Article

Securing Audio Using AES-based Authenticated Encryption with Python

Jessy Ayala [1]

[1] New York University, Tandon School of Engineering; ja3995@nyu.edu

Featured Application: Securing communication of audio files utilizing symmetric authenticated encryption.

Abstract: The focus of this research is to analyze the results of encrypting audio using various authenticated encryption algorithms implemented in the Python cryptography library for ensuring authenticity and confidentiality of the original contents. The Advanced Encryption Standard (AES) is used as the underlying cryptographic primitive in conjunction with various modes including Galois Counter Mode (GCM), Counter with Cipher Block Chaining Message Authentication Code (CCM), and Cipher Block Chaining (CBC) with Keyed-Hashing for encrypting a relatively small audio file. The resulting encrypted audio shows similarity in the variance when encrypting using AES-GCM and AES-CCM. There is a noticeable reduction in variance of the performed encodings and an increase in the amount of time it takes to encrypt and decrypt the same audio file using AES-CBC with Keyed-Hashing. In addition, the corresponding encrypted using this mode audio spans a longer duration. As a result, AES should either have GCM or CCM for an efficient and reliable authenticated encryption integration within a workflow.

Keywords: AES; Audio analysis; Authenticated encryption; Cryptography; Python

1. Introduction

Cryptography is used worldwide for adhering to the security CIA triad: confidentiality, integrity, and availability. In an environment where mobile devices have become ubiquitous, voice messages are more common than one may think. In 2018, it was reported that about 200 million voice messages are delivered over WhatsApp daily [1]. Being able to intercept and alter voice messages without the other party knowing can result in unintended altercations. Furthermore, IoT devices that rely on direct communication with humans can also have consequences if voice commands are modified by an adversary. Cryptographic implementations rooted in authentication can be used to help keep these communications genuine.

The framework of authenticated encryption schemes assures authenticity and confidentiality of information. These schemes can be constructed in many ways, which includes joining symmetric and block ciphers to meet modern security standards. The Advanced Encryption Standard (AES), originally coined as Rijndael in 1998, is a symmetric block cipher that is practically impossible to break by brute-force. AES has proven to be useful for maintaining confidentiality and integrity of message data in multimedia when combined with digital signatures [2]. It is worthwhile to further analyze how AES can be used with different modes to encrypt audio and provide perspective on which implementations prove to be more effective than the others.

Authors in [3] demonstrate the efficiency of using AES over DES for audio file encryption via simulations utilizing tools provided in MATLAB; however, this fails to capture the dimension of authenticity to ensure messages are not altered by the time they reach the other party involved. Furthermore, performing encryption on sensitive data is inherently safer when done using pre-built libraries as opposed to hands-on, reconstructed implementations. The Python cryptography library provides such a framework [4]. In this paper, built-in authenticated encryption primitives from this library are used for encrypting audio and the results are primarily analyzed for their variance, speed, and reliability.

2. Materials and Methods

The approach presented is intended to give an overview of various AES-based authenticated encryption algorithms as methods for encrypting audio. An original audio file is used as a reusable input for encryption by a handful of cryptographic primitives. In this section, a brief description of each algorithm used for investigation is provided in the context of the Python *cryptography* library. In addition, initial analysis of the original audio file as a baseline. Source code can be found on Zenodo [5].

2.1. AES Primitives for Analysis

2.1.1. AES Galois Counter Mode (AES-GCM)

The AES-GCM is an authenticated encryption scheme with the support of providing integrity for associated data as an extra layer of security. AES-GCM requires a key of either 128, 192, or 256 bits. Once generated using *AESGCM.generate_key* and passed into the *AESGCM* constructor, a random 12-byte one-time nonce, recommended length, needs to be created. Messages can then be encrypted using the nonce, data for encryption, and the unencrypted associated data.

