# Reversible fingerprinting for genomic information

Daniel Naro, Jaime Delgado, Silvia Llorente

*Departament d'Arquitectura de Computadors (DAC),*

*Universitat Politècnica de Catalunya · UPC BarcelonaTECH,*

*C/Jordi Girona, 1-3, 08034 Barcelona*

{dnaro, jaime.delgado, silviall}@ac.upc.edu

http://dmag.ac.upc.edu

*Corresponding author*
Name: Silvia Llorente
e-mail: silviall@ac.upc.edu
Phone: +34 93 401 74 09

**Abstract** – New genome sequencing technologies have simplified the generation of genomic data, making them more common but in turn a likely target of attack. Security strategies have been devised such as restricting the amount of information that can be queried or using new encryption techniques. These solutions might not be enough if the entire file has to be shared, as the recipient might leak the accessible information. This contribution addresses this issue using watermarking. Each read in a genomic file is modified depending on its content and a secret key. This allows generating different watermarked instances of the original file. Each watermark acts as a fingerprint: if a leak occurs, the unique modifications of the instance points to who originated the unauthorized publication. Using the key, the modifications can be undone. This allows sharing a leak-discouraging version with which the relevance of a file can be assessed, and can be reversed to the original if needed.

**Keywords:** Watermarking, Genomic Information, Information Leakage, MPEG-G, Fingerprinting

## 1 Introduction

The genomic information obtained from sequencing the DNA (DeoxyriboNucleic Acid) [1], [2] of an individual contains sensitive information. As with other biometric measures such as the palm print, the genome cannot be modified in order to mitigate the leakage of sensitive information. This information not only identifies the sequenced individual for life and informs about possible health issues, but also gives information about blood relatives and the diseases they might have to face.

This privacy issue motivates the development of new ways of protecting the genomic information: for example, through the anonymization offered by beacons [3], cryptography [4] or access rules [5].

In the case of the beacons, the interaction with the data changes. Only the genomic information for given regions of the genome is stored in the beacons, which can then be queried with statistical requests such as the frequency of a mutation in a population suffering from a given disease. This is a way to query the data without making it available in its entirety, similar to the biometric identification solution described in [6], where features are scattered across multiple storages.

The use of beacons alters how the data is used, as only the statistics for an entire population are known. For some studies, however, the entire genome is required and, in those cases, the researcher must have access to the entire data, not only to statistical information. In this case, and if it appears to be wrongfully released, we are interested in a method to identify who is responsible for the leakage, and this is what we propose in this paper with the use of watermarking.

From the world of audio, images and video (in summary, multimedia) we are familiar with the idea of watermarking: inserting some alteration which can be identified and hardly undone. This is a first step in addressing the introduced problem. With one mark identifying each one of the known data-holders, i.e. all individuals in possession of one instance of a genomic information file, we know that we will be able to find who broke the rules by publishing the genomic sequence. This is referred to as fingerprinting [7]. Nevertheless, as genomic information has many important applications and each modification could have an effect on the conclusions of a study, we prefer a modification method which has as limited effect as possible, whilst identifying multiple copies of the shared information, in order to detect possible information leaks.

The structure of this paper is as follows. First of all, we introduce some genomic concepts needed to understand the method proposed and which kind of information we are processing. Then, we review in which use cases a watermarking or fingerprinting method might be useful to protect genomic information. Then, we introduce different properties a watermark could have, both in terms of features and resistance to modifications. After that, we present our proposal to watermark / fingerprint genomic information in a deterministic and reversible manner, allowing to generate a new file with a minimum amount of changes, taking as input a genomic information file, a key and a set of parameters, and returning a modified genomic information file. We qualify the method as reversible based on the fact that, if the key is available, it is possible to restore the original state of the genomic information. Next, we present some results to the application of such method. Finally, we draw some conclusions and future work.

# 2 Background

## 2.1 Introduction to genomic concepts

The genomic information is stored in each cell as multiple molecules of DNA, which can be interpreted as a sequence of nucleotides. In DNA, there are four types of nucleotides, A (adenine), C (cytosine), G (guanine) and T (thymine) [1], [2]. During the life of a cell, the DNA molecules are translated into proteins. Some portions of the DNA molecule encode the protein sequence: three nucleotides at a time are read and translated into an amino acid, the building block of proteins.

Within a species, the genomic information is almost the same, however some mutations can occur: some nucleotides might change, be inserted or be removed from the DNA, leading to changes in the resulting proteins.

Research in this field is interested in finding mutations explaining certain diseases or advantages. In order to do so, the genomes of individuals are compared with one another. This allows the definition of a reference genome. Then, it is possible to determine the difference between one individual's genome and the reference defined.

