java implementation of the MPEG-G ISO/IEC 23092 standard. In *ELIXIR All Hands*, Lisboa, 2019.

[64] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, jul 2003.

[65] Jaime Delgado, Silvia Llorente, Daniel Naro, Luis Torres, and Francesc Tarres. Representing compressed and secure genomic information. In *International Conference of the European Federation for Medical Informatics*, apr 2018.

[66] Claudio Alberti, Tom Paridaens, Jan Voges, Daniel Naro, Junaid J Ahmad, Massimo Ravasi, Daniele Renzi, Giorgio Zoia, Paolo Ribeca, Idoia Ochoa, Marco Mattavelli, Jaime Delgado, and Mikel Hernaez. An introduction to MPEG-G, the new ISO standard for genomic information representation. *bioRxiv*, jan 2018.

[67] ISO/IEC JTC 1/SC 29. *ISO - ISO/IEC 14496-12:2015 - Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*. ISO, 5 edition, 2015.

[68] Jaime Delgado, Daniel Naro, Silvia Llorente, Josep Lluís Gelpí, and Romina Royo. Metadata to Describe Genomic Information. In Adrien Ugon, Daniel Karlsson, Gunnar O. Klein, and Anne Moen, editors, *Studies in health technology and informatics*, volume 247, pages 621–625, Gothenburg, 2018. IOS Press.

[69] Daniel Naro, Jaime Delgado, and Silvia Llorente. Side-channel attack on a partially encrypted MPEG-G file. *Submitted to Multimedia Tools and Applications*.

[70] Daniel Naro, Jaime Delgado, and Silvia Llorente. Reversible fingerprinting for genomic information. *Multimedia Tools and Applications*, 2020.

[71] Takeshi Imamura, Blair Dillaway, Ed Simon, Kelvin Yiu, and Magnus Nyström. *XML Encryption Syntax and Processing Version 1.1.* W3C, 2013.

[72] Donald Eastelake, Joseph Reagle, David Solo, Frederick Hirsch, Magnus Nyström, Thomas Roessler, Kelvin Yiu, Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. *XML Signature Syntax and Processing Version 1.1.* W3C, 2013.

[73] Kathleen Moriarty, Burt Kalisky, and Andreas Rusch. RFC 2898 PKCS #5: Password-Based Cryptography Specification Version 2.1. Technical report, Internet Engineering Task Force, 2017.

[74] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Journal of Machine Learning Research*, 2011.

[75] Han Fang, Yiyang Wu, Giuseppe Narzisi, Jason A ORawe, Laura T Jimenez Barrón, Julie Rosenbaum, Michael Ronemus, Ivan Iossifov, Michael C. Schatz, and Gholson J. Lyon. Reducing INDEL calling errors in whole genome and exome sequencing data. *Genome Medicine*, 6(10):89, dec 2014.

[76] Michael J. Clark, Rui Rong Chen, Hugo Y K Lam, Konrad J. Karczewski, Rui Rong Chen, Ghia Euskirchen, Atul J. Butte, and Michael Snyder. Performance comparison of exome DNA sequencing technologies. *Nature Biotechnology*, 29(10):908–914, oct 2011.

[77] Alison M. Meynert, Louise S. Bicknell, Matthew E. Hurles, Andrew P. Jackson, and Martin S. Taylor. Quantifying single nucleotide variant detection sensitivity in exome sequencing. *BMC Bioinformatics*, 14(1):195, dec 2013.

[78] Genohub. Recommended Coverage and Read Depth for NGS Applications. https://genohub.com/recommended-sequencing-coverage-by-application/, 2019.

[79] Michael L Metzker. Sequencing technologies — the next generation. *Nature Reviews Genetics*, 11:31, dec 2009.

[80] Kai Song, Li Li, and Guofan Zhang. Coverage recommendation for genotyping analysis of highly heterologous species using next-generation sequencing technology. *Scientific Reports*, 6(1):35736, dec 2016.

[81] European Nucleotide Archive. Run: ERR317482 Illumina HiSeq 2000 paired end sequencing, 2019.

[82] Google. Run: ERR194146, "Utah residents (CEPH) with Northern and Western European ancestry", 2019.

[83] Namita Agarwal, Amit Kumar Singh, and Pradeep Kumar Singh. Survey of robust and imperceptible watermarking. *Multimedia Tools and Applications*, 78(7):8603–8633, apr 2019.

[84] Jaime Delgado. Privacy, metadata and APIs in compressed genomic information: The MPEG-G case. In *GA4GH & MPEG Genome Compression Workshop*, oct 2018.

[85] Jaime Delgado, Silvia Llorente, and Daniel Naro. Adding security and privacy to genomic information representation. In Amnon Shabo, Inge Madsen, Hans-Ulrich Prokosch, Kristiina Häyrinen, Klaus-Hendrik Wolf, Fernando Martin-Sanchez, Matthias Löbe, and Thomas M. Deserno, editors, *Studies in Health Technology and Informatics*, volume 258, pages 75–79, Hanover, apr 2019. IOS Press.

