applied sciences

MDPI

Article

A Mobile-Oriented System for Integrity Preserving in Audio Forensics

Diego Renza *,†, Jaime Andres Arango † and Dora Maria Ballesteros †

Department of Telecommunications Engineering, Universidad Militar Nueva Granada, Carrera 11 No. 101-80, Bogotá 110111, Colombia

* Correspondence: diego.renza@unimilitar.edu.co; Tel.: +57-1-650-0000

† These authors contributed equally to this work.

Received: 4 June 2019; Accepted: 29 July 2019; Published: 31 July 2019

Abstract: This paper addresses a problem in the field of audio forensics. With the aim of providing a solution that helps Chain of Custody (CoC) processes, we propose an integrity verification system that includes capture (mobile based), hash code calculation and cloud storage. When the audio is recorded, a hash code is generated in situ by the capture module (an application), and it is sent immediately to the cloud. Later, the integrity of the audio recording given as evidence can be verified according to the information stored in the cloud. To validate the properties of the proposed scheme, we conducted several tests to evaluate if two different inputs could generate the same hash code (collision resistance), and to evaluate how much the hash code changes when small changes occur in the input (sensitivity analysis). According to the results, all selected audio signals provide different hash codes, and these values are very sensitive to small changes over the recorded audio. On the other hand, in terms of computational cost, less than 2 s per minute of recording are required to calculate the hash code. With the above results, our system is useful to verify the integrity of audio recordings that may be relied on as digital evidence.

Keywords: hash function; integrity; audio forensics; cloud solution; mobile application

1. Introduction

Audio forensics consists of "acquiring, analysing and evaluating audio recordings that may ultimately be presented as admissible evidence in a court of law or some other official venue" [1]. In this scenario, such recordings are susceptible of becoming what is known as digital evidence, i.e., suitable proofs in the form of binary data [2,3]. However, the validity of this evidence type is subject to its reliability from the point of view of its integrity. The integrity of digital content refers to ensuring that the information obtained from the original source is complete and to the fact that it has not undergone any type of manipulation from the moment of its acquisition until its final disposition [4]. However, determining the integrity of an audio signal is not a simple task given the simplicity and great availability of audio editing tools in the market. In this sense, the admissibility of digital audio evidence in legal processes is one of the key tasks in the audio forensics area, and consequently the integrity verification process is of great interest today.

Thus, nowadays, there are several methods that allow demonstrating the integrity of digital evidence such as hash functions, digital signatures, encryption, cyclic redundancy check (CRC), and watermarks, among others [4]. The digital signatures allow guaranteeing the authorship and integrity of a document by means of the implementation of the hash function in the document to be signed, which will then be encrypted with a specific private key [4]; encryption is a process in which the information is modified in order to protect it [4]; the cyclic redundancy check (CRC) is used commonly to check that the data has not been altered during the transmission process [5]; and watermarks

provide an information concealment technique in which a representative mark of the author is inserted into the file that is intended to be shared, seeking to protect the copyright [6,7]. On the other hand, a hash function (also known as digest function) allows producing a distinctive summary from an entry based on its content, generating a fixed length text string (hash code). Since hash functions are highly sensitive to changes in content, these types of functions are quite useful to demonstrate the integrity of audio signals. Recent proposals include cloud-based solutions for crime reporting [8], cloud based integrity verification by means of hash functions [9], integrity verification using blockchain [10], and evidence retrieval based on semantic methodologies [11].

Even though hash functions applicable to any type of content have been designed, in some cases, it may be advisable to design specific functions according to the content type; for multimedia files, hash functions can be designed to be applied in audio, image or video signals [12]. In addition, it is necessary to consider that the cryptographic hash functions designed for forensic files must comply with the following fundamental properties: the first property indicates that it should not be possible to recover the original signal from the hash code (preimage resistance); the second and third properties define collision resistance, and they establish that it must be computationally infeasible for a given input, to find a different input with the same hash code. In the same way, it must be difficult to find two different files that generate the same hash code [13].

Currently, there are several mechanisms that seek to attack hash functions such as brute force attack and cryptanalytical attack [14–16]; therefore, the number of output bits in the hash code should be considered when designing the system. It is also necessary to clarify that collisions will always exist since the mapping process is not one to one [17]. Moreover, it has been shown that certain hash functions widely used in the forensic context, such as MD5 and SHA-1, may be susceptible to collisions [18,19].