The first step to obtain the genomic information from one individual is to sequence a biological sample. Nowadays, this can be done with Next Generation Sequencing (NGS) machines, like Illumina devices [8]. These machines sequence the genome by obtaining small subsequences of contiguous nucleotides forming the DNA molecules. We call chromosome to each DNA molecule and we refer to these subsequences as reads. Chromosomes are grouped by pairs: one chromosome is inherited from the mother and the other one from the father. The output is stored in text files in a format called FASTQ [9], representing the collection of generated reads as the sequence of obtained nucleotides and the confidence with which the nucleotide is identified (i.e. a measure of the quality of the read for each nucleotide). The order of appearance of the reads is random within the FASTQ file, respect to the species genome. This means that to be able to process the genomic information stored in the FASTQ file, several steps have to be taken, as described next.

The next step in the genome sequencing pipeline is to align the FASTQ file information to a reference genome. A reference genome is assembled by scientists as a representative example of a species' set of genes. The result of the alignment process is stored in a SAM file (Sequence Alignment Map) [10]. The information generated for each read during the alignment is the position where the read is stored (i.e. on which chromosome and where on the chromosome), and the differences between the reference genome and the read.

The main differences that can be found during alignment are mutations, insertions and deletions. A mutation means that a nucleotide on the read is different from the nucleotide on the reference for that location. An insertion means that some nucleotides have been added. Finally, a deletion means that some nucleotides are missing. There are other possible differences, which are further explained in [10]. The way of representing these differences inside a SAM file is to use CIGAR (Concise Idiosyncratic Gapped Alignment Report) strings.

A CIGAR string contains information about the reference sequence, the read sequences, and the type of operations (for example, match (M), insertion (I) and/or deletion (D)). A match means that the position of the nucleotide in the read is equal to the position on the reference genome, but does not allow to discriminate between the case where the two nucleotides are the same or not (this can be done either by comparison with the reference genome, using an auxiliary tag bounded to the read in the SAM, or by using a newer set of symbols in the CIGAR allowing to discriminate the two cases).

In summary, the SAM file stores the genomic information contained in the FASTQ file, indicating where it is regarding the reference genome. In case there is not a perfect alignment, the differences between original genome and reference genome are stored, being summarized in a CIGAR string, an example of which is shown in Table 1. In the example, a fake read is aligned to a fake reference. The meaning of the values of the CIGAR operations row in Table 1 are as follows:

- the first two nucleotides of the sequence match the ones from the reference (i.e. we have two M operations),
- 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).

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 1.

*Table 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 | | M | M | I | M | D | M | M | | | | | |
| Read CIGAR (as stored in SAM) | 2M1I1M1D2M | | | | | | | | | | | | |

Additionally, when a read's beginning or end does not match to the reference sequence, part of its nucleotides can be clipped. This is indicated with the type S operation (for soft-clipping) in the CIGAR string which means that the nucleotides are left in the read even though they are not mapped. This could be due to different reasons: sequencing errors at the beginning or end of the read, lack of information in the reference genome for the location, or more complex reasons (for example one part of a chromosome has been copied to another chromosome).

It is worth noting that the text-based information stored in a SAM file can be compressed into a BAM file (Binary Alignment Map) [10] in order to reduce the size of the information stored. SAM or BAM files also contain metadata regarding the process followed to generate them, like the tools used to perform the alignment.

Nevertheless, sequencing machines can make mistakes when identifying nucleotides, and as such, the fact that one nucleotide appears altered in one read does not mean that the studied individual has a mutation at that location. Instead, enough reads are sequenced such that there are multiple reads mapping to a same position, in order to have enough evidence. Then, all reads are analyzed in order to determine if mutations are really present. This information is stored in a VCF file

(Variant Call Format) [11], listing each mutation with its position, its type, and if it affects to one or two copies of the chromosome.

## 2.2 Watermarking in genomics

### 2.2.1 Watermarking

Watermarking techniques can be used for a broad set of tasks. In [12], 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:

- A simplification in the access control: if the image's metadata (e.g. information about the patient) is merged with the image by the means of the watermark, protecting the image and controlling the access to it already ensures the protection and the 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 opening new ways to attach information to it.
- Using a watermark to carry information identifying the origin, such as a signature.

The work presented here has something in common with [12], as we are using the new channel obtained thanks to watermarking to merge an identification of the receiver to 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:

- 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 on less significant regions, or, on the contrary, modify the content without considering its impact. We can find in [13] an example of limited effect modifications in images, as the proposed algorithm only modifies the least significant bits of certain elements.

The modification of the least significant bit is also employed in [14]. The authors include a new signal to the wireless transmission of an ECG (electrocardiogram): the proposal of the new signal is to indicate the identity of the patient within the ECG signal in order. 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 [15]: their contribution addresses the need for watermarking images. In order to preserve the clinically relevant portions of the image, they discriminate its regions as of interest (Region Of Interest, ROI) or not (Non-Region Of Interest, NROI). By applying the watermarking techniques to only the NROI regions, the potential drawbacks due to the file modification are mitigated.

In order to know if the watermark will be resistant to file modifications, we first identify actions which can be performed when modifying a genomic file, and the implications on the security method. They are as follows:

- 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 file: the watermark should not be invalidated by the presence of new non-watermarked data.
- Modifying without semantic changes (file changed but not its meaning, e.g. the read ids are changed): the watermarking method should rely on a minimal set of information, unlikely to be modified.
- Modifying with semantic changes: 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 [16], 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.

