Computer Science and Information Technologies
Vol. 2, No. 2, July 2021, pp. 87~94
ISSN: 2722-3221, DOI: 10.11591/csit.v2i2.p87-94

Designing a secured audio based key generator for cryptographic symmetric key algorithms

Avinash Krishnan Raghunath[1], Dimple Bharadwaj[2], M Prabhuram[3], Aju D[4]
[1,2,3]M-Tech, CSE Information Security, Vellore Institute of Technology, Vellore, Tamil Nadu, India
[4]Associate Professor, SCOPE, Vellore Institute of Technology, Vellore, Tamil Nadu, India

Article Info

Article history:

Received Oct 18, 2020
Revised Feb 2, 2021
Accepted Mar 5, 2021

Keywords:

Cryptography
Deterministic
Entropy keys
Key generator
TRNG

ABSTRACT

Cryptography is a technique to secure data transmissions and ensure confidentiality, authenticity and integrity of data exchanged over the digital networks by utilizing mathematical algorithms to transform the plain text (original message) to cipher text (encrypted message) using a key or seed value. The general con-sensus regarding the use of non-deterministic true random numbers (TRN) which are generated from the physical environment such as entropy keys, at-mospheric noise, etc., as a public or private key has received limited encour-agement due to the demanding hardware requirements needed to extract the necessary data from the environment. Therefore, this research aims at design-ing and developing a lightweight program to generate a True Random Number (TRNG) key using live audio recordings which is further randomized using system date and time. These TRNs can be used to replace the deterministic pseudo random number cryptographic keys that are presently used by indus-tries for a symmetric key encryption algorithm which devolves the algorithm to being conditionally secured. Using the audio based TRNG key would render the same encryption algorithm as unconditionally secured.

Corresponding Author:

Avinash Krishnan Raghunath
M-Tech, CSE Information Security
Vellore Institute of Technology
Vellore, Tamil Nadu, India
Email: avish1190@gmail.com

1. INTRODUCTION

The most secured cryptography technique considered today in use is the One-Time Pad (OTP) algorithm which is used to generate a random key every time encryption is performed. The random number key generator used for OTP is a pseudo random number key generator that follows logical instructions from a computer by utilizing a mathematical formula to produce a number that appears random and is used as a key to encrypt the data [1]. The number generated appears to have some degree of randomness, but after some research and statistical analysis, it was found that these numbers are deterministic in nature and are computationally predetermined [2]. Such algorithms utilize a seed value to generate keys using some defined mathematical formula. Under perfect cryptanalysis, if the start point of the random number generator sequence is known then that generator fails as one can predict the keys and can also predetermine the periodicity of the generator [3]. Hence such generators are only successful and have shelf life till their starting point and mathematical function used are unknown, thus they are termed pseudo random number generator.

To resolve this flaw, the focus was moved to utilize true random number generators as a source for the key value. These True Random Number Generators (TRNG) utilize the data extracted from the physical environment such as entropy keys, atmospheric noise, wave noise, etc. However, such a data extraction has a

heavy hardware requirement (e.g. hypersensitive microphones, highly efficient entropy detectors, high capacity storage space, etc.) which requires a massive financial investment. Therefore, the general consensus of industries in utilizing TRNG keys has received very little encouragement due to the heavy hardware financial investment just for capturing input data for a TRN key generator.

The primary goal of this research work is to design and develop a lightweight program that is capable of using the current laptops or PCs hardware to generate a True Random Number (TRNG) key using live audio recordings which is further randomized using system date and time. These TRNs can be used to replace the deterministic pseudo random number cryptographic keys that are presently used by industries for symmetric key encryption algorithms which devolves the algorithm to being conditionally secured (cipher text can be decrypted to plain text without key knowledge over a long duration of time, that can be more than the message's time to live). Using the audio based TRNG key would render the same encryption algorithm as unconditionally secured (cipher text doesn't contain sufficient data to uniquely determine the plain text without key knowledge). Audio Recordings in general are capable enough to produce truly random numbers. However, we have tried to consider a special use case where a user ends up with a live recording without actually speaking (may occur if the user uses a headset with a faulty mic), which would result in a static audio recording. To accommodate this scenario, the modulo integer value of the current system date and time is utilized to further randomize the program output. Upon execution, the lightweight Audio based TRN Key Generator program (proof of concept was created using JAVA 13) produces a randomized 64-bit binary key which can be directly used as a key input in any cryptographic symmetric key algorithm such as Advanced Encryption Standard (AES-256), Triple Data Encryption Algorithm (TDES or 3DES), One-Time-Pad (OTP), etc. Future enhancements are planned to variablize the final output key size so the program can produce keys of variable length such as 256 bits, 512 bits, 1024 bits, etc.