[86] Jaime Delgado, Silvia Llorente, and Daniel Naro. Protecting Privacy of Genomic Information. In *Studies in health technology and informatics*, volume 235, pages 318–322, apr 2017.

[87] Daniel Naro, Jaime Delgado, and Silvia Llorente. Improving queries on genomic information by segregation. *Draft for Patterns.*

[88] OASIS Standard. eXtensible Access Control Markup Language (XACML) Version 3.0. Technical report, OASIS Standard, 2013.

[89] OASIS standard. OASIS Open standards. Open source., 2019.

[90] Elisa Bertino, Amani Abu Jabal, Seraphin Calo, Dinesh Verma, and Christopher Williams. The challenge of access control policies quality. *Journal of Data and Information Quality*, 10(2):1–6, sep 2018.

[91] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334. IEEE, may 2007.

[92] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits. *SIAM Journal on Computing*, 45(3):882–929, jan 2016.

[93] Jaime Delgado, Silvia Llorente, and Daniel Naro. Providing FAIR principles while preserving patient's privacy. *Submitted to Methods of Information in Medicine.*

[94] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C. t Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan Van Der Lei, Erik Van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. Comment: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 2016.

[95] Jaime Delgado, Silvia Llorente, and Daniel Naro. xACS: Management of privacy in genomic and medical information. In *6th International Workshop on Genome Privacy and Security (GenoPri'19)*, Boston, Massas, oct 2019.

[96] Jaime Delgado, Silvia Llorente, Daniel Naro, Sara Rodríguez-Cubillas, Łukasz Roguski, Josep Lluís Gelpí, Dmitry Repchevsky, Leonor Frías, Oscar Flores, Jordi Portell, Francesc Julbe, Paolo Ribeca, Mikel Hernaez, and Idoia Ochoa. M39175 GENIFF (GENomic Information File Format), a proposal for a Secure Genomic Information Transport Layer (GITL) based on the ISO Base Media File Format. Technical report, ISO/IEC JTC1 SC29 WG11, Chengdu, China, 2016. http://dmag.ac.upc.edu/downloads/mpegg/m39175_GENIFF_Response_genome_transport.pdf.

[97] Jaime Delgado, Silvia Llorente, Daniel Naro, Lukasz Roguski, and Dmitry Repchevsky. M39939 Genomic Information Compression and Storage CE4: DMAG-UPC's GENIFF v.2 implementation. Technical report, ISO/IEC JTC1 SC29 WG11, Geneva, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m39939_CE4_DMAG_UPC_development.pdf.

[98] Jaime Delgado, Silvia Llorente, Daniel Naro, Łukasz Roguski, and Dmitry Repchevsky. M39940 GENIFF (GENomic Information File Format) v2. Technical report, ISO/IEC JTC1 SC29 WG11, Geneva, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m39940_Geniffv2.pdf.

[99] Jaime Delgado, Silvia Llorente, Daniel Naro, Łukasz Roguski, and Dmitry Repchevsky. M39941 User's Guide for DMAG-UPC's GENIFF v.2 software used for Genome information CEs. Technical report, ISO/IEC JTC1 SC29 WG11, Geneva, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m39941_CE4_DMAG_UPC_User_Guide.pdf.

[100] Daniel Naro, Jaime Delgado, and Silvia Llorente. M39943 Genomic Information Compression and Storage CE4: Cross-check of HEGIF. Technical report, ISO/IEC JTC1 SC29 WG11, Geneva, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m39943_CE4_DMAG-UPC_cross_checking.pdf.

[101] Daniel Naro, Jaime Delgado, and Silvia Llorente. M39944 GENIFF (GENomic Information File Format) v2: Security and signature issues. Technical report, ISO/IEC JTC1 SC29 WG11, Geneva, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m39944_DMAG_UPC_GENIFF_Signatures.pdf.

[102] Jaime Delgado, Silvia Llorente, Daniel Naro, and Giorgio Zoia. M39945 Towards a WD for a format on genomic information compression and storage. Technical report, ISO/IEC JTC1 SC29 WG11, Geneva, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m39945_CE4_Format_integration.pdf.

[103] Jaime Delgado, Silvia Llorente, Daniel Naro, and Claudio Alberti. M40494 Genomic Information Representation. Proposal for Part 3 on Protection, Application Programming Interfaces and Metadata. Technical report, ISO/IEC JTC1 SC29 WG11, Macao, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m40494_Proposal_WD_Part_3_Genome.pdf.

[104] Jaime Delgado, Daniel Naro, and Silvia Llorente. M41069 Genomic Information Representation. Proposal for Metadata representation. Technical report, ISO/IEC JTC1 SC29 WG11, Torino, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m41069_MetadataProposal_DMAG.pdf.

