ITC 4/50
Information Technology and Control
Vol. 50 / No. 4 / 2021
pp. 786-807
DOI 10.5755/j01.itc.50.4.29464

Analysis of Cryptography Algorithms Implemented in Android Mobile Application

Received 2021/04/27 | Accepted after revision 2021/11/24

http://dx.doi.org/10.5755/j01.itc.50.4.29464

HOW TO CITE: Salkanovic, A., Ljubic, S., Stankovic, L., Lerga, J. (2021). Analysis of Cryptography Algorithms Implemented in Android Mobile Application. *Information Technology and Control*, 50(4), 786-807. https://doi.org/10.5755/j01.itc.50.4.29464

Analysis of Cryptography Algorithms Implemented in Android Mobile Application

Alen Salkanovic, Sandi Ljubic

University of Rijeka, Faculty of Engineering; Vukovarska 58, 51000 Rijeka, Croatia;
e-mails: alen.salkanovic@riteh.hr, sandi.ljubic@riteh.hr
University of Rijeka, Center for Artificial Intelligence and Cybersecurity; Radmile Matejcic 2, 51000 Rijeka, Croatia

Ljubisa Stankovic

Faculty of Electrical Engineering; University of Montenegro; Džordža Vašingtona bb, 81000 Podgorica, Montenegro; e-mail: ljubisa@ucg.ac.me

Jonatan Lerga

University of Rijeka; Faculty of Engineering; Vukovarska 58, 51000 Rijeka, Croatia; e-mail: jonatan.lerga@riteh.hr
University of Rijeka, Center for Artificial Intelligence and Cybersecurity, Radmile Matejcic 2, 51000 Rijeka, Croatia

Corresponding authors: jonatan.lerga@riteh.hr

This paper evaluates the performances of numerous encryption algorithms on mobile devices running the Android operating system. The primary objective of our research was to measure and compare the relative performances of tested algorithm implementations (Data Encryption Standard (DES), 3DES, Advanced Encryption Standard (AES), ChaCha20, Blowfish, and Rivest Cipher 4 (RC4)) on the Android platform. The algorithms were compared in terms of CPU utilization by measuring the time required to encrypt and decrypt variable size text files. Besides evaluating the six common symmetric encryption ciphers, a comparison has been conducted for several Password-Based Encryption (PBE) algorithms. Diverse cipher transformations were evaluated for each algorithm by utilizing various feedback modes and padding schemes. Two smartphone devices were used for testing, with different versions of the Android operating system and hardware specifications. The summarized performance outcomes for various cipher transformations are presented to demonstrate the effectiveness of each algorithm.

KEYWORDS: encryption algorithms, software security, performance analysis, mobile devices, Android platform.

1. Introduction

Cryptography (encryption and decryption) includes procedures employed for keeping private information safe while storing and transmitting data. Its primary purposes are to protect sensitive data from unauthorized persons and exchange information over a public channel. In addition, cryptography is used for user authentication. Encryption ciphers can be divided into two categories: symmetric key and asymmetric key encryption.

In symmetric-key ciphers, both sender and receiver use the same private key (used for encryption and decryption). This means that the key must be exchanged between them using some trustworthy procedure. Also, this approach requires a new key for each pair of persons who communicate over a public channel.

On the other hand, asymmetric key ciphers use two related keys (public and private keys). The public key is open and freely distributed, while its paired private key is kept as a secret. The public key is used by the sender for encryption of data, and the private key is used by the receiver to decrypt received encrypted data. Hence, there is no need for a new pair of keys for each pair of persons who communicate over a public channel.

In this article, we focus on common encryption methods used on contemporary smartphones. The performance of several algorithms was assessed using two mobile devices with varied hardware specifications and different versions of the Android operating system. Our research concentrates on six relevant symmetric encryption ciphers provided by Android platform: AES, ChaCha20, RC4, DES, TripleDES, and Blowfish. Additionally, performances of various Password-Based Encryption (PBE) algorithms were evaluated.

The reason for employing these algorithms is the usage of the *Cipher* class, which implements the encryption and decryption capabilities of a cryptographic cipher on Android platform, additionally providing various cipher transformations [4]. That is why we selected cipher transformations which the Android operating system supports. The *Cipher* class the fundamental component of the Java Cryptographic Extensions (JCE) framework. The supported cipher transformations we have tested in this research are described in the Cipher section of the Java Cryptography Architecture Standard Algorithm Name Documentation [5].

The mobile application was developed to assess the cryptographic algorithms by considering CPU utilization during the encryption-decryption process. The application allows users to encrypt and decrypt files of various types and sizes and to adjust algorithm parameters, such as key length, mode of operation (feedback mode), and padding scheme. In addition, several encryption-decryption cycles and the delay between the cycles can be specified. Two Samsung smartphones at our disposal were evaluated: Samsung Galaxy S9 Plus, which has enhanced hardware, and Samsung Galaxy A20e, with somewhat poorer performances. We consider the following contribution as compared to other relevant works:

- extensive performance evaluation of several different cipher modes of operation (feedback modes), as well as supported padding schemes and key lengths
- assessing the encryption algorithms utilizing smartphone devices with varying hardware capabilities (devices with different computational performance levels)
- analyzing encryption algorithms on earlier as well as newer versions of the Android operating system (including the Chacha20 cipher, which is accessible only on recent Android version 9 and above, i.e., starting from API level 28+)
- comparison of several Password-Based Encryption (PBE) algorithms with different parameters provided
- implementation of an application for the Android operating system that provides encryption and decryption capabilities for a variety of file types using configurable cipher parameters

The rest of the paper is structured as follows. A brief overview of encryption algorithms and implementation details are described in Sections 3 and 4. Simulation results and discussion on ciphers' performances are given in Section 5. Conclusion is found in Section 6.

2. Related Work

Several research papers have already addressed the issue of performance analysis of encryption algorithms on different platforms. Namely, in [21], Nadeem and Javed implemented and compared the performance of

four popular encryption algorithms: Data Encryption Standard (DES), TripleDES (3DES), Advanced Encryption Standard (AES) and Blowfish. Two different machines were used, based on Pentium processors, and running the Windows operating system.

Salama et al. [32] performed a comparison based on CPU process time, clock cycles and battery power consumption on a laptop computer. The authors presented simulation results to demonstrate the effectiveness of each encryption algorithm. Their findings demonstrate that the Blowfish algorithm outperforms all other algorithms in terms of processing time, while the 3DES takes more time than the DES cipher.

Sahu and Kushwaha [31] and Panda [25] compared the performance of three symmetric cryptographic algorithms (AES, DES and Blowfish) with different data types. Again, the outcomes of this research corroborated our own. Namely, when comparing encryption and decryption speed, i.e., throughput, the Blowfish cipher performed better than the DES algorithm.

The efficiency of a Blowfish cipher is also demonstrated by Patil et al. [26] and Thirupalu and Reddy [38]. Among AES, DES and 3DES the findings indicate that Blowfish is the best choice as it consumes least CPU time and memory. However, the authors stated that AES is the ideal algorithm when cryptographic strength plays an important role in the application.

Nie et al. [22] studied the encryption security of the algorithms Blowfish and DES, as well as their performance and power consumption. It has been shown that the Blowfish encryption method may be more suitable for application security in wireless networks. Moreover, the results show that the Blowfish algorithm outperforms the DES algorithm in terms of speed and power consumption.

Moreover, Haque et al. [15] tested several algorithms with different parameters including data blocks, varying keys (128, 192, 256 bits) and file sizes. The basic concept behind performance testing is to identify the optimal algorithms for better utilization on resource constraint and mobile devices.

Idrus et al. [36], investigated how several encryption algorithms behave on different web browsers. The goal of the study was to find out which algorithm works best on a particular desktop web browser.

In addition, Ratnadewi et al. [28] developed an application to write and read data on a smart card using the cryptographic methods DES and 3DES on systems with NFC technology. Unsurprisingly, data writing and reading were faster using the DES cipher. The authors concluded that the difference in execution time between the two encryption methods grew as the length of the processed data increased.

Olaleye [24] compared cryptographic algorithms based on performance in mobile cloud computing. The work examined AES, DES, RSA and ECC cryptographic algorithms. The test results showed that DES encryption was faster than AES when smaller file sizes were utilized.

In addition to all mentioned, special attention should be paid to mobile devices. Indeed, today's mobile devices have become powerful computing platforms capable of running relatively complex software applications. Since communication via mobile devices is increasing, their use is also subject to communication and cyber-attacks. This, together with the rapid development of mobile platforms, contributes to the increasing demand for encryption algorithms suitable for mobile devices to protect them from increasingly serious security threats. As a result, various encryption ciphers are now available on mobile platforms to secure information in both communication transmission process and data storage.

To further compare our work with similar studies, it is feasible to refer to the work of Grgić et al. [14]. The Android platform is used to analyze the implementation of three symmetric ciphers (DES, 3DES and AES) in different operating modes. Similarly, the performance of the cryptosystems was evaluated under different conditions using a variety of variable factors, including cipher, key size, plaintext size, and thread count. However, unlike our research, the authors considered only two modes of operation: ECB and CTR. The authors confirmed when using the ECB mode, the AES cipher outperformed both the DES and 3DES cryptosystems.

With reference to smartphone devices, Tayde and Siledar [37] utilized AES cipher on the Android operating system to test its performance in encrypting and decrypting images and text files. The implemented application allows the user to encrypt the file before it is transmitted over the network. The authors provided support for different types of file encryption such as text, docx, pdf and image encryption.

Malina et al. [18], the authors analyzed lightweight and classical software-oriented block ciphers on the Android platform. They compared the performance of two forms of implementation by native JAVA cryptography APIs and by an external cryptography provider.

Gonzalez et al. [13] studied Android's cryptography framework and integrated tools. The authors evaluated the cryptographic services offered by a number of modules, including Bouncy Castle, Crypto, and HarmonyJSSE. It was found that external modules are the best approach for providing cryptographic capabilities.

Rouaf and Yousif [30] evaluated five algorithms (AES, DES, TEA, RSA and REA) in terms of execution time and battery consumption. The experiment was conducted on three Android devices with different hardware specifications than the ones used in our study. According to these results, AES proved to be the most efficient encryption method, followed by DES algorithm.

