# INTERNATIONAL ORGANISATION FOR STANDARDISATION
# ORGANISATION INTERNATIONALE DE NORMALISATION
# ISO/IEC JTC1/SC29/WG11
# CODING OF MOVING PICTURES AND AUDIO

**ISO/IEC JTC1/SC29/WG11 MPEG2018/M43545**
**July 2018, Ljubljana, SI**

**Source:** **DMAG-UPC**
**Status:** **Proposal**
**Title:** **Side-channel attack on MPEG-G in Descriptor Stream Contiguous mode**
**Authors:** **Daniel Naro, Jaime Delgado (Distributed Multimedia Applications Group - Universitat Politècnica de Catalunya, Barcelona)**

**Table of Contents**

# 1 Problem

In Descriptor Stream Contiguous mode (DSC), MPEG-G allows to encrypt all data of chosen descriptors. This might give a false impression that by encoding the streams storing the mutation position and mutation types of class M or class I, for example, all mutations are protected.

It is however possible to use the descriptor of the position as a side-channel to deduce the position of the mutations. We here present this side-channel attack in a simplified file having only reads with no mutation in respect to the reference (class P), or having one or more mutations (only base substitutions, i.e. reads of class M). In this simplified file, the reads have a fixed and known length. We assume that who wanted to protect the file, did not want the mutations to be readable, and assumed that encrypting the streams regarding the position and the type of the mutation in each read is enough. In other words, in this attack against the simplified file, only the position and class of each read is readable, but yet information concerning the mutations can be retrieved.

# 2 Attack's background

Let us imagine there is a mutation on both chromosomes at the position 5. In this case all sequenced reads which have one base pair mapped on position 5 will be different from the reference, as they will contain the mutation. Table 1 summarizes a set of reads ordered by start position, indicating by the color if they belong to class P or M (blue for P, red for M) and for how many position they span. Each column corresponds to a position on the reference, while each row corresponds to a read.

*Table 1*

| 0 | 1 | 2 | 3 | 4 | 5 (mutation) | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |

One way to understand Table 1 is to imagine the genome of the individual as a discrete time signal, where time is used to represent the person's genome. In this signal, t=0 corresponds to position 0, t=1 to position 1 and so on. This signal is such that each position is equal to 0 except for those positions where there is a mutation, in which case it is equal to 1. In the example introduced in Table 1, the signal is defined as:

$$signal[t] = \begin{cases} 1, & t = 5 \\ 0, & otherwise \end{cases}$$

Convoluting this signal with a rectangular filter, integrating from the start position of the read over a size of read length into the "future" (i.e. positions in the genome ahead of the current one), it will return the class to which the read belongs (if the result is one, the class is M, otherwise P). It should

be noted that the result of this convolution is clipped to one: if the convolution detects multiple mutations, the read belongs to the same class M as if only one was detected.

Using the previously obtained data, we summarize the available information by integrating for each position over the reads available for that position. I.e. we count how many reads of class M are mapped with a given position. We can understand this as a convolution with a rectangular filter integrating over the last read length positions. Table 2 adds to Table 1 one row corresponding to this result, i.e. the number of reads of class M mapped on the position associated to the column.

*Table 2*

| **0** | **1** | **2** | **3** | **4** | **5** (mutation) | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 4 | 3 | 2 | 1 | 0 |

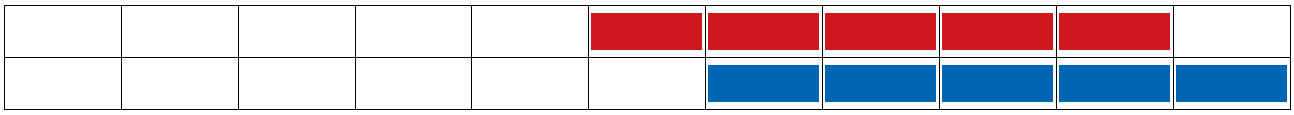
We have now derived two convolutions to determine what count of reads with mutation to expect for a given position knowing the genome. We should note that an approximate solution to both convolutions can be obtained using the Z-transform, therefore there is a way to reverse from the previous table to discover that there is a mutation at position 5. It should be noted however, that due to the clipping in case of multiple mutations, we have to expect non exact results, thus undermining the efficiency of the method.