For the audio case, some hash functions are oriented to be perceptual, which are characterized by tolerating changes in the signal that do not significantly alter its content, consequently the generated hash code does not undergo modifications. Some examples of perceptual hash functions apply the non-negative matrix factorization (NNMF) of Mel-frequency Cepstral coefficients (MFCC) and then, through Principal Component Analysis (PCA), obtain a hash code of the audio signal [20]. It is possible to find solutions that use balanced multiwavelets (BMW) using five levels of decomposition and division of coefficients in the sub-bands [21]; it is also possible to find solutions based on MDCT (Modified Discrete Cosine Transform) that allow the extraction of characteristics of the signal [22]. Another variant is based on the ordering of the spectral coefficients obtained from the FFT calculation (Fast Fourier Transform) of the voice signal and a Gaussian noise signal generated from the statistics of the original signal [23].

Regarding the methods to calculate the hash code of an audio signal, the existing solutions are aimed at generating the summary of the signal after the recording (for example, once the evidence has been collected). An alternative to this approach is to generate the hash code of the audio signal immediately after its recording, as well as to store the code in a secure manner. In this sense, this paper presents the design and development of a solution capable of calculating a hash code; it is designed for speech signals without compression, and it can be used for forensic purposes. The solution operates in real time (at the time of the events) and allows one to guarantee the integrity of audio signals recorded on a mobile device with Android operating system and storage of the hash code in Microsoft Azure. After the recording and generation of the code, the solution allows verifying the integrity of the file by means of a query to the metadata previously stored in the cloud.

2. Proposed Scheme

The proposed function to obtain and save the hash code of an audio signal in real time is composed of three modules: (i) recording of the input signal; (ii) hash code calculation; and (iii) storage of the code in the database. The general outline of the system is shown in Figure 1. The specific operation of the system is explained below.

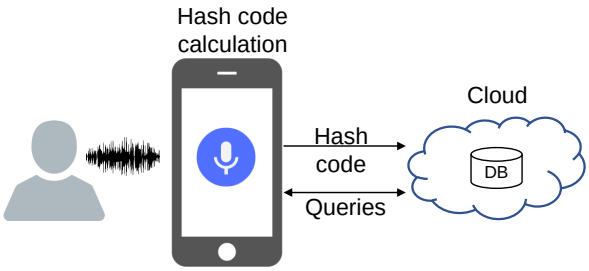

Figure 1. General outline of the proposed scheme.

2.1. Input Signal

Taking into account that the proposed solution is oriented to operate in real time (generate and store the hash code immediately after recording the file), the first block of the proposed method is to record the audio file in the same Android application that performs the hash code calculation. For the recording process, the following parameters were defined in java: each audio is acquired with sampling frequency of 8000 Hz and 16 bits per sample, and it is stored in wav format. The number of samples and the duration will depend on the needs of the user. In addition, the *MediaRecorder* library available in Android Studio was used to make the recordings.

The result of the recording process is the signal $S(n)$ (Equation (1)), where N is the number of samples of the signal.

$$S(n), 1 \leq n \leq N \tag{1}$$

2.2. Hash Code Calculation (Hash Function)

After recording the audio signal S, the first step in obtaining the hash code consists of resizing the vector S (input signal), in such a way that the data are structured in an array of 64 columns. The number of rows of this matrix will depend on the number of samples of the original signal, being equal to $N/64$, where N is the number of samples of the original signal. In the case N is not a multiple of 64, the original signal will be completed with zeros until the criterion is met. It should be noted that the vector of the original signal only considers the samples of the signal, that is, it discards the header of the file (44 bytes). In addition, the selection of 64 columns is due to the fact that a smaller number of columns can produce a greater number of repeated characters in the hash code, while the definition of a greater number of columns can cause an unnecessary number of zeros to be added (so that the number of samples is a multiple of the number of columns). Thus, the output of the reshaping block is a $N/64 \times 64$ matrix (s^r), as shown in Equation (2).

