International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-4, February 2020

Using Smartphone Sensors to Generate Cryptographic Keys

Ruhillo Alaev

***Abstract*: The paper proposes an algorithm for generating random numbers based on sensors of a gyroscope, magnetometer, accelerometer on mobile devices, and discusses the sensors of a gyroscope, magnetometer and accelerometer.**

The random bit generator uses smartphone sensors as entropy source. It collects raw data from smartphone sensors and processes they by given algorithm.

The degree of randomness of random bits generated using the proposed algorithm was tested using statistical tests NIST SP 800-22, and a test result was given.

It is important to ensure the safe storage, transmission and use of data in these information systems. To solve these problems, cryptographic methods are used more often than other methods of data protection. Cryptographic algorithms are the basis of cryptographic data protection methods. Most cryptographic algorithms are publicly available and the key plays the role of a secret value when applied. The generations of these secret keys and sometime private keys are based on random number generations.

***Keywords*: mobile device, sensor, random number generation, algorithm, cryptographic key, gyroscope, magnetometer, accelerometer.**

I. INTRODUCTION

Currently, the use of electronic public services and other organizations in the Republic of Uzbekistan is becoming increasingly popular. Some of these services require an electronic digital signature. By increasing the number of smartphone users, smartphones are becoming increasingly popular in using these services. To use the services on mobile devices that require digital signatures, electronic digital signature software must be installed on these devices. Besides, these software programs must perform the generation, export, import, destruction, creation and verification of cryptographic keys, as well as support the O'zDSt 1105:2009 data encryption algorithm, the O'zDSt 1106:2009 hash function, the creation and verification algorithm digital signature O'zDSt 1092:2009. The generation of the symmetric key of the O‘zDSt 1105:2009 algorithm and the private key of the O‘zDSt 1092:2009 algorithm is solved by generating random numbers.

Revised Manuscript Received on February 28, 2020.
***** Correspondence Author
Alaev Ruhillo*, Department of Information Security, National university of Uzbekistan Named After Mirzo Ulugbek, Tashkent, Uzbekistan. Email: mr.ruhillo@gmail.com

A complete set of operations necessary to create, maintain, protect, and control the use of cryptographic keys are called key management. Key management is carried out by the public key infrastructure [13].

A number of scientists around the world have conducted random number generation research. In particular, the National Institute of Standards and Technology (NIST) has developed a set of statistical tests for random and pseudo random number generators for cryptographic applications, which provide an opportunity to evaluate the sequence of random bits. It consists of 15 statistical tests, the purpose of which is testing random and pseudo random number generators and determining whether or not a generator is suitable for a particular cryptographic application: frequency (monobit) test, frequency test within a block, runs test, est for the longest run of ones in a block, binary matrix rank test, discrete fourier transform (spectral) test, non-overlapping template matching test, overlapping template matching test, maurer's "universal statistical" test, linear complexity test, serial test, approximate entropy test, cumulative sums (cusum) test, cumulative sums (cusum) test, random excursions test, random excursions variant test. Random bit generators receive initial data from various sources of entropy. In [2], the design principles and requirements for the entropy sources used by Random Bit Generators, and the tests for the validation of entropy sources were specified. Entropy source data is processed by deterministic random number generators [3]. The design and architecture of random bit generators are described in [4].

An assessment of the entropy of thermometers, magnetometers, accelerometers, and other sensors of a smartphone, as well as the calculation of the minimum entropy, was presented in [5]. In this work, it is indicated that magnetometers, accelerometers, and vibration sensors showed good results when used as a source of entropy in a private experiment.

The work [6] shows the generation of random bits based on smartphone sensors. In this work, 32-bit values were collected from motion sensors and then hashed using the SHA1, SHA256, SH384, SHA512 algorithm. The hash value is the output of a random bit generator.

One of the basic requirements for random bit generators is the inability to predict the sequence of bits generated by this generator. A study was conducted to predict the outputs of random sequence generators [7]. A fast software method for estimating a sequence of random bits was widely described in [8]. Typically, on the Android platform, OpenSSL is used to generate random numbers. The work [9] showed the weakness of the OpenSSL library in generating random numbers on the Android platform.

It should be noted that the values obtained from the sensors of the smartphone cannot be used directly in cryptography.

They should be processed based on certain algorithms, such as pseudo random number generators, hashing algorithms [12]. However, using only hashing algorithms is not efficient enough, since it is difficult to achieve high marks in static testing presented in [1]. In the aforementioned studies, an algorithm for processing values received from smartphone sensors and generating an effective sequence of random bits is not described in detail.

This work shows the processing of values obtained from a gyroscope, magnetometer, accelerometer, and the generation of effective random bits.

Given that most smartphones have gyroscope, magnetometer and accelerometer sensors, these sensors are considered here as sources of entropy for generating random bits. It was discovered that when using the accelerometer to generate random bits, the entropy of a stationary accelerometer cannot be reduced in the presence of a variety of environmental variations or even under adversarial manipulations [10]. In mobile operating systems, each mobile application must have certain permissions so that it can use a specific source as a source of entropy (for example, camera [11], wireless network, and microphone) [14]. Granting multiple permissions to a mobile application only to generate random bits is an impractical way.

II. PROPOSED ALGORITHM

A three-axis accelerometer sensor returns the current acceleration along three-axis (x, y, z) in m/s^2, a three-axis gyroscope sensor returns the rate of device rotation along three-axis (x, y, z) in radian/s, and a magnetometer sensor returns the magnetic field for all three physical axes (x, y, z) in micro-Tesla (uT).When changing the values of the sensors of the gyroscope, magnetometer, accelerometer, the new values are transmitted to the random bit generator.

Definitions and Notations

aX, aY, aZ - is a set of real numbers, where the x, y, z values of the accelerometer sensor are stored, respectively.

gX, gY, gZ - is a set of real numbers, where the x, y, z values of the gyroscope sensor are stored, respectively.

mX, mY, mZ - is a set of real numbers, where the x, y, z values of the magnetometer sensor are stored, respectively.

randomBitsLen – number of random bits to be generated

TB – an array of bytes, the generated bits are divided into 8-bit sequences and stored in this array

xk – *k-bit* of *x, k* ≥ *0*

seed – used to initialize the random bit generator

flag – natural number, in the initial state *flag = 0*

& – conjunction

| – disjunction

|| – concatenation

^ – XOR operation

>> right shift operator

<< left shift operator

<<< cyclic left shift operator

Random bits generation

The random bits generation process consists of 4 stages, which were implemented as functions: *Initialization, Generation, Transmission and Completion.*

All numbers are in little-endian format.

Initialization(seed)

Step 1: initialization of arrays: *seed, aX, aY, aZ, gX, gY, gZ, mX, mY, mZ.*

Generation (randomBitsLen)

Step 1: start the generator to read data from sensors.
When the sensor values change, the new values are read from the sensors and are checked as shown in Fig. 1.

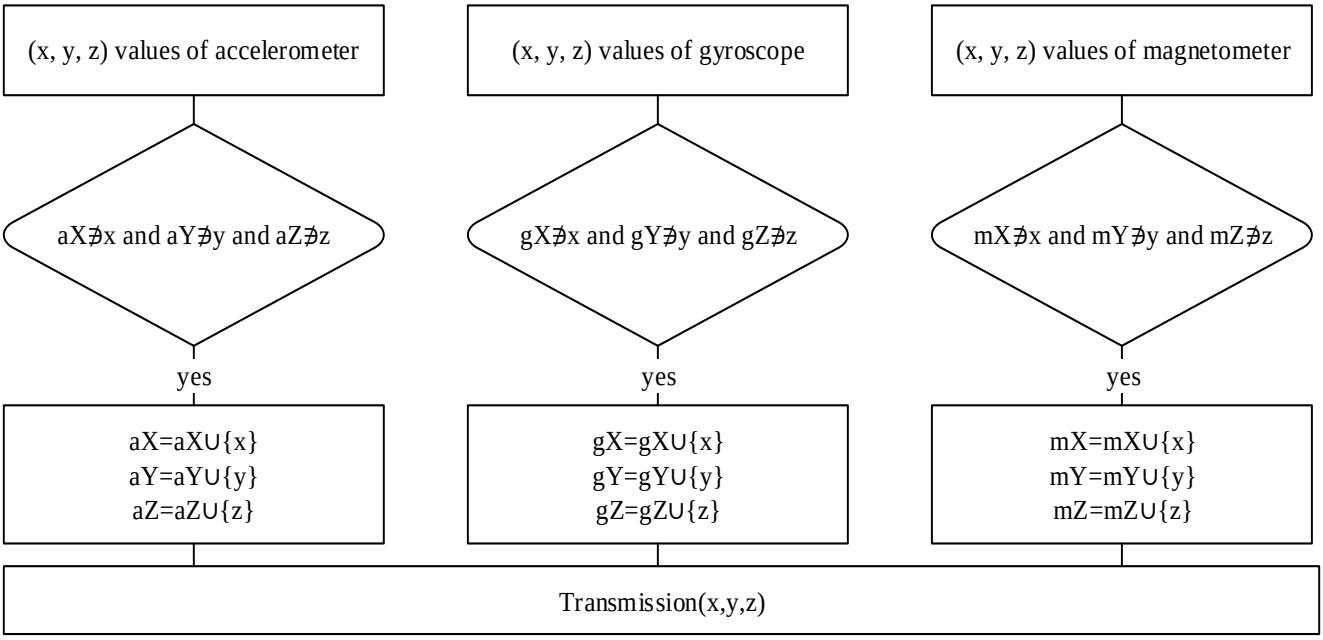

Fig. 1.Reading data from sensors of the gyroscope, magnetometer, accelerometer

Function Transmission (X, Y, Z)

Step 1: If $X \neq 0$ and $Y \neq 0$ and $Z \neq 0$, then go to step 2, else go to step 6.

Step 2: $B = Processing\ (X, Y, Z)$

Step 3: If $B = 0$, then go to step 6, else 4 bytes of B are denoted as $b_0b_1b_2b_3$.

Step 4: The values of $b_0b_1b_2b_3$ are assigned to *TB array as shown below:*

$TB[i] = b_0$, $TB[i+1] = b_1$,
$TB[i+2] = b_2$, $TB[i+3] = b_3$, $i+=4$;

Step 5: If $randomBitsLen \geq i*8$, then call *Completion* ().

Step 6: Exit

Function Processing (X, Y, Z), returns a 32-bit sequence of random bits.

Step 1: The 32-bit value of X, Y, Z is denoted as a binary sequence $x_0x_1x_3\ldots x_{31}$; $y_0y_1y_3\ldots y_{31}$; $z_0z_1z_3\ldots z_{31}$ respectively.

Step 2: 24 bits are selected from each value. Because this range of bits changes more. As a result, the length of the new binary S sequence will be 24 * 3 = 72:

$S = x_8||x_9||x_{10}||\ldots||x_{31}||y_8||y_9||y_{10}||\ldots||y_{31}||z_8||z_9||z_{10}||\ldots||z_{31}$

Step 3: $flag = flag + 1 \bmod 2$

Step 4: $offset = flag * 8$, from the binary sequence S, starting from the position of $flag * 8$, 64 bits are selected. Thus, the various parts of the binary S sequence are selected in turn.

$R = s_{offset}||s_{offset+1}||s_{offset+2}||\ldots||s_{offset+63}$

Step 5: Add *current time* to R:

$R = R +$ *(current time 64 bit)* $\bmod 2^{64}$

Step 6: Assign right 32 bits of the number R to $R1$.

$R1 = R \ \&\ 0x00000000FFFFFFFF$;

if $R1 \neq 0$, then go to step 7 else go to step 11.

Step 7: 64 bits of the S we assign to R in step 4, now the rest 8 bits of the S we assign to Qu and Qch:

$t = flag * 64 \bmod 72$,
$Qu = s_t||s_{t+1}||s_{t+2}||s_{t+3}$,
$Qch = s_{t+4}||s_{t+5}||s_{t+6}||s_{t+7}$

Step 8: The left 16 bits of the number $R1$ are assigned to Ru, the second 16 bits to Rch.

$Ru = R1 >> 16$;
$Rch = R1 \ \&\ 0x0000FFFF$;

Step 9: $R1 = ((Ru <<< Qu) << 16) \mid (Rch <<< Qch)$;

Step 10: Assign left 32 bits of the number R to $R2$.

$R2 = R >> 32$;

Step 11: $R2 = (R2 >> 16) \wedge (R2 \ \&\ 0x0000FFFF)$;

Step 12: Add right 4 bytes of number $R2$ modulo 32, and assign the result to Q:

$Q = (R2>>12)+((R2\&0x0F00)>>8)+((R2\&0x00F0)>>4)+(R2\&0x000F) \bmod 32$;

Step 13: Cyclic left shift the bits of number $R1$ by Q places:

$R1 = R1 <<< Q$

Step 14: The value of $R1$ is returned as a result.

Function Completion()

Step 1: Stopping the process of reading data from sensors.

Step 2: The *TB* array is divided into 32-byte blocks $d_1d_2d_3\ldots d_n$.

Step 3: $d_0 = seed$. Each d_j block, d_{j-1} and the block index j are concatenated using the || operator, and hashed by the *SHA256* algorithm [15] (*SHA-3* algorithm [16]).

$d_j = sha256(d_j||d_{j-1}||j),\ 1 \leq j \leq n$

Step 4: All d_j blocks are concatenated and sequentially written to the *TB* array. The values of 32 elements of each d_j array are written to the *TB* array sequentially. If the entire value of the last d_n block does not fit into the *TB* array, then the first part of the d_n block is written to the *TB* array, the rest part is ignored.

Step 5: The *TB* array is returned as a result.

III. RESULTS AND DISCUSSIONS

Based on this algorithm, a mobile application was created and a series of tests were carried out on *Samsung Galaxy S3* and *Samsung Galaxy A6* smartphones. In the mobile application, one or several sources of entropy (gyroscope, magnetometer and accelerometer) are selected and the sequence length of the generated random bits is set (see Fig. 2). After clicking on the "*Generate*" button, the application starts reading data from the sources of entropy. This process continues until generating a sufficient number of random bits.

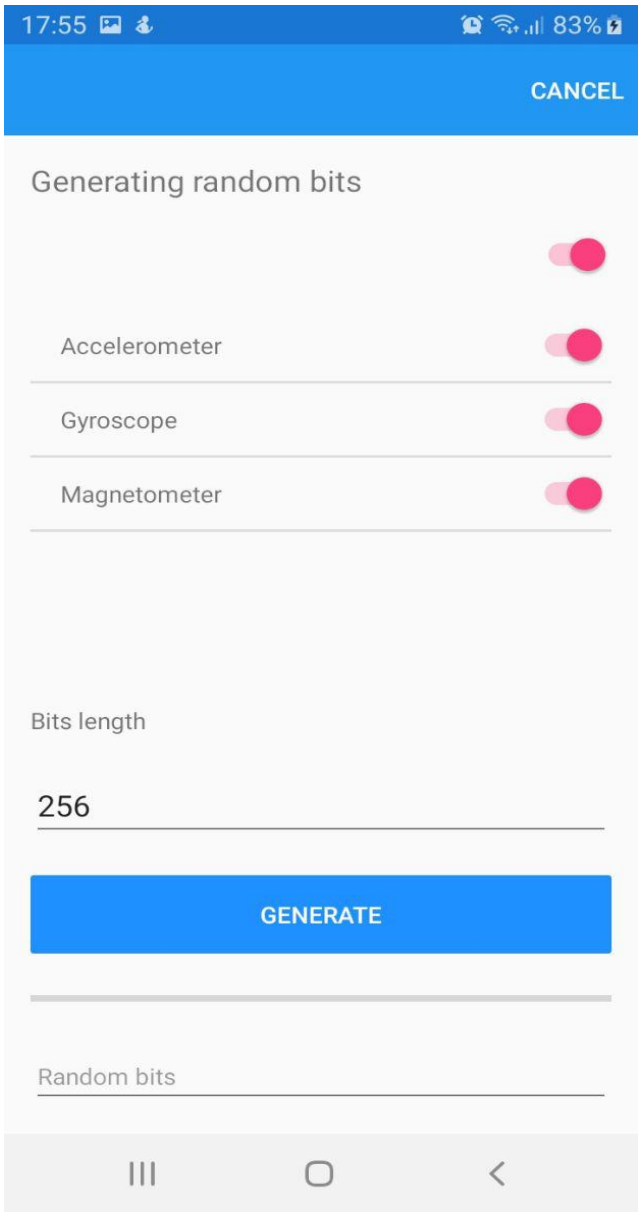

Fig. 2. Mobile application interface

After completing the generation of random bits, the application displays the result as a binary string and as a graphic (see Fig. 3).

All 32 bits of the *X*, *Y*, *Z* value received from the sensors of the smartphone are not used. Only 24 bits are used, which changes more.

In addition, in the *Processing* function in step 5, another source of entropy (the current time) is used. This increases the degree of randomness of the generated random bits, and also complicates the prediction of the generator.

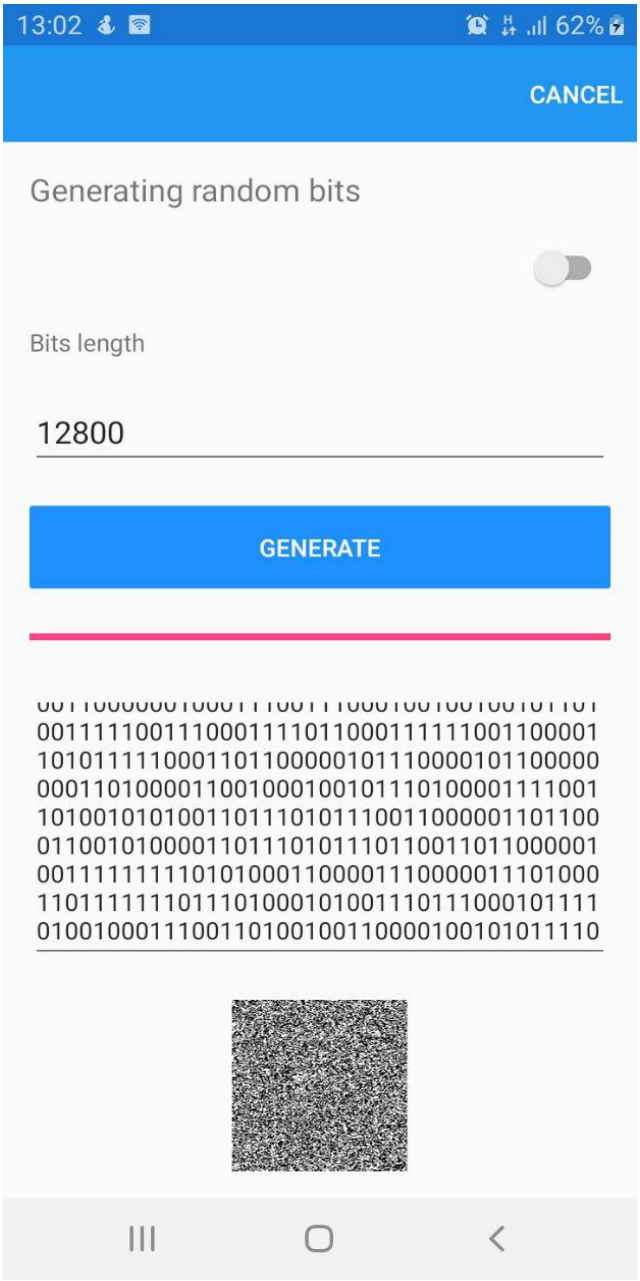

Fig. 3.Generated Random Bits

During the experiments, 50 sequences of random bits with a length of 12800 bits were generated. The generated random bits were tested using the statistical tests given in [1]. The *P-value* of all generated random bits is bigger than 0,01 (see Table I).

Table- I: The test result of random bits

P-VALUE	PROPORTION	STATISTICAL TEST
0.236810	49/50	Frequency
0.816537	49/50	BlockFrequency
0.191687	49/50	CumulativeSums
0.289667	50/50	CumulativeSums
0.883171	50/50	Runs
0.383827	50/50	LongestRuns

In the experiment, the received raw data from the sensors of the smartphone were processed in 3 ways. In the first method, the hash algorithm was not used, that is, in the Completion function, step 3 did not execute. The second method used the *SHA256* hash algorithm, that is, the *SHA256* hash algorithm was used in the Completion function in step 3. The third method used the *SHA-3* hashing algorithm, that is, in the *Completion* function in step 3, the *SHA-3* hashing algorithm was used. In all three cases, the $P\text{-}value \geq 0.01$. The degree of randomness of the generated bits using hash algorithms was high.

IV. CONCLUSION

The random bits generation algorithm is proposed in this study which uses smartphone sensors (gyroscope, magnetometer, and accelerometer) as entropy sources. In the experiments, results were obtained, indicating that the generated random bits using the proposed algorithm can be used in cryptography for generating strong keys.

The generated random bits were tested using the statistical tests given in [1]. The test result showed that the generated random bit sequences received the $P\text{-}value \geq 0.01$.

Also, the proposed algorithm can be used to develop custom random number generators in mobile applications when full control over generation is required.

This work serves to create integrated public key infrastructure software that supports the state standard algorithms “O'zDSt 1106:2009 - The hash function” and “O'zDSt 1092:2009 - The processes of generating and verifying electronic digital signatures”, as well as allowing the use of these algorithms in computers and mobile devices.

REFERENCES

1. Rukhin, Andrew & Soto, Juan & Nechvatal, James & Smid, Miles & Barker, Elaine & Leigh, Stefan & Levenson, Mark & Vangel, Mark & Banks, David & Heckert, Alan & Dray, James & Vo, San & Bassham, Lawrence. (2010). NIST Special Publication 800-22: A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. NIST Special Publication 800-22.
2. Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A. McKay, Mary L. Baish, Mike Boyle. (2018). NIST Special Publication 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation. https://doi.org/10.6028/NIST.SP.800-90B.
3. Barker, E., Kelsey, J.: NIST SP 800-90A Rev. 1 Recommendation for random number generation using deterministic random bit generators (2015).
4. Barker, E., Kelsey, J.: NIST Special Publication 800-90C. Recommendation for Random Bit Generator (RBG) Constructions (2012).

5. Hennebert, C., Hossayni, H., & Lauradoux, C. (2013). Entropy harvesting from physical sensors. In WiSec 2013 - Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks (pp. 149–154). https://doi.org/10.1145/2462096.2462122.
6. Loutfi, J., Chehab, A., Elhajj, I. H., & Kayssi, A. (2014). Smartphone sensors as random bit generators. In Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA (Vol. 2014, pp. 773–780). IEEE Computer Society. https://doi.org/10.1109/AICCSA.2014.7073279.
7. Г. П. Акимова, Е. В. Пашкина, А. В. Соловьев. Методологический подход к оценке качества случайных чисели последовательностей. Труды ИСА РАН 2008. Т. 38.
8. Р.Р. Вильданов, Р.В. Мещеряков, С.С. Бондарчук. Тесты псевдослучайных последовательностей и реализующее их программное средство. Доклады ТУСУРа, No 1 (25), часть 2, июнь 2012.
9. Kim, S. H., Han, D., & Lee, D. H. (2013). Predictability of Android OpenSSL's pseudo random number generator. In Proceedings of the ACM Conference on Computer and Communications Security (pp. 659–668). https://doi.org/10.1145/2508859.2516706
10. Voris, J., Saxena, N., & Halevi, T. (2011). Accelerometers and randomness: Perfect together. In WiSec'11 - Proceedings of the 4th ACM Conference on Wireless Network Security (pp. 115–126). https://doi.org/10.1145/1998412.1998433
11. Zhang, X., Qi, L., Tang, Z., & Zhang, Y. (2014). Portable true random number generator for personal encryption application based on smartphone camera. Electronics Letters, 50(24), 1841–1843. https://doi.org/10.1049/el.2014.2870
12. Viega, J. (2003). Practical random number generation in software. In Proceedings - Annual Computer Security Applications Conference, ACSAC (Vol. 2003-January, pp. 129–140). IEEE Computer Society. https://doi.org/10.1109/CSAC.2003.1254318
13. Buchmann, J. A., Karatsiolis, E., & Wiesmaier, A. (2013). Introduction to public key infrastructures. Introduction to Public Key Infrastructures (pp. 1–187). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-40657-7
14. Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011). Android permissions demystified. In Proceedings of the ACM Conference on Computer and Communications Security (pp. 627–636). https://doi.org/10.1145/2046707.2046779
15. Function, N. H., Bertoni, G., Daemen, J., Assche, G. V., Nakamoto, S., Paper, W., … Tuấn, Đ. M. (2016). Description of SHA-256, SHA-384 AND SHA-512. ACM Transactions on Programming Languages and Systems, 9(July), 9. https://doi.org/10.1007/s10838-008-9062-0
16. National Institute of Standards and Technology (NIST). (2014). SHA-3 Standard : Permutation-Based Hash and Extendable-Output Functions. Standard.

AUTHORS PROFILE

Alaev Ruhillo, Research Scholar in department of Information security of National University of Uzbekistan named after Mirzo Ulugbek. Research Field: Software security engineering, Applied Cryptography, Network security, Web security