In addition, Montoya et al. [20] compared three algorithms (AES, Serpent and Twofish) in terms of computational cost on smartphone and tablet devices. In the experiment, two response variables were considered (CPU usage and memory consumption), which are closely related to battery life. The aim of the study was to determine which cipher is optimal for implementation in smartphones and tablets. The AES was confirmed as the one with the best performance on the tablet, while the Twofish performed better on the smartphone.

Mentioning device batteries, the authors evaluated different algorithms in terms of battery consumption in several other studies. When developing a mobile application, increased battery consumption could discourage programmers from using a particular encryption cipher. For this reason, power consumption should be one of the main requirements when designing these algorithms for smartphones.

Masoud et al. [19] provided a performance evaluation of four different symmetric encryption algorithms and analyzed the power consumption of smartphones when using different encryption methods. The authors found that the power consumption mainly depends on the file size, an encryption technique and the battery capacity of the smartphone. Depending on the version of AES used, this encryption resulted in the highest reduction in battery life.

Finally, let us conclude this literature review with the work done by Yao et al. [39], who addressed the energy consumption of ten security algorithms. The work investigates the trade-off between the security level of the algorithms and the power consumption on mobile devices.

In the next section, before presenting simulation results, we provide a brief description of the tested cryptography algorithms.

3. Encryption Methods

3.1. DES/3DES

Data Encryption Standard (DES) is a symmetric-key algorithm for data encryption, developed by IBM in the early 1970s and published in a slightly modified version as an official Federal Information Processing Standard for the United States in 1977. Due to its short key length of 56 bits, DES is today unsafe for most of the modern applications. The algorithm is vulnerable to various attacks, including differential and linear cryptanalysis.

Triple-DES (3DES) uses the DES algorithm three times, each time with a different 56 bits key [34]. This symmetric key-block cipher applies the DES cipher in triplicate by encrypting with the first key (key1), decrypting with the second key (key2), and encrypting with the third key (key3). A two-key variant is also available, using the identical in the first and the third step (same key1 and key3). Eventually, this variant was retired in 2015.

When implementing the 3DES algorithm and test it using an Android application, a symmetric key must be generated. Key size can be select as one of the three options: 56, 112, or 168 bits. A similar method is also utilized in the AES (using the keyGenerator class). The initialization vector must be provided when using the Cipher class to encrypt and decrypt the data (when the cipher is initialized using the init method). For this purpose, the SecureRandom class is used, as well as the nextBytes method to generate random bytes. The length of the initialization vector is equal to the algorithm's block size (8 bytes, 64 bits), not to

the length of the key. Block cipher mode of operation and padding schemes can also be selected when setting the algorithm's parameters using the Android application. It is important to note that due to its vulnerability, 3DES is likely to be disallowed after 2023.

Tables and figures should appear within the text rather than in appendices. All illustrations should be numbered in Arabic numerals. Place a caption at the top. Figures must have a caption at the top as well.

3.2. AES

The research for replacing the DES started in 1997 since it was found vulnerable and not satisfying increasing security levels due to its small key size. Advanced Encryption Standard (AES), also known as Rijndael, was established by the National Institute of Standards and Technology in 2001. The AES is a symmetric-key block cipher with a block size set to 128 bits, specified in the AES standard.

Due to the fact that symmetric encryption uses the identical key for data encryption and decryption, the implemented Android application generates the same key for both the sender and the receiver [27]. The keys were created using KeyGenerator class, which provides the functionality of a secret (symmetric) key generator [16]. Objects created this way are reusable, and the same KeyGenerator object can be used again to generate further keys. For key generation, two different options are available:

- Algorithm-Independent Initialization - uses the init method to generate the key. Two types of arguments are required: key size and a source of randomness. The randomness source is a SecureRandom implementation – a class that provides a cryptographically strong random number generator (RNG).
- Algorithm- Specific Initialization - this option is used when a set of algorithm-specific parameters already exists. Two init methods that have an argument for the transparent specification of cryptographic parameters are used.

To generate a secret key using the KeyGenerator for the AES, three different key sizes can be provided as an argument: 128-, 192- and 256-bit size. Namely, these are key sizes accepted by the AES standard. Finally, interface SecretKey groups all secret key interfaces and generates a symmetric key with the help of a KeyGenerator. The optional step includes an initialization vector, an arbitrary number used with a secret key. Initialization vector prevents repetition in data encryption and can be agreed on in advance or transmitted independently. In order to implement the encrypt function, a cipher object must be created, along with a provided transformation string that describes the cryptographic algorithm. Feedback mode and padding scheme may also be provided. The feedback mode masks the patterns which exist in encrypted data. A padding scheme is used if block cipher modes require their input to be an exact multiple of the block size, filling up the missing bits of the block. As mentioned before, the same generated secret key is used for decryption.

The main advantage of the AES encryption algorithm is fast execution (implemented in both hardware and software). Only one key is needed to both encrypt and decrypt data. However, if the key is somehow obtained by the attacker, cracking is a matter of time. Due to the key size, the time necessary to encrypt and decrypt the message hinders efficient communication. Also, each recipient must receive the key using a different channel than the message itself.

3.3. BLOWFISH

Blowfish is a symmetric-key block cipher included in many encryption products. The algorithm provides a reasonable encryption rate in hardware applications [22]. It is mostly used when the key does not often change, for example, cipher data file or communication link. Key size is variable, ranging from 32 bits up to 448 bits. Implemented Android application uses 128-, 256- and 448-bits key size and allows testing algorithm's performances. Compared to other block ciphers, Blowfish is very slow when changing keys. Pre-processing equivalent is required for each new key, equivalent to encrypting about 4 kilobytes of text. Due to the use of 64-bit block size, Blowfish is vulnerable to birthday attacks.

3.4. RC4

RC4 (Rivest Cipher 4), also called ARC4 or ARCFOUR, is a software-optimized variable-key-size stream cipher that generates a pseudorandom stream of bits (a keystream) [12]. The algorithm has two phases: key generation and encryption. The first step is the most difficult and is used to generate an encryp-

tion key. The key is then used to generate variable encryption that uses two arrays, state and keys, and the results of merging operations [35].

The symmetric key algorithm is used identically for encryption and decryption. RC4 uses a variable-length key from 1 to 256 bytes. However, RC4 can turn into very insecure cryptosystems if one keystream is used twice or when the start of the output keystream is not removed. The simplicity of RC4 makes it vulnerable to different security attacks.

3.4. CHACHA20

ChaCha20 is a symmetric encryption algorithm, a member of a family of fast symmetric stream ciphers. ChaCha variants are improved versions of the Salsa20 algorithm, using very similar stream ciphers. A new round function increases performance on some architectures [2]. It uses a pseudo-random number generator, as its main purpose is data encryption. Unlike block ciphers used in other encryption algorithms, padding is not necessary. The stream can be defined as a one-time pad, an encryption technique that requires a one-time pre-shared key. However, the same nonce must never be used with the same key twice.

It is important to notice that Android provides Cipher transformations for API levels 28+ and above (Android Pie or newer). Two modes are available (NONE and Poly1305) with no padding.

The ChaCha20 stream cipher with integrated Poly1305 authenticator requires a 256-bit key and random 96-bit nonce. The extremely high-performance cipher is useful on mobile devices and is implemented by the most modern crypto libraries. Salsa20 and its variant are not patented.

In comparison to the AES algorithm, a stream cipher is used in the ChaCha20 instead of a block cipher. The stream cipher converts plain text into cipher text by taking one byte of plain text at a time, instead of taking plain text's block. Additionally, 8 bits are used rather than 64 bits or more. In software-only implementations, when the CPU does not provide dedicated AES instructions, ChaCha20-Poly1305 is almost three times faster than the AES. Also, ChaCha20 is not vulnerable to cache-collision timing attacks, unlike the AES [3].

To implement this algorithm in the developed Android application, a special method for creating 96-bit nonce is developed. The nonce itself is not a secret, but it has to be unique for a given key. For example, a 96-bit counter can be defined. When invoked, the method increments the counter by one. The nonce is then prepended with the message cipher. A secret key for this algorithm was generated using the KeyGenerator class, similar to the AES implementation.

The algorithm is simple to implement securely in software, in contrast to AES. This is very important since most mobile platforms do not support AES acceleration [11]. For Android devices, even with no hardware acceleration, ChaCha20 is fast and secure (taking advantage of some acceleration features in the ARM chips). It is able to leverage common CPU instructions, including ARM vector instructions, on both mobile and wearable devices. Also, spending less time and computational power on decryption is especially important to save battery life. However, the weakest point for encryption is the asymmetric handshake that begins the session. If the key for symmetric encryption is lost, the rest of the session is insecure, regardless of the method of encryption. The algorithm is currently supported by mobile internet browsers, like Google Chrome.

4. Implementation Details

4.1. Application Details

An Android mobile application was developed for the purpose of testing of encoding and decoding using symmetric block and stream ciphers. While implementing the application, recent functionality changes affecting cryptography were taken into consideration. Starting with Android version 9 (Android P), the Bouncy Castle, a popular collection of APIs used in cryptography, is deprecated for implementations of many algorithms [8].

Consequently, instead of explicitly requesting the Bouncy Castle provider, the default implementation should be used instead. Furthermore, when using password-based encryption (PBE) ciphers, an explicit initialization vector must be provided. As of Android 9 (API level 28), the Crypto Java Cryptography Architecture (JCA) provider has been removed [7].

The mobile application measuring the algorithm performances have been implemented using Java pro-

gramming language. The Cipher class forms the core of the Java Cryptography Extension (JCE) framework. It provides the cryptographic cipher functionality for encryption and decryption, as well as hashing of private data. Stream and block ciphers are two major types of ciphers. These symmetric key ciphers represent the methods used for transforming plain text into ciphertext. While the stream cipher encrypts plain text one byte at a time, the block cipher splits the data into blocks of a set length. The plain text conversion is then performed by encrypting bits or bytes in each block simultaneously.

Various transformations can be provided to the Cipher object, specifying the mode of operation and padding scheme. A block cipher mode of operation describes how to repeatedly apply a cipher's single-block operation to securely encrypt or decrypt amounts of data larger than a block. A padding algorithm is used for the block modes that require the input to be split into blocks and the final block to be padded to the block size.