Pseudo Random Number Generators (PRNG) are used to generate symmetric or asymmetric keys to be used in encryption algorithms. There are many ways PRNGs can be compromised. One of the possible attacks to compromise the PRNGs is to attack the respective algorithm through which the random generator is working and making the keys predictable, thus making PRNG a deterministic machine. S. Indarjani, G. Supriyatno, A. Nugraha and I. M. M. Astawa [4] tested the Pseudo random number generators with Insertion Attack and effects of it on PRNG using NIST (National Institute of Standards and Technology) Randomness Tool. Previous researches concluded that inserting a single bit for attacking PRNG implemented AES had 17.7% failed tests for AES-128 and 24.44% for AES-192 [5]. Researchers from Indonesia conducted tests on four algorithms of PRNGs, AES standard PRNG, ANSI X9.31 i.e. NIST recommended RNG, Dragon Stream Cipher and Rabbit Stream Cipher using concepts of NIST randomness test tool with a level of significance = 0.01 and concluded that the Dragon Stream Cipher algorithm to be the strongest among all four [6]. AES was found to be severely compromised at the block level and concluded that insertion attack using bits can compromise the characteristic randomness of target sequences [7]. Some PRNGs can be compromised using a known plain text attack.

Ahmad Amro and El-Sayed M. El-Alfy [8] tried to improve the PRNG based text Encryption and analysed the PRNGs for the known plain text attack. They proposed to improve security using PRNG based on stream cipher. But the proposed model could not withstand attacks and failed to cipher text only attack that was using the brute force method to crack especially when the keys domain is very small. Chosen plain text attack is also possible and chosen cipher text attack can be successfully accomplished under certain circumstances [9]. Thus, the proposed model does not have enough efficiency to serve its purpose. To overcome all these shortcomings, it is better to switch to True Random Number Generators which uses resources, either natural or hardware, like temperature, noise or keystrokes. E. M. M. Manucom, B. D. Gerardo and R. P. Medina [10] from the Philippines have suggested a way to improve the randomness to generate keys for One Time Pad and tested them for various frequency tests, mono bit tests, within a block test and run test. They studied and analyzed the work of Easttom, where the author has analyzed and stated the Lehmer algorithm, a linear congruential generator (LCG) which generates a pseudo random number and uses the following formula, $X_{t+1} = ((c+X_t*a) \bmod n)$, where n is a prime number.

Another well-known algorithm for generating PRNG that utilizes the middle square algorithm has some more limitations. Foremost, the seed value of this algorithm is zero, which reduces the succeeding key values to zero. Periodicity of some seed values is very small i.e. shorter cycles with frequent repetitions. Later, a true random generator using mouse movements has been proposed using the OR and AND operations to mouse coordinates to generate the respective keys. They used correlation analysis to measure the degree of relationship of keys, plain text and cipher text [11]. Such True Random Number generators are considered difficult for practical implementation. Moreover, the key distribution process is considered to be very impractical and overhead [12]. This research work tries to come up with a solution to the non-deterministic generating process and uncontrolled entropy by using a human voice as a cryptographic key to generate an unconditionally secure key generator. In one of the research articles published by Damir Omerasevic and his

team, the randomness of various media file types such as FLV, MP3, WAV, JPEG, and so on was compared so as to identify a viable candidate to be used for producing random keys which later be used in session key generation [13]. The authors have performed a C program implementation for the analysis methodologies by using the first part of the random number sequence test program. And the additional test scripts were generated to process the tests at a faster pace. The main purpose was to identify the randomness of different media files so as to be utilized for the encryption keys [14].