The next step is to calculate the mean of each column in the matrix s^r, obtaining a 1×64 vector (s^m), as shown in Equation (3).

$$s^r = \begin{bmatrix} s(1) & \cdots & s(64) \\ \vdots & \ddots & \vdots \\ s(n-64) & \cdots & s(N) \end{bmatrix} \tag{2}$$

$$s^m(j) = \frac{64}{N} \sum_{i=1}^{N/64} s^r(i,j) \tag{3}$$

Then, the Discrete Fourier Transform (DFT) of the s^r vector is computed through the use of the Fast Fourier Algorithm (FFT). Since the execution time for FFT is faster for powers of two, and considering voice signals with sampling frequency of 8 kHz, 2^{13} was selected as the number of points in FFT, i.e., an 8192-point DFT is used. In addition, since the phase presents a greater variability than the magnitude in the whole spectrum, the argument of the DFT is taken as shown in Equations (4) and (5).

$$f(j\omega) = FFT(s^m, 8192) \tag{4}$$

$$\varphi = arg\left(f\left(j\omega\right)\right) \tag{5}$$

To reduce the size of the data, the vector φ is resized to an array of 128 rows and 64 columns, and then it is converted to a row vector containing the sum of each column. In other words, the total sum of the elements in each column of this matrix is calculated in such a way that a vector φ^s of 64 elements is obtained, as shown in Equations (6) and (7).

$$\varphi^f = \begin{bmatrix} s\left(1\right) & \cdots & s\left(64\right) \\ \vdots & \ddots & \vdots \\ s\left(8128\right) & \cdots & s\left(8192\right) \end{bmatrix} \tag{6}$$

$$\varphi^s\left(j\right) = \sum_{i=1}^{128} \left|\varphi^f\left(i,j\right)\right| \tag{7}$$

Finally, to obtain the hash code, the integer and the fractional parts (considering only four significant figures) are taken (Equation (8)). These five digits interpreted as an integer value are converted to hexadecimal, and the least significant digit of each element is concatenated to obtain the 64-hex-digit hash code.

$$c = hex\left(\left\lfloor \varphi^s \cdot 10^4 \right\rfloor\right) \tag{8}$$

2.3. Cloud Storage

After generating the hash code of the signal, the metadata of the signal are stored in a database that is hosted in the cloud. These metadata include parameters such as the name, the date of creation of the recording, the duration of the recording, and the code and the time used to calculate the code. It should be noted that the storage created in Windows Azure is an SQL database with 32 MB of storage capacity; moreover, when the user finishes recording the audio signal, the metadata are stored in the database and cannot be deleted by the mobile user, i.e., after storage, the data are read-only.

Figure 2 shows the block diagram of the proposed module.

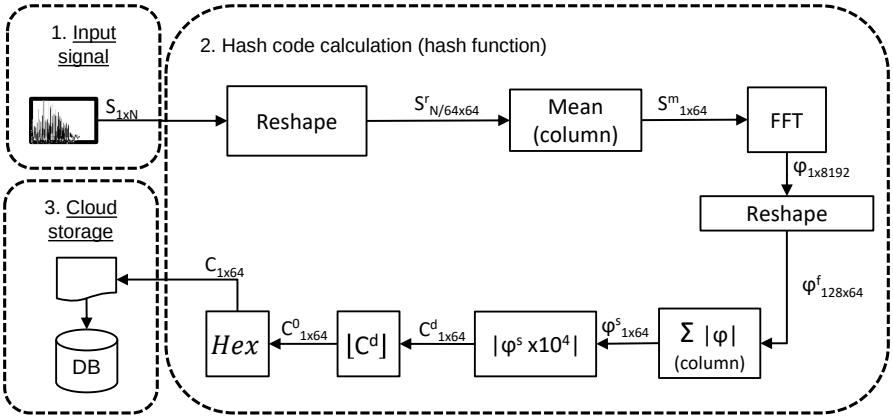

Figure 2. Block diagram of the proposed integrity verification module.

3. Implementation and Evaluation of the Method

To validate the proposed method, several tests were accomplished, focused on evaluating the performance of the hash function in terms of: (i) resistance to collisions, which implies determining if two different inputs produce the same output; (ii) runtime used by the mobile device to calculate the hash code with respect to the duration of the signal; and (iii) sensitivity of the code with respect to changes in the input signal.

For the evaluation of the aforementioned parameters, two mobile devices with the characteristics mentioned in Table 1 were used.

Table 1. Characteristics of the mobile devices used for the tests.

Specs\Model	Moto G	Nexus 5X
Processor	1.2 GHz Qualcomm Snapdragon 400, Quad-core	1.8 GHz Qualcomm Snapdragon 800, Hexa-core
OS	Android 5.1	Android 7.0
Memory	1 GB RAM	2 GB RAM

In each mobile device, 70 audio signals of different duration were recorded. To structure the results, seven ranges of duration were defined, with ten recordings in each range per device. The first range includes recordings of up to 10 s, while the second includes recordings between 10 and 20 s, doubling the duration limit in each new range. In this sense, there are 140 total recordings, as shown in Table 2.