### 2.2.2 Use of watermarking in genomic information

The described process to sequence and align the genome is important for many use cases. For example, it can be used to perform research on some disease, to identify the best treatment or, when the genome is not from a human being, 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 because of intellectual property or to hold bad actors accountable in case of data leaks. To this end, it is interesting to be able to modify in some way the described data (i.e. watermarking it), in order to identify leakers in case of an audit. We have to make clear the difference between watermarking a DNA molecule as in a living cell or including a mark in the result of sequencing such molecule.

[17] and [18] 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 upon the consequence of changing the nucleotide. In the process of translating the DNA to a protein, each group of three nucleotides is translated to 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 to 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 [17], [18] 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 [19]. They modify sequential data before sending it to every recipient, in such a way that the ability to reconstruct the original data is minimal, the modifications are overall 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.

[19] 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 is whether the mutation is present in none, one or the two chromosomes of the individual. As such, [19] 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 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 with which modifications, 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 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 them 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 data modified: 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.

## 2.3 Use case for watermarking in genomics

Our use case is intended to apply watermarking as a fingerprinting to genomic information. Nevertheless, several situations can be covered, as described next.

Either the genomic data owner or the genomic data custodian can receive a request for a copy of the data. The request might be extended with metadata regarding the intentions of the researcher (e.g. conduct a study on cancer or use it as input for a genealogy search), allowing to check with the owner's policy if the request should be replied with a positive answer.

In addition to this protection, the genomic data owner or custodian might prefer to slightly alter the file during the transfer by including a fingerprint. If the alteration is unique for each request, the data owner is then able to audit the result of leakages. If the leaked data matches the transferred one (exactly the same reads are present, or the number of identical mutations is above a certain threshold), it is worth comparing the present variations to the one inserted willingly. In the case where they match the ones sent to one of the requesters, the owner is at least aware of who breached the link of trust.

Another application could be to limit the temptation of separating the genomic data from associated metadata and privacy rules. In the upcoming MPEG standard for genomic data, ISO/IEC 23092, MPEG-G [20], [21], [22], it will be possible to convey such type of fields alongside the genomic information. By modifying the privacy rules, an attacker could repurpose a file without permission. In this case, the watermarking could be generated using the privacy rules as input instead of the genomic data. In the case where some auditing agent detects data supplied with a privacy rules field which is not the input used for the watermark, a flag could be raised indicating a modification of the content.

Finally, if for a given case the data request is just for a brief showing in order to decide if this could be a valuable input, but protection methods such as cryptography are for any reason non suitable, watermarking could be an approach to ensure that the data has a low value outside the very scope of the request. The ideas used in [23] could be of interest in such a use case: the same concepts of optimizing the trade-off between privacy and utility and how to measure both ideas are relevant to this scenario.

# 3 Fingerprinting of reads with mutations

## 3.1 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 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 Subsection 2.1. 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 sequence:

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

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

## 3.2 Description

The assumption on top of 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 2, where we separate with hyphens each field used for the description.

*Table 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 | |

### 3.3 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).

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.

### 3.4 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. We refer with secure transformation to such an operation which, given a description, generates deterministically a secret.

Both during the watermark auditing or the watermark removal we want to obtain the same result from the secure transformation. To this end, we do not consider the numbering of the read to be a suitable input since, for example, if at some point the user generates a new file containing only a subset of the reads (perhaps all the reads aligned to one reference sequence only), this input is lost. As we would require an initialization vector per read (in order to maintain the independence of reads), such 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 Section 4.

### 3.5 Perform the modifications

The result of the fingerprinting will be a modification of the soft clipping at the beginning of the read (see Subsection 2.1 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 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 in the pool of possible lengths. On the contrary, if we plot the histogram of soft clip lengths (Figure 1), we observe that short lengths are far more likely. As such, it is preferable to construct the decoding of the length of the soft clip operation using the observed Cumulative Distribution Function.