There have been other proposals, utilizing the biometric properties such as retinal scan, face scan, fingerprints, and so on for random number generation [15]. But one of the major concerns for using these techniques is its larger processing time that delays the encryption process. Therefore, the randomness for media files was investigated thoroughly so that it is used for the encryption. True Random Number Generators (TRNGs) uses an energy source and combine them with a processing function to generate random numbers. TRNG sources can vary from computer hardware parts that produce sound during processing, to the atmospheric or natural phenomenon like winds noise, traffic noise, and so on. And this results in producing desired outputs that cannot be determined using the mathematical functions and henceforth they are known as the non-deterministic processes. For implementing the TRNG for key generation, the media file types were tested to access their randomness using various tests such as Entropy test to measure randomness, Serial correlation test to check relations between variables over different intervals of time, Arithmetic mean test and Lempel-Ziv compression test [16]. The respective test results indicated that the YouTube videos extracted in the form of FLV format generate the highest randomness which is closely followed by MP3 and JPEG file types [17]. With respect to the research world carried out by various authors, it is observed that these file types are the best candidate to be considered for generating random numbers and can therefore be used for creating cryptographic encryption keys.

Pseudo Random Number Generators (PRNG) have their pros and cons just like True Random Number generators (TRNG). By comparing both of these random number generators, insights on their strengths are provided in the application perspective. Pseudo-RNG's execution does not proceed in an expected manner when randomized events such as roulette wheel, lottery number extraction or a simple dice roll are considered [18]. The term "pseudo" in a PRNG can give a certain degree of visibility when we compare it with TRNG. Similar to the logical or binary instructions for a processor, PRNG's are recognized as algorithms that make use of various mathematical concepts and formulae to generate a series of numbers. The generated series of numbers provides the illusion of being completely randomized [19]. Due to extensive years of research being invested in developing and improvising various PRNG algorithms, programs that deploy these algorithms can sometimes replicate the true random sequences [20]. However, there is always room for improvement since these number sequences are never truly in random nature.

A detailed comparison of TRNG and PRNG algorithms provides a better insight with respect to its comparison and algorithm effectiveness. For example, when the outcome of PRNG algorithms provides the result of dice rolls in numbers, at first glance these results might appear random [21]. However, the statistical analysis of those results provided enough evidence that these randomized number outcomes are pre-determined and not truly random. This proves the fact that the outcomes of these algorithms can easily be predicted, controlled, standardized and measured [22]. TRNG on the other hand shows completely different behaviour due to the fact that its results are unpredictable and completely randomized. If one’s expectation is that a personal computer should be able to generate a randomized number series then the computer has to rely on events that occur in physical nature or natural phenomenon [23]. This might include the entropy of various radioactive isotopes, naturally occurring audio and video disturbances in the environment or the wave pattern of a water body such as oceanic waves [24]. There is one minor issue with the cost efficiency of TRNG when compared with PRNG because the input devices are subjected to physical deterioration due to wear and tear which is not noticed in PRNGs [10], [25]. Consequently, the research work that is carried out aims to prove that with current technological infrastructure available on a personal computer (PC), TRNG can be implemented in real life systems such as conferencing solutions, audio call, VOIP call and so on.

2. PROPOSED METHODOLOGY

By considering the existing work as a baseline, the live audio recording of the sender is utilized to generate the respective MP3 audio file, which is then converted into Base 64 format. Subsequently, the characters in the Base 64 text file are extracted and stored in an integer array which is mixed with the modulo integer value of the current system date and time to further randomize the output key. The final output is converted to a Binary String array of 64-bit length and returned to be printed on the console window. The basic walkthrough is to setup and synthesize a 3 seconds audio session whereby the message sender is asked to speak out a keyword to a microphone so that the sender’s voice is captured and recorded into a mp3 file format. The respective recorded mp3 file in-turn is converted to base 64 format which provides the random set of UTF-16

characters. Here, even though the three second audio cost you in approximately 100 to 300 kilobytes of data storage, it is compressed and combined with the modulo integer value of system data and time to produce a 64-bit length cryptographic key.