Immediately after recording the audio signals, collision resistance, computational cost and sensitivity were evaluated. For the tests performed, stored metadata (Microsoft Azure) of each recording were used, such as recording name, creation date, hash code, and recording duration and compile time. On the other hand, seven audio recordings (one for each time range) were randomly selected, which were used to make minor modifications that later allowed generating and comparing the hash code of the modified signal with respect to the original one.

Table 2. Time ranges of the recordings acquired by each device.

Duration (s)	Number of Recordings per Device
0–10	10
10–20	10
20–40	10
40–80	10
80–160	10
160–320	10
320–640	10

3.1. Collision Evaluation

A collision refers to the event in which two different inputs (voice signals) produce the same hash code. Since it is mathematically impossible to design a hash function without collisions, it is important to test the proposed function to determine its variability. The theoretical value of the collision probability can be calculated through Equation (9) [24],

$$P = \left(\frac{1}{P}\right)^{L}, \tag{9}$$

where L is the number of digits of the hash code and P is the base of the numeral system used to represent the digits in the hash code. For the proposed scheme, $L = 64$ and $P = 16$. Thus, the theoretical probability of collision is given by,

$$P = \left(\frac{1}{16}\right)^{64} \approx 8.64 \times 10^{-78}. \tag{10}$$

This means that, theoretically, you expect to find a collision every $\approx 1.16 \times 10^{77}$ tests.

To verify the presence of collisions, the Hamming distance (HD) is calculated between each pair of codes of the test signals. The percentage of coordinates that differ from each other (HD) is given by Equation (11).

$$d_{st} = (\#(x_{sj} \neq x_{tj}/n)). \tag{11}$$

If the result obtained in HD is 0, it implies that the two hash codes are identical, whereas, if the obtained value is 1, it means that the hash codes are completely different.

In a complementary way, each pair of evaluated codes are compared to each other using the Pearson Correlation Coefficient (PCC) given by Equation (12), in order to determine which is the linear dependence between two codes. The goal here is to evaluate the degree in which a pair of hash codes are linearly related, in order to evaluate the similarity between them.

$$PCC = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}}, \tag{12}$$

where A and B are matrices or vectors of the same size. The result of the correlation can vary between -1 and 1; if the absolute PCC value is 1, it indicates a perfect linear relationship, whereas a PCC close to 0 means that there is no linear relationship between the data. The sign indicates the direction of the relationship, i.e., if both variables tend to increase or decrease at the same time, the coefficient will be positive, whereas if one variable tends to increase and the other to decrease, the coefficient will be negative. In practice, when evaluating the collision resistance in binary hash codes by means of the correlation coefficient, $PCC \neq 1$ is expected.

3.2. Computational Cost Evaluation

Since all the processing is done directly on the mobile device, it is important to consider the time it takes to execute the proposed function according to the duration of the input signal. Accordingly, it was decided to register the time it takes for the application to calculate the hash code for each recording. Then, an analysis of runtime versus signal duration was done.

3.3. Sensitivity Evaluation

To determine how the proposed hash function behaves against small modifications made to the input signal, six types of modifications were made in selected recordings. After this, the codes of the original signal were compared with those of the modified signals by using HD and PCC. The modifications made to the test signals are listed below:

1. Modifying the amplitude of a sample
2. Reversing the signal in time
3. Muting a word
4. Cutting out an audio fragment
5. Changing from 16 to 8 quantization bits
6. Changing from 16 to 24 quantization bits

For the first modification, it was decided to change only the tenth sample of the signal to 0; if the amplitude of this sample is already 0, the next sample is changed. For the second modification, the signal reversion was performed without modifying any of the samples. In the third modification, a word lasting less than 1 s was silenced. In the fourth modification (trimming), a fragment of the signal was eliminated (lasting less than 1 s). Finally, in the fifth and sixth modifications, the 16-bit signal was changed to 8 and 24 bits, respectively.