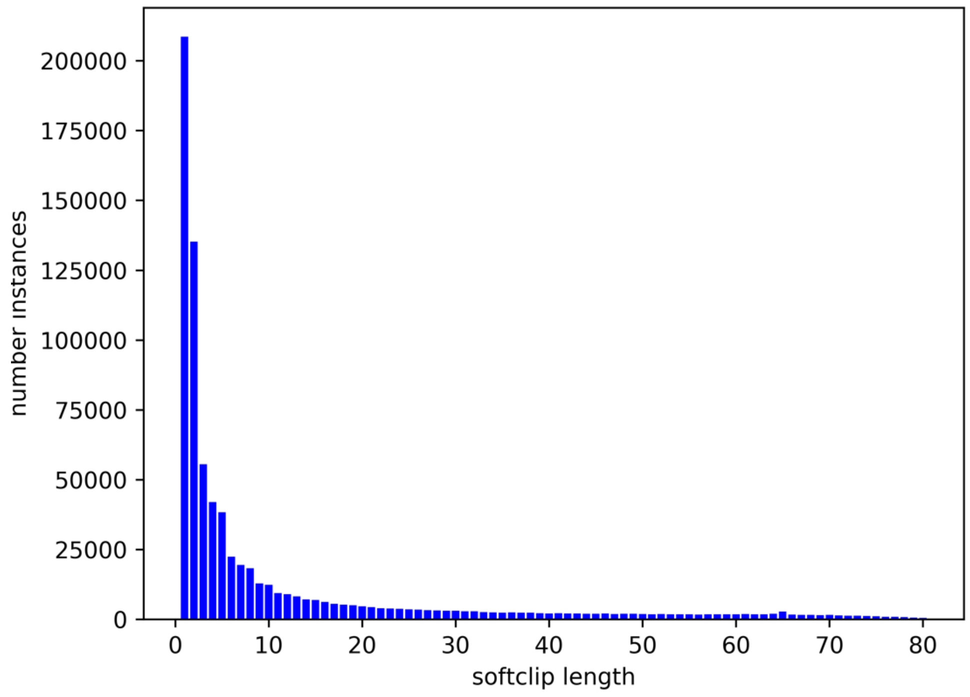

**Fig. 1.** Histogram of softclip lengths

With this mechanism, we can now first read from the result of the secure transformation a guess for the current soft clip operation size $guess_{size}$, and then the length of the one in which we should transform $transformation_{size}$. This distribution function is likely to return for the two random variables the value zero, thus reducing the likelihood of executing a transformation. Additionally, we also decode base pair strings: $guess_{string}$ and $transformation_{string}$. 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 2 we show how the bytes are interpreted.

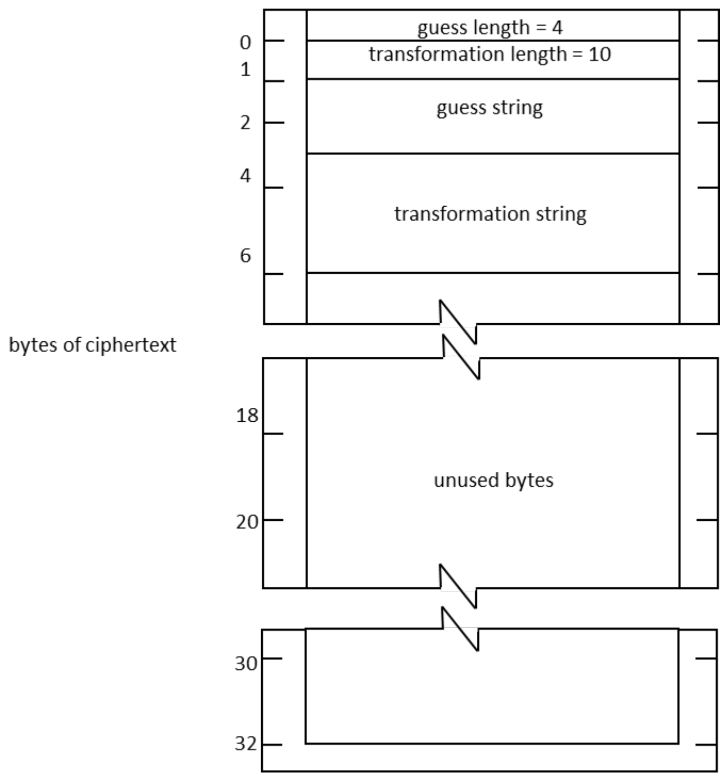

**Fig. 2.** Interpretation of the secure transformation output.

The operation is performed if *guess* $_{size}$ and *guess* $_{string}$ 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, *transformation* $_{size}$ and *transformation* $_{string}$ 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 *guess* $_{size}$ and *transformation* $_{size}$ 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.

```
**Data**: read, secure_transformation_output
**Result**: Watermarked read
$real\_softclip_{length}$ = getSoftclipLength(read);
$real\_softclip_{string}$ = getSoftclipString(read);
$guess_{size}$ = readSoftclipLength(secure_trans_output[0]);
$trans_{size}$ = readSoftclipLength(secure_trans_output[1]);
$guess_{string}$ = readSoftclipString(secure_trans_output[2]);
$trans_{string}$ = readSoftclipString(secure_trans_output[3]);
**if** $real\_softclip == guess$ **then**
    fingerprint(read, from=$guess$, to=$trans$);
**else**
    **if** $real\_softclip == trans$ **then**
        fingerprint(read, from=$trans$, to=$guess$);
    **end**
**end**
```

**Alg. 1.** Pseudocode for reading watermarking function parameters.

### 3.5.1 Fingerprinting operation

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 $max(guess_{length}, transformation_{length})$ 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 3 for a visual representation of the case where $transformation_{length} < guess_{length}$. In the inverse case, we would copy the reference genome in the undoing operation. Figure 4 shows how a read, its soft-clipping and its corresponding CIGAR string is modified after applying the transformation operation.