Depending on the API level, the Android operating system provides various cipher transformations. When creating Cipher object, a user may provide a transformation string describing the operation(s) to be performed. The name of a cryptographic algorithm is always included in a string and may be followed by a cipher mode and padding scheme. If the block mode or padding are not given, provider-specific default values for the mode and padding scheme are used [4].

4.2. Utilized Software and Hardware

To conduct the experiment, we had two mobile devices at our disposal. The first device was the Samsung Galaxy A20e, which was released in May 2019. The smartphone is equipped with an Exynos 7884 chipset (Octa-core CPU (2 x 1.6GHz & 6 x 1.35GHz)) and 3 gigabytes of random-access memory (RAM). This device is running Android Q (10), with the final API Level 29.

The second device available for testing was the Samsung Galaxy S9 Plus (a variant for European markets). This smartphone is somewhat older and has been launched in March 2018. It is powered by Exynos 9810 chipset (Octa-core CPU ((4x2.7 GHz Mongoose M3 & 4x1.8 GHz Cortex-A55))) together with 6 gigabytes of RAM. The smartphone is running Android Pie (9), with API version 28. It should be pointed out that several cipher transformations available on Android OS are supported only at API level 28 or higher.

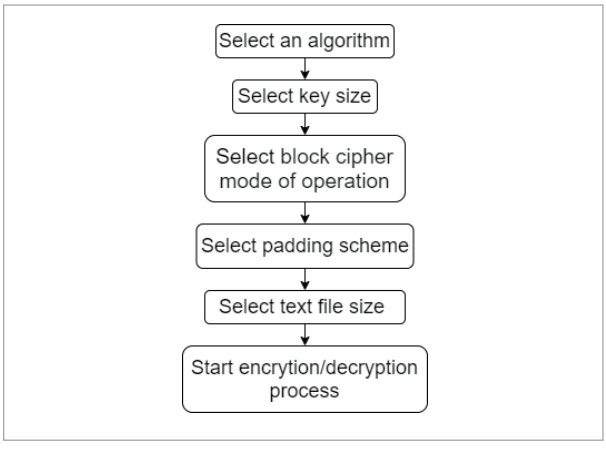
4.3. Supported Algorithms

The implemented application allows users to test the performances of various encryption algorithms. Android supports the following symmetric encryption algorithms: DES, 3DES (DESede), AES, AES_128, AES_256, BLOWFISH, RC4, and ChaCha20. As mentioned, the supported cipher transformations are described in the Cipher section of the Java Cryptography Architecture Standard Algorithm Name Documentation [5].

Furthermore, the users can specify several different cipher operating modes (feedback modes) for each algorithm, as well as the supported padding schemes. Besides, various key lengths can be selected for the encryption process. Key size selection has been programmatically restricted to ensure a particular algorithm supports the chosen key length.

The application is capable of performing encryption and decryption processes on several different file types. Supported document types include text files, PDF documents, Excel, and PowerPoint file formats. Besides, images, audio, and video files can also be encrypted and decrypted using the abovementioned algorithms. The flowchart of general encryption-decryption process using the implemented Android mobile application is illustrated in Figure 1.

Figure 1
Flowchart illustrating the general process for initiating the encryption/decryption of text files using an implemented Android mobile application



Password-Based Encryption (PBE) refers to the type of symmetric key encryption and decryption technique. The developed mobile application implements various PBE algorithms intended for performance testing. To encrypt a file using the PBE, the user selects an arbitrary password, which is used along with a generated salt (key). The chosen password is then exchanged between the parties. The decryption process again involves the same password along with the salt. Therefore, an encryption key could be generated by providing a user-specified string or even a random value. The decryption process again involves the same password along with the salt.

The test application allows users to choose between two iteration count values. The iteration count represents the number of times the password is hashed during the derivation of the symmetric key. The higher number, the more difficult it is to validate a password guess and then derive the correct key.

4.4. Measuring the Performances

Our research's primary objective was to measure and compare the relative performance of several algorithm implementations on the Android operating system. To ensure a fair comparison, we have maintained the same style consistently throughout the source code. The code is structured so that each algorithm has its own method to perform the encryption and decryption process. The whole procedure involves the following steps:

- reading the plain data from the file on the device's internal memory,
- encrypting the data using the specified cipher algorithm,
- writing the encrypted data to the newly created file on the internal memory,
- reading the encrypted data from the file,
- decrypting the data using the corresponding algorithm,
- writing the decrypted data to the newly created file on the internal memory of a device.

To measure the CPU usage of the procedure, the threadCpuTimeNanos method from Debug class has been used. The obtained value indicates the amount of time that the current thread has spent executing code or waiting for certain types of I/O. However, the nanosecond resolution does not imply nanosecond accuracy [9].

While measuring algorithms' performance, the time required to initialize the cipher and set up the key also has also been taken into consideration. The delay of several seconds has been utilized between every 100 encryption and decryption cycles. During the testing procedure, all network-based services on mobile devices were turned off.

The following section provides a discussion on the obtained simulation results for all tested symmetric ciphers for various file sizes and mobile devices.

5. Performance Results

In this section, the performance results of encryption algorithms are presented. The simulation results show differences in encryption and decryption times when using diverse cipher transformations, i.e., various block cipher modes, padding algorithms, and key sizes. As mentioned, CPU utilization is defined as the time (in milliseconds) needed to encrypt and decrypt the documents stored on the internal memory of a device. To summarize the results, we have given performances for text files. Different sizes of text files have been encrypted and decrypted using various ciphers: 0.01, 0.1, 1, and 10 megabytes. The implemented application allows users to determine the number of total encryption and decryption cycles, as well as the custom delay between different runs. In this experiment, we have used a constant number of 100 cycles and a delay of two seconds. Once all cycles had been completed successfully, the application automatically calculated the average CPU time. Subsequently, the final result was written to the log file.

5.1. Results for DES & 3DES Ciphers

DES and TripleDES (3DES, DESede, or TDES) are block cipher algorithms supported on the Android operating system starting from API level 1. The same keys are used for encryption and decryption. However, the security level of the DES is considered today relatively low due to its key size of 56 bits [21]. The initial key actually consists of 64 bits. However, 8 bits are reserved for parity checking and eventually discarded. Hence,

the effective key length is 56 bits. The drawback of this algorithm is that it is easily prone to Brute Force Attack in which the hacker attempts to break the key by applying all possible combinations [1].

3DES cipher supports double and triple length keys. For Two-Key 3DES, the user must specify a key size of 128 bits. Given that each byte's MSB is a parity bit, the strength is in fact 112 bits. Three-Key TripleDES uses a total key length of 192 bits, providing effective security of 168 bits. 3DES is vulnerable to meet-in-the-middle attack and the block collision attack.

In recent years, the cipher has been superseded by the AES algorithm. Besides being stronger than 3DES, the AES is also significantly faster.

Several feedback modes combined with two padding schemes were tested for both DES and 3DES ciphers. The particular cipher transformations correspond to those utilized with AES and Blowfish algorithms. Tables 1 and 2 provide DES cipher results obtained from both mobile devices.

Tables 3 and 4 present the results for the 3DES cipher tested on Galaxy S9 Plus and Galaxy A20e devices, respectively. This algorithm assuredly yielded the slowest encryption-decryption times among all other ciphers we have tested. Two different key lengths were used, together with six modes of operation and two padding schemes.

Table 1

Comparative execution times (in ms) of DES algorithm on Galaxy S9+ (56-bit key)

Cipher	Padding	Mode	0.01mb	0.1mb	1mb	10mb
DES	PKCS5Padding	CBC	7.106	23.424	170.703	1461.402
		CFB	9.630	31.466	199.434	1884.228
		CTR	7.417	28.444	251.707	1769.271
		CTS	7.025	22.889	152.697	1449.570
		ECB	7.118	22.250	149.386	1258.129
		OFB	7.634	34.231	186.897	1757.705
	ISO10126Padding	CBC	7.575	26.541	150.881	1408.635
		CFB	7.855	30.916	195.341	1880.133
		CTR	7.425	32.858	180.297	1594.242
		CTS	7.124	24.582	154.587	1452.622
		ECB	6.856	22.239	136.674	1226.751
		OFB	7.386	28.413	191.473	1775.506

Table 2

Comparative execution times (in ms) of DES algorithm on Galaxy A20e (56-bit key)

Cipher	Padding	Mode	0.01mb	0.1mb	1mb	10mb
DES	PKCS5Padding	CBC	17.524	63.869	622.650	5693.122
		CFB	15.860	79.268	791.506	7265.053
		CTR	17.816	70.037	692.653	6398.734
		CTS	13.936	63.165	624.407	5692.895
		ECB	12.902	54.865	534.253	4834.796
		OFB	15.235	74.309	740.798	6806.912
	ISO10126Padding	CBC	14.506	63.807	623.620	5692.004
		CFB	16.050	79.081	792.926	7269.239
		CTR	15.243	69.404	694.158	6317.854
		CTS	14.286	63.612	624.542	5705.297
		ECB	13.062	54.977	534.165	4839.931
		OFB	15.162	73.785	738.414	6741.632

The results show that different key sizes affect CPU utilization. Namely, when we utilized a device with higher hardware specifications (Galaxy S9 Plus), the encryption-decryption process was somewhat faster if we provided the 192-bit key. However, the differences in results are not so prominent, as it can be observed from Table 4.

Regarding the cipher transformations using the DES, the outcomes are comparable to the AES algorithm with a 128-bit key.

It can be observed that two different mobile devices attained similar results when the DES or AES ciphers are utilized. For both provided padding schemes (PKCS5Padding and ISO10126Padding), the Cipher Feedback (CFB) mode yielded the slowest encryption-decryption times. Again, Cipher block chaining (CBC) and Electronic codebook (ECB) modes proved to be the fastest modes of operation. Compared with other block cipher algorithms, like AES or Blowfish, 3DES was the slowest option. For both padding schemes, CFB mode required the most CPU time to encrypt and decrypt the text files.