4. Results

4.1. Collision Resistance

As discussed above, to verify the presence of collisions, HD was calculated between each pair of codes. To determine the number of possible combinations without repetition of each pair of codes, the binomial coefficient was calculated for a set of n elements taken in groups of size r, where n and r are two positive integers, being n greater than or equal to r. For the number of tests performed, n corresponds to the number of audio recordings, i.e., 140. Likewise, since the evaluation was done between each pair of recordings, r corresponds to two elements. In conclusion, for the number of tests performed, the quantity of possible combinations among all the codes is 9730, as shown in Equations (13)–(15).

$$C(n,r) = \frac{n!}{r!\,(n-r)!} \tag{13}$$

$$C(140,2) = \frac{140!}{2!\,(140-2)!} \tag{14}$$

$$C(140,2) = 9730 \tag{15}$$

To validate in a practical way the collision resistance, the histograms of the comparison data between each pair of codes for the 140 analyzed signals (9730 cases) are shown in Figures 3 and 4. Regarding Hamming distances, the maximum value for HD was 0.61 and the minimum value was 0.39, with a standard deviation of 0.0309. This means that, in the worst case, 39% of different coordinates were obtained between the two compared codes. It is possible to observe that all the results obtained were grouped around 0.4999.

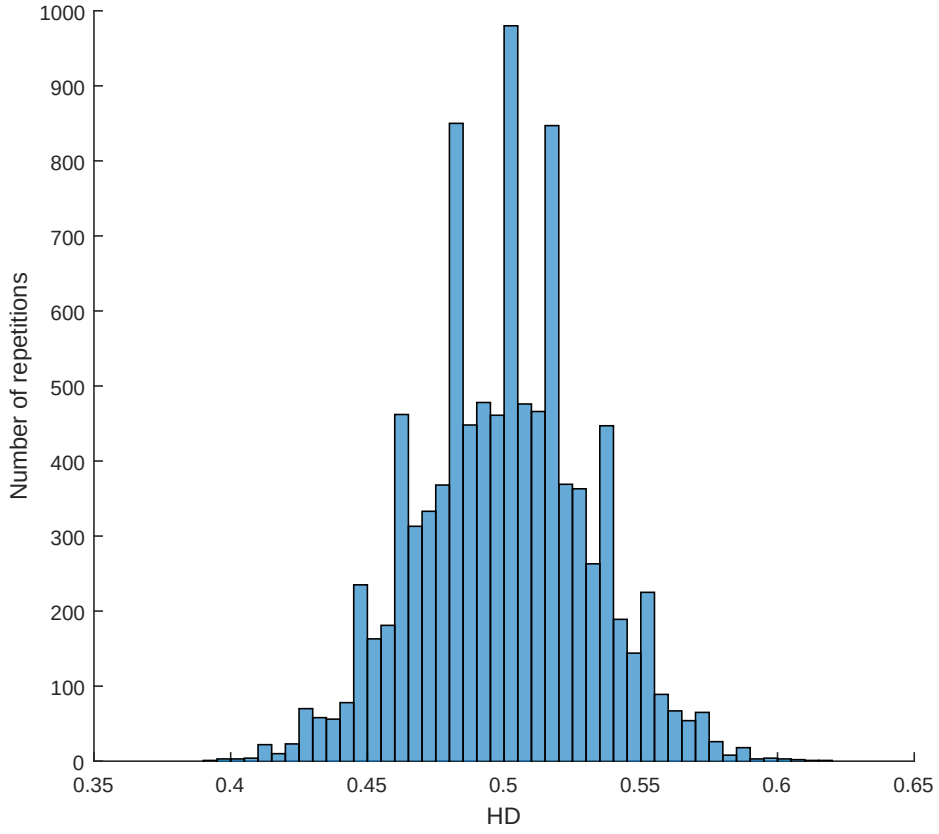

Figure 3. Histogram of HD between all the obtained codes.

When calculating the *PCC* and making the histogram of the data, it is possible to observe that most of the data are concentrated in values around 0, which implies that there is no linear relationship between the compared codes (Figure 4).