### 3.5.2 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 $transformation$.

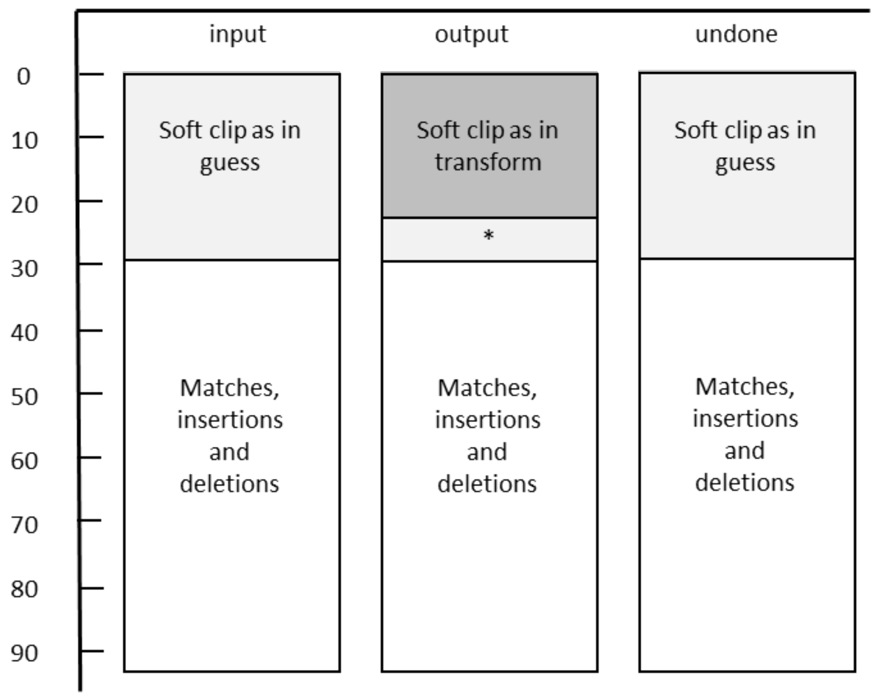
input
output
undone
0
10
20
30
40
50
60
70
80
90
Soft clip as in guess
Soft clip as in transform
*
Soft clip as in guess
Matches, insertions and deletions
Matches, insertions and deletions
Matches, insertions and deletions

**Fig. 3.** Soft clip contraction operation

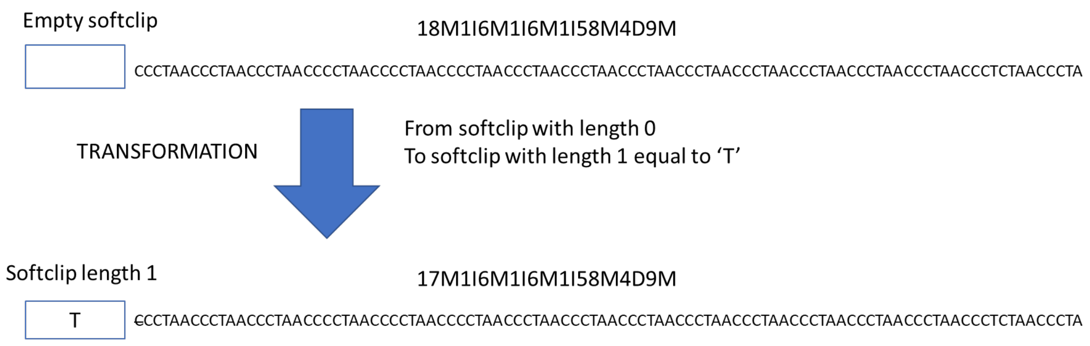
Empty softclip
18M1I6M1I6M1I58M4D9M
CCCTAACCCTAACCCTAACCCCTAACCCCTAACCCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTCTAACCCTA
TRANSFORMATION
From softclip with length 0
To softclip with length 1 equal to 'T'
Softclip length 1
17M1I6M1I6M1I58M4D9M
T
~~C~~CCTAACCCTAACCCTAACCCCTAACCCCTAACCCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTCTAACCCTA

**Fig. 4.** Example of application of transformation operation to a read and its associated CIGAR String

## 4 Results and Discussion

### 4.1 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 if the leaked data matches the modified result. In case of granting access to the unmodified version, only the fingerprinting key is required by the recipient 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 [8]. The file has a low coverage

(2.3). Coverage refers to the average number of identifications for a single nucleotide, as described in [24]. The file is part of the database of test material [25] 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 $guess_{length}$ and $transform_{length}$ 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.

## 4.2 Resistance to collusion attacks

### 4.2.1 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 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 there are different outcomes 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 where 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 5 shows how the risk of collusion increases as the number of watermarked files increases: as the number of watermarked files increases, the ratio of cases where the non-watermarked variant is the most frequent variant increases, 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 [19] 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 key. As soon as a read is watermarked with one of the keys, the iterations stop.