3. RESEARCH METHOD

3.1. Architectural diagram

Figure 1 showcases the architectural diagram for the Audio Recorder Key Generator program. The program starts with the initialization of class and static variable then the audio recorder libraries are initialized. Post Audio line support verification, live audio recording is commenced and the audio packets are written and saved in wav format. After the recording is concluded, then the wav file is read into a byte array and subsequently encoded in Base 64 format which is stored in a text file locally. Finally, the characters in the Base 64 text file are extracted and stored in an integer array which is mixed with the modulo integer value of the current system date and time to further randomize the output key. The final output is converted to a Binary String array of 64-bit length and returned to be printed on the console window. Since the 64-bit key is stored in a variable, it can be used as a function/method return value to be utilized as an input key value in any cryptographic symmetric key algorithm such as AES-256, 3DES, OTP, etc. Audio Recordings in general are capable enough to produce truly random numbers. However, we have tried to consider a special use case where a user ends up with a live recording without actually speaking (may occur if the user uses a headset with a faulty mic), which would result in a static audio recording. To accommodate this scenario, the modulo integer value of the current system date and time is utilized to further randomize the program output.

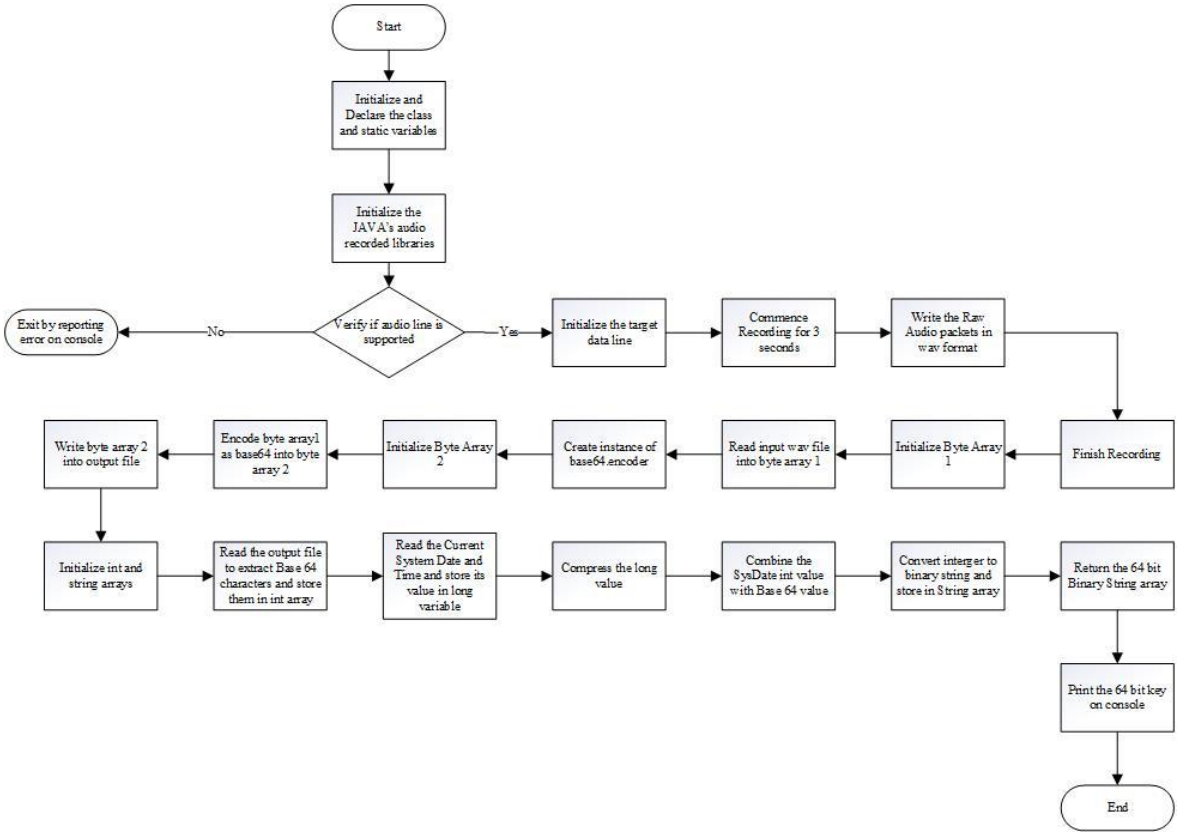

Figure 1. Architectural flow diagram

3.1. Implementation

This section provides details about the implementation of the Audio Recorder Key Generator using Java 13.0.2 version. The output of the program generates a 64-bit binary key that is used for the management and transportation of key in symmetric key algorithms. Since the 64-bit key is created using the current system date and time, the recorded audio of the user and the captured environmental sounds, the output provides a

64-bit True Random Number as an encryption key. This key can be used in any symmetric key encryption algorithm such as AES, DES, RC4, Blowfish, etc. Furthermore, to fortify the randomness of the key, the current system date and time is included in the key.

3.2. Algorithm

The following algorithmic flow would provide a walkthrough of the Secured Audio Based Key Generator.

Algorithm: Secured Audio Based Key Generator

1. Start
2. Initialize and declare: class, static variables
3. Initialize: Audio recorder libraries, target data
4. Rec → Aud (3 sec);
5. Cap_Aud = Aud (.wav format);
6. Stop Rec
7. Initialize: Byt_Ary (Byte Arrays)
8. Read Cap_Aud → Byt_Ary;
9. Encode Byt_Ary → Base64;
10. Txt = Base64;
11. Initialize: String, Int_Ary (int array)
12. Read Txt;
13. Txt_Cnt = Txt (type casted in to Int_Ary);