2.1.2. AES Counter with Cipher Block Chaining Message Authentication Code (AES-CCM)

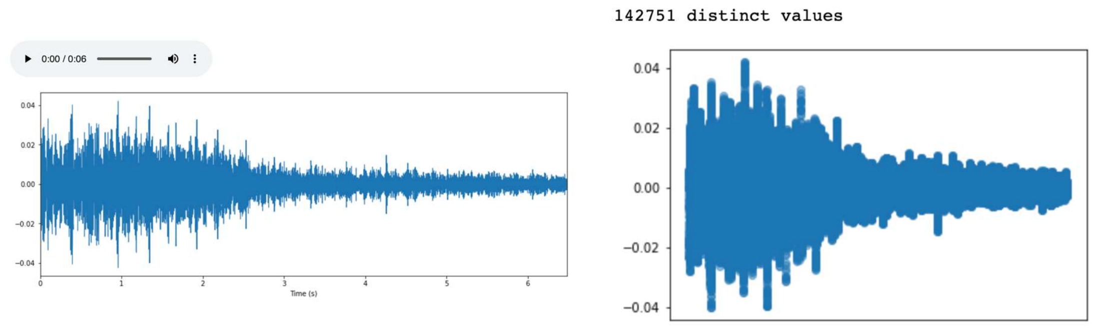
Like AES-GCM, AES-CCM is also an authenticated encryption scheme with the support of providing integrity for associated data as an extra layer of security. AES-CCM requires a key of either 128, 192, or 256 bits. In addition, an authentication tag of either 4, 6, 8, 10, 12, 14, or 16 bytes is required. For secure purposes, 16-bytes is the default length. Once the key generated using *AESCCM.generate_key* and the authentication tag length are set, these are passed into the *AESCCM* constructor. Afterwards, a random 7 to 13 byte one-time nonce needs to be created. Messages can then be encrypted using the nonce, data for encryption, and the unencrypted associated data.

2.1.3. Fernet: AES Cipher Block Chaining (AES-CBC)

Fernet is an algorithm that utilizes multiple cryptographic primitives; furthermore, Fernet is an implementation of AES-CBC using a 128-bit key for encryption with PKCS7 padding and HMAC using SHA-256 for authentication. For HMAC purposes, a password and salt are used as a part for generating the key for AES-CBC to use in the *Fernet* constructor. Afterwards, encrypting the message is as simple as passing it to the Fernet *encrypt* method.

2.2. Original Audio

The algorithms described are used to encrypt bytes of an audio file. The duration of the file, in seconds, the original wave plot, scatter plot, and number of distinct data points generated from the audio file are shown Figure 1(a-b) respectively.



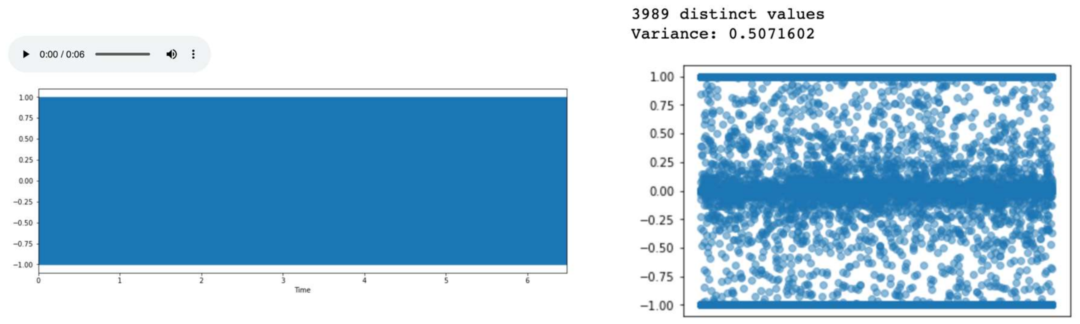
(**a**) (**b**)

Figure 1. Original audio file representations (**a**) Time duration and wave plot; (**b**) Number of distinct values and scatterplot.