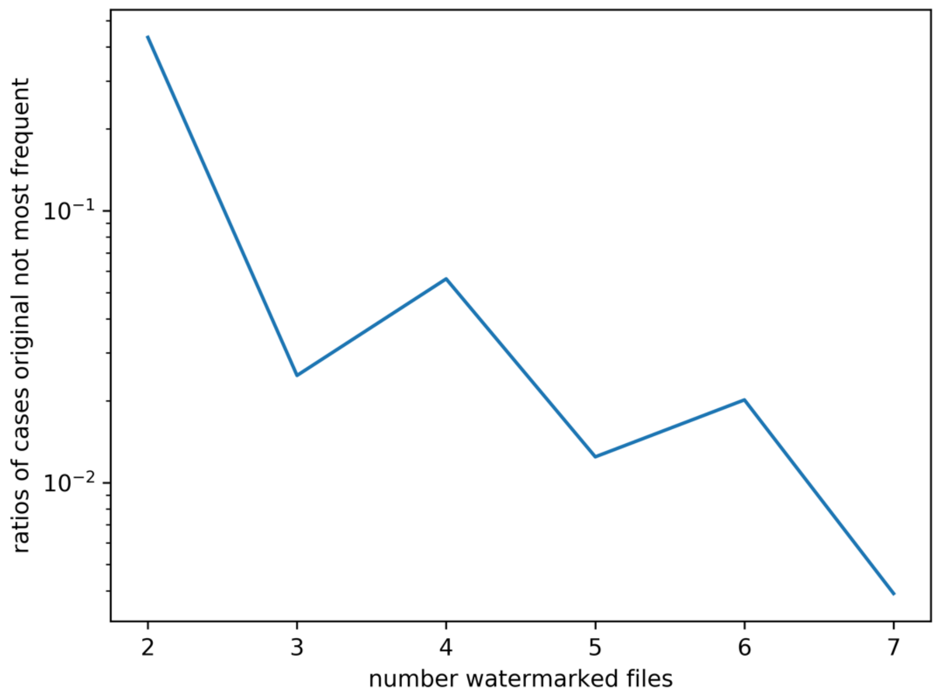

**Fig. 5.** Evolution of the ratio of cases where the original non-watermarked variant is not the most frequent across all watermarked file.

As for each read the likelihood of it being modified is low, we can think as if each key was modifying a distinct subset of reads, without any overlapping with any other set. In these conditions, what we want is a configuration such that for each recipient we have selected a unique set of keys, that 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 version, even when 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, possibly harming conclusions. If the number of recipients is known before-hand the task is trivial, and the collusion strategy of selecting the most frequent version will always fail.

### 4.2.2 An improvement using an ILP

In the case where the number of recipients is unknown, some concessions have to be made. Let us assume that the set of keys for a new user is decided upon receiving the new request. We only know which keys were used for the previous recipients.

Similar to [19], we construct an Integer Linear Program (ILP), which aims at minimizing the associated cost 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 to all previous sets employed.

Table 3 shows the result for this iterative process: in order to obtain a column, all previous columns were 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 source of leaking as it appears in a minority of sets, but in subsequent calls the key is always selected, therefore it eventually reaches a point where it is not a source anymore. However, this happens when all combinations are used, therefore the problem rises again at the next iteration. The process can be observed for recipient 2, 4, 8, and 16.

*Table 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 | | | | | | | | | | | | | | | | |

We generate the fingerprints for each of the recipients decided by the ILP, the results for which are summarized in Figure 6. We can see that as the number of recipients increases, the average number of modified reads for each recipient 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 enough recipients.