14. Read CSD=ComIntVal_Crnt_SysDate;
15. Read CST=CompIntVal_Crnt_SysTime;
16. Merge Mer=CSD, CST, Txt_Cnt;
17. Compress Mer → 64-bit binary string array;
18. Return → 64-bit key;
19. Print → Console window;
20. Stop

The overall program is divided into the following functions, exhibited in Figure 2

- StartRecording – Responsible for capturing the user's audio packet for 3 seconds.
- FinishRecording – Concludes the recording session after 3 seconds.
- WavToBase64 – Responsible for converting wav file to Base 64 text file.
- KeyGen – Responsible for generating the 64-bit key from Base 64 text file and the current system date and time.
- Modulo255 – Responsible for returning the modulus 255 integer value which is also less than 255.

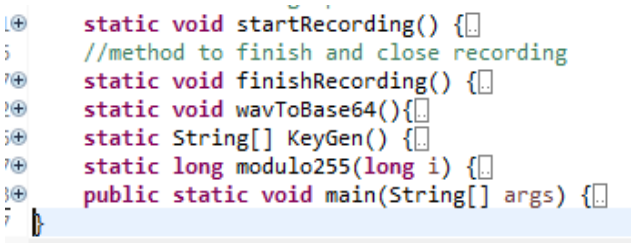

Figure 2. Class method snippet of audio recorder key generator program

4. RESULTS AND DISCUSSION

Our lightweight program is capable of using the current laptops or PCs hardware to generate a True Random Number (TRNG) key using live audio recordings which is further randomized using system date and time. These TRNs can be used to replace the deterministic pseudo random number cryptographic keys that are presently used by industries for symmetric key encryption algorithms which devolves the algorithm to being conditionally secured (cipher text can be decrypted to plain text without key knowledge over a long duration of time, that can be more than the message's time to live). Using the audio based TRNG key would render the

same encryption algorithm as unconditionally secured (cipher text doesn't contain sufficient data to uniquely determine the plain text without key knowledge). Audio Recordings in general are capable enough to produce truly random numbers. However, we have tried to consider a special use case where a user ends up with a live recording without actually speaking (may occur if the user uses a headset with a faulty mic), which would result in a static audio recording. To accommodate this scenario, the modulo integer value of the current system date and time is utilized to further randomize the program output. Upon execution, the lightweight Audio based TRN Key Generator program (proof of concept was created using JAVA 13) produces a randomized 64-bit binary key which can be directly used as a key input in any cryptographic symmetric key algorithms. The respective program was executed for 20 run cycles to showcase the randomness of the 64-bit key output. The main highlight of the implemented code is the incorporation of Audio Recording and the Current System Date that is responsible for generating a truly random number of bits. Table 1 displays the list of twenty 64-bit keys generated utilizing the developed code which throws some insight on the TRNG capabilities of the formulated Secured Audio Recorder Key Generator.