2.3. *Software and Hardware Usage*

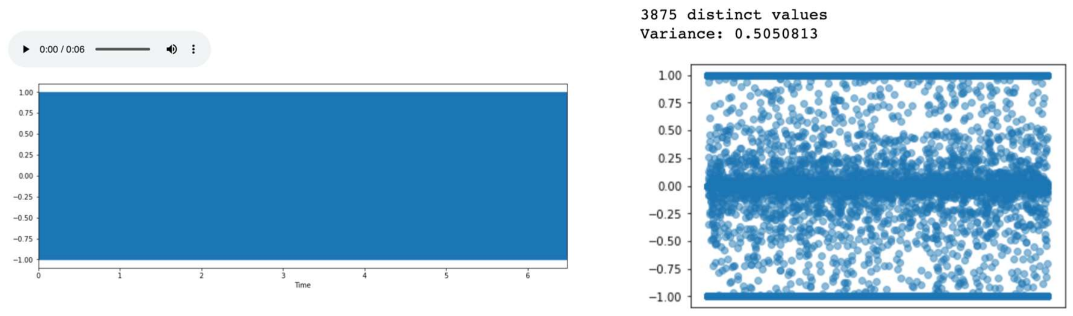
Python 3.9.1 is used for the simulations performed and the version of the cryptography library used is 3.4.7 [4]. Jupyter Notebook is used as the simulation environment. The various package versions installed can be found at the bottom of the *encrypted-audio-analysis.ipynb* file [5]. For hardware, a 2019 MacBook Pro installed with Catalina 10.15.4 is used.

3. Results



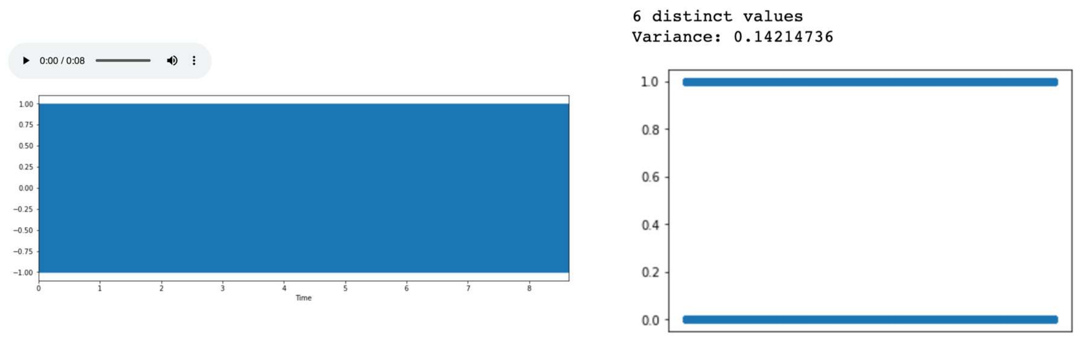
(**a**) (**b**)

Figure 2. Representations of the original file encrypted with AES-GCM (**a**) Time duration and wave plot; (**b**) Number of distinct values and scatterplot.



(**a**) (**b**)

Figure 3. Representations of the original file encrypted with AES-CCM (**a**) Time duration and wave plot; (**b**) Number of distinct values and scatterplot.



```
6 distinct values
Variance: 0.14214736
```

(**a**) (**b**)

Figure 4. Representations of the original file encrypted with Fernet AES-CBC (**a**) Time duration and wave plot; (**b**) Number of distinct values and scatterplot.

From a listening standpoint, all encrypted audio files consist of white noise as wave plots show this in Figure 2(a), Figure 3(a), and Figure 4(a). This can also be heard when playing the encrypted audio in the respective cells in the encrypted-audio-analysis.ipynb file [5]; thus, using any of the investigated primitives will produce incoherent mp4 files for transfer.

The number of distinct values is drastically smaller when using AES-CBC for encryption. This can be seen when comparing the wave scatterplots shown in Figure 2(b), Figure 3(b), and Figure 4(b), as there are roughly 650 times more distinct outputs in the AES-GCM and AES-CCM encrypted audio files. The variance is also smaller by a scale factor of approximately 3.64 for AES-CBC when compared to AES-GCM and AES-CCM. The exact values of each encrypted output array can be explored when running the cells in [5]; however, it remains $|\text{AES-CBC}_{\text{ENCRYPT}}| < |\text{AES-GCM}_{\text{ENCRYPT}}| \sim |\text{AES-CCM}_{\text{ENCRYPT}}|$ in a given iteration.