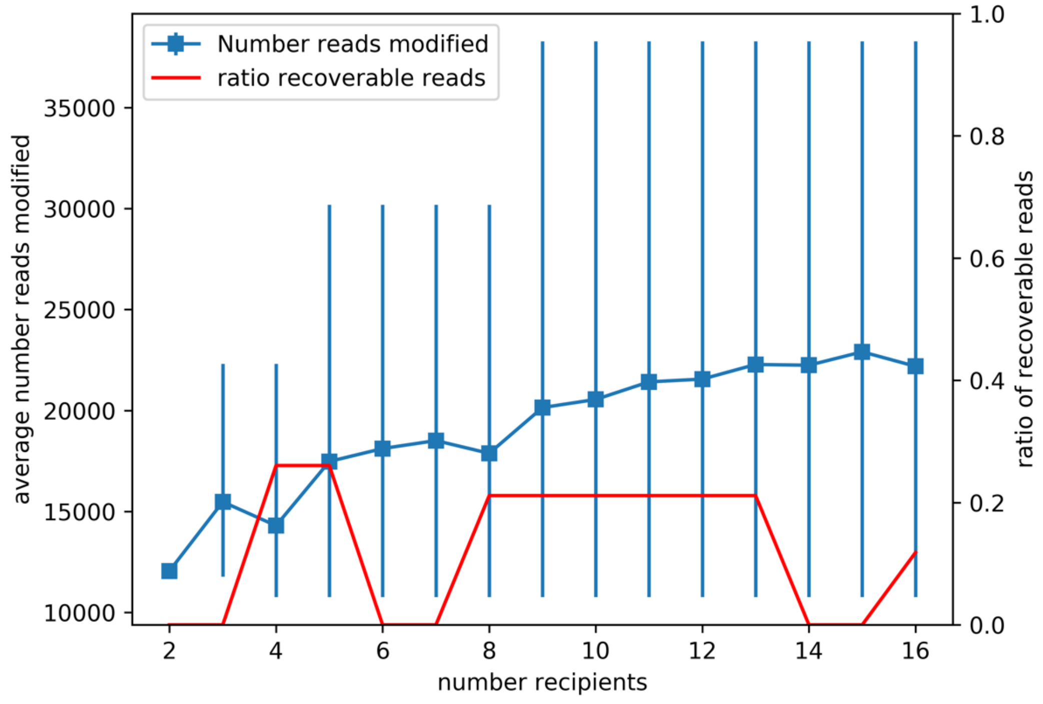

**Fig. 6.** Number of reads modified and leakage ratio versus number of recipients.

There is a trade-off between the use of multiple keys to avoid collusion attack 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.

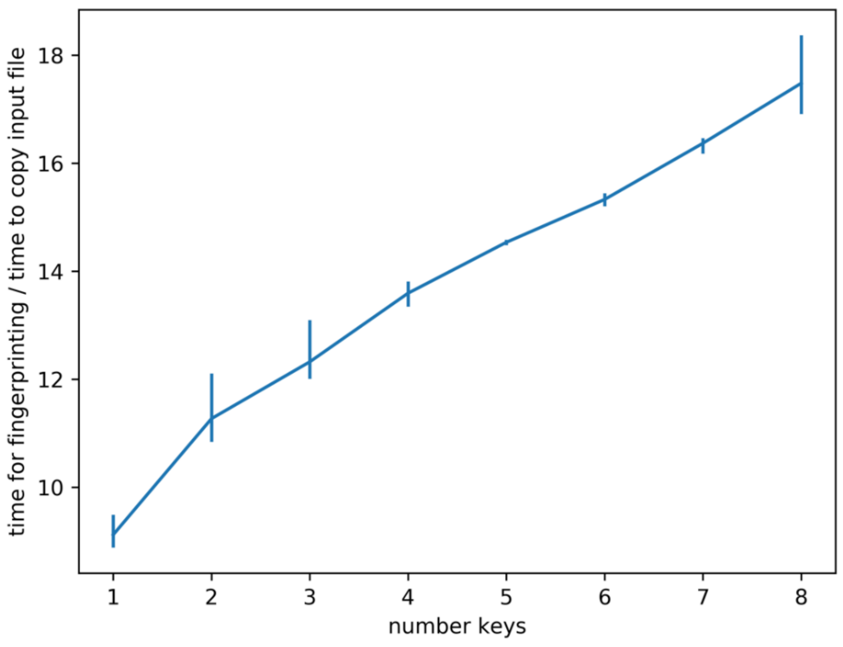

**Fig. 7.** Time required to fingerprint a file compared to copying it

## 4.3 Generalization of the description

Our next step is to slightly modify the behavior of the algorithm in order to improve it. In this way, a broader set of cases are accepted as input. We use a file with variable read length, some of which have a length greater than 255 (average depth of 8.1 reads per base). Thus, the byte used to store the position of the mutation is overflowed: we convert this to a two bytes system. The maximal length of the description is kept at 32 bytes; therefore, we are able to store less mutations in it. The observed likelihood of a read being marked, given 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 [26], 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 build, 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. However, this same approach decreases greatly the value of the published information, as it modifies blindly the data 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.

Similarly, and as mentioned in Section 3, it is possible that different recipients of the same data, but with different fingerprints, could collude in order to work the fingerprinting back. In this case, the different attackers could compare the different variants of the same read. The most common variant is most likely not to be marked by any fingerprint. As such, a collusion of three or more actors could allow to work back the security measure: considering a marking likelihood of 25% there is a probability of 84% that a read will be at most worked once in the three copies.

# 5 Conclusions

We have presented a method to modify a file of aligned genomic information, in order to introduce recognizable changes, but which are hard to identify as such without the required parameters. These changes can be undone in the case where 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 closest algorithms 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 [27]. As in the case of image watermarking, we have to be concerned with the possible transformations done over the file. However, despite the similarity in the objectives (either mark the ownership of the intellectual property or identifying 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 [21].

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

## *Acknowledgements*

The work presented in this paper has been partially supported by the Spanish Research Agency/ERDF(EU), through the project Secure Genomic Information Compression (GenCom, TEC2015-67774-C2-1-R, TEC2015-67774-C2-2-R), and by the Generalitat de Catalunya (2017 SGR 1749).

## References

[1] E. S. Lander et al., “Initial sequencing and analysis of the human genome”, Nature, vol. 409, no. 6822, pp. 860–921, Feb. 2001.