Table 1. Test run cycles of secured audio recorder key generator

Run Cycle	Key Generated
1	11000010 11000010 11000010 11001001 11000001 11000001 11000011 11000001
2	10110001 10110001 10110001 10110001 10101100 10101111 10101111 10110101
3	11001001 11001001 11001001 11001001 11001000 11001000 11001000 11001001
4	11011100 11011101 11011100 11011101 11011110 11011101 11011110 11011101
5	11101101 11101101 11101101 11101100 11101101 11101100 11101100 11101100
6	00000101 00000101 00000101 00000101 00000100 00000100 00000100 00000110
7	00010111 00010110 00010110 00010110 00010111 00010110 00010111 00010111
8	00101000 00101000 00101000 00100110 00100111 00101001 00100111 00101000
9	00111100 00111100 00111011 00111100 00111101 00111011 00111100 00111100
10	01001111 01001111 01001111 01001111 01001111 01001111 01001111 01010001
11	01011111 01011111 01011111 01011111 01011111 01011110 01011110 01011110
12	01110000 01110000 01110001 01101111 01110000 01110000 01110000 01101111
13	01111110 01111110 01111110 01111110 01111110 01111111 01111110 01111110
14	10010001 10010001 10010001 10010001 10010001 10010000 10010001 10010000
15	10011110 10011110 10011110 10011110 10011110 10011110 10011110 10011110
16	10110100 10110100 10110100 10110100 10110100 10110011 10110011 10110100
17	11000001 11000001 11000001 11000001 11000001 11000001 11000001 11000001
18	11001111 11001111 11001111 11001110 11001111 11001111 11001111 11001111
19	11011100 11011100 11011100 11011100 11011100 11011011 11011100 11011100
20	11101001 11101001 11101001 11101000 11101000 11101000 11101001 11101001

As showcased in Table 1, the program was executed 20 times to generate 20 different 64-bit keys as the program output. Due to the combination of the audio voice captured during runtime along with the current system date and time, the keys generated have an average difference of 30 plus bits giving a greater degree of random bits generated. For each test run cycle, the program captures the live audio of the user which is encoded in wav format and is stored locally. After the recording is concluded, the wav file is read into a byte array and subsequently encoded in Base 64 format which is stored in a text file that is later on extracted and stored in an integer array which is mixed with the modulo integer value of the current system date and time to further randomize the output key. The final output is converted to a Binary String array of 64-bit length that is stored in a variable and returned to be printed on the console window. Figure 3 highlights the evidence of few keys generated during test cycle 16-20. As displayed, the program is individually executed each time to produce a 64-bit key output. This results in the reuse of the wav file and base 64 text file by overwriting over it. Hence no overhead is involved with the space occupied by wav file and base 64 text file in space constrained devices during the program usage.