Table 1. Compiled statistics as a result of using the different primitives for encryption and decryption

Metric	AES-GCM (Fig. 2)	AES-CCM (Fig. 3)	AES-CBC (Fig. 4)
Encryption Duration	0.002 seconds	0.002 seconds	0.010 seconds
Encrypted File Duration	6 seconds	6 seconds	8 seconds
Variance	0.5072	0.5051	0.1421
Number of Distinct Values	3989	3875	6
Decryption Duration	0.001 seconds	0.001 seconds	0.006 seconds

Table 1 shows negligible differences between the AES-GCM and AES-CCM algorithms when referencing the various statistics collected after encrypting the original audio file. When these algorithms are used for encrypting and decrypting the audio file, it takes about the same amount of time to perform the respective cipher operations. When running the corresponding algorithms using AES-CBC, the scale factor for encryption is 5 and the scale factor for decryption is 6.

The encrypted file duration for AES-CBC is two seconds longer than the ones generated using AES-GCM and AES-CCM, as they both maintain the duration of the original audio. As for the distinct values and variance produced, AES-GCM and AES-CCM show similar outputs. Minimal differences can be attributed to the random nonce used when the primitives are generating keys. When using AES-CBC for encryption, the number of distinct values and variance is drastically smaller. This may be ideal as the mapping encrypted space is smaller; however, the duration it takes and resulting file size imply otherwise when intended for implementation.

4. Discussion

Based on the presented findings, any of these algorithms can be chosen for encrypting small audio files when it comes to the context of sending and receiving messages; however, encrypting with AES-CBC results in a larger file size, which may cause hassle when trying to communicate a longer voice message to one another. AES-GCM or AES-CCM are recommended for audio encryption, choosing one of these two algorithms should then be decided based on the underlying specifications desired for integrating within a communication workflow. Since authenticated encryption algorithms are used as the underlying framework, confidentiality and integrity are assumed; thus, meeting standards for guaranteed genuine communication between two parties and preventing audio tampering attacks.

Limitations of this brief study include only using cryptographic primitives from the Python *cryptography* library and the usage of only a single original audio file for analysis. Future directions include running multiple simulations with audio files of various sizes, using other metrics of analysis, and determining if there may be cases where an algorithm like AES-CBC should be used for encryption. Implementing these algorithms in a communication system for the purpose of practical usage and performing analyses from a consumer perspective to measure quality would determine the extent in which utilizing such algorithms are practical.

Selecting suitable cryptographic primitives for encrypting different types of media can be difficult. The results from this brief study give insight to how using different primitives from the same framework can affect the variance, speed, and distinctness of encrypted output using a relatively small audio file. When using the Python *cryptography* library, AES-GCM or AES-CCM should be used for encrypting audio communication to achieve larger variance and faster encryption and decryption times for successfully maintaining confidentiality and integrity.

Funding: This research received no external funding

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data supporting reported results can be found on Zenodo [5].

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

References

1. S. Keach, "How voice messages are taking over texts on WhatsApp and iMessage," *The Sun*, 2018. Available: https://www.thesun.co.uk/tech/6815812/texts-voice-messages-whatsapp-imessage-switching/ (accessed on 18 June 2021)
2. J. Kim and S. R. Basavarasu, "On the voice and image data encryption using advanced encryption standard (AES) in counter mode for multimedia broadcasting," 2015 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 2015, pp. 1-4, doi: 10.1109/BMSB.2015.7177191
3. H. Bijlani, D. Gupta and M. Lovanshi, "Audio Encryption Optimization," *2018 8th International Conference on Communication Systems and Network Technologies (CSNT)*, 2018, pp. 199-203, doi: 10.1109/CSNT.2018.8820275
4. Python Cryptographic Authority, (2014) cryptography (Version 3.4.7). https://github.com/pyca/cryptography
5. J. Ayala, jayala-29/Encrypted-Audio-Analysis: Zenodo, 2021. doi: 10.5281/ZENODO.5043451