[105] Daniel Naro, Jaime Delgado, and Silvia Llorente. M41070 Comments and issues in current MPEG-G Working Drafts (Parts 1 and 2). Technical report, ISO/IEC JTC1 SC29 WG11, Torino, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m41070_UPCCommentsWD.pdf.

[106] Daniel Naro, Jaime Delgado, and Silvia Llorente. M41072 Description of DMAG-UPC's MPEG-G generation and decoding tool. Technical report, ISO/IEC JTC1 SC29 WG11, Torino, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m41072_UPCTool.pdf.

[107] Jaime Delgado, Daniel Naro, and Silvia Llorente. M41073 Use case for referencing external existing genomic information from MPEG-G. Technical report, ISO/IEC JTC1 SC29 WG11, Torino, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m41073_UPC_UseCases_ExternalFiles.pdf.

[108] Jaime Delgado, Daniel Naro, and Silvia Llorente. M41730 MPEG-G metadata: Issues and mapping with EGA profiles. Technical report, ISO/IEC JTC1 SC29 WG11, Macao, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m41730_IssuesAndMappingEGAProfile.pdf.

[109] Jaime Delgado, Daniel Naro, and Silvia Llorente. M41731 Genomic Information Representation Metadata. Revision after 119th meeting. Technical report, ISO/IEC JTC1 SC29 WG11, Macao, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m41731_MetadataRev.pdf.

[110] Jaime Delgado, Silvia Llorente, and Daniel Naro. M41733 Genomic Information Representation. Proposal for WD of Part 3 on Protection, Application Programming Interfaces and Metadata. Technical report, ISO/IEC JTC 1/SC 29/WG 11, Macao, 2017. http://dmag.ac.upc.edu/downloads/mpegg/m41733_Proposal_WD_Part_3_Genome_Compression_API.pdf.

[111] Jaime Delgado, Silvia Llorente, and Daniel Naro. M42106 Proposal for CD of MPEG-G Part 3: Genomic Information Metadata and Application Programming Interfaces (APIs). Technical report, ISO/IEC JTC1 SC29 WG11, Gwangju, 2018. http://dmag.ac.upc.edu/downloads/mpegg/m42106_Genomic_Information_Part3_Metadata_APIs_CD_text.pdf.

[112] Jaime Delgado, Silvia Llorente, and Daniel Naro. M42588 Proposal for a Study of ISO/IEC CD 23092-3 Genomic Information Metadata and APIs. Technical report, ISO/IEC JTC1 SC29 WG11, San Diego, CA, 2018. http://dmag.ac.upc.edu/downloads/mpegg/m42588_StudyOf_CD_23092_3_Genomic_Information_Metadata_and_APIs.pdf.

[113] Daniel Naro and Jaime Delgado. M42590 Comments on Draft Text of ISO/IEC DIS 23092-1 Transport and Storage of Genomic Information. Technical report, ISO/IEC JTC1 SC29 WG11, San Diego, CA, 2018. http://dmag.ac.upc.edu/downloads/mpegg/m42590_Comments_on_part_1.pdf.

[114] Daniel Naro, Jaime Delgado, and Silvia Llorente. M43542 MPEG-G Part 3: Proposed Draft DIS. Technical report, ISO/IEC JTC1 SC29 WG11, Ljubljana, 2018. http://dmag.ac.upc.edu/downloads/mpegg/m43542_ISO_IEC_23092_3_proposed_draft_DIS.pdf.

[115] Jaime Delgado, Daniel Naro, and Silvia Llorente. M43543 MPEG-G Part 3 ballot: Pre-analysis of ballot comments. Technical report, ISO/IEC JTC1 SC29 WG11, Ljubljana, 2018.

[116] Daniel Naro and Jaime Delgado. M43545 Side-channel attack on MPEG-G in Descriptor Stream Contiguous mode. Technical report, ISO/IEC JTC1 SC29 WG11, Ljubljana, 2018. http://dmag.ac.upc.edu/downloads/mpegg/m43545_SideChannelAttack_DMAG_UPC.pdf.

[117] Jaime Delgado and Daniel Naro. M44845 Text proposal for ISO/IEC DIS 23092-3 Genomic Information Metadata and APIs. Technical report, ISO/IEC JTC1 SC29 WG11, Macao, 2018.

[118] Jaime Delgado, Daniel Naro, and Silvia Llorente. M47222 Draft text proposal for ISO/IEC FDIS 23092-3 Genomic Information Metadata and APIs. Technical report, ISO/IEC JTC1 SC29 WG11, Geneva, 2019. http://dmag.ac.upc.edu/downloads/mpegg/m47222_Proposed_FDIS_23092_3.pdf.

[119] Samtools. htsjdk. https://samtools.github.io/htsjdk/, 2020.