Table 3
Text files encryption-decryption times (in milliseconds) using 3DES cipher with 128- and 192-bit key sizes on Samsung Galaxy S9 Plus device

Cipher	Key size	Padd.	Mode	0.01mb	0.1mb	1mb	10mb
TripleDES	128 bits	PKCS5Padding	CBC	8.014	38.017	278.893	2636.159
			CFB	15.855	45.836	392.096	4782.926
			CTR	10.496	40.952	355.713	3707.193
			CTS	8.719	40.190	328.008	3098.009
			ECB	12.063	41.688	334.711	3469.213
			OFB	9.188	52.387	364.591	3805.269
		ISO10126Padding	CBC	12.135	45.153	326.483	3001.291
			CFB	8.996	49.457	387.517	4661.737
			CTR	9.024	51.992	361.987	3958.649
			CTS	8.431	34.875	337.515	3560.789
			ECB	10.191	50.670	625.069	3626.286
			OFB	9.288	46.560	375.603	4303.548
	192 bits	PKCS5Padding	CBC	8.757	38.029	286.819	2439.856
			CFB	9.193	52.461	389.830	3569.959
			CTR	8.810	49.678	460.091	3625.130
			CTS	7.083	47.226	481.949	3045.118
			ECB	9.218	37.355	347.111	3218.112
			OFB	11.005	62.211	518.042	4156.313
		ISO10126Padding	CBC	8.215	47.372	480.989	3581.662
			CFB	11.972	43.547	394.526	3521.521
			CTR	9.060	52.343	380.980	3832.504
			CTS	9.459	45.882	348.359	3109.964
			ECB	8.544	45.172	333.698	3317.770
			OFB	9.130	54.028	416.254	4246.282

Similar to the AES algorithm, the fastest mode of operation was the CBC mode.

The 3DES cipher unarguably achieved the lowest performances of all tested algorithms on both devices.

As shown in Table 4, this algorithm was especially inefficient on less-powerful Galaxy A20e device. When the results are compared to the ones obtained from Galaxy S9+ device, the encryption-decryption process was significantly slower.

Table 4
Text files encryption-decryption times (in milliseconds) using 3DES cipher with 128- and 192-bit key sizes on Samsung Galaxy A20e device

Cipher	Key size	Padd.	Mode	0.01mb	0.1mb	1mb	10mb
TripleDES	128 bits	PKCS5Padding	CBC	16.462	63.903	624.220	5733.306
			CFB	21.330	149.835	1556.935	14280.901
			CTR	23.559	140.603	1449.789	13323.153
			CTS	20.792	133.651	1382.435	12717.155
			ECB	20.917	124.516	1292.808	11822.519
			OFB	21.504	144.113	1494.541	13745.526
		ISO10126Padding	CBC	21.353	133.939	1373.051	12689.928
			CFB	22.325	150.211	1558.152	14272.954
			CTR	21.501	140.012	1449.044	13321.714
			CTS	23.388	135.618	1393.489	12786.035
			ECB	20.647	125.334	1292.286	11831.700
			OFB	21.111	144.385	1507.103	13743.422
	192 bits	PKCS5Padding	CBC	14.426	63.931	629.426	5797.143
			CFB	21.508	150.925	1566.009	14346.761
			CTR	23.434	140.513	1460.812	13357.382
			CTS	20.983	134.858	1396.065	12785.659
			ECB	19.700	125.912	1304.041	11899.184
			OFB	22.403	143.398	1503.530	13750.740
		ISO10126Padding	CBC	19.931	134.445	1393.031	12798.459
			CFB	24.800	151.578	1561.663	14332.682
			CTR	21.513	139.643	1460.804	13350.047
			CTS	20.411	134.723	1395.843	12787.808
			ECB	19.315	126.119	1302.672	11900.799
			OFB	23.142	144.245	1502.863	13742.847

5.2. Results for AES Cipher

Two different key lengths (128-bit and 256-bit keys) were tested with AES cipher. Block cipher modes of operation have been combined with different pad-

ding schemes for each key size. Also, two padding algorithms have been utilized: PKCS5Padding and ISO10126Padding. AES with Galois/Counter Mode (AES-GCM) does not require data being padded out to a specific block size, hence avoiding the need to use the padding mechanism.

Tables 5-6 show the elapsed times required to encrypt and decrypt four different text files with different sizes. These results refer to the Samsung Galaxy S9 Plus device using 128-bit and 256-bit AES, respectively.

As indicated in the Tables 5-6, the fastest encryption-decryption times were acquired when using the CBC or ECB modes and PKCS5Padding algorithm. Although this can be expected, as simplicity and speed are the main advantages of ECB, it is the easiest to cryptanalysis and not semantically secure [33].

The slowest results were achieved when the CFB and PKCS5Padding were utilized for both 128- and 256-bit key length.

In case when CBC or ECB modes are combined with the ISO10126Padding, more time is required for the complete process.

Table 5

Encryption-decryption times (in ms) on Samsung Galaxy S9+ using AES cipher with a 128-bit key

Cipher	Padding	Mode	0.01mb	0.1mb	1mb	10mb
AES (128)	PKCS5Padding	CBC	8.292	20.516	85.123	733.766
		CFB	11.567	38.686	215.520	1884.358
		CTR	14.258	37.402	182.577	1668.644
		CTS	13.798	32.784	159.036	1435.431
		ECB	7.884	20.730	91.778	719.447
		OFB	9.539	30.989	213.572	1798.168
	ISO10126Padding	CBC	10.957	33.691	169.657	1441.750
		CFB	13.438	36.003	213.456	1878.245
		CTR	12.754	32.868	196.588	1823.418
		CTS	11.585	30.594	164.050	1442.604
		ECB	11.041	28.875	160.071	1365.735
		OFB	12.569	44.374	197.944	1788.293
	NoPadding	GCM	7.881	10.716	92.967	1629.124

Table 6

Encryption-decryption times (in ms) on Samsung Galaxy S9+ using AES cipher with a 256-bit key

Cipher	Padding	Mode	0.01mb	0.1mb	1mb	10mb
AES (256)	PKCS5Padding	CBC	8.576	20.575	92.641	665.134
		CFB	13.504	47.553	241.369	2789.291
		CTR	12.620	48.380	230.024	2529.401
		CTS	12.124	42.508	200.889	2130.827
		ECB	8.352	21.756	77.921	641.334
		OFB	13.092	48.520	230.189	2075.595
	ISO10126Padding	CBC	13.116	38.012	193.416	1724.681
		CFB	13.539	41.555	232.291	2157.963
		CTR	13.222	44.977	229.124	2531.739
		CTS	12.489	35.969	210.265	2242.854
		ECB	12.035	38.694	193.067	1670.217
		OFB	13.092	48.520	230.189	2075.595
	NoPadding	GCM	6.829	11.797	102.707	1745.916

To test the performances of the AES cipher on a lower hardware specification device, we have used the Samsung Galaxy A20e device running Android Q (10).

When compared to the Galaxy S9 Plus model, this device was significantly slower when the largest file (10 mb) was utilized in the encryption-decryption process. When compared to the Galaxy S9 Plus model, this device was significantly slower when the largest file (10 mb) was utilized in the encryption-decryption process. However, the difference is not so prominent when smaller text files were employed.

As shown in the Tables 7-8, similar results were obtained in general, where CBC and ECB modes combined with PKCS5Padding yielded the fastest encryption-decryption times.

Moreover, when PKCS5Padding or ISO101216 padding algorithms were applied, the AES cipher transformation using CFB mode proved to be the slowest option.

Additionally, it can be observed that the GCM produced very similar results for both padding algorithms and key lengths. The same could also be said

Table 7
Encryption-decryption times (in ms) on Samsung Galaxy A20e using AES cipher with a 128-bit key

Cipher	Padding	Mode	0.01mb	0.1mb	1mb	10mb
AES (128)	PKCS5Padding	CBC	14.049	27.142	212.166	1835.062
		CFB	16.947	82.375	785.214	7154.419
		CTR	16.566	74.959	717.230	6533.891
		CTS	15.308	65.525	614.725	5592.708
		ECB	10.397	26.920	215.073	1860.548
		OFB	16.302	76.980	729.950	6650.728
	ISO10126Padding	CBC	17.047	67.172	617.375	5586.215
		CFB	17.566	82.456	784.940	7154.308
		CTR	16.814	75.646	717.202	6536.435
		CTS	15.757	65.830	617.002	5594.179
		ECB	15.137	62.667	579.265	5230.725
		OFB	16.810	76.414	730.378	6668.537
	NP	GCM	11.019	17.737	130.136	3867.815

Table 8
Encryption-decryption times (in ms) on Samsung Galaxy A20e using AES cipher with a 256-bit key

Cipher	Padding	Mode	0.01mb	0.1mb	1mb	10mb
AES (256)	PKCS5Padding	CBC	14.281	27.864	211.326	1833.323
		CFB	19.096	96.815	935.547	8565.483
		CTR	18.080	90.030	868.727	7933.309
		CTS	17.601	80.532	767.471	7004.101
		ECB	10.888	27.210	215.363	1867.484
		OFB	17.843	92.601	883.657	8067.842
	ISO10126Padding	CBC	19.030	79.689	769.828	6996.850
		CFB	19.002	97.238	935.561	8560.270
		CTR	18.254	89.922	868.763	7940.664
		CTS	17.438	80.564	767.642	7000.774
		ECB	17.300	76.405	731.040	6631.829
		OFB	18.276	93.241	883.265	8062.224
	NP	GCM	10.536	17.213	129.367	3843.505

for the Output Feedback (OFB) and CypherText Stealing (CTS) modes.

5.3. Results for Blowfish Cipher

Blowfish algorithm is a 64-bit block cipher with a variable-length key from 32-bits (4 bytes) to 448-bits (56 bytes). The symmetric key encryption algorithm is suitable and efficient for hardware implementation [23]. However, like most other ciphers, the Blowfish is often used in software implementations (from Android API level 10+).

The cipher can be used as a drop-in replacement for the DES or International Data Encryption Algorithm (IDEA). The Blowfish algorithm is considered secure and fast. However, the keys should be chosen to be big enough to withstand a brute force attack (e.g., at least 16 bytes).

To test the Blowfish performances, we have provided five different key lengths, including lower and upper key limits. The cipher transformation refers to a combination of two padding schemes and 6 modes of operations.

Tables 9-10 provide performance results obtained with Blowfish cipher on Galaxy S9 Plus device, using PKCS5Padding and ISO101216Padding padding schemes, respectively.