## 3 Performing the attack

Let us now suppose that we have an MPEG-G file with reads of class P and M, and we are only aware of the position and length of each read, since the rest is encrypted. The input data might look like:

*Table 3*

| **0** | **1** | **2** | **3** | **4** | **5** (mutation) | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |



Although not the same, we obtain something similar to Table 1. We use this to derive the equivalent to Table 2:

*Table 4*

| 0 | 1 | 2 | 3 | 4 | 5 (mutation) | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| 0 | 0 | 1 | 3 | 3 | 5 | 5 | 4 | 1 | 1 | 0 |

Although we have not obtained exactly Table 2, we see that the result is very similar.

As described in section 2, we could attempt to reverse the convolutions in order to obtain the signal modelling the genome. However, the approximations in the process and the non-uniform coverage of the genome makes this technique more challenging. We could improve the signal, for example by dividing by the average coverage, or the number of reads mapped to the position.

An alternative proposal is to train a convolutional neural network, returning a decision for each position in the genome. The training information is a collection of pairs of the genome (in our example the signal of length 11, equal to 0 for all positions except at position 5) and partially encrypted data (in our example the contents of Table 3). This network is then used to reverse the input, thus obtaining the supposedly protected genome. It should be noted that the trained neural network can be reused. For example, a neural network built and trained for Illumina data can be reused for any file sequenced with such a machine. Furthermore, the training can be performed using synthetic genomes, thus lowering the entry barrier.

## 4 Computational tests

A first version of the attack was implemented using solely convolutions. This method was used on simple, short genomes, with only one mutation, present on both chromosomes, with pair of reads. Each segment was 100 base pair long. The method detected correctly the mutation, but the ratio of false positive was too high.

A second method was developed using machine learning (neural networks). The solver was using a convolutional neural network, namely two successive convolutions, followed by a fully connected graph. Tests were performed using the Tensorflow framework, running on a GeForce

GTX 1070. In order to adequate to the available memory, the simulated read lengths were reduced to only one segment, 50 base pairs long. This number could be increased either reducing the size of input batches, or by increasing the available memory.

After training the network, on 15000 genomes of 10000 base pair length, the method was tested on a separate set of cases. The system was trained to return one of three values for each position: 0 (indicating no mutation), 1 (indicating one mutation, without specifying the chromosome), 2 (indicating a mutation on both chromosomes). The best result obtained were of 96.95% correctly inferred values. Further results are summarized in Table 5.

*Table 5*

| Value | Explanation | Result |
|---|---|---|
| True positive | Method returns 1 or 2, correct is 1 or 2 | 1.0 |
| False positive | Method returns 1 or 2, correct is 0 | 0.0 |
| False negative | Method returns 0, correct is 1 or 2 | 0.03 |

It should be noted that this method has a theoretical minimal value of false negative greater than 0: if two mutations are on the same chromosome(s) and closer from one another than read length base pairs, than any mutation on the same chromosomes placed between the two will not introduce any modification to the previously described tables. It is therefore shadowed by the other two and undetectable.

# 5 Conclusions

Although this is yet just a theoretical attack, it proves the existence of a threat. The standard should clearly state that trying to protect a genome in Descriptor Stream Contiguous (DSC) mode by just encrypting the streams containing the mutation positions and types might be easily defeated.

As in other cases in cryptography and security, users should be made aware of the existence of this strategy, but also about its pitfalls.

This attack is defeated either by using Access Unit Contiguous (AUC) mode (because all position data are already encrypted), or by encrypting the position descriptor stream when in DSC mode.