[2] J. C. Venter et al., “The Sequence of the Human Genome”, Science (80), vol. 291, no. 5507, pp. 1304–1351, Feb. 2001.

[3] M. Fiume et al., “Federated discovery and sharing of genomic data using Beacons”, Nat. Biotechnol., vol. 37, no. 3, pp. 220–224, Mar. 2019.

[4] H. Tang et al., “Protecting genomic data analytics in the cloud: state of the art and opportunities”, BMC Med. Genomics, vol. 9, no. 1, pp. 1–9, Dec. 2016.

[5] J. Delgado, S. Llorente, and D. Naro, “Protecting Privacy of Genomic Information”, Stud. Health Technol. Inform., vol. 235, pp. 318–322, 2017.

[6] L. Leng, A. B. Teoch, and M. Li, “Simplified 2dpalmhash code for secure palmprint verification”, Multimed. Tools Appl., vol. 76, no. 6, pp. 8373-8398, Mar. 2017.

[7] R. Popa, “An Analysis of Steganographic Techniques”, The “Politehnica” University of Timisoara, 1998.

[8] M. L. Metzker, “Sequencing technologies — the next generation”, Nat. Rev. Genet., vol. 11, p. 31, Dec. 2009.

[9] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice, “The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants”, Nucleic Acids Res., vol. 38, no. 6, pp. 1767–1771, 2009.

[10] H. Li et al., “The Sequence Alignment/Map format and SAMtools”, Bioinformatics, 2009.

[11] P. Danecek et al., “The variant call format and VCFtools”, Bioinformatics, vol. 27, no. 15, pp. 2156–2158, 2011.

[12] A. Giakoumaki, S. Pavlopoulos, and D. Koutsouris, “Multiple Image Watermarking Applied to Health Information Management”, IEEE Transactions on Information Technology in Biomedicine, vol. 10, no. 4, pp. 722–732, 2006.

[13] N. Provos and P. Honeyman, “Hide and seek: An introduction to steganography”, IEEE Secur. Priv., vol. 1, no. 3, pp. 32–44, May 2003.

[14] A. Ibaida, I. Khalil, and R. van Schyndel, “A low complexity high capacity ECG signal watermark for wearable sensor-net health monitoring system”, 2011 Computing in Cardiology, pp.393–396, 2011.

[15] D. S. Chauhan, A. K. Singh, B. Kumar, and J. P. Saini, "Quantization based multiple medical information watermarking for secure e-health", Multimed. Tools Appl., vol. 78, no. 4, pp. 3911–3923, 2019.

[16] H. Zarrabi, M. Hajabdollahi, S. M. R. Soroushmehr, N. Karimi, S. Samavi, and K. Najarian. "Reversible Image Watermarking for Health Informatics Systems Using Distortion Compensation in Wavelet Domain", 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 798–801, 2018.

[17] D. Heider and A. Barnekow, "DNA-based watermarks using the DNA-Crypt algorithm", BMC Bioinformatics, vol. 8, no. 1, p. 176, 2007.

[18] D. Heider and A. Barnekow, "DNA watermarks: A proof of concept", BMC Mol. Biol., 2008.

[19] A. Yilmaz and E. Ayday, "Collusion-Secure Watermarking for Sequential Data", eprint arXiv:1708.01023, Aug. 2017.

[20] J. Delgado, S. Llorente, and D. Naro, "Protecting Privacy of Genomic Information", Stud. Health Technol. Inform., vol. 235, pp. 318–322, Apr. 2017.

[21] ISO/IEC JTC 1/SC 29/WG 11, "MPEG-G, ISO/IEC 23092 Genomic Information Representation", 2019. [Online]. Available: https://mpeg.chiariglione.org/standards/mpeg-g. [Accessed: 12-September-2019].

[22] C. Alberti et al., "An introduction to MPEG-G, the new ISO standard for genomic information representation", bioRxiv, p. 426353, Jan. 2018.

[23] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Reconciling Utility with Privacy in Genomics", in Proceedings of the 13th Workshop on Privacy in the Electronic Society - WPES '14, pp. 11–20, 2014.

[24] K. Song et al., "Coverage recommendation for genotyping analysis of highly heterologous species using next-generation sequencing technology", Scientific Reports, vol. 6, Oct. 2016.

[25] Run: ERR317482, "Illumina HiSeq 2000 paired end sequencing", 2019. Available: https://www.ebi.ac.uk/ena/data/view/ERR317482. [Accessed: 12-September -2019].

[26] Run: ERR194146, "Utah residents (CEPH) with Northern and Western European ancestry", 2019. https://storage.googleapis.com/genomics-public-

data/platinum-genomes/bam/NA12877_S1.bam. [Accessed: 12- September - 2019].

[27] N. Agarwal, A. K. Singh, and P. K. Singh, “Survey of robust and imperceptible watermarking”, Multimed. Tools Appl., vol. 78, no. 7, pp. 8603–8633, Apr. 2019.