As can be seen, the differences in computational time between the two padding schemes are not prominent. Regarding the comparison to other tested ciphers, it can be observed that the Blowfish algorithm was outperformed by both tested stream ciphers (ChaCha20/NONE/NoPadding and RC4). However, the ChaCha20 encryption-decryption times were somewhat slower when Poly1305 feedback mode was utilized on the Galaxy S9 Plus device.

When compared to the AES, the faster encryption-decryption times were achieved with AES only in a situation when CBC or ECB modes (combined with PKCS5Padding scheme) were utilized.

For other modes of operation and padding schemes, the Blowfish performed faster than AES. This cipher also performed significantly better than the DES and 3DES block ciphers.

Furthermore, results obtained from a lower performance device (Galaxy A20e) show the RC4 stream cipher was faster than the Blowfish block cipher. Unlike results from the Galaxy S9 Plus device, when the Cha-

Table 9

Execution times (in ms) on Galaxy S9 Plus using the Blowfish cipher and PKCS5Padding scheme

Cipher	Mode	Key size	Mode	0.01mb	0.1mb	1mb	10mb
Blowfish	PKCS5Padding	32 bits	CBC	5.854	20.229	141.412	1131.290
			ECB	5.769	19.392	141.143	1136.098
			CTR	5.849	20.297	140.236	1138.478
			CTS	6.001	19.557	143.498	1141.933
			OFB	5.591	19.693	148.471	1150.964
			CFB	6.512	21.295	141.510	1132.870
		64 bits	CBC	6.065	21.246	164.881	1356.558
			ECB	6.336	21.693	145.674	1112.461
			CTR	5.912	21.302	151.798	1118.786
			CTS	6.366	21.458	150.408	1133.411
			OFB	6.303	23.171	149.976	1226.121
			CFB	6.278	21.832	153.535	1364.487
		128 bits	CBC	6.400	20.407	134.003	1138.683
			ECB	6.611	20.090	200.049	1185.560
			CTR	6.527	19.504	138.542	1140.158
			CTS	6.328	21.963	138.067	1144.860
			OFB	5.741	19.605	144.946	1132.322
			CFB	6.306	21.492	142.299	1135.524
		256 bits	CBC	6.437	19.335	137.193	1142.071
			ECB	6.374	19.697	143.697	1145.442
			CTR	6.450	19.557	140.677	1145.143
			CTS	6.614	19.464	140.181	1142.245
			OFB	6.445	19.985	143.174	1145.868
			CFB	5.304	19.648	141.276	1137.993
		448 bits	CBC	6.602	21.162	151.377	1154.152
			ECB	6.581	19.753	144.526	1146.532
			CTR	6.810	20.518	139.537	1153.289
			CTS	6.555	19.738	141.133	1156.952
			OFB	6.893	21.289	138.389	1156.594
			CFB	6.488	21.334	138.893	1154.016

Table 10

Execution times (in ms) on Galaxy S9 Plus using the Blowfish cipher and ISO10126Padding scheme

Cipher	Mode	Key size	Mode	0.01mb	0.1mb	1mb	10mb
Blowfish	ISO10126 Padding	32 bits	CBC	6.269	22.117	142.938	1122.703
			ECB	5.970	19.555	143.396	1134.827
			CTR	6.033	19.570	143.298	1146.477
			CTS	5.857	19.237	147.967	1144.946
			OFB	5.928	19.294	141.954	1144.196
			CFB	6.460	21.526	148.725	1169.140
		64 bits	CBC	6.449	21.563	144.054	1122.483
			ECB	6.446	21.656	152.972	1116.980
			CTR	6.627	25.284	198.592	1422.841
			CTS	6.503	23.900	151.577	1125.730
			OFB	6.121	26.943	214.862	1227.251
			CFB	6.341	21.676	152.057	1366.054
		128 bits	CBC	6.517	19.591	139.851	1139.217
			ECB	6.677	19.658	143.144	1447.182
			CTR	6.537	20.261	211.909	1366.366
			CTS	6.540	19.892	144.159	1142.271
			OFB	5.948	20.305	150.111	1148.841
			CFB	6.659	21.828	141.618	1157.637
		256 bits	CBC	6.489	19.307	139.146	1144.456
			ECB	6.212	17.803	139.923	1145.977
			CTR	6.389	20.301	138.888	1140.367
			CTS	6.542	19.365	139.545	1144.296
			OFB	6.535	20.674	144.840	1144.207
			CFB	6.551	19.352	140.601	1145.893
		448 bits	CBC	6.549	25.230	145.262	1159.551
			ECB	6.510	18.973	142.215	1146.360
			CTR	6.806	20.077	136.908	1145.345
			CTS	6.526	19.565	142.961	1144.872
			OFB	6.588	19.705	141.146	1145.214
			CFB	6.631	28.426	185.758	1286.251

Table 11

Execution times on Galaxy A20e – Blowfish/ PKCS5Padding

Cipher	Mode	Key size	Mode	0.01mb	0.1mb	1mb	10mb
Blowfish	PKCS5Padding	32 bits	CBC	12.617	53.433	512.164	4572.016
			ECB	12.474	53.532	511.581	4583.087
			CTR	12.135	53.027	515.317	4573.389
			CTS	11.996	52.743	511.898	4572.733
			OFB	12.226	53.062	511.496	4573.029
			CFB	15.240	53.701	514.172	4574.275
		64 bits	CBC	15.236	54.172	511.211	4602.183
			ECB	12.522	53.438	512.532	4610.807
			CTR	12.383	54.221	512.986	4609.036
			CTS	11.568	53.781	513.601	4611.579
			OFB	12.176	53.421	513.691	4602.532
			CFB	12.051	53.196	513.171	4609.774
		128 bits	CBC	12.198	52.979	511.949	4571.927
			ECB	12.405	53.291	512.451	4582.290
			CTR	12.394	54.315	512.899	4567.503
			CTS	12.298	53.779	513.442	4567.218
			OFB	12.466	52.691	512.847	4566.304
			CFB	12.533	53.114	512.686	4572.589
		256 bits	CBC	15.668	53.326	511.889	4609.992
			ECB	15.383	53.790	509.001	4585.996
			CTR	12.481	53.436	510.949	4597.795
			CTS	12.089	54.371	510.067	4596.939
			OFB	12.287	54.318	510.857	4596.757
			CFB	12.356	53.374	510.655	4597.682
		448 bits	CBC	14.885	54.143	513.035	4626.042
			ECB	15.390	59.229	514.685	4623.143
			CTR	12.081	54.727	512.355	4606.192
			CTS	12.328	54.411	512.502	4599.046
			OFB	12.412	54.115	511.376	4594.384
			CFB	12.399	53.870	514.822	4574.546

Table 12

Execution times on Galaxy A20e – Blowfish/ ISO10125Padding

Cipher	Mode	Key size	Mode	0.01mb	0.1mb	1mb	10mb
Blowfish	ISO10126 Padding	32 bits	CBC	15.772	53.759	511.823	4566.677
			ECB	11.952	53.133	512.358	4575.922
			CTR	12.125	53.509	511.783	4576.918
			CTS	12.235	52.981	512.146	4571.054
			OFB	12.296	54.389	512.645	4572.524
			CFB	12.425	52.966	512.578	4574.776
		64 bits	CBC	12.348	55.184	512.223	4653.591
			ECB	12.486	52.770	513.138	4613.991
			CTR	12.430	53.715	513.576	4608.366
			CTS	12.476	53.638	513.132	4605.202
			OFB	12.321	53.001	512.426	4604.305
			CFB	12.500	53.692	512.933	4605.954
		128 bits	CBC	12.534	53.275	513.890	4566.651
			ECB	12.542	53.955	513.203	4565.278
			CTR	12.415	53.838	513.151	4566.292
			CTS	12.431	54.018	512.774	4566.395
			OFB	12.272	53.707	513.937	4569.615
			CFB	12.278	53.973	514.334	4582.273
		256 bits	CBC	12.506	53.358	512.671	4617.118
			ECB	12.370	54.076	541.170	4594.027
			CTR	12.387	53.188	510.560	4592.074
			CTS	12.265	53.655	510.923	4597.685
			OFB	12.428	53.351	511.144	4596.208
			CFB	12.447	54.687	510.613	4594.699
		448 bits	CBC	12.130	53.908	514.253	4651.642
			ECB	11.435	53.734	514.261	4627.384
			CTR	12.341	54.574	511.171	4602.165
			CTS	12.355	53.836	512.159	4604.905
			OFB	12.463	54.071	511.610	4601.230
			CFB	12.471	52.950	514.262	4574.347

Cha20 stream cipher was combined with different modes of operation (NONE and Poly1305), the Blowfish algorithm was outperformed in both cases. When compared to the PBE algorithms, this cipher proved to be the faster option on both smartphone devices.

Tables 11-12 provide results obtained by the Blowfish cipher on Galaxy A20e device, using PKCS5Padding and ISO101216Padding padding schemes, respectively. It can be observed that the obtained CPU utilization times are very similar for both padding schemes.

5.4. Results for RC4 (ARC4) Cipher

The RC4 (or ARCFOUR, ARC4) is a stream cipher using variable-sized keys. The supported key length is between 1 byte and 256 bytes, conventionally used from 40 bits (5 bytes) to 2048 bits (256 bytes).

Due to the fact that this cipher does not encrypt in blocks, the number of output bytes is exactly equal to the number of input bytes. Also, RC4 does not utilize initialization vectors, which are only applied in block cipher algorithms. As mentioned, the Bouncy Castle versions of parameters and many algorithms have been deprecated as of Android 9. For this reason, several changes related to cryptography were introduced. Additional implementations of algorithm parameters are provided in Conscrypt [6]. The Conscrypt implementation of the RC4 cipher allows users to specify either RC4/ECB/NoPadding or RC4/NONE/NoPadding. However, it should be mentioned that using NONE as a mode of operation is supported only on API level 28 and higher.

The RC4 outperformed all ciphers we have tested. Table 13 presents the outcomes from the Samsung Galaxy S9 Plus device. Very similar results were obtained for all four text file sizes. It can be observed that neither the key size nor padding scheme significantly affected encryption-decryption times.

Table 14 provides the RC4 encryption-decryption times acquired by the less powerful Galaxy A20e device. The identical algorithm modes and key sizes were employed as in the Galaxy S9 Plus model. Likewise, the results show no significant variations regarding CPU time when different key lengths or cipher modes are utilized. Similar outcomes were acquired with the ECB and NONE modes of operation.