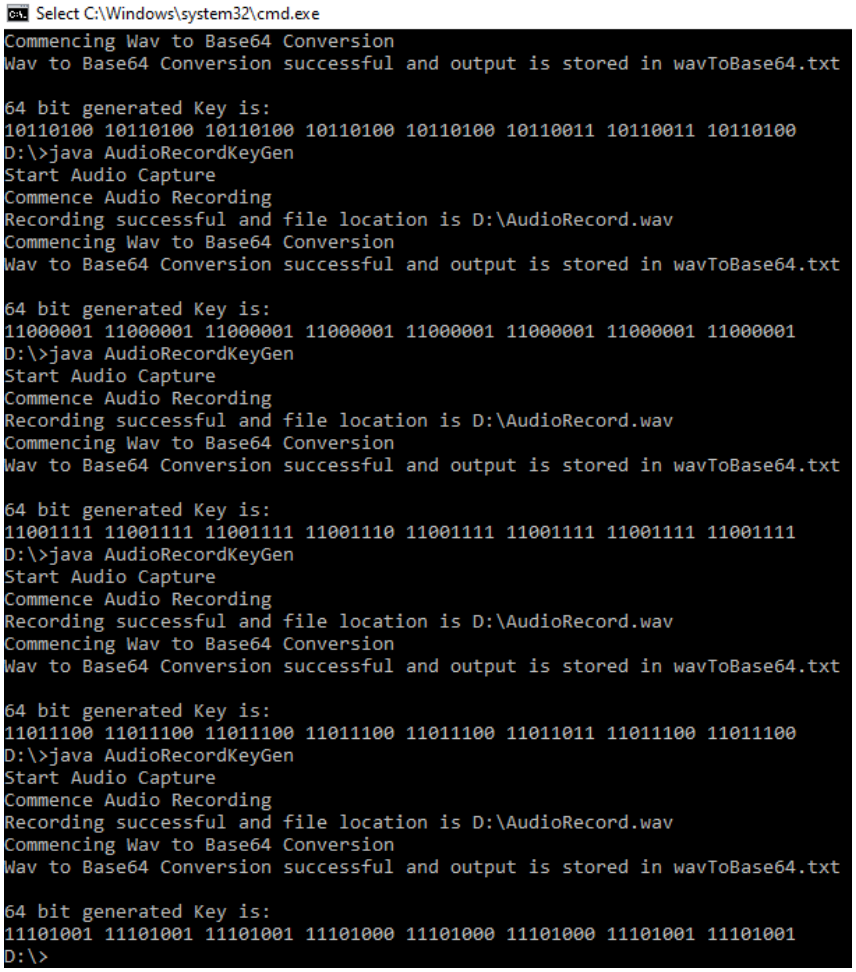

Figure 3. Multiple execution of audio recorder key generator

5. CONCLUSION

TRNG as the term suggests is a true random number generator that utilizes the entropy of physical entities for its randomness. Based on the same principle, the research work showcases that two audio records, recorded by the same person in the same physical posture never have a 100% match. In such scenarios, audio recording over a layered approach with current system date and time are perfect candidates for generating random numbers which in turn can be used as symmetric keys in the encryption algorithm. Due to the combination of audio record with system date and time, the degree of randomness of the key generated in each execution run is more than 30 bits at different bit positions for each key generated. Furthermore, the program can be executed in a resource constraint device with ease since the wav file (approx. 120 KB file size) and the base 64 text file are reused each time the program is executed to generate the 64-bit key. Henceforth, this research work aims at proving this hypothesis by real time implementation of this scenario.

ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisors Dr. Aju D, Dr. Saravanan R, and Monalisa Kushwaha, you all have been a tremendous inspiration for me.

REFERENCES

[1] G. Ramesh, U. R and T. E, "A Survey on Various Most Common Encryption Techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 7, pp. 226-233, 2012.

[2] M. Kaur and S. Kaur, "Survey of Various Encryption Techniques for Audio Data," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 5, pp. 1314-1317, 2014.