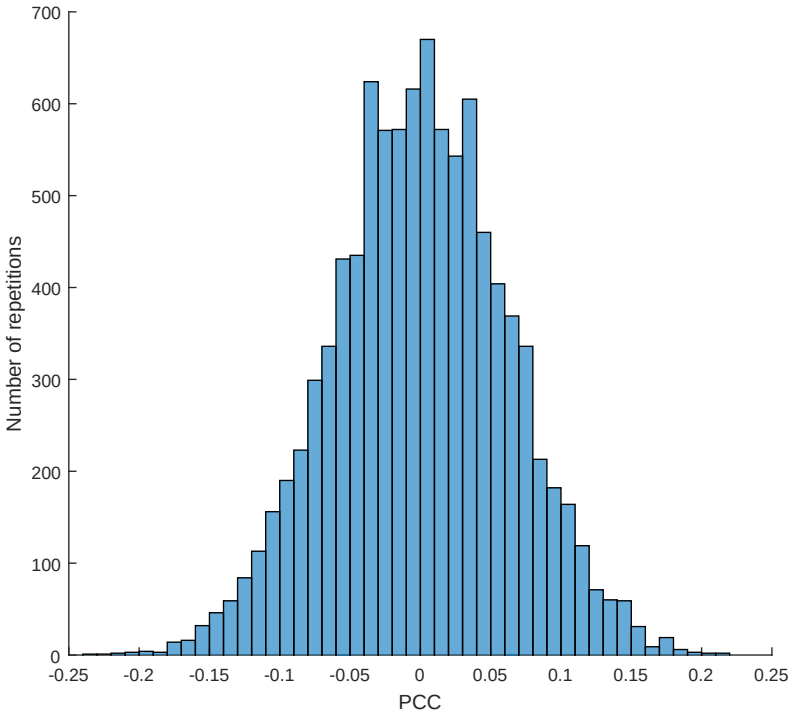

Figure 4. Histogram of *PCC* between all the obtained codes.

According to the above, it was found that for the tests performed no collisions were detected.

4.2. Computational Cost

With the metadata stored in the database corresponding to the runtime of the hash function, it was possible to determine the computing time according to the mobile device and the duration of the voice recording. This relationship is shown in Figure 5, where the *X*-axis has a logarithmic scale (base 2). Based on the obtained data, it can be said that the execution time will depend on the characteristics of the device, i.e. the lower are the resources, the longer is the execution time, although in general terms the execution time is low.

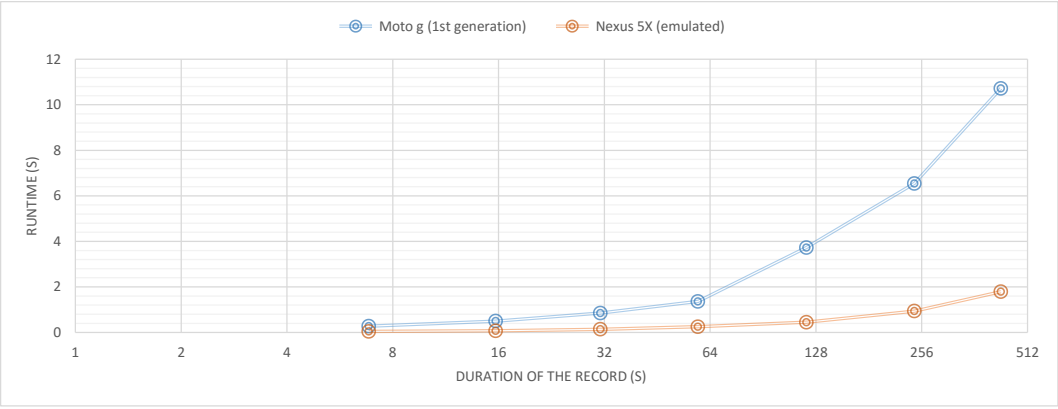

Figure 5. Runtime of the hash function based on recording duration, according to the mobile device.

In Figure 5, it is possible to determine that the runtime of the proposed hash function for the mobile device Moto g is related to Equation (16), where x is the duration of the input recording.

$$y \approx -1 \times 10^{-5}x^2 + 0.0303x - 0.0365 \quad (16)$$

It is also possible to determine that the execution time of the proposed hash function for Nexus 5x is related to Equation (17).

$$y \approx 2 \times 10^{-6}x^2 + 0.0033x + 0.0308 \quad (17)$$

4.3. Sensitivity

After making the six changes described above for each of the selected signals, the codes of the original signals and the modified signals were compared, obtaining the results in Tables 3 and 4.

Table 3. Results obtained in the Hamming distance comparison between the hash codes of the original recording and the tampered recording.