In case when larger text files are encrypted, the difference between devices is more prominent. However, it can be seen that results for smaller text files do not differ significantly on different tested mobile devices

Table 13

Results of the encryption-decryption process (in ms) using the RC4 cipher, tested on Galaxy S9 Plus

Cipher	Mode	Key size [bits]	0.01mb	0.1mb	1mb	10mb
RC4	NONE	40	6.057	16.321	81.074	690.160
		64	4.482	12.660	114.194	682.013
		128	5.579	11.714	80.024	688.027
		256	5.238	11.429	79.768	695.925
		512	5.594	17.840	78.545	686.746
		1024	5.480	18.637	78.044	694.424
		2048	5.353	13.617	79.490	687.553
	ECB	40	5.333	12.711	89.268	716.045
		64	5.876	17.949	80.210	690.065
		128	5.750	12.706	91.216	715.133
		256	6.043	12.565	78.509	693.585
		512	5.505	17.372	80.104	685.431
		1024	5.507	17.734	79.616	684.982
		2048	5.787	17.924	99.176	764.131

Table 14

Results of the encryption-decryption process (in ms) using the RC4 cipher, tested on Galaxy A20e

Cipher	Mode	Key size [bits]	0.01mb	0.1mb	1mb	10mb
RC4	NONE	40	8.388	26.033	219.039	1920.510
		64	9.690	25.221	219.065	1925.623
		128	8.406	24.672	218.916	1937.155
		256	11.673	25.129	219.809	1929.744
		512	8.928	24.861	219.548	1927.004
		1024	8.287	25.977	219.097	1924.540
		2048	9.603	24.491	218.696	1922.298
	ECB	40	12.054	26.309	217.068	1924.771
		64	12.130	26.590	221.323	1965.724
		128	8.943	25.761	218.692	1926.733
		256	11.986	25.917	215.167	1912.890
		512	8.777	26.020	220.483	1962.694
		1024	8.695	26.442	220.398	1962.855
		2048	8.482	26.136	218.081	1925.217

5.5. Results for ChaCha20 Cipher

The Android operating system provides support for the ChaCha stream cipher with 20 rounds, 96-bit nonce, and 32-bit counter, as described in [3]. To encrypt the plain text, the cipher uses a 256-bit key and IV (initialization value, nonce + initial count). The nonce and secret key must be unique for each encryption.

For the encryption process, the key, nonce, and counter must be the same. Unlike the AES, it is not vulnerable to cache-collision-timing attacks. The ChaCha20 cipher has been reported faster than the AES in software-only implementations when the CPU does not provide dedicated AES instructions [17]. However, it must be noted that the cipher is supported only on API levels 28 or higher. The ChaCha20 is a relatively new stream cipher that can replace the older, insecure RC4 stream cipher on Android devices. The Figures 2-3 illustrate the text files encryption-decryption times utilizing ChaCha20 and ChaCha20-Poly1305 cipher transformations on the Samsung Galaxy S9 Plus and Galaxy A20e, respectively.

Four different text file sizes were tested in encryption-decryption process: 10, 1, 0.1, and 0.01 megabytes.

The ChaCha20 cipher obtained similar results as the AES algorithm with 128-bit key size on the Galaxy S9 Plus device. The results are even more comparable when CBC or ECB modes are combined with the PKCS5Padding algorithm. Additionally, the ChaCha20-Poly1305 acquired somewhat faster times when compared to the remaining AES modes and paddings. However, this cipher required considerably more time to encrypt-decrypt the text files when compared to the ChaCha20.

Figure 2

Difference between ChaCha20 and ChaCha20-Poly1305 cipher transformations on the Samsung Galaxy S9 Plus. The encryption-decryption process includes four different file sizes (0.01, 0.1, 1, and 10 megabytes). The results are in milliseconds

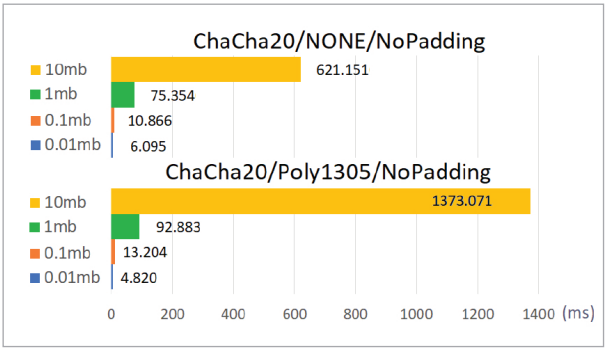
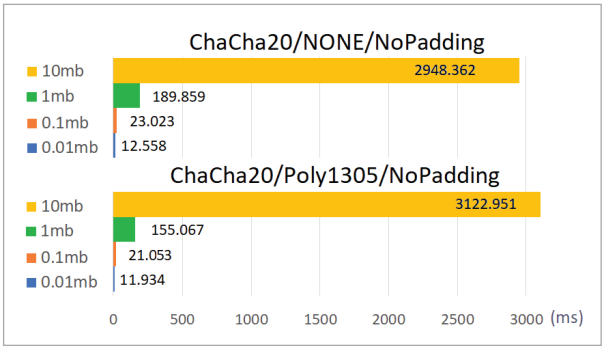


Figure 3

Difference between ChaCha20 and ChaCha20-Poly1305 cipher transformations on the Samsung Galaxy A20e. The encryption-decryption process includes four different file sizes (0.01, 0.1, 1, and 10 megabytes). The results are in milliseconds



Regarding the Galaxy A20e device with lower hardware specifications, the results obtained using ChaCha20 were significantly enhanced when compared to the AES cipher. This is true for both 128- and 256-bit key lengths. As depicted in Figure 3, both the ChaCha20 and ChaCha20-Poly1305 encrypted and decrypted the text files significantly faster than the AES on a less powerful device.

The only case when the AES was faster is for the PKCS5Padding algorithm along with CBC or ECB modes. When compared to another tested stream cipher (RC4), ChaCha20/NONE/NoPadding cipher is faster only on the device with higher hardware specifications.

The Figures 4-5 present CPU utilization times (in milliseconds) of an encryption-decryption process on

Figure 4

CPU utilization times (in milliseconds) of an encryption-decryption process on Galaxy S9 Plus device using ChaCha20 cipher, grouped by the same text file sizes

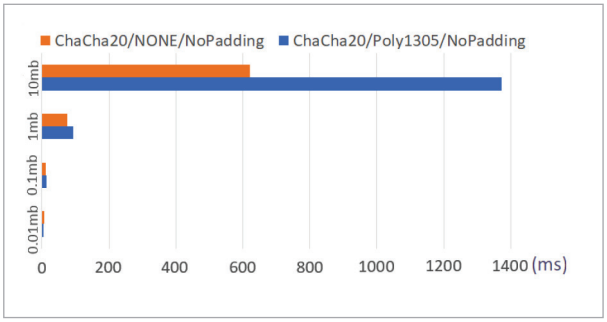


Figure 5
CPU utilization times (in milliseconds) of an encryption-decryption process on Galaxy A20e device using ChaCha20 cipher, grouped by the same text file sizes

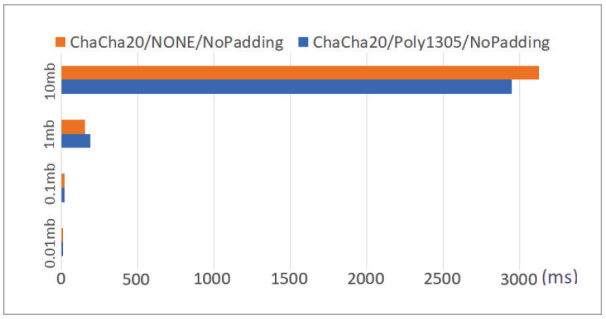
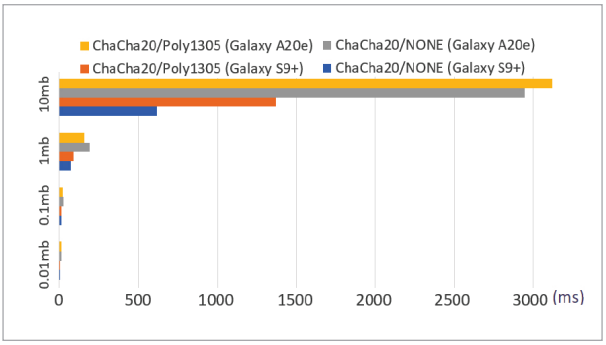


Figure 6
CPU utilization times (in milliseconds) of an encryption-decryption process on both devices using ChaCha20 cipher, grouped by the same text file sizes



Galaxy S9 Plus and Galaxy A20e devices, respectively. The results are obtained using ChaCha20 cipher and grouped by the same text file sizes.

For comparison, Figure 6 shows the same results simultaneously for both devices.

5.6. Results for PBE Algorithms

The PBE (Password-Based Encryption) algorithms are techniques that allow generating a secret key based on a password provided by the end-user. This feature is convenient if the users want to encrypt files based on the easily memorable passwords [10]. In order to produce key bytes as random and unpredictable as possible, additional parameters (i.e., salt and iteration count) are provided. The salt is a randomization system aiming to prevent dictionary attacks.

Creating a table of possible passwords ("dictionary attack") would be challenging since there would be several possible keys for each password. Consequently, searching through passwords separately for each salt would be limited.

The iteration count complicates the key derivation function by performing a number of iterations, greatly increasing the cost of exhaustive password search attacks. A minimum of 1000 iterations is recommended [29].

The PBE algorithm combines the specified message digest or pseudo-random function and symmetric encryption algorithm [5]. In total, we have tested 17 different combinations of PBE algorithm parameters, defined as *PBEWith<digest>And<encryption>*. For every tested algorithm, two different values (1000 and 10000) were provided as an iteration count.

Table 15 presents the outcomes of the PBE encryption-decryption process using the Samsung Galaxy S9 Plus device. The fastest encryption time was obtained by utilizing SHA as message digest and RC4 as encryption algorithm. When compared to other ciphers, ChaCha20 (NONE used as feedback mode) and RC4 algorithms achieved slightly faster performances.