[3] G. Gupta and R. Chawla, "Review on Encryption Ciphers of Cryptography in Network Security," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 7, 2012.
[4] S. Indarjani, G. Supriyatno, A. Nugraha and I. M. M. Astawa, "Insertion attack effects on some PRNGs based on NIST randomness tests tool: Case study on ANSI-X9.17, ANSIX9.31, Dragon and Rabbit algorithms," *2014 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, Bandung, 2014, pp. 181-186, doi: 10.1109/IC3INA.2014.7042624.
[5] M. A. Kumar and D. K. S, "Investigating the Efficiency of Blowfish and Rejindael (AES) Algorithms," *I. J. Computer Network and Information Security*, vol. 2, no. 22, pp. 22-28, 2012.
[6] R. S and P. A, "Recent Developments in Signal Encryption–A Critical Survey," *International Journal of Scientific and Research Publications*, vol. 2, no. 6, pp. 1-7, 2012.
[7] A. K. *Mandal*, C. Parakash and A. Tiwari, "Performance evaluation of cryptographic algorithms: DES and AES," *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*, Bhopal, 2012, pp. 1-5, doi: 10.1109/SCEECS.2012.6184991.
[8] A. Amro and E. M. El-Alfy, "Known-plaintext attack and improvement of PRNG-based text encryption," 2016 7th *International Conference on Information and Communication Systems (ICICS)*, Irbid, 2016, pp. 233-238, doi: 10.1109/IACS.2016.7476117.
[9] J. P and R. H. S. A. H. R., "Information Hiding Using Audio Steganography–A Survey," *The International Journal of Multimedia & Its Applications (IJMA)*, vol. 3, no. 3, pp. 86-96, 2011.
[10] E. M. M. Manucom, B. D. Gerardo and R. P. Medina, "Analysis of Key Randomness in Improved One-Time Pad Cryptography," 2019 *IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, Xiamen, China, 2019, pp. 11-16, doi: 10.1109/ICASID.2019.8925173.
[11] D. Sharma, "Five Level Cryptography in Speech Processing using Multi Hash and Repositioning of Speech Elements," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 653-658, 2012.
[12] S. Sharma and P. K. Pateriya, "A Study on Different Approaches of Selective Encryption Technique," *International Journal of Computer Science and Communication Networks*, vol. 2, no. 6, pp. 658-662, 2012.
[13] D. Omerasevic, N. Behlilovic, S. Mrdovic and A. Sarajlic, "Comparing randomness on various video and Audio Media File types," *2013 21st Telecommunications Forum Telfor (TELFOR)*, Belgrade, 2013, pp. 381-384, doi: 10.1109/TELFOR.2013.6716249.
[14] D. Omerasevic, N. Behlilovic and S. Mrdovic, "CryptoStego-a novel approach for creating cryptographic keys and messages," *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, Bucharest, 2013, pp. 83-86, doi: 10.1109/IWSSIP.2013.6623455.
[15] L. Ballard, S. Kamara and M. K. Reiter, "The practical subtleties of biometric key generation," 17th *USENIX Security Symposium*, 2008, pp. 61-74.
[16] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," in *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337-343, May 1977, doi: 10.1109/TIT.1977.1055714.
[17] G. A. Spanos and T. B. Maples, "Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video," *Proceedings of Fourth International Conference on Computer Communications and Networks* - IC3N'95, Las Vegas, NV, USA, 1995, pp. 2-10, doi: 10.1109/ICCCN.1995.540095.
[18] R. A. Gandhi and A. M. Gosai, "A Study on Current Scenario of Audio Encryption," *International Journal of Computer Applications*, vol. 116, no. 7, pp. 13-17, 2015.
[19] V. Makwana and N. Parmar, "Encrypt an Audio file using Combine Approach of Transformation and Cryptography," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4473-4476, 2014.
[20] A.-q. Majdi and Y. H. Lin, "Simple Encryption/Decryption Application," *International Journal of Computer Science and Security*, vol. 1, no. 1, pp. 33-40, 2013.
[21] T. Addabbo, A. Fort, S. Rocchi and V. Vignoli, "Chaos based generation of true random bits," in *Intelligent Computing Based on Chaos*, Kocarev, L. and Galias, Z. and Lian, S, Berlin, Heidelberg, Germany: Springer, 2009, pp. 355-377.
[22] P. S and R. E, "Throughput Analysis of Symmetric Algorithms," *International Journal of Advanced Networking and Applications*, vol. 4, no. 2, pp. 1574-1577, 2012.
[23] Wenjun Zeng and S. Lei, "Efficient frequency domain selective scrambling of digital video," in *IEEE Transactions on Multimedia*, vol. 5, no. 1, pp. 118-129, Mar 2003, doi: 10.1109/TMM.2003.808817.
[24] S. Sharma, H. Sharma and L. Kumar, "Power Spectrum Encryption and Decryption of an Audio File," *International Journal of Research in Computer Science*, vol. 1, no. 1, pp. 1-4, 2013.
[25] S. Sharma, L. Kumar and H. Sharma, "Encryption of an Audio File on Lower Frequency Band for Secure Communication," *International Journal of Computer Science and Software engineering*, vol. 3, no. 7, 2013.