Recording/ Modification	1	2	3	4	5	6	7	Mean	Standard Deviation
Modification 1	0.52	0.51	0.48	0.46	0.47	0.10	0.48	0.43	0.15
Modification 2	0.48	0.47	0.54	0.46	0.43	0.53	0.45	0.48	0.04
Modification 3	0.52	0.53	0.50	0.47	0.52	0.49	0.51	0.51	0.02
Modification 4	0.51	0.49	0.48	0.54	0.52	0.47	0.43	0.49	0.04
Modification 5	0.45	0.48	0.50	0.48	0.50	0.53	0.46	0.49	0.02
Modification 6	0.49	0.48	0.48	0.52	0.49	0.49	0.50	0.49	0.01

For the first modification made, the only recording that did not obtain a significant change was number six (Recording 6, Modification 1 in Table 3, i.e., $HD = 0.10$). However, even though the change consisted in modifying the value of a single sample of the signal, the hash code obtained when applying the modification differs by 10%. The remaining five modifications guarantee that at least 40% of the bits of the hash code change. The above implies that the proposed function is quite sensitive to the input data (i.e., a change of a sample in the input signal generates a significant change in the output).

Table 4. Results obtained in the *PCC* comparison between the hash codes of the original recording and the modified recording.

Recording/ Modification	1	2	3	4	5	6	7	Mean	Standard Deviation
Modification 1	−0.05	−0.02	0.05	0.09	0.06	0.80	0.03	0.14	0.29
Modification 2	0.03	0.06	−0.07	0.07	0.13	−0.07	0.11	0.04	0.08
Modification 3	−0.05	−0.05	−0.01	0.06	−0.03	0.02	−0.02	−0.01	0.04
Modification 4	−0.02	0.02	0.04	−0.08	−0.05	0.06	0.14	0.02	0.07
Modification 5	0.09	0.03	−0.01	0.03	0.00	−0.05	0.07	0.02	0.05
Modification 6	0.02	0.03	0.04	−0.05	0.02	0.02	0.00	0.01	0.03

5. Conclusions

A system for audio integrity based on mobile applications and cloud storage is proposed. In the mobile device, a hash function specifically designed for audio recordings is executed. Once the hash code is obtained, it is transmitted to a database in the cloud.

The performance evaluation of the proposed hash function was focused on collision resistance and sensitivity. After 9730 tests, no collisions were found, not even between two pairs of very similar recordings (for example, an audio recording and its altered version with only one modified sample). Among all the recordings, the lowest *HD* (Hamming Distance) was 0.4; in the case of a very similar

signal, the lowest *HD* was 0.1. This means that our proposed hash function is very sensitive to the input signal.

On the other hand, the computational cost to calculate the hash code was measured. With two different mobile devices, a nonlinear relationship was obtained between the duration of the recording and the execution time. However, in both cases, it takes up to 2 s for each minute of the audio recording.

As a main conclusion, our proposed system is feasible to be used as a useful tool for audio integrity within a forensic field. A user can record a conversation with the app and use it as evidence within a legal process. Legal authority can verify the integrity of the evidence with the hash code stored in the cloud.

Author Contributions: Conceptualization, D.R.; Formal analysis, D.M.B.; Investigation, J.A.A.; Methodology, D.R.; Software, J.A.A.; Validation, D.M.B.; Writing—original draft, D.R. and J.A.A.; and Writing—review and editing, D.M.B.

Funding: This research was funded by Universidad Militar Nueva Granada-Vicerrectoría de Investigaciones grant number IMP-ING-2136.

Acknowledgments: The authors would like to thank the reviewers for their comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Maher, R.C. Audio forensic examination. *IEEE Signal Process. Mag.* **2009**, *26*, 84–94. [CrossRef]
2. ISO/IEC. *Information Technology–Security Techniques–Incident Investigation Principles and Processes, ISO/IEC FDIS 27043*; Technical Report; ISO/IEC: Geneva, Switzerland, 2014.
3. ISO/IEC. *Information Technology–Security Techniques–Guidelines for Identification, Collection, Acquisition and Preservation of Digital Evidence, ISO/IEC 27037*; Technical Report; ISO/IEC: Geneva, Switzerland, 2012.
4. SWGIT. *Best Practices for Maintaining the Integrity of Digital Images and Digital Video, SWGIT Document Section 13, Version 1.1*; Technical Report; Scientific Working Group on Imaging Technology: 2012. Available online: https://www.swgit.org/pdf/Section%2013%20Best%20Practices%20for%20Maintaining%20the%20Integrity%20of%20Digital%20Images%20and%20Digital%20Video?docID=54 (accessed on 25 June 2019).
5. Tanenbaum, A.S. *Redes de Computadoras*; Pearson Educación: Madrid, Spain, 2003.
6. Renza, D.; Lemus, C.; Ballesteros L, D.M. Authenticity verification of audio signals based on fragile watermarking for audio forensics. *Expert Syst. Appl.* **2018**, *91*, 211–222. [CrossRef]
7. Ballesteros, L.D.; Renza, D.; Rincon, R. Gray-scale images within color images using similarity histogram-based selection and replacement algorithm. *J. Inf. Hiding Multimed. Signal Process.* **2015**, *6*, 1156–1166.
8. Shih, T.F.; Chen, C.L.; Syu, B.Y.; Deng, Y.Y. A cloud-based crime reporting system with identity protection. *Symmetry* **2019**, *11*, 255. [CrossRef]
9. Camacho, S.; Ballesteros, L.D.; Renza, D. A cloud-oriented integrity verification system for audio forensics. *Comput. Electr. Eng.* **2019**, *73*, 259–267. doi:10.1016/j.compeleceng.2018.11.022. [CrossRef]
10. Tian, Z.; Li, M.; Qiu, M.; Sun, Y.; Su, S. Block-DEF: A secure digital evidence framework using blockchain. *Inf. Sci.* **2019**, *491*, 151–165. doi:10.1016/j.ins.2019.04.011. [CrossRef]
11. Amato, F.; Cozzolino, G.; Moscato, V.; Moscato, F. Analyse digital forensic evidences through a semantic-based methodology and NLP techniques. *Future Gener. Comput. Syst.* **2019**, *98*, 297–307. [CrossRef]
12. Singh, M.; Garg, D. Choosing Best Hashing Strategies and Hash Functions. In Proceedings of the 2009 IEEE International Advance Computing Conference, Patiala, India, 6–7 March 2009; pp. 50–55.
13. Steinebach, M.; Yannikos, Y.; Zmudzinski, S.; Winter, C. Advanced multimedia file carving. In *Handbook of Digital Forensics of Multimedia Data and Devices*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2015; pp. 219–269.
14. Balasubramanian, K.; Rajakani, M. Problems in Cryptography and Cryptanalysis. In *Algorithmic Strategies for Solving Complex Problems in Cryptography*; IGI Global: Hershey, PA, USA, 2018; pp. 23–39. doi:10.4018/978-1-5225-2915-6.ch002.

15. Stevens, M.; Bursztein, E.; Karpman, P.; Albertini, A.; Markov, Y. The First Collision for Full SHA-1. In *Advances in Cryptology–CRYPTO 2017*; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 570–596. doi:10.1007/978-3-319-63688-7_19.
16. Ghonaim, W.; Ghali, N.I.; Hassanien, A.E.; Banerjee, S. An improvement of chaos-based hash function in cryptanalysis approach: An experience with chaotic neural networks and semi-collision attack. *Memetic Comput.* **2013**, *5*, 179–185. [CrossRef]
17. Sobti, R.; Geetha, G. Cryptographic hash functions: A review. *IJCSI Int. J. Comput. Sci. Issues* **2012**, *9*, 461–479.
18. Wang, X.; Yu, H. How to break MD5 and other hash functions. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2005, Aarhus, Denmark, 22–26 May 2005; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3494, pp. 19–35.
19. Stevens, M. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, 26–30 May 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 245–261.
20. Chen, N.; Xiao, H.D.; Wan, W. Audio hash function based on non-negative matrix factorisation of mel-frequency cepstral coefficients. *IET Inf. Secur.* **2011**, *5*, 19–25. [CrossRef]
21. Ghouti, L.; Bouridane, A. A Robust Perceptual Audio Hashing using Balanced Multiwavelets. In Proceedings of the 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, Toulouse, France, 14–19 May 2006; Volume 5, p. V.
22. Jiao, Y.; Yang, B.; Li, M.; Niu, X. MDCT-Based Perceptual Hashing for Compressed Audio Content Identification. In Proceedings of the 2007 IEEE 9th Workshop on Multimedia Signal Processing, Crete, Greece, 1–3 October 2007; pp. 381–384.
23. Ballesteros, D.M.; Renza, D.; Ortiz, H.D. Función resumen perceptual para verificación de integridad en audio forense. *Ing. Y Cienc.* **2017**, *13*, 167–183. [CrossRef]
24. Renza, D.; Mendoza, S.; Ballesteros L, D.M. High-uncertainty audio signal encryption based on the Collatz conjecture. *J. Inf. Secur. Appl.* **2019**, *46*, 62–69. [CrossRef]