The AES cipher combined with the PKCS5Padding scheme and two modes of operation (CBC or ECB) also performed faster. In case when 3DES cipher was used together with SHA message digest, the poorer outcomes were obtained. Table 16 shows the results from Samsung Galaxy A20e device. Again, the PBE with an RC4 encryption algorithm and SHA message digest yielded the fastest outcomes, while SHA and 3DES cipher resulted in the slowest encryption-decryption times.

However, the difference between various PBE algorithm parameters is more prominent when compared to results obtained by the Galaxy S9 Plus device. As can be observed, the CPU utilization was not significantly affected by diverse iteration count values for large file sizes.

5.7. Discussion

In this section, we provide a concluding analysis of the comparison of all tested algorithms with respect to different file sizes, utilized devices, and cipher types (block or stream). Also, we compare our obtained results with the with the results of the other authors.

Table 15
Encryption-decryption times (in ms) using the PBE (Password-Based Encryption) algorithms on the Galaxy S9 Plus device

Cipher	Iter. count	0.01mb	0.1mb	1mb	10mb
PBEWithSHA AndAES_128	1 000	19.806	34.212	173.277	1434.596
	10 000	78.297	89.563	229.934	1499.064
PBEWithSHA AndAES_256	1 000	26.652	39.760	207.809	1820.101
	10 000	108.075	124.430	304.382	1874.394
PBEWithSHA256 AndAES_128	1 000	20.702	34.790	170.002	1457.359
	10 000	81.494	97.489	235.637	1672.277
PBEWithSHA256 AndAES_256	1 000	14.145	35.740	264.713	1883.595
	10 000	95.215	117.602	336.835	2047.895
PBEWithMD5 AndAES_128	1 000	6.292	22.948	166.090	1588.924
	10 000	6.579	33.068	256.305	1550.301
PBEWithMD5 AndAES_256	1 000	10.111	27.987	200.267	1759.074
	10 000	6.391	26.593	211.016	1994.764
PBEWithSHA AndRC4_40	*1 000*	*9.542*	*20.872*	*105.707*	*870.129*
	10 000	52.521	54.980	146.929	910.693
PBEWithSHA AndRC4_128	1 000	15.096	21.221	116.367	890.555
	10 000	50.801	53.756	146.635	909.834
PBEWithMD5 AndDES	1 000	17.238	31.646	165.883	1461.346
	10 000	53.142	65.061	206.404	1554.473
PBEWithSHA1 AndDES	1 000	14.731	28.157	176.767	1486.448
	10 000	51.330	73.169	288.389	1614.502
PBEWithSHA And2-KeyTripleDES	1 000	20.999	42.252	333.897	2999.435
	10 000	67.138	92.318	385.760	3121.791
PBEWithSHA And3-KeyTripleDES	1 000	20.861	59.786	466.927	3128.143
	10 000	96.863	124.102	437.986	3717.820
PBEWithSHA AndTwoFish	1 000	22.285	34.766	158.179	1234.499
	10 000	108.584	123.751	247.335	1557.562
PBEWithSHA AndRC2_40	1 000	22.097	37.535	217.236	1832.921
	10 000	80.745	98.921	286.657	1939.613
PBEWithSHA AndRC2_128	1 000	20.068	37.899	226.817	1885.972
	10 000	78.938	97.417	287.953	1968.008
PBEWithMD5 AndRC2	1 000	17.552	35.627	215.023	1877.257
	10 000	51.924	64.403	249.711	1916.396
PBEWithSHA1 AndRC2	1 000	15.072	35.318	224.487	2018.025
	10 000	55.614	82.214	326.354	2017.591

Table 16
Encryption-decryption times (in ms) using the PBE (Password-Based Encryption) algorithms on the Galaxy A20e device

Cipher	Iter. count	0.01mb	0.1mb	1mb	10mb
PBEWithSHA AndAES_128	1 000	32.654	80.360	628.963	5518.971
	10 000	187.389	234.848	788.164	5680.162
PBEWithSHA AndAES_256	1 000	40.116	104.833	789.520	6977.135
	10 000	270.890	335.590	1005.797	7201.701
PBEWithSHA256 AndAES_128	1 000	30.140	81.144	631.012	5532.806
	10 000	191.064	239.089	791.905	5670.330
PBEWithSHA256 AndAES_256	1 000	34.353	95.141	787.093	6968.974
	10 000	192.724	253.075	935.032	7139.641
PBEWithMD5 AndAES_128	1 000	13.390	63.409	615.560	5517.451
	10 000	13.126	63.991	617.144	5512.636
PBEWithMD5 AndAES_256	1 000	14.928	78.685	771.684	6959.989
	10 000	14.885	77.984	771.528	6964.944
PBEWithSHA AndRC4_40	1 000	19.166	36.574	260.300	2257.256
	10 000	97.078	115.399	343.639	2330.458
PBEWithSHA AndRC4_128	1 000	19.066	36.873	260.910	2256.598
	10 000	96.665	116.136	342.699	2326.426
PBEWithMD5 AndDES	1 000	22.836	71.877	636.696	5702.136
	10 000	103.523	153.235	715.764	5779.626
PBEWithSHA1 AndDES	1 000	21.278	72.077	636.491	5708.364
	10 000	99.160	150.132	711.094	5769.855
PBEWithSHA And2-KeyTripleDES	1 000	38.997	151.666	1411.665	12811.084
	10 000	194.703	306.684	1572.867	12916.847
PBEWithSHA And3-KeyTripleDES	1 000	48.454	161.034	1403.666	12771.224
	10 000	279.104	397.151	1650.404	12995.879
PBEWithSHA AndTwoFish	1 000	38.210	81.119	540.867	4661.399
	10 000	274.354	315.177	779.612	4895.116
PBEWithSHA AndRC2_40	1 000	31.900	100.047	847.923	7582.186
	10 000	190.087	259.983	997.371	7731.347
PBEWithSHA AndRC2_128	1 000	33.797	99.846	845.920	7563.939
	10 000	188.177	256.292	1002.332	7723.197
PBEWithMD5 AndRC2	1 000	26.158	92.270	838.803	7570.298
	10 000	104.060	171.107	916.466	7638.832
PBEWithSHA1 AndRC2	1 000	24.607	92.863	838.941	7574.212
	10 000	101.501	169.478	916.441	7650.544

Regarding the smallest text file size (0.01 megabytes), the RC4 cipher proved to be the fastest among all other ciphers on both tested devices. Considering the other three text file sizes (in particular, 0.1, 1, and 10 megabytes), the AES cipher outperformed all other algorithms on a less powerful Galaxy A20e device. Furthermore, when compared to the DES cipher, AES algorithm was found to be slower option for smaller file sizes. This conclusion confirms our findings with the results obtained by Olaleye [24].

Although tested on two different platforms (desktop and mobile), the results of Nadeem and Javed [21] study are very comparable to ours which were obtained on mobile Android devices. Among the four algorithms tested, the results presented show that Blowfish is the fastest algorithm. The same is true for the results of our performance tests, especially for larger file sizes. The algorithms AES and DES performed quite similarly on both platforms, while the 3DES cipher was the slowest option, as in our study.

Rouaf and Yousif [30] additionally validated the AES and DES results we have obtained in our study. However, they have performed an experiment using three Android smartphones with hardware specifications that varied from those utilized in our study.

Additionally, Haque et al. [15] and Salama et al. [32] reached the same conclusions, in which our findings also corroborate theirs. Namely, it was unsurprising that a smaller key size (128 bits) results in a quicker execution time for AES. Also, as with our findings obtained on Android platform, larger key sizes (256 bits) resulted in greater AES time consumption. As a result, a higher key size should be used to support the smaller packet sizes typically required by mobile devices.

As with the ChaCha20 algorithm, given the large text files (i.e., 1 and 10 megabytes), the stream cipher was confirmed to be the fastest option on a device with more powerful hardware specifications (Galaxy S9 Plus). On a smartphone with less powerful hardware, only the RC4 cipher obtained better performance.

Similar to Nie et al. [22] research who compared Blowfish and DES ciphers, results we have obtained also match the authors' findings. Namely, the Blowfish has outperformed DES algorithm on both devices and across all file sizes we have evaluated. Our findings on mobile platform are also relatable to a comparison conducted by Patil et al. [26] and Thirupalu and Reddy [38]. Between AES, DES, and 3DES, the results suggest that Blowfish is the optimal option because to its minimal CPU and memory requirements.

The outcomes of research obtained by Sahu and Kushwaha [31] and Panda [25] are similarly comparable to our results related to the AES and Blowfish ciphers. As was the case in our research on Android platform, AES and Blowfish produced findings that were generally similar. Also, when encryption and decryption speeds are compared, the Blowfish cipher outperformed the DES cipher.

The TripleDES cipher assuredly has the lowest performance in view of all four text file sizes when compared to other algorithms tested on both smartphones. While the results we got are exclusive to the Android mobile OS, the findings in related works using 3DES method obtained on desktop computers are consistent with ours.

Similar to our research, Grgić et al. [14] utilized Android mobile platform to evaluate the performance of three symmetric ciphers (AES, DES, and 3DES). However, unlike our research, the authors considered only two modes of operation: ECB and CTR. Regardless, their results and conclusions further corroborated ours. Namely, when employed in the ECB mode, the AES cipher surpasses both the DES and 3DES cryptosystems.

Furthermore, the PBE algorithm with AES (128) encryption cipher and MD5 message digest yielded the fastest outcomes when the smallest text file is taken into consideration. For other file sizes, the best performances were obtained using the PBE algorithm with RC4 encryption cipher and SHA message digest. The slowest results were acquired when the PBE algorithm combined TripleDES cipher and SHA message digest as algorithm parameters.

With regard to different device models, the best option on a less powerful Galaxy A20e device was the AES block cipher with 128- or 256-bit keys provided. However, the stream ciphers (ChaCha20 and RC4) achieved the leading results on the Galaxy S9 Plus device. As mentioned, TripleDES was the slowest cipher on both devices, utilizing the most CPU time.

Relating to the types of encryption algorithm, the AES proved to be the fastest block cipher on both devices. Apropos of the stream ciphers, the RC4 algorithm was the fastest option on the slower Galaxy A20e model,

while the ChaCha20 attained the best performances on the Galaxy S9 Plus device.

6. Conclusion

This paper provides a detailed study of the popular symmetric key encryption algorithms available on the Android operating system. Various ciphers are designed to provide different features and performance characteristics. We have tested six block and stream ciphers, as well as Password-Based Encryption (PBE) ciphers with different algorithms' parameters. Their performances were compared by measuring CPU utilization while encrypting and decrypting text files of varying sizes. Two different devices were utilized: Samsung Galaxy S9 Plus (Android 9) and Samsung Galaxy A20e (Android 10).

Results obtained by both devices clearly show the advantage of RC4 and ChaCha20 stream ciphers over other encryption algorithms. RC4 proved to be the fastest option on the less powerful Galaxy A20e model, while it was outperformed by ChaCha20 cipher on S9 Plus device.

Regarding the block ciphers, the AES algorithm obtained the best performances on both devices. The 3DES cipher yielded the poorest performances, especially in the case of a device with lower hardware specifications. When referring to PBE ciphers, the best performances were obtained with the RC4 encryption algorithm and SHA message digest. In case when SHA was combined with 3DES, the algorithm performs encryption-decryption process considerably slower than other ciphers.

Acknowledgement

This work was fully supported by the Croatian Science Foundation under the project IP-2018-01-3739, EU Horizon 2020 project "National Competence Centres in the Framework of EuroHPC (EUROCC)", IRI2 project "ABsistemDCiCloud" (KK.01.2.1.02.0179), Center for Artificial Intelligence and Cybersecurity, University of Rijeka under the projects uniri-tehnic-18-17 and uniri-tehnic-18-15, Croatian-Slovenian bilateral project BI-HR/20-21-043, and European Cooperation in Science and Technology (COST) under the project CA17137.

References

1. Agrawal, M., Mishra, P. A Comparative Survey on Symmetric Key Encryption Techniques. International Journal on Computer Science and Engineering, 2012, 4(5).
2. Bernstein, D. J. Chacha, a Variant of Salsa20. https://cr.yp.to/chacha/chacha-20080120.pdf. Accessed on November 30, 2020.
3. Bernstein, D. J. Rfc 7539 - Chacha Stream Cipher. https://tools.ietf.org/html/rfc7539. Accessed on November 12, 2020.
4. Cipher. https://developer.android.com/reference/javax/crypto/Cipher. Accessed on November 6, 2020.
5. Cipher section of the Java Cryptography Architecture Standard Algorithm Name Documentation. https://docs.oracle.com/javase/8/docs/technotes/guides/security//StandardNames.html#Cipher. Accessed on November 30, 2020.
6. Conscrypt - A Java Security Provider. https://github.com/google/conscrypt. Accessed on November 6, 2020.
7. Cryptography. https://developer.android.com/guide/topics/security/cryptography. Accessed on November 6, 2020.
8. Cryptography Changes in Android P. https://android-developers.googleblog.com/2018/03/cryptography-changes-in-android-p.html. Accessed on November 5, 2020.
9. Debug.ThreadCpuTimeNanos. https://developer.android.com/reference/android/os/Debug#threadCpuTimeNanos (). Accessed on November 5, 2020.
10. Dhany, H., Izhari, F., Fahmi, H., Tulus, T., Sutarman, M. Encryption and Decryption using Password Based Encryption, MD5, and DES. International Conference on Public Policy, Social Computing and Development 2017 (ICOPOSDev 2017), 2018. https://doi.org/10.2991/icoposdev-17.2018.57
11. Do the ChaCha: Better Mobile Performance with Cryptography. https://blog.cloudflare.com/do-the-chacha-better-mobile-performance-with-cryptography/. Accessed on November 30, 2020.
12. Fenyi, A., Davis, J. G., Riverson, K. Comparative Analysis of Advanced Encryption Standard, Blowfish and Rivest Cipher 4 Algorithms. International Journal of Innovative Research and Development, 2014.

13. González, D., Esparza, O., Muñoz, J., Alins, J., Mata, J. Evaluation of Cryptographic Capabilities for the Android Platform. International Conference on Future Network Systems and Security, 2015. https://doi.org/10.1007/978-3-319-19210-9_2

14. Grgić, K., Kovačevic, Z., Čik, V. K. Performance Analysis of Symmetric Block Cryptosystems on Android Platform. 2017 International Conference on Smart Systems and Technologies (SST), 2017, 155-159. https://doi.org/10.1109/SST.2017.8188687

15. Haque, M. E., Zobaed, S., Islam, M. U., M. Areef, F. Performance Analysis of Cryptographic Algorithms for Selecting Better Utilization on Resource Constraint Devices. 2018 21st International Conference of Computer and Information Technology (ICCIT), 2018, 1-6.https://doi.org/10.1109/ICCITECHN.2018.8631957

16. KeyGenerator, Android Developers Website. https://developer.android.com/reference/javax/crypto/KeyGenerator. Accessed on November 30, 2020.

17. Langley, A., Chang, W., Mavrogiannopoulos, N., Strömberg-son, J. and Josefsson, S. ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS), 2016, RFC7905, 1-8. https://doi.org/10.17487/RFC7905

18. Malina, L., Clupek, V., Martinasek, Z., Hajny, J., Oguchi, K., Zeman, V. Evaluation of Software-Oriented Block Ciphers on Smartphones. FPS 2013: Revised Selected Papers of the 6th International Symposium on Foundations and Practice of Security, 2013, 8352, 353-368. https://doi.org/10.1007/978-3-319-05302-8_22

19. Masoud, M., Jannoud, I., Ahmad, A., Alshoubaki, H. The Power Consumption Cost of Data Encryption in Smartphones. IEEE OSSCOM 2015, 2015. https://doi.org/10.1109/OSSCOM.2015.7372685

20. Montoya B., A. O., Muñoz G., M. A., Kofuji, S. T. Performance Analysis of Encryption Algorithms on Mobile Devices. 2013 47th International Carnahan Conference on Security Technology (ICCST), 2013, 1-6. https://doi.org/10.1109/CCST.2013.6922058

21. Nadeem, A., Javed, M. A Performance Comparison of Data Encryption Algorithms. Information and Communication Technologies, 2005, 2015. https://doi.org/10.1109/ICICT.2005.1598556

22. Nie, T., Song, C., Zhi, X. Performance Evaluation of DES and Blowfish Algorithms. 2010 International Conference on Biomedical Engineering and Computer Science, 2010, 1-4. https://doi.org/10.1109/ICBECS.2010.5462398

23. Nie, T., Zhang, T. A study of DES and Blowfish Encryption Algorithm. TENCON 2009 - 2009 IEEE Region 10 Conference, 2009. https://doi.org/10.1109/TENCON.2009.5396115

24. Olaleye, S. B. A Security Study to Compare Cryptographic Algorithms Based on Performance in Mobile Cloud Computing. Journal of Basic and Applied Engineering Re-search, 2017, 58-63.

25. Panda, M. Performance Analysis of Encryption Algorithms for Security. 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016, 278-284. https://doi.org/10.1109/SCOPES.2016.7955835

26. Patil, P., Narayankar, P., Narayan D.G., Meena S.M. A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA, and Blowfish, 2016, Procedia Computer Science 78:617-624. https://doi.org/10.1016/j.procs.2016.02.108

27. Rachmat, N., Samsuryadi Performance Analysis of 256-bit AES Encryption Algorithm on Android Smartphone. Journal of Physics Conference Series, 2019, 1196:012049. https://doi.org/10.1088/1742-6596/1196/1/012049

28. Ratnadewi, R., Adhie, R., Hutama, Y., Ahmar, A., Setiawan, M. Implementation Cryptography Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES) Method in Communication System Based Near Field Communication (NFC). Journal of Physics: Conference Series, 2018. https://doi.org/10.1088/1742-6596/954/1/012009

29. RFC 2898 - Password-Based Cryptography. https://datatracker.ietf.org/doc/html/rfc2898. Accessed on November 29, 2020.

30. Rouaf, M. T. and Yousif, A. Performance Evaluation of Encryption Algorithms in Mobile Devices. 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), 2021, 1-5. https://doi.org/10.1109/CCST.2013.6922058

31. Sahu, S. K., Kushwaha, A. Performance Analysis of Symmetric Encryption Algorithms for Mobile Ad Hoc Network. International Journal of Computer Networks and Communications, 2016, 8(2), 89-101. https://doi.org/10.5121/ijcnc.2016.8208

32. Salama, D.-D., Abd Elkader, H., Hadhoud, M. M. Performance Evaluation of Symmetric Encryption Algorithms. International Journal of Network Security, 2009, 10(3)

33. Schneier, B. Applied Cryptography: Protocols, Algorithms and Source Code in C, 20th Anniversary Edition, Wiley, USA.

34. Srinivasa, O. Performance Analysis of DES and Triple DES. International Journal of Computer Applications, 2015, 130(14), 30-34. https://doi.org/10.5120/ijca2015907190

35. Sumartono, I., Siahaan, A. P. U., Mayasari, N. An Overview of the RC4 Algorithm. IOSR Journal of Dental and Medical Sciences, 2016, 18(6), 2278-661. http://dx.doi.org/10.9790/0661-1806046773

36. Syed Idrus, S. Z., Aljunid, S., Asi, S., Sudin, S., Ahmad, R. Performance Analysis of Encryption Algorithms' Text Length Size on Web Browsers. IJCSNS. 8. 20., 2008.

37. Tayde, S., Siledar, S. File Encryption, Decryption Using AES Algorithm in Android Phone. International Journal of Advanced Research in Computer Science and Software Engineering 5, 2015.

38. Thirupalu, U., Reddy, E. K. Performance Analysis of Cryptographic Algorithms in the Information Security. International Journal of Engineering Research and Technology, 2020.

39. Yao, H., Lian, L., Fan, Y., Qingzhong, L., Yan, X. 2013 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Networks, 2013, 405-409. http://dx.doi.org/10.1109/MSN.2013Brodić, D. Advantages of the Extended Water Flow Algorithm for Handwritten Text Segmentation. In: Kuznetsov, S. O. et al. (Eds.), Pattern Recognition and Machine Intelligence, Lecture Notes in Computer Science, 6744, Springer, Belin-Heidelberg, 2011, 418-423. https://doi.org/10.1007/978-3-642